pywws Documentation

Release 13.12.dev1117

Jim Easterbrook

19 apr 2017

Indice

Software Python per Stazione Meteo USB senza filo
http://pythonhosted.org/pywws
http://jim-easterbrook.github.com/pywws/
http://github.com/jim-easterbrook/pywws/

Questo documento è tradotto nelle seguenti lingue (Le versioni diverse dall' inglese possono non essere complete o aggiornate):

- English
- Français – translated by Jacques Desroches
- Italiano – translated by Edoardo

CAPITOLO 1

Introduzione

pywws è un programma sviluppato in Python per leggere, memorizzare ed elaborare i dati delle stazioni meteo USB senza filo per i seguenti tipi Elecsa AstroTouch 6975, Watson W-8681, WH-1080PC, WH1080, WH1081, WH3080 ecc. Suppongo che tutti i modelli che funzionano con il software EasyWeather per Windows siano compatibili, ma non posso garantirlo.

Il programma è concepito per funzionare con bassi consumi, piccole memorie come un router. Il suo scopo è creare grafici e pagine Web con i dati letti dalla stazione meteo, generalmente aggiornati ad ogni ora. Si possono inviare i dati a servizi come Weather Underground e postare i messaggi su Twitter.

Ho scritto il programma per le mie necessità, ma ho fatto in modo che sia adattabile alle necessità altrui. Voi potete modificare alcuni o tutti i moduli, o scriverne dei nuovi, per ottenere esattamente quello che desideri.La ragione per cui è stato scelto Python è che rende le modifiche facili. Non abbiate paura, mettetevi alla prova..

CAPITOLO 2

Requisiti

Il software necessario per eseguire pywws dipende da cosa si intende fare con esso. È necessario Python 2.5 o successiva – Python 3 è parzialmente supportato, alcune funzionalità dipendono dalle librerie che non sono ancora state portate in Python 3.

Per ulteriori dettagli, vedi *Dipendenze*.

CAPITOLO 3

Scaricare una copia di pywws

Semplicemente per installare pywws usate il comando `pip` scaricando direttamente dal sito Python Package Index (PyPI). Nota questo metodo richiede i privilegi di 'root', usate il comando `sudo`:

```
sudo pip install pywws
```

Se non hai i privilegi di root, o non vuoi installare pywws nel sistema, si può scaricare il file zip o tar.gz da PyPI ed estrarre i file in una qualsiasi directory del tuo computer.

I file di PyPI contengono una versione istantanea del software - una nuova versione viene emessa ogni pochi mesi. Se si desidera mantenere aggiornati con gli sviluppi più recenti della pywws, è consigliabile utilizzare `git` per clonare il repository pywws:

```
git clone https://github.com/jim-easterbrook/pywws.git
```

Dopo averlo fatto è possibile compilare i file di localizzazione della documentazione e lingua (che richiederanno le dipendenze `sphinx` e `gettext`):

```
cd pywws
python setup.py msgfmt
python setup.py build_sphinx
```

Questo è facoltativo- - la documentazione è disponibile online se preferisci utilizzare pywws in Inglese.

Per ulteriori dettagli, vedi *Come iniziare con pywws*.

Aggiornare pywws

Il metodo utilizzato per aggiornare pywws dipende da come originariamente l' hai ottenuta. Se hai scaricato un file zip o tar. gz, hai solo bisogno di fare la stessa cosa di nuovo, con la nuova versione, quindi eliminare il vecchio download quando hai finito di impostare quella nuova. (Si noti che l'aggiornamento è molto più facile se non conservate i template, moduli utente e dati meteo nella stessa directory del file scaricato). Gli utenti `git` hanno bisogno del comando `git pull`.Se si è utilizzato `pip` è necessario utilizzare l'opzione di aggiornamento:

```
sudo pip install pywws -U
```

Alcune nuove versioni di pywws hanno cambiato ciò che è memorizzato nei file di dati di riepilogo oraria, giornaliera o mensile. Queste nuove versioni non sono compatibili con i dati elaborati da versioni precedenti. Il *pywws.Reprocess* Rigenera tutti i dati di riepilogo. Questo dovrebbe essere eseguito dopo ogni aggiornamento importante.

CAPITOLO 4

Documentazione

La documentazione è inclusa con pywws ed è anche disponibile online. Un buon punto di partenza è *Come iniziare con pywws* che descrive in dettaglio come installare pywws.

Se avete domande o non risposte nella documentazione, unitevi alla pywws Google mailing list / discussion group e chiedete lì. Si noti che il primo messaggio del gruppo non apparirà immediatamente - nuovi poster devono essere approvati da un moderatore, per evitare messaggi di spam.

Contenuto

Licenza Pubblica Generale GNU

```
                    GNU GENERAL PUBLIC LICENSE
                       Version 2, June 1991

 Copyright (C) 1989, 1991 Free Software Foundation, Inc.
     51 Franklin Street, Fifth Floor, Boston, MA  02110-1301  USA
 Everyone is permitted to copy and distribute verbatim copies
 of this license document, but changing it is not allowed.

                            Preamble

  The licenses for most software are designed to take away your
freedom to share and change it.  By contrast, the GNU General Public
License is intended to guarantee your freedom to share and change free
software--to make sure the software is free for all its users.  This
General Public License applies to most of the Free Software
Foundation's software and to any other program whose authors commit to
using it.  (Some other Free Software Foundation software is covered by
the GNU Library General Public License instead.)  You can apply it to
your programs, too.

  When we speak of free software, we are referring to freedom, not
```

```
price.  Our General Public Licenses are designed to make sure that you
have the freedom to distribute copies of free software (and charge for
this service if you wish), that you receive source code or can get it
if you want it, that you can change the software or use pieces of it
in new free programs; and that you know you can do these things.

  To protect your rights, we need to make restrictions that forbid
anyone to deny you these rights or to ask you to surrender the rights.
These restrictions translate to certain responsibilities for you if you
distribute copies of the software, or if you modify it.

  For example, if you distribute copies of such a program, whether
gratis or for a fee, you must give the recipients all the rights that
you have.  You must make sure that they, too, receive or can get the
source code.  And you must show them these terms so they know their
rights.

  We protect your rights with two steps: (1) copyright the software, and
(2) offer you this license which gives you legal permission to copy,
distribute and/or modify the software.

  Also, for each author's protection and ours, we want to make certain
that everyone understands that there is no warranty for this free
software.  If the software is modified by someone else and passed on, we
want its recipients to know that what they have is not the original, so
that any problems introduced by others will not reflect on the original
authors' reputations.

  Finally, any free program is threatened constantly by software
patents.  We wish to avoid the danger that redistributors of a free
program will individually obtain patent licenses, in effect making the
program proprietary.  To prevent this, we have made it clear that any
patent must be licensed for everyone's free use or not licensed at all.

  The precise terms and conditions for copying, distribution and


    GNU GENERAL PUBLIC LICENSE
   TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION
  0. This License applies to any program or other work which contains
a notice placed by the copyright holder saying it may be distributed
under the terms of this General Public License.  The "Program", below,
refers to any such program or work, and a "work based on the Program"
means either the Program or any derivative work under copyright law:
that is to say, a work containing the Program or a portion of it,
either verbatim or with modifications and/or translated into another
language.  (Hereinafter, translation is included without limitation in
the term "modification".)  Each licensee is addressed as "you".
Activities other than copying, distribution and modification are not
covered by this License; they are outside its scope.  The act of
running the Program is not restricted, and the output from the Program
is covered only if its contents constitute a work based on the
Program (independent of having been made by running the Program).
Whether that is true depends on what the Program does.
  1. You may copy and distribute verbatim copies of the Program's
source code as you receive it, in any medium, provided that you
conspicuously and appropriately publish on each copy an appropriate
copyright notice and disclaimer of warranty; keep intact all the
```

```
notices that refer to this License and to the absence of any warranty;
and give any other recipients of the Program a copy of this License
along with the Program.
You may charge a fee for the physical act of transferring a copy, and
you may at your option offer warranty protection in exchange for a fee.
  2. You may modify your copy or copies of the Program or any portion
of it, thus forming a work based on the Program, and copy and
distribute such modifications or work under the terms of Section 1
above, provided that you also meet all of these conditions:
    a) You must cause the modified files to carry prominent notices
    stating that you changed the files and the date of any change.
    b) You must cause any work that you distribute or publish, that in
    whole or in part contains or is derived from the Program or any
    part thereof, to be licensed as a whole at no charge to all third
    parties under the terms of this License.
    c) If the modified program normally reads commands interactively
    when run, you must cause it, when started running for such
    interactive use in the most ordinary way, to print or display an
    announcement including an appropriate copyright notice and a
    notice that there is no warranty (or else, saying that you provide
    a warranty) and that users may redistribute the program under
    these conditions, and telling the user how to view a copy of this
    License.  (Exception: if the Program itself is interactive but
    does not normally print such an announcement, your work based on
    the Program is not required to print an announcement.)
These requirements apply to the modified work as a whole.  If
identifiable sections of that work are not derived from the Program,
and can be reasonably considered independent and separate works in
themselves, then this License, and its terms, do not apply to those
sections when you distribute them as separate works.  But when you
distribute the same sections as part of a whole which is a work based
on the Program, the distribution of the whole must be on the terms of
this License, whose permissions for other licensees extend to the
entire whole, and thus to each and every part regardless of who wrote it.
Thus, it is not the intent of this section to claim rights or contest
your rights to work written entirely by you; rather, the intent is to
exercise the right to control the distribution of derivative or
collective works based on the Program.
In addition, mere aggregation of another work not based on the Program
with the Program (or with a work based on the Program) on a volume of
a storage or distribution medium does not bring the other work under
the scope of this License.
  3. You may copy and distribute the Program (or a work based on it,
under Section 2) in object code or executable form under the terms of
```

```
Sections 1 and 2 above provided that you also do one of the following:
    a) Accompany it with the complete corresponding machine-readable
    source code, which must be distributed under the terms of Sections
    1 and 2 above on a medium customarily used for software interchange; or,
    b) Accompany it with a written offer, valid for at least three
    years, to give any third party, for a charge no more than your
    cost of physically performing source distribution, a complete
    machine-readable copy of the corresponding source code, to be
    distributed under the terms of Sections 1 and 2 above on a medium
    customarily used for software interchange; or,
    c) Accompany it with the information you received as to the offer
    to distribute corresponding source code.  (This alternative is
    allowed only for noncommercial distribution and only if you
    received the program in object code or executable form with such
    an offer, in accord with Subsection b above.)
The source code for a work means the preferred form of the work for
making modifications to it.  For an executable work, complete source
code means all the source code for all modules it contains, plus any
associated interface definition files, plus the scripts used to
control compilation and installation of the executable.  However, as a
special exception, the source code distributed need not include
anything that is normally distributed (in either source or binary
form) with the major components (compiler, kernel, and so on) of the
operating system on which the executable runs, unless that component
itself accompanies the executable.
If distribution of executable or object code is made by offering
access to copy from a designated place, then offering equivalent
access to copy the source code from the same place counts as
distribution of the source code, even though third parties are not
compelled to copy the source along with the object code.
  4. You may not copy, modify, sublicense, or distribute the Program
except as expressly provided under this License.  Any attempt
otherwise to copy, modify, sublicense or distribute the Program is
void, and will automatically terminate your rights under this License.
However, parties who have received copies, or rights, from you under
this License will not have their licenses terminated so long as such
parties remain in full compliance.
  5. You are not required to accept this License, since you have not
signed it.  However, nothing else grants you permission to modify or
distribute the Program or its derivative works.  These actions are
prohibited by law if you do not accept this License.  Therefore, by
modifying or distributing the Program (or any work based on the
Program), you indicate your acceptance of this License to do so, and
all its terms and conditions for copying, distributing or modifying
the Program or works based on it.
  6. Each time you redistribute the Program (or any work based on the
Program), the recipient automatically receives a license from the
original licensor to copy, distribute or modify the Program subject to
these terms and conditions.  You may not impose any further
restrictions on the recipients' exercise of the rights granted herein.
You are not responsible for enforcing compliance by third parties to
this License.
  7. If, as a consequence of a court judgment or allegation of patent
infringement or for any other reason (not limited to patent issues),
conditions are imposed on you (whether by court order, agreement or
otherwise) that contradict the conditions of this License, they do not
excuse you from the conditions of this License.  If you cannot
distribute so as to satisfy simultaneously your obligations under this
```

```
License and any other pertinent obligations, then as a consequence you
may not distribute the Program at all.  For example, if a patent
license would not permit royalty-free redistribution of the Program by
all those who receive copies directly or indirectly through you, then
the only way you could satisfy both it and this License would be to
refrain entirely from distribution of the Program.
If any portion of this section is held invalid or unenforceable under
any particular circumstance, the balance of the section is intended to
apply and the section as a whole is intended to apply in other
circumstances.
It is not the purpose of this section to induce you to infringe any
patents or other property right claims or to contest validity of any
such claims; this section has the sole purpose of protecting the
integrity of the free software distribution system, which is
implemented by public license practices.  Many people have made
generous contributions to the wide range of software distributed
through that system in reliance on consistent application of that
system; it is up to the author/donor to decide if he or she is willing
to distribute software through any other system and a licensee cannot
impose that choice.
This section is intended to make thoroughly clear what is believed to
be a consequence of the rest of this License.
  8. If the distribution and/or use of the Program is restricted in
certain countries either by patents or by copyrighted interfaces, the
original copyright holder who places the Program under this License
may add an explicit geographical distribution limitation excluding
those countries, so that distribution is permitted only in or among
countries not thus excluded.  In such case, this License incorporates
the limitation as if written in the body of this License.
  9. The Free Software Foundation may publish revised and/or new versions
of the General Public License from time to time.  Such new versions will
be similar in spirit to the present version, but may differ in detail to
address new problems or concerns.
Each version is given a distinguishing version number.  If the Program
specifies a version number of this License which applies to it and "any
later version", you have the option of following the terms and conditions
either of that version or of any later version published by the Free
Software Foundation.  If the Program does not specify a version number of
this License, you may choose any version ever published by the Free Software
Foundation.
  10. If you wish to incorporate parts of the Program into other free
programs whose distribution conditions are different, write to the author
to ask for permission.  For software which is copyrighted by the Free
Software Foundation, write to the Free Software Foundation; we sometimes
make exceptions for this.  Our decision will be guided by the two goals
of preserving the free status of all derivatives of our free software and
of promoting the sharing and reuse of software generally.



    NO WARRANTY
  11. BECAUSE THE PROGRAM IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY
FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW.  EXCEPT WHEN
OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES
PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED
OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF
MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.  THE ENTIRE RISK AS
TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU.  SHOULD THE
```

```
PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING,
REPAIR OR CORRECTION.
  12. IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING
WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR
REDISTRIBUTE THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES,
INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING
OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED
TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY
YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER
PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE
POSSIBILITY OF SUCH DAMAGES.


     END OF TERMS AND CONDITIONS

    How to Apply These Terms to Your New Programs
  If you develop a new program, and you want it to be of the greatest
possible use to the public, the best way to achieve this is to make it
free software which everyone can redistribute and change under these terms.
  To do so, attach the following notices to the program.  It is safest
to attach them to the start of each source file to most effectively
convey the exclusion of warranty; and each file should have at least
the "copyright" line and a pointer to where the full notice is found.
    <one line to give the program's name and a brief idea of what it does.>
    Copyright (C) <year>  <name of author>
    This program is free software; you can redistribute it and/or modify
    it under the terms of the GNU General Public License as published by
    the Free Software Foundation; either version 2 of the License, or
    (at your option) any later version.
    This program is distributed in the hope that it will be useful,
    but WITHOUT ANY WARRANTY; without even the implied warranty of
    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
    GNU General Public License for more details.
    You should have received a copy of the GNU General Public License
    along with this program; if not, write to the Free Software
    Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA  02110-1301  USA
Also add information on how to contact you by electronic and paper mail.
If the program is interactive, make it output a short notice like this
when it starts in an interactive mode:
    Gnomovision version 69, Copyright (C) year  name of author
    Gnomovision comes with ABSOLUTELY NO WARRANTY; for details type `show w'.
    This is free software, and you are welcome to redistribute it
    under certain conditions; type `show c' for details.
The hypothetical commands `show w' and `show c' should show the appropriate
parts of the General Public License.  Of course, the commands you use may
be called something other than `show w' and `show c'; they could even be
mouse-clicks or menu items--whatever suits your program.
You should also get your employer (if you work as a programmer) or your
school, if any, to sign a "copyright disclaimer" for the program, if
necessary.  Here is a sample; alter the names:
  Yoyodyne, Inc., hereby disclaims all copyright interest in the program
  `Gnomovision' (which makes passes at compilers) written by James Hacker.
  <signature of Ty Coon>, 1 April 1989
  Ty Coon, President of Vice
This General Public License does not permit incorporating your program into
proprietary programs.  If your program is a subroutine library, you may
consider it more useful to permit linking proprietary applications with the
library.  If this is what you want to do, use the GNU Library General
```

```
Public License instead of this License.
modification follow.
```

Dipendenze

La lista di altri software da cui dipende pywws sembra spaventosamente lunga a prima vista. Tuttavia, molti di questi pacchetti non sono necessari nella maggior parte degli utenti. Che cosa avete bisogno dipende da cosa si vuole fare con pywws. Ricordate, è un "kit of parts" piuttosto che un'applicazione monolitica.

Si può essere in grado di installare la maggior parte di questi utilizzando il gestore dei pacchetti del sistema operativo. Questo è molto più facile che scaricando e compilando il file di origine da siti Web del progetto. Si noti che alcune distribuzioni di Linux utilizzano nomi differenti per alcuni pacchetti, ad esempio in Ubuntu, pyusb viene chiamato python-usb.

In alternativa, può essere in grado di installare le versioni più recenti di alcune delle librerie dal Python Package Index (PyPI). Vi consiglio l'installazione di pip (il pacchetto può essere chiamato python-pip) o easy_install. Questo semplifica l'installazione del software da PyPI. Per esempio, per installare PyUSB da PyPI utilizzando il comando `pip` command:

```
sudo pip install pyusb
```

Nota: alcune di queste librerie possono avere le loro proprie dipendenze che potrebbe essere necessario installare. Segui i link per saperne di più su ciascuna libreria.

Indispensabile

- Python versione 2.5 o superiore.

Python 3 è supportato, ma alcune cose potrebbero non funzionare correttamente. Se avete un problema con Python 3, si prega di inviare un messaggio al mailing list o presentare un <https://github.com/jim-easterbrook/pywws/issues>‘_.

Libreria USB

Per recuperare dati da una stazione meteo pywws ha bisogno di una libreria che permette di comunicare via USB. C'è una varietà di librerie USB che possono essere utilizzate. Non tutte sono disponibili su tutte le piattaforme informatiche, che possono limitare la vostra scelta.

Su MacOS X il sistema operativo driver generico hid "claims" per la stazione meteo, impedisce di lavorare a libusb. Questo limita gli utenti Mac all'opzione 3 o 4.

- USB library opzione 1 (preferito, tranne in MacOS)
  - PyUSB versione 1.0
  - libusb versione 0.1 o versione 1.0
- USB library opzione 2 (Se PyUSB 1.0 non è disponibile)
  - PyUSB versione 0.4
  - libusb versione 0.1
- USB library opzione 3 (preferito per MacOS)
  - hidapi
  - ctypes (Incluso in molte installazioni Python)

- USB library opzione 4
  - hidapi
  - cython-hidapi
  - cython

Disegnare grafici

Il modulo *pywws.Plot* usa `gnuplot` per disegnare grafici. Se si vogliono produrre i grafici dei dati meteo, ad esempio, da includere in una pagina web, è necessario installare l'applicazione `gnuplot`:

- gnuplot v4.2 o superiore

Trasferimento sicuro di file (sftp)

Il modulo *pywws.Upload* può utilizzare "ftp over ssh" (sftp) per caricare i file sul vostro sito web. Il caricamento normale utilizza i moduli Python standard, ma se si desidera utilizzare sftp è necessario installare questi due moduli:

- paramiko
- pycrypto

Postare su Twitter

Il modulo *pywws.ToTwitter* è utilizzato per inviare messaggi delle condizioni meteo a Twitter. Per postare su Twitter richiede questi moduli:

- python-twitter v1.0 o superiore **o** tweepy v2.0 o superiore
- simplejson
- python-oauth2
- httplib2

Cambiato nella versione 13.10_r1086: Riabilitato uso della libreria `tweepy` come un'alternativa a `python-twitter`. `python-oauth2` è ancora richiesto da *pywws.TwitterAuth*.

Cambiato nella versione 13.06_r1023: Pywws precedentemente utilizzava la libreria `tweepy` invece di `python-twitter` e `python-oauth2`.

Per creare nuove traduzioni di lingua

pywws può essere configurato per utilizzare lingue diverse dall'inglese, e la documentazione può anche essere tradotta in altre lingue. Vedere *Come utilizzare pywws in un altra lingua* per ulteriori informazioni. Il pacchetto `gettext` è necessario per estrarre le stringhe da tradurre e compilare i file di traduzione.

- gettext

Per 'compilare' la documentazione

La documentazione di pywws è scritto in "testo RiStrutturato". Un programma chiamato `Sphinx` è utilizzato per convertire questo formato di scrittura in HTML per l'uso di un browser web. Se si desidera creare una copia locale della documentazione (in modo non sia necessario fare affidamento sulla versione online, o per provare una traduzione su cui stai lavorando) è necessario installare `Sphinx`.

- sphinx

Commenti o domande? È possibile iscriversi alla mailing list pywws http://groups.google.com/group/pywws e farci sapere.

Storico aggiornamenti

```
pywws - Python software for USB Wireless Weather Stations
http://github.com/jim-easterbrook/pywws
Copyright (C) 2008-13  Jim Easterbrook  jim@jim-easterbrook.me.uk

This program is free software; you can redistribute it and/or
modify it under the terms of the GNU General Public License
as published by the Free Software Foundation; either version 2
of the License, or (at your option) any later version.

This program is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
GNU General Public License for more details.

You should have received a copy of the GNU General Public License
along with this program; if not, write to the Free Software
Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA  02110-1301, USA.

Changes in v13.12:
    1/ Changed API of user calibration module.
    2/ Can use python-twitter *or* tweepy library.
    3/ Added a script to run live logging as a UNIX daemon process.
    4/ Changed data store to use separate read and write caches.
    5/ Various bug fixes and minor improvements.

Changes in v13.10:
    1/ Changed Twitter library from tweepy to python-twitter.
    2/ Added ability to do uploads asynchronously.
    3/ Rearranged and improved documentation.
    4/ Various bug fixes and minor improvements.

Changes in v13.06:
    1/ Substantially rearranged directories, getting rid of 'code' and 'code3'.
    2/ Removed 'makefile' - everything is now done via 'setup.py'.
    3/ Removed 'RunModule.py' - use 'python -m pywws.module' now.
    4/ Separated storage of config (weather.ini) and status (status.ini).
    5/ Replaced toservice.py "rapid fire" mode with a separate config file for
       Weather Underground rapid fire.
    6/ Added 2 more low-level USB access modules.
    7/ Various bug fixes and minor improvements.

Changes in v13.03:
    1/ Added 'rain days' to monthly data. (Reprocess required when upgrading.)
    2/ Extended template syntax to include comments.
    3/ Added 'humidity index' function.
    4/ Added French translation of documentation.
    5/ Reduced frequency of saving data files.
    6/ Various bug fixes.
```

```
Changes in v12.12:
    1/ Added support for Python 3.
    2/ Added French documentation translation.
    3/ Used 'binary search' to speed up data access.
    4/ Various bug fixes.

Changes in v12.11:
    1/ Moved development from Google code to GitHub.
    2/ Made software attempt to avoid USB activity at times when it is assumed
       the weather station might be writing to its memory. This might solve
       the USB lockup problem, but it's too early to tell.

Changes in v12.10:
    1/ Added a 'winddir_text' function for use in templates.
    2/ Added <ytics> and <y2tics> options to graph plots.
    3/ Various bug fixes.

Changes in v12.07:
    1/ Added Open Weather Map to the services.
    2/ Fixed problem with Weather Underground uploads that started on 1st June.
    3/ Various bug fixes and software structure improvements.

Changes in v12.05:
    1/ Made 'fixed block' data available to template calculations.
    2/ Fixed buggy auto-detection of 3080 weather stations.
    3/ Added a function to generate the Zambretti forecast code letter.
    4/ Added a program to test USB communication reliablility.
    5/ Various bug fixes and software structure improvements.

Changes in v12.02:
    1/ Separated out low level USB communications to enable use of different
       libraries. Now works on recent versions of Mac OS.
    2/ Added humidity, pressure & wind data to summary data.
    3/ Merged Weather Underground and UK Met Office uploaders into one combined
       module. Added more 'service' uploaders.
    4/ Various bug fixes and software structure improvements.

Changes in v11.10:
    1/ Complete restructuring of documentation.
    2/ Added a user defined 'calibration' process.
    3/ Sets 'locale' according to language setting.
    4/ Added ability to upload to UK Met Office 'WOW'.
    5/ Various bug fixes and software structure improvements.
    6/ New language files: French, Danish.

Changes in v11.05:
    1/ Added support for '3080' family stations that have illuminance and
       UV sensors.
    2/ Broadened the range of tasks that can be done with 'live' data.
    3/ Various bug fixes and software structure improvements.

Changes in v11.02:
    1/ Various bug fixes and software structure improvements.
    2/ Improved wind direction averaging.
    3/ Added conversion functions for common things such as C to F.
    4/ Added a YoWindow module.
    5/ Improved Zambretti forecaster.
```

```
Changes in v10.12:
    1/ Various bug fixes and software structure improvements.
    2/ Added a 'goto' instruction to Template.py.
    3/ Added a 'Zambretti' forecast function to Template.py. This should
       be treated as an experiment, and not relied upon for accuracy.

Changes in v10.10:
    1/ Added 'catchup' mode to ToUnderground.py.
    2/ Created 'Tasks.py' to handle common tasks.
    3/ Made better use of Python's logger for info and error
       messages.
    4/ Changed over from 'python-twitter' to 'tweepy' for Twitter
       access. Twitter authorisation using OAuth now works.
    5/ Added 'LiveLog.py' live logging program.
    6/ Added 'SetWeatherStation.py' to do some configuration of weather
       station. No longer need EasyWeather to set logging interval!
    7/ Added 'Rapid Fire' ability to ToUnderground.py.
    8/ Added plain text versions of HTML documentation.
    9/ Many bug fixes and minor improvements.

Changes in v10.08:
    1/ Added internal temperature to daily and monthly summaries.
       Run Reprocess.py when upgrading from earlier versions.
    2/ Added 'prevdata' to Template.py. Allows calculations that
       compare values from different times.
    3/ Made 'pressure_offset' available to calculations in Plot.py
       and Template.py. This is only useful when using 'raw' data.
    4/ Improved synchronisation to weather station's clock when
       fetching stored data.

Changes in v10.06:
    1/ Improved localisation code.
    2/ Minor bug fixes.
    3/ Added Y axis label angle control to plots.

Changes in v10.04:
    1/ Changed version numbering to year.month.
    2/ Allowed "upload" to a local directory instead of ftp site.
    3/ Added "calc" option to text templates (Template.py).
    4/ Added -v / --verbose option to Hourly.py to allow silent operation.
    5/ Added internationalisation / localisation of some strings.
    6/ Made 'raw' data available to text templates.
    7/ Added ability to upload to Weather Underground.
    8/ Added dual axis and cumulative graph capability.

Changes in v0.9:
    1/ Added lowest daytime max and highest nighttime min temperatures
       to monthly data.
    2/ Added average temperature to daily and monthly data.
    3/ Added 'terminal' element to Plot.py templates for greater control
       over output appearance.
    4/ Added 'command' element to Plot.py templates for even more
       control, for advanced users.
    5/ Added secure upload option.
    6/ Minor speed improvements.

Changes in v0.8:
```

```
    1/ Added meteorological day end hour user preference
    2/ Attempts at Windows compatibility
    3/ Corrected decoding of wind data at speeds over 25.5 m/s
    4/ Improved speed with new data caching strategy

Changes in v0.7:
    1/ Several bug fixes, mostly around new weather stations with not
       much data
    2/ Added min & max temperature extremes to monthly data
    3/ Added template and workspace directory locations to weather.ini
    4/ Increased versatility of Plot.py with layout and title elements

Changes in v0.6:
    1/ Added monthly data
    2/ Changed 'pressure' to 'abs_pressure' or 'rel_pressure'

Changes in v0.5:
    1/ Small bug fixes.
    2/ Added start time to daily data
    3/ Replaced individual plot programs with XML "recipe" system

Changes in v0.4:
    1/ Can post brief messages to Twitter.
    2/ Now time zone aware. Uses UTC for data indexing and local time
       for graphs and text data files.

Changes in v0.3:
    1/ Now uses templates to generate text data
    2/ Added 28 day plot
    3/ Minor efficiency improvements
    4/ Improved documentation

Changes in v0.2:
    1/ Now uses Python csv library to read and write data
    2/ Creates hourly and daily summary files
    3/ Includes rain data in graphs
```

Guide per gli utenti

Contenuti:

Come iniziare con pywws

Installare le dipendenze

Prima di fare qualsiasi cosa con pywws è necessario installare Python e alcune periferiche USB (in modo da consentire alle librerie Python per accedere alla stazione meteo). Vedere *Dipendenze* per maggiori dettagli

Scaricare il software pywws

Creare una directory per tutti i file meteo correlati e posizionarsi nella directory. Per esempio (su un sistema Linux o simile sistema operativo):

```
mkdir ~/weather
cd ~/weather
```

È possibile installare pywws direttamente da PyPI utilizzando `pip` o `easy_install`, oppure è possibile scaricare ed estrarre i file nella directory meteo. Questo ha il vantaggio che si può facilmente leggere i moduli Python e altri file. Consente inoltre di eseguire software pywws senza i privilegi 'root' solitamente necessari per installare il software.

Installazione facile

Si tratta di una semplice riga di comando

```
sudo pip install pywws
```

Le directory dov'è installato dipendere dal sistema operativo e dalla versione Python. I moduli pywws sono installati nella directory 'site-packages' (esempio `/usr/lib/python2.7/site-packages`). In genere vengono installati gli script in `/usr/bin` e vengono installati i file di esempio in `/usr/share/pywws`, ma potrebbe essere usato altre directory (come `/usr/local/share`).

Aggiornare pywws con una semplice riga di comando:

```
sudo pip install pywws -U
```

Scaricare ed estrarre

È possibile scaricare una versione di snapshot PyPI, o potete usare `git` per ottenere il massimo fino alla data della versione aggiornata di pywws.

Per scaricare uno snapshot, visita http://pypi.python.org/pypi/pywws/ e scaricare uno dei file .tar.gz o .zip. Nella vostra directory meteo, quindi estrarre tutti i file, ad esempio:

```
cd ~/weather
tar zxvf pywws-12.11_95babb0.tar.gz
```

o:

```
cd ~/weather
unzip pywws-12.11_95babb0.zip
```

Questo dovrebbe creare una directory (chiamata `pywws-12.11_95babb0` in questo esempio) contenente tutti i file di origine pywws. E' conveniente per creare un link a questa strana directory denominata:

```
cd ~/weather
ln -s pywws-12.11_95babb0 pywws
```

In alternativa, per ottenere l'ultima versione di sviluppo di pywws usare `git clone`:

```
cd ~/weather
git clone https://github.com/jim-easterbrook/pywws.git
```

Dopo la clonazione è possibile utilizzare `setup.py` per compilare i file delle lingue e la documentazione, se è stato installato il pacchetto `gettext` e `sphinx`:

```
cd ~/weather/pywws
python setup.py msgfmt
python setup.py build_sphinx
```

Dopo il download ed estrazione o la clonazione dei repository, è possibile utilizzare ' setup.py ' per compilare e installare tutto:

```
cd ~/weather/pywws
python setup.py build
sudo python setup.py install
```

Questo è opzionale, e installa nella stessa directory come usando `pip` se vuoi. Se non fai questo processo di installazione, sarai solo in grado di eseguire i moduli pywws dalla directory pywws.

L'aggiornamento dei file clonati è fatto con `git pull`, dopo di che si può ricompilare e reinstallare se desiderate:

```
cd ~/weather/pywws
git pull
```

L'aggiornamento di una versione scaricata è lo stesso processo come per la prima installazione. Scarica il .tar.gz o .zip file, estrai il suo contenuto, quindi elimina il link che punta al vecchio download e creane uno che punta al nuovo download. Una volta che siete soddisfatti e la nuova versione funziona OK, è possibile eliminare il vecchio download interamente.

(Solo per utenti 3 Python) Tradurre pywws per Python 3

Se la vostra versione di Python predefinita è 3. x e avete installato pywws usando `pip`, o ha funzionato `python setup.py install`, il codice sarà già stato tradotto da Python 2 a Python 3 come parte del processo di installazione. In caso contrario, è necessario utilizzare setup.py per fare la traduzione e creare un'installazione di Python 3

```
cd ~/weather/pywws
rm -Rf build
python3 setup.py build
sudo python3 setup.py install
```

Testare la connessione della stazione meteo

Finalmente si è pronti per testare l'installazione pywws. Collegare la stazione meteo (se non è già connessa) quindi eseguire il modulo *pywws.TestWeatherStation*. Se avete scaricato ma non installato pywws, non dimenticate di passare alla directory pywws prima. Ad esempio:

```
cd ~/weather/pywws
python -m pywws.TestWeatherStation
```

Se tutto funziona correttamente, questo dovrebbe visualizzare dei numeri sullo schermo, ad esempio:

```
0000 55 aa ff ff ff ff ff ff ff ff ff ff ff ff ff ff 05 20 01 51 11 00 00 00 81 00 00␣
↪0f 00 00 60 55
0020 ea 27 a0 27 00 00 00 00 00 00 00 10 10 12 13 45 41 23 c8 00 32 80 47 2d 2c 01 2c␣
↪81 5e 01 1e 80
0040 96 00 c8 80 a0 28 80 25 a0 28 80 25 03 36 00 05 6b 00 00 0a 00 f4 01 18 03 00 00␣
↪00 00 00 00 00
0060 00 00 4e 1c 63 0d 2f 01 73 00 7a 01 47 80 7a 01 47 80 e4 00 00 00 71 28 7f 25 bb␣
↪28 bd 25 eb 00
```

```
0080 0c 02 84 00 0e 01 e3 01 ab 03 dc 17 00 10 08 21 08 54 10 03 07 22 18 10 08 11 08␣
↪30 10 04 21 16
00a0 26 08 07 24 17 17 08 11 01 06 10 09 06 30 14 29 09 01 06 07 46 09 06 30 14 29 09␣
↪01 06 07 46 08
00c0 08 31 14 30 10 05 14 15 27 10 01 26 20 47 09 01 23 05 13 10 01 26 20 47 09 01 23␣
↪05 13 10 02 22
00e0 11 06 10 02 22 11 06 08 07 07 19 32 08 12 13 22 32 08 09 07 08 48 01 12 05 04 43␣
↪10 02 22 14 43
```

Ci sono diverse ragioni perché questo potrebbe non funzionare. Molto probabilmente è un problema di 'permessi'. Questo può essere testato eseguendo il comando come root:

```
sudo python -m pywws.TestWeatherStation
```

Se questo funziona, allora bisogna consentire all'account utente normale di accedere alla stazione meteo impostando una regola 'udev'. Vedere la pagina del wiki di compatibilità http://code.google.com/p/pywws/wiki/Compatibility for more details.

Se avete qualsiasi altro problema, per favore chiedi aiuto pywws mailing list: http://groups.google.com/group/pywws

Configurare la tua stazione meteo

Se non lo hai già fatto, imposta la tua stazione meteo per visualizzare la corretta pressione atmosferica relativa. (Vedere il manuale per informazioni su come effettuare questa operazione). pywws ottiene l'offset tra pressione relativa e assoluta dalla stazione, quindi questa deve essere impostata prima di utilizzare pywws.

Cercando su internet tra i bollettini meteo di una stazione vicina, idealmente una ufficiale come un aeroporto, è possibile ottenere la corretta pressione relativa dalla vostra posizione. Questo è meglio farlo quando il tempo è calmo e la pressione è costante su una vasta area.

Impostare l'intervallo di registrazione della stazione meteo

La tua stazione meteo probabilmente lasciato la fabbrica con un intervallo di registrazione di 30 minuti. In questo modo la stazione memorizzare circa 11 settimane di dati. La maggior parte degli utenti pywws usano i loro computer per leggere i dati dalla stazione ogni ora, o più spesso che solo bisogno la stazione per archivi dati sufficienti per coprire i fallimenti del computer. L'intervallo consigliato è di 5 minuti, che consente ancora 2 settimane di stoccaggio. Per impostare l'intervallo usa *pywws.SetWeatherStation*:

```
python -m pywws.SetWeatherStation -r 5
```

Registrare i dati della stazione meteo

Innanzitutto, scegliere una directory per archiviare tutti i dati della stazione meteo. Questo verrà scritto abbastanza frequentemente, quindi un disco rigido è preferibile a una memory stick, come questi hanno un numero limitato di scritture. Nella maggior parte dei casi è adatta la home directory, per esempio:

```
mkdir ~/weather/data
```

Questa directory viene definita nella documentazione pywws altrove come directory di dati.

Assicurarsi che il computer abbia la giusta data ora e fuso orario, poiché questi vengono usati per etichettare i dati della stazione meteo. Se non hai già fatto, vale la pena di impostazione NTP per sincronizzare il computer a un 'time server'.

La prima volta che si esegue *pywws.LogData* sarà creato un file di configurazione nella directory dati chiamato 'weather.ini' e poi chiudi il programma. È necessario modificare il file di configurazione e cambiare la linea `ws type = Unknown` in `ws type = 1080` o `ws type = 3080`. (Se la tua stazione meteo visualizza illuminamento solare avete un modello 3080, tutti gli altri sono 1080). Quindi eseguire nuovamente *pywws.LogData*. Ciò può richiedere diversi minuti, siccome esso copierà tutti i dati memorizzati nella memoria della vostra stazione. Il programma *pywws.LogData* ha un'opzione 'verbose' che aumenta la quantità di messaggi vengono visualizzati durante l'esecuzione. Questo è utile quando si esegue manualmente, ad esempio:

```
python -m pywws.LogData -vvv ~/weather/data
```

(Sostituire `~/weather/data` con la directory dei dati, se è diversa.)

Ora dovreste avere alcuni file di dati che si possono guardare. Ad esempio:

```
more ~/weather/data/weather/raw/2012/2012-12/2012-12-16.txt
```

(Sostituire l'anno, il mese e il giorno con quelli che i vostri dati dovrebbero avere.)

Convertire i vecchi dati di EasyWeather (opzionale)

Se era in esecuzione EasyWeather prima di decidere di utilizzare pywws, è possibile convertire i dati che EasyWeather aveva registrato nel formato pywws. Trovare il file EasyWeather.dat e poi convertirlo:

```
python -m pywws.EWtoPy EasyWeather.dat ~/weather/data
```

Impostare alcune opzioni di configurazione

Dopo l'esecuzione *pywws.LogData* ci dovrebbe essere un file di configurazione nella directory dati chiamata 'weather.ini'. Aprire questa con un editor di testo. Si dovrebbe trovare qualcosa come il seguente:

```
[config]
ws type = 1080
logdata sync = 1
pressure offset = 9.4
```

È necessario aggiungere una nuova voce nella sezione `[config]` chiamata `day end hour`. Questo dice a pywws che convenzione si desidera utilizzare nel calcolo dei dati di riepilogo giornalieri. Nel Regno Unito, la 'giornata meteorologica' è solitamente dalle 09.00 alle 09:00 GMT (10.00 alle 10.00 BST durante l'estate), quindi utilizzare un valore di ora di fine giornata di 9. In altri paesi il valore 24 (o 0) potrebbe essere più adatto. Si noti che il valore è impostato nel periodo invernale locale. Non è necessario cambiarla quando l'ora legale è in vigore.

Dopo la modifica, il file weather.ini dovrebbe apparire qualcosa di simile a questo:

```
[config]
ws type = 1080
logdata sync = 1
pressure offset = 9.4
day end hour = 9
```

È inoltre possibile modificare il valore `pressure offset` per regolare il calcolo di pywws della pressione relativa (al livello del mare) dal valore assoluto della stazione. In futuro se si modifica la pressione offset o l'ora di fine giornata, è necessario aggiornare tutti i dati memorizzati tramite il comando *pywws.Reprocess*.

Per maggiori dettagli sulle opzioni del file di configurazione, vedere *weather.ini - configurazione del formato del file*.

Cambiato nella versione 13.10_r1082: inserito `pressure offset` un elemento di configurazione. In precedenza è sempre stata letta dalla stazione meteo.

Elaborare i dati grezzi(raw)

pywws.LogData copia solo i dati grezzi dalla stazione meteo. Per fare qualcosa di utile con quei dati è probabilmente necessario di riepiloghi ogni ora, giornalieri e mensili. Sono creati da *pywws.Process*. Ad esempio:

```
python -m pywws.Process ~/weather/data
```

Ora dovreste avere alcuni file elaborati da guardare:

```
more ~/weather/data/weather/daily/2012/2012-12-16.txt
```

Se cambi l'impostazione di configurazione `day end hour`, è necessario rielaborare tutti i dati meteo. È possibile farlo eseguendo *pywws.Reprocess*:

```
python -m pywws.Reprocess ~/weather/data
```

Ora siete pronti per impostare la registrazione ad intervalli o continua, come descritto in *Come impostare 'hourly' per la registrazione oraria con pywws* o *Come impostare una registrazione 'live' con pywws*.

Leggere la documentazione

La directory doc nella directory di origine pywws contiene le versioni HTML (a meno che non hai fatto un'installazione diretta con 'pip '). I file HTML possono essere letti con qualsiasi browser web. Iniziare con l'indice (*pywws*) e seguire i link da lì.

Commenti o domande? Si prega di iscriversi per al mailing list pywws http://groups.google.com/group/pywws e facci sapere.

Come impostare 'hourly' per la registrazione oraria con pywws

Introduzione

Ci sono due abbastanza differenti modalità di funzionamento con pywws. Tradizionalmente con *Hourly* viene eseguito a intervalli regolari (di solito un'ora) da cron. Questo è adatto per siti Web abbastanza statici, ma aggiornamenti più frequenti possono essere utili per i siti come Weather Underground (http://www.wunderground.com/). Il più recente programma *LiveLog* gira continuamente e può i caricare dati ogni 48 secondi.

Nota che sebbene questo documento (e il nome del programma) si riferisce alla registrazione 'hourly'('oraria'), è possibile eseguire *Hourly* come più frequente o meno come ti piace, ma non cercare di eseguirlo più frequente per raddoppiare l'intervallo di registrazione. Ad esempio, se l'intervallo di registrazione è di 10 minuti, non eseguire *Hourly* più frequente che ogni 20 minuti.

Guida introduttiva

Prima di tutto, è necessario installare pywws e assicurarsi che si possono ottenere i dati dalla tua stazione meteo. Vedere *Come iniziare con pywws* per i dettagli.

Esecuzione una prova con *Hourly* dalla riga di comando, con un alto livello di verbosità in modo da poter vedere ciò che sta accadendo:

```
python -m pywws.Hourly -vvv ~/weather/data
```

Entro cinque minuti (supponendo di aver impostato un intervallo di registrazione di 5 minuti) si dovrebbe vedere un messaggio 'live_data new ptr', seguita dal recupero di eventuali nuovi dati dalla stazione meteo ed elaborarli.

Percorsi dei file di configurazione

Aprire il file weather.ini con un editor di testo. Si dovrebbe avere una sezione di `[paths]` simile al seguente (dove `xxx` è il tuo nome utente):

```
[paths]
work = /tmp/weather
templates = /home/xxx/weather/templates/
graph_templates = /home/xxx/weather/graph_templates/
local_files = /home/xxx/weather/results/
```

Modificare queste per soddisfare le vostre preferenze di installazione. `work` è una directory temporanea utilizzata per archiviare i file intermedi, `templates` è la directory per i tuoi file modello di testo e `graph_templates` è la directory per i tuoi file grafici di modello e `local_files` è una directory dove viene messo l'output del modello che non è stato caricato in un sito web. Non usare le directory di esempio pywws per questi, perchè saranno sovrascritti quando si aggiorna pywws.

Copiare i modelli di testo e grafico nelle directory appropriate. Si possono trovare alcuni esempi forniti con pywws utili per iniziare. Se hai installato pywws con `pip` gli esempi dovrebbero essere in `/usr/share/pywws` o `/usr/local/share/pywws` o simili.

Configurazione dell'esecuzione periodica

In weather.ini si dovrebbe avere le sezioni `[logged]`, `[hourly]`, `[12 hourly]` e `[daily]` simili ai seguenti:

```
[logged]
services = []
plot = []
text = []

[hourly]
...
```

Questo specifica cosa *Hourly* dovrebbe fare quando viene eseguito. Attività nella sezione `[logged]` vengono eseguite ogni volta che ci nuovi dati registrati, le attività nella sezione `[hourly]` vengono effettuate ogni ora, le attività nella sezione `[12 hourly]` vengono eseguite due volte al giorno e le attività nella sezione `[daily]` vengono eseguite una volta al giorno.

La voce `services` è un elenco di servizi meteo online per inviare i dati meteo. Le voci `plot` e `text` sono elenchi di file di modello grafico e file di testo per essere elaborati e, opzionalmente, caricati su un sito web. Aggiungere i nomi dei file di modello e servizi meteo alle voci appropriate, ad esempio:

```
[logged]
services = ['underground', 'metoffice']
plot = []
text = []
```

```
[hourly]
services = []
plot = ['7days.png.xml', '24hrs.png.xml', 'rose_24hrs.png.xml']
text = [('tweet.txt', 'T'), '24hrs.txt', '6hrs.txt', '7days.txt']

[12 hourly]
services = []
plot = []
text = []

[daily]
services = []
plot = ['28days.png.xml']
text = [('forecast.txt', 'T'), 'allmonths.txt']
```

Si noti l'uso del flag `'T'` – questo dice a pywws si inviare il risultato a Twitter invece di caricarlo sul proprio sito ftp.

È possibile verificare che tutti questi stiano lavorando rimuovendo tutte le linee `last update` da status.ini, quindi eseguire nuovamente *Hourly*:

```
python -m pywws.Hourly -v ~/weather/data
```

Cambiato nella versione 13.06_r1015: Aggiunto il flag `'T'`. Precedentemente i modelli Twitter sono stati indicati separatamente in voci `twitter` in `[hourly]` e di altre sezioni. La sintassi precedente funziona ancora, ma è obsoleta.

Cambiato nella versione 13.05_r1009: L'ultimo aggiornamento delle informazioni è era memorizzato in precedenza in weather.ini, con `ultimo aggiornamento` le voci sono in sezioni diverse.

Utilizzare gli script di utilità

L'installazione di pywws include un breve script `pywws-hourly.py` che viene installato nel `/usr/bin` o `/usr/local/bin` o simili. Si dovrebbe essere in grado di utilizzare questo script per eseguire *Hourly*:

```
pywws-hourly.py -v ~/weather/data
```

Eseguire con cron job

La maggior parte dei sistemi UNIX/Linux hanno un demone 'cron' che può eseguire programmi in certi momenti, anche se non si è connessi al computer. Si modifica un file 'crontab' per specificare cosa eseguire e quando per farlo funzionare. Ad esempio, per eseguire *Hourly* a zero minuti di ogni ora:

```
0 * * * *       pywws-hourly.py /home/jim/weather/data
```

Questo potrebbe funzionare, ma probabilmente non sarà possibile ottenere eventuali messaggi di errore per dirvi che cosa è andato storto. È molto meglio eseguire uno script che esegue *Hourly* e poi invia tramite e-mail qualsiasi output prodotto. Ecco lo script che uso:

```
#!/bin/sh
#
# weather station logger calling script

if [ ! -d /data/weather/ ]; then
  exit
```

```
  fi

log=/var/log/log-weather

pywws-hourly.py -v /data/weather >$log 2>&1

# mail the log file
/home/jim/scripts/email-log.sh $log "weather log"
```

Sarà necessario modificare questo file un bel pò per soddisfare i vostri percorsi di file e così via, ma dà un'idea di cosa fare.

Commenti o domande? Si prega di iscriversi per al mailing list pywws http://groups.google.com/group/pywws e facci sapere.

Come impostare una registrazione 'live' con pywws

Introduzione

Ci sono due abbastanza differenti modalità di funzionamento con pywws. Tradizionalmente con *Hourly* viene eseguito a intervalli regolari (di solito un'ora) da cron. Questo è adatto per siti Web abbastanza statici, ma aggiornamenti più frequenti possono essere utili per i siti come Weather Underground (http://www.wunderground.com/). Il più recente programma *LiveLog* gira continuamente e può i caricare dati ogni 48 secondi.

Guida introduttiva

Prima di tutto, è necessario installare pywws e assicurarsi che si possono ottenere i dati dalla tua stazione meteo. Vedere *Come iniziare con pywws* per i dettagli.

Esecuzione di prova *LiveLog* dalla riga di comando, con un alto livello di verbosità in modo da poter vedere ciò che sta accadendo:

```
python -m pywws.LiveLog -vvv ~/weather/data
```

Entro cinque minuti (supponendo di aver impostato un intervallo di registrazione di 5 minuti) si dovrebbe vedere un messaggio 'live_data new ptr', seguita dal recupero di eventuali nuovi dati dalla stazione meteo ed elaborarli. Lasciate che *LiveLog* funzionare per un minuto o due o più, poi fermare il processo digitando '<Ctrl>C'.

Percorsi dei file di configurazione

Aprire il file weather.ini con un editor di testo. Si dovrebbe avere una sezione di `[paths]` simile al seguente (dove `xxx` è il tuo nome utente):

```
[paths]
work = /tmp/weather
templates = /home/xxx/weather/templates/
graph_templates = /home/xxx/weather/graph_templates/
local_files = /home/xxx/weather/results/
```

Modificare queste per soddisfare le vostre preferenze di installazione. `work` è una directory temporanea utilizzata per archiviare i file intermedi, `templates` è la directory per i tuoi file modello di testo e `graph_templates` è la

directory per i tuoi file grafici di modello e `local_files` è una directory dove viene messo l'output del modello che non è stato caricato in un sito web. Non usare le directory di esempio pywws per questi, perchè saranno sovrascritti quando si aggiorna pywws.

Copiare i modelli di testo e grafico nelle directory appropriate. Si possono trovare alcuni esempi forniti con pywws utili per iniziare. Se hai installato pywws con `pip` gli esempi dovrebbero essere in `/usr/share/pywws` o `/usr/local/share/pywws` o simili.

Configurazione dell'esecuzione periodica

In weather.ini si dovrebbe avere una sezione `[live]` simile alla seguente:

```
[live]
services = []
plot = []
text = []
```

Questa sezione specifica cosa pywws dovrebbe fare ogni volta che ottiene una nuova lettura dalla stazione meteo, cioè ogni 48 secondi. La voce `services` è un elenco di servizi meteo online dove caricare dati, ad esempio `['underground_rf']`. Le voci `plot` e `text` sono elenchi di file di modello grafici e file di testo che devono essere elaborati e, opzionalmente, caricati sul tuo sito web. Probabilmente si dovrebbe lasciare tutto questo vuoto tranne che per i servizi `services`.

Se si utilizza YoWindow (http://yowindow.com/) si può aggiungere una voce alla sezione `[live]` per specificare il file di YoWindow, ad esempio:

```
[live]
services = ['underground_rf']
text = [('yowindow.xml', 'L')]
...
```

Si noti l'uso del flag `'L'` – questo dice a pywws si inviare il risultato alla directory "local files" invece di caricarlo sul tuo sito ftp.

Se non li hai già, create quattro altre sezioni nel file weather.ini: `[logged]`, `[hourly]`, `[12 hourly]` e `[daily]`. Queste sezioni dovrebbero avere voci simili alla sezione `[live]` e specificare cosa fare ogni volta i dati vengono registrati (5-30 minuti, a seconda l'intervallo di registrazione), ogni ora, due volte al giorno e una volta al giorno. Aggiungere i nomi dei file di modello alle voci appropriate, ad esempio:

```
[logged]
services = ['underground', 'metoffice']
plot = []
text = []

[hourly]
services = []
plot = ['7days.png.xml', '24hrs.png.xml', 'rose_24hrs.png.xml']
text = [('tweet.txt', 'T'), '24hrs.txt', '6hrs.txt', '7days.txt']

[12 hourly]
services = []
plot = []
text = []

[daily]
services = []
```

```
plot = ['28days.png.xml']
text = [('forecast.txt', 'T'), 'allmonths.txt']
```

Si noti l'uso del flag `'T'` – questo dice a pywws si inviare il risultato a Twitter invece di caricarlo sul proprio sito ftp.

Cambiato nella versione 13.06_r1015: Aggiunto il flag `'T'`. Precedentemente i modelli Twitter sono stati indicati separatamente in voci `twitter` in `[hourly]` e di altre sezioni. La sintassi precedente funziona ancora, ma è obsoleta.

Cambiato nella versione 13.05_r1013: aggiunto il modello `'yowindow.xml'`. Precedentemente il file yowindow era generato da un modulo separato, richiamato dalla voce `yowindow` nella sezione `[live]`. La sintassi precedente funziona ancora, ma è obsoleta.

Upload asincrono

Nuovo nella versione 13.09_r1057.

Il caricamento dei dati in siti web o 'services' a volte può richiedere del tempo, in particolare se un sito è andato off line e i tempi di caricamento sono lunghi. In condizioni di normale funzionamento pywws attende che tutti i caricamenti siano elaborati prima di scaricare ulteriori dati dalla stazione meteo. Questo può portare alcune volte a dati di essere mancanti.

L'elemento `asynchronous` nella sezione `[config]` di weather.ini può essere impostato su `True` per dire a *LiveLog* per fare questi caricamenti in un processo separato. Questa funzionalità è ancora un po' sperimentale – provatelo a vostro rischio.

Utilizzare gli script di utilità

L'installazione di pywws include un breve script `pywws-livelog.py` che viene installato in `/usr/bin` o `/usr/local/bin` o simili. Si dovrebbe essere in grado di utilizzare questo script per eseguire *LiveLog*:

```
pywws-livelog.py -v ~/weather/data
```

Esecuzione in background

Al fine di avere :py:mod: *LiveLog* in esecuzione dopo aver terminato di utilizzare il computer deve essere eseguito come un processo in background. Nella maggior parte dei sistemi Linux/UNIX è possibile farlo mettendo una e commerciale ('&') alla fine della riga di comando. Per esempio:

```
pywws-livelog.py ~/weather/data &
```

Tuttavia, sarebbe utile sapere cosa è andato storto se il programma si blocca per qualsiasi motivo. Con *LiveLog* è possibile memorizzare i messaggi in un file di log specificato con il opzione `-l`:

```
pywws-livelog.py -v -l ~/weather/data/pywws.log ~/weather/data &
```

Riavvio automatico

Ci sono diversi modi per configurare un sistema Linux per avviare un programma quando la macchina si avvia. In genere, questi comportano di mettere un file in `/etc/init.d/`, che richiede i privilegi di root. Il problema è leggermente più difficile da garantire se un programma si riavvia quando si blocca. La mia soluzione per entrambi i problemi è quello di eseguire il seguente script da cron, ogni ora.

```
#!/bin/sh

pidfile=/var/run/pywws.pid
datadir=/data/weather
logfile=$datadir/live_logger.log

# exit if process is running
[ -f $pidfile ] && kill -0 `cat $pidfile` && exit

# email last few lines of the logfile to see why it died
if [ -f $logfile ]; then
  log=/var/log/log-weather
  tail -40 $logfile >$log
  /home/jim/scripts/email-log.sh $log "weather log"
  rm $log
  fi

# restart process
pywws-livelog.py -v -l $logfile $datadir &
echo $! >$pidfile
```

Questo memorizza l'id di processo del *LiveLog* del pidfile. Se il processo è in esecuzione, lo script non serve a nulla. Se il processo si è bloccato, mi invia un e-mail con le ultime 40 righe del file di log (usando uno script che crea un messaggio e lo passa al sendmail) e poi si riavvia *LiveLog*. Avrete bisogno di modificare molto questo per soddisfare le posizioni dei file e così via, ma dà un'idea di cosa fare.

Commenti o domande? Si prega di iscriversi per al mailing list pywws http://groups.google.com/group/pywws e facci sapere.

Come integrare pywws con vari servizi meteorologici

Questa guida dà brevi istruzioni su come utilizzare pywws con alcuni altri servizi meteorologici e software. Non è completo, e alcuni servizi (come Twitter) sono trattati più dettagliatamente altrove.

YoWindow

YoWindow è un display widget meteo in grado di visualizzare i dati da internet o dalla tua stazione meteo. Per visualizzare i dati della tua stazione pywws devi scrivere in un file locale, in genere ogni 48 secondi quando vengono ricevuti i dati nuovi. Questo è facile da fare:

1. Arrestare tutti i software pywws
2. Copiare il modello di esempio ' yowindow.xml ' nella directory modelli di testo.
3. Se non hai già fatto, modifica `weather.ini` e imposta la voce `local_files` nella sezione `[paths]` una directory adatta per il vostro file di yowindow.
4. Aggiungere il modello yowindow per le attività di `[live]` in `weather.ini`. Impostare il flag `'L'` così il risultato è copiato in una directory locale invece di essere caricata su un sito ftp:

   ```
   [live]
   text = [('yowindow.xml', 'L')]
   ```

5. Riavviare pywws in registrazione 'live'.

È possibile controllare se il file è aggiornato ogni 48 secondi usando `more` o `cat` per visualizzare sullo schermo il file.

Infine configurare yowindow per utilizzare questo file. Vedere http://yowindow.com/pws_setup.php per le istruzioni su come effettuare questa operazione.

Twitter

Vedere *Come configurare pywws per pubblicare messaggi su Twitter* per le istruzioni complete.

UK Met Office

Gestito dal modulo *pywws.toservice*. Vedi *Weather Underground* per le istruzioni di configurazione generale.

- Create account: https://register.metoffice.gov.uk/WaveRegistrationClient/public/register.do?service=weatherobservations
- API: http://wow.metoffice.gov.uk/support/dataformats#automatic
- Esempi della sezione `weather.ini`:

```
[metoffice]
site id = 12345678
aws pin = 987654
```

Open Weather Map

Gestito dal modulo *pywws.toservice*. Vedi *Weather Underground* per le istruzioni di configurazione generale.

- Creare un account: http://openweathermap.org/login
- API: http://openweathermap.org/API
- Esempi della sezione `weather.ini`:

```
[openweathermap]
lat = 51.501
long = -0.142
alt = 10
user = Elizabeth Windsor
password = corgi
id = Buck House
```

Il comportamento predefinito è quello di utilizzare il nome utente per identificare la stazione meteo. Tuttavia, è possibile per un utente avere più di una stazione meteo, quindi c'è un parametro di `name` non documentato in API che può essere utilizzato per identificare la stazione. Questo appare come `id` in `weather.ini`. Assicuratevi di che non scegliere un nome che è già in uso.

PWS Weather

Gestito dal modulo *pywws.toservice*. Vedi *Weather Underground* per le istruzioni di configurazione generale.

- Creare un account: http://www.pwsweather.com/register.php
- API basate sul protocollo WU: http://wiki.wunderground.com/index.php/PWS_-_Upload_Protocol

- Esempi della sezione `weather.ini`:

```
[pwsweather]
station = ABCDEFGH1
password = xxxxxxx
```

Weather Underground

- Creare un account: http://www.wunderground.com/members/signup.asp
- API: http://wiki.wunderground.com/index.php/PWS_-_Upload_Protocol
- Esempi della sezione `weather.ini`:

```
[underground]
station = ABCDEFGH1
password = xxxxxxx
```

Weather Underground (or Wunderground) è uno dei più longevi siti meteo nel mondo. Il modulo *pywws.toservice* gestisce la comunicazione per una vasta gamma di servizi on-line.

Il primo passo è di impostare un account Weather Underground, utilizzare la scheda "Add A Station" e fornire dettagli della stazione come la sua posizione e il tipo. Si dovrebbe quindi avere una ID per la stazione e password, prendere nota di questi.

Ora arrestare qualsiasi pywws software in esecuzione, quindi provare a utilizzare direttamente *pywws.toservice*:

```
python -m pywws.toservice ~/weather/data underground
```

Questo deve fallire, siccome non è stata impostata la stazione ID e la password, ma crea le voci in `weather.ini` per la modifica. Modificare `weather.ini` e la sezione `[underground]`:

```
[underground]
station = unknown
password = unknown
```

Sostituire i valori `unknown` con stazione ID e password.

Ora provate dinuovo *pywws.toservice*:

```
python -m pywws.toservice ~/weather/data underground
```

Se questo ha funzionato, allora è possibile caricare i vostri ultimi 7 giorni di dati. Si noti che questo potrebbe richiedere molto tempo, soprattutto se avete una breve 'intervallo di registrazione'. In primo luogo modificare `status.ini` e rimuovere la voce `underground` nella sezione `[last update]`. Quindi eseguire *pywws.toservice* con l'opzione 'catchup' e l'elevato livello di dettaglio così può vederla lavorare:

```
python -m pywws.toservice -vvc ~/weather/data underground
```

Una volta che tutto funziona, è possibile aggiungere 'underground' alla sezione `[logged]` nel file `weather.ini`:

```
[logged]
services = ['underground']
```

"RapidFire" aggiornamenti

Weather Underground ha un secondo upload URL per aggiornamenti in tempo reale appena 2,5 secondi. Se si esegue pywws con 'live logging' (vedere *Come impostare una registrazione 'live' con pywws*) è possibile utilizzare questa opzione per inviare gli aggiornamenti ogni 48 secondi, con l'aggiunta di 'underground_rf' nella sezione `[live]` delle attività `weather.ini`:

```
[live]
services = ['underground_rf']

[logged]
services = ['underground']
```

Assicurarsi di che avere attivo un servizio 'underground' in `[logged]` o `[hourly]`. In questo modo vengono inviati i record 'catchup' per colmare eventuali lacune se vostra stazione passa alla modalità offline per qualche motivo.

Commenti o domande? Si prega di iscriversi per al mailing list pywws http://groups.google.com/group/pywws e facci sapere.

Come configurare pywws per pubblicare messaggi su Twitter

Installare le dipendenze

Postare su Twitter richiede alcuni software aggiuntivi. Vedere *Dipendenze* - *Postare su Twitter*.

Creare un account su Twitter

Si potrebbero postare aggiornamenti meteo all'account 'normale' di Twitter, ma penso che sia meglio avere un conto separato solo per i bollettini meteo. Questo potrebbe essere utile a qualcuno che vive nella tua zona, ma non vuole sapere quello che avevi per la prima colazione.

Autorizzare pywws per inviare al tuo account di Twitter

Se si esegue pywws su un dispositivo di bassa potenza come un router, potrebbe essere più facile per eseguire questo passaggio di autorizzazione su un altro computer, sempre che ci sia '' python-oauth2 '' installato. Utilizzare una directory 'data' vuota – il file `weather.ini` verrà creato i cui contenuti possono essere copiati nel file `weather.ini` reale utilizzando qualsiasi editor di testo.

Assicurarsi che non vi siano altri software pywws in esecuzione, quindi su esegui *TwitterAuth*:

```
python -m pywws.TwitterAuth /data/weather
```

(Sostituire `/data/weather` con la vostra directory dei dati.)

In questo modo, si apre una finestra del browser web (o un URL da copiare nel il browser web) dove è possibile accedere al proprio account Twitter e autorizzare pywws. Il browser visualizzerà 7 cifre che è necessario copiare nel programma *TwitterAuth*. Se ha successo, il file `weather.ini` avrà una sezione `[twitter]` con le voci `secret` and `key`. (Non rivelare a nessun altro.)

Aggiungere i dati di localizzazione (opzionale)

Modificare il file `weather.ini` e aggiungere voci `latitude` e `longitude` nella sezione `[twitter]`. Ad esempio:

```
[twitter]
secret = xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
key = xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
latitude = 51.501
longitude = -0.142
```

Creare un modello

I messaggi Twitter vengono generati utilizzando un modello, proprio come la creazione di file da caricare su un sito Web. Copia il modello di esempio 'tweet.txt' directory dei modelli testo, poi testalo:

```
python -m pywws.Template /data/weather ~/weather/templates/tweet.txt tweet.txt
cat tweet.txt
```

(Sostituire `/data/weather` e `~/weather/templates` con la vostra directory di dati e modelli). Se è necessario modificare il modello (per esempio per cambiare la lingua utilizzata o l'unità di misura) puoi modificarlo subito o più tardi.

Pubblicare il tuo primo Meteo Tweet

Ora tutto è pronto per *ToTwitter* essere eseguito:

```
python -m pywws.ToTwitter /data/weather tweet.txt
```

Se questo funziona, il vostro nuovo account Twitter, ha pubblicato il suo primo Meteo report. (È necessario eliminare il file tweet.txt).

Aggiungere aggiornamenti Twitter alla tua attività oraria

Modificare il file `weather.ini` e modificare la sezione `[hourly]`. Ad esempio:

```
[hourly]
services = []
plot = ['7days.png.xml', '24hrs.png.xml', 'rose_12hrs.png.xml']
text = [('tweet.txt', 'T'), '24hrs.txt', '6hrs.txt', '7days.txt']
```

Si noti l'uso del flag `'T'` – questo dice a pywws si inviare il risultato a tweet invece di caricarlo sul il sito ftp.

Invece si potrebbe cambiare le sezioni `[logged]`, `[12 hourly]` o `[daily]`, ma credo che ogni `[hourly]` è più appropriato per gli aggiornamenti di Twitter.

Cambiato nella versione 13.06_r1015: Aggiunto il flag `'T'`. Precedentemente i modelli Twitter sono stati indicati separatamente in voci `twitter` in `[hourly]` e di altre sezioni. La sintassi precedente funziona ancora, ma è obsoleta.

Commenti o domande? Si prega di iscriversi per al mailing list pywws http://groups.google.com/group/pywws e facci sapere.

Come utilizzare pywws in un altra lingua

Introduzione

Alcune parti del pywws possono essere configurate per utilizzare la lingua locale anziché l'Inglese Britannico. Ciò richiede un file di lingua appropriata che contiene le traduzioni delle varie stringhe utilizzate in pywws. Il progetto pywws si affida agli utenti di fornire queste traduzioni. Questo documento descrive come creare un file di lingua.

La documentazione pywws può anche essere tradotta in altre lingue. Questo è ancora molto lavoro, ma potrebbe essere molto utile agli utenti potenziali che non leggono molto bene l'inglese.

Dipendenze

Così come il software pywws è necessario installare il pacchetto di utilità di internazionalizzazione GNU `gettext`. Questo è disponibile dai repository standard per la maggior parte delle distribuzioni Linux, o potete scaricarlo da http://www.gnu.org/software/gettext/ se è necessario compilarlo da soli.

Scegli il tuo codice di lingua

I computer utilizzano tag della lingua IETF (vedere http://en.wikipedia.org/wiki/IETF_language_tag). Ad esempio, nel Regno Unito si usa ' en_GB '. Questo ha due parti: `en` per l'inglese e `GB` per la versione britannica. Per trovare il codice corretto per la vostra lingua, consultare l'elenco presso http://www.gnu.org/software/gettext/manual/gettext.html#Language-Codes.

Guida introduttiva

La directory pywws dovrebbe già avere una sottodirectory denominata traduzioni. Questo contiene l'insieme dei file di lingua esistente, ad esempio `translations/sv/pywws.po` contiene le traduzioni svedesi. Se una di queste lingue è quello che ti serve, quindi modificare il file weather.ini e aggiungere una voce `language` alla sezione `[config]`, ad esempio:

```
[config]
day end hour = 21
gnuplot encoding = iso_8859_1
language = sv
```

Potrebbe essere ancora necessario compilare e installare il file della lingua scelta. Questo viene fatto con `setup.py`:

```
python setup.py msgfmt
```

Se non c'è già un file per la lingua, il resto di questo documento spiega come crearne uno.

Creare un file di lingua

Il primo passo è quello di creare un file contenente le stringhe che hai bisogno di tradurre. Ad esempio, per creare un file di origine per la lingua francese (codice `fr`):

```
python setup.py xgettext
python setup.py msgmerge --lang=fr
```

Questo ti chiederà di confermare il tuo indirizzo email, quindi creare un file `pywws.po` nella directory `translations/fr`. Ora si dovrebbe modificare `pywws.po`, riempiendo in ogni linea di `msgstr` con una traduzione della linea `msgid` immediatamente di sopra di esso. Il motivo di includere l'indirizzo e-mail è di consentire a chiunque abbia domande circa la vostra traduzione di mettersi in contatto con voi. Sentitevi liberi di mettere in un indirizzo non valido, se siete preoccupati per la privacy.

Dopo avere modificato i file di lingua devono essere compilati in modo che pywws possa usarli. Questo viene fatto con il comando `msgfmt`:

```
python setup.py msgfmt
```

Non dimenticate di fare questo ogni volta che si modifica un file di lingua.

Controllare la traduzione pywws

Il modulo *Localisation* può essere utilizzato per fare un test rapido della lingua del file di installazione:

```
python -m pywws.Localisation -t fr
```

Questo dovrebbe produrre output con qualcosa di simile a questo:

```
Locale changed from (None, None) to ('fr_FR', 'UTF-8')
Translation set OK
Locale
  decimal point: 23,2
  date & time: lundi, 17 décembre (17/12/2012 16:00:48)
Translations
  'NNW' => 'NNO'
  'rising very rapidly' => 'en hausse très rapide'
  'Rain at times, very unsettled' => 'Quelques précipitations, très perturbé'
```

Modificare la voce lingua nel file `weather.ini` per utilizzare il tuo codice lingua (per esempio `it`), quindi provare a utilizzare *Plot* e tracciate un grafico. L'asse x del grafico dovrebbe ora essere tradotto nella tua lingua, utilizzando la traduzione che hai fornito per 'Time', 'Day' or 'Date'.

Tradurre la documentazione

Il sistema utilizzato per tradurre le stringhe utilizzate in pywws è utilizzabile anche per tradurre la documentazione. Il comando per estrarre le stringhe dalla documentazione è molto simile:

```
python setup.py xgettext_doc
```

Si noti che questo richiede il pacchetto sphinx utilizzato per la 'compilazione' della documentazione. Dopo l'estrazione delle stringhe, creare i file di origine per la tua lingua. In questo esempio la lingua è l'italiano, con il codice di due lettere `it`:

```
python setup.py msgmerge --lang=fr
```

Questo crea quattro file (`index.po`, `essential.po`, `guides.po` e `api.po`) che contengono stringhe di testo (spesso interi paragrafi) estratti da diverse parti della documentazione.

Questi file possono essere modificati in modo simile a `pywws.po`. Riempire ogni `msgstr` con una traduzione di `msgid` di sopra di esso. Si noti che alcune stringhe (ad esempio gli URL e i collegamenti ad altre parti della documentazione) non dovrebbero essere tradotti. In questi casi, lasciare vuoto il `msgstr`.

Tradurre tutta la documentazione pywws è un sacco di lavoro. Tuttavia, quando la documentazione è 'compilata' le stringhe non tradotte ritornano al loro originale inglese. Ciò significa che una traduzione parziale potrebbe ancora essere utile – Io consiglio di iniziare con la documentazione front page, `index.po`.

Visualizzazione della documentazione tradotta

Prima convertire il file appena modificato della lingua:

```
python setup.py msgfmt
```

Quindi eliminare la vecchia documentazione (se esiste)e ricostruire utilizzando il tuo linguaggio:

```
rm -Rf doc/html/fr
LANG=fr python setup.py build_sphinx
```

Si noti che il comando `build_sphinx` non ha un opzione `--lang`, per la lingua è impostata una variabile temporanea di ambiente.

Infine è possibile visualizzare la documentazione tradotta tramite un browser web per leggere il file `doc/html/it/index.html`.

Aggiornare i file di lingua

Siccome pywws è in sviluppo, possono essere aggiunte nuove stringhe che richiederà i file di traduzione per essere anche sviluppato. Questo è abbastanza facile da fare. Innanzitutto è necessario ri-estrarre le stringhe per essere tradotte, quindi unirle nei file lingua esistenti. Questo è fatto ripetendo i comandi utilizzati per creare i file:

```
python setup.py xgettext
python setup.py xgettext_doc
python setup.py msgmerge --lang=fr
```

Ciò dovrebbe aggiungere le nuove stringhe nel file di lingua, senza modificare le stringhe già tradotte.

Se l'origine della lingua inglese è cambiato dall'ultima traduzione, alcune stringhe possono essere contrassegnate da gettext come '#, fuzzy '. È necessario controllare che la traduzione è ancora corretta per queste stringhe – il cambiamento può essere banale (per esempio una correzione ortografica) ma potrebbe essere abbastanza significativo. Quando hai controllato (e corretti se necessario) la traduzione, rimuovere la riga '#, fuzzy '.

Inviare a Jim la traduzione

Sono sicuro che ti piacerebbe che gli altri possano beneficiare del lavoro che hai fatto nel tradurre pywws. Per favore, per favore, inviate una copia del vostro file di lingua (per esempio `pywws.po`) a jim@jim-easterbrook.me.uk. Quando si invia una nuova traduzione che è necessario includere i dettagli di quale versione pywws si basa - il modo più semplice per farlo è di includere il valore di `commit` dal file `pywws/version.py` nella tua e-mail.

Commenti o domande? Si prega di iscriversi per al mailing list pywws http://groups.google.com/group/pywws e facci sapere.

weather.ini - configurazione del formato del file

Quasi tutta la configurazione di pywws è tramite un unico file nella directory dei dati: weather.ini. Questo file ha una struttura simile a quella del file Microsoft Windows INI. Esso è diviso in "sections"', ognuna delle quali ha un numero di voci "name = value". L'ordine in cui appaiono spesso delle sezioni non è importante.

Qualsiasi editor di testo può essere utilizzato per modificare il file. (Non provare a modificarlo durante l'esecuzione di qualsiasi altro software pywws). In molti casi sarà pywws inizializzare le voci a valori ragionevoli.

Un altro file, status.ini, viene utilizzato per memorizzare alcune informazioni che pywws utilizza internamente. È descritto alla fine di questo documento. Nell'uso normale non è necessario modificarlo.

Le seguenti sezioni sono attualmente in uso:

- config: varie configurazioni di sistema.
- paths: directory in cui sono memorizzati i modelli ecc.
- live: attività da fare ogni 48 secondi.
- logged: attività da fare ogni volta la stazione registra un record di dati.
- hourly: attività da fare ogni ora.
- 12 hourly: attività da fare ogni 12 ore.
- daily: attività da fare ogni giorno.
- ftp: configurazione per il caricamento di un sito Web.
- twitter: configurazione per postare su Twitter
- underground, metoffice, temperaturnu etc: configurazione di postare sui 'services'.

config: varie configurazione di sistema

```
[config]
ws type = 1080
day end hour = 21
pressure offset = 9.4
gnuplot encoding = iso_8859_1
template encoding = iso-8859-1
language = en
logdata sync = 1
rain day threshold = 0.2
asynchronous = False
usb activity margin = 3.0
gnuplot version = 4.2
```

`ws type` è il "tipo" della stazione meteo. Esso deve essere impostato `1080` per la maggior parte delle stazioni meteorologiche, o `3080` se la vostra console stazione visualizza illuminazione solare.

`day end hour` è la fine del "giornata meteorologica", nell'ora locale, senza tenere conto dell'ora legale. Valori tipici sono 21, 9, o 24. È necessario aggiornare tutti i vostri dati memorizzati eseguendo *`pywws.Reprocess`* per rielaborare tutto dopo aver modificato questo valore.

`pressure offset` è la differenza tra la pressione dell'aria assoluta e relativa (al livello del mare). È copiata dalla stazione meteo, supponendo che è stata impostata fino a visualizzare la corretta pressione relativa, ma è possibile modificare il valore in weather.ini per calibrare la vostra stazione. È necessario aggiornare tutti i vostri dati memorizzati tramite l'esecuzione di *`pywws.Reprocess`* dopo aver modificato questo valore.

Cambiato nella versione 13.10_r1082: inserito `pressure offset` un elemento di configurazione. In precedenza è sempre stata letta dalla stazione meteo.

`gnuplot encoding` è la codifica del testo utilizzato per tracciare i grafici. Il valore predefinito di `iso_8859_1` che consente di usare il simbolo del grado, il che è utile per un'applicazione meteo! Altri valori potrebbero essere necessari se la lingua comprende i caratteri accentati. I possibili valori dipendono dalla vostra installazione gnuplot così alcuni esperimenti possono essere necessari.

`template encoding` è la codifica del testo utilizzato per i modelli. Il valore predefinito è `iso-8859-1`, che è la codifica utilizzata nell'esempio dei modelli. Se è necessario creare modelli con un diverso set di caratteri, è necessario modificare questo valore in base ai vostri modelli.

`language` è utilizzata per localizzare pywws. E' facoltativa, siccome pywws solitamente utilizza il computer come lingua di default impostata dalla variabile di ambiente LANG. Le lingue disponibili sono quelle della sottodirectory `translations` del vostra installazione pywws. Se si imposta un'altra lingua non presente, pywws tornerà in Inglese.

`logdata sync` imposta la qualità di sincronizzazione utilizzato da *pywws.LogData*. Impostare a 0 è più veloce ma imprecisa o 1 più lento ma preciso.

`rain day threshold` è la quantità di pioggia (in mm) che deve cadere in un giorno per poter qualificare come un giorno di pioggia nei dati di riepilogo mensili. È necessario aggiornare tutti i vostri dati memorizzati tramite l'esecuzione di *pywws.Reprocess* dopo aver modificato questo valore.

Nuovo nella versione 13.09_r1057: `asynchrouous` controlla l'utilizzo di un processo separato in upload di *pywws.LiveLog*.

Nuovo nella versione 13.10_r1094: `usb activity margin` controls the algorithm that avoids the "USB lockup" problem that affects some stations. It sets the number of seconds either side of expected station activity (receiving a reading from outside or logging a reading) that pywws does not get data from the station. If your station is not affected by the USB lockup problem you can set `usb activity margin` to 0.0.

Nuovo nella versione 13.11_r1102: `gnuplot version` tells *pywws.Plot* and *pywws.WindRose* what version of gnuplot is installed on your computer. This allows them to use version-specific features to give improved plot quality.

paths: directory in cui sono memorizzati modelli ecc.

```
[paths]
templates = /home/$USER/weather/templates/
graph_templates = /home/$USER/weather/graph_templates/
user_calib = /home/$USER/weather/modules/usercalib
work = /tmp/weather
local_files = /home/$USER/weather/results/
```

Queste tre voci specificano dove i tuoi modelli testo e modelli di grafici vengono archiviati, dove devono essere creati i file temporanei, dove l'output dei file (che non sono stati caricati) dovrebbe essere messo e (se ne hai uno) la posizione del tuo modulo di calibrazione.

live: attività da fare ogni 48 secondi

```
[live]
services = ['underground_rf']
text = [('yowindow.xml', 'L')]
plot = []
```

Questa sezione specifica le attività che devono essere effettuate per ogni campione di dati durante la 'live logging', cioè ogni 48 secondi. È improbabile che si vorrebbe fare altro che caricare questo spesso a Weather Underground o aggiornare il file di YoWindow.

`services` è un elenco di 'servizi' per caricare i dati. Ognuno qui elencato deve avere un file di configurazione `pywws/services/`. See *pywws.toservice* per ulteriori dettagli.

`text` and `plot` sono elenchi di modelli di testo e grafici che devono essere elaborati ed, opzionalmente, caricati sul tuo sito Web.

Cambiato nella versione 13.05_r1013: aggiunto il modello `'yowindow.xml'`. Precedentemente il file yowindow era generato da un modulo separato, richiamato dalla voce `yowindow` nella sezione `[live]`. La sintassi precedente funziona ancora, ma è obsoleta.

logged: attività da fare ogni volta la stazione registra un record di dati

```
[logged]
services = ['underground', 'metoffice']
text = []
plot = []
```

Questa sezione specifica le attività che devono essere effettuate ogni volta che viene registrato un record di dati in modalità 'live logging' o ogni volta che viene eseguito un attività di cron.

`services` è un elenco di 'servizi' per caricare i dati. Ognuno qui elencato deve avere un file di configurazione `pywws/services/`. See *pywws.toservice* per ulteriori dettagli.

`text` and `plot` sono elenchi di modelli di testo e grafici che devono essere elaborati ed, opzionalmente, caricati sul tuo sito Web.

hourly: le attività da fare ogni ora

```
[hourly]
services = []
text = [('tweet.txt', 'T'), '24hrs.txt', '6hrs.txt', '7days.txt', 'feed_hourly.xml']
plot = ['7days.png.xml', '24hrs.png.xml', 'rose_12hrs.png.xml']
```

Questa sezione specifica le attività che devono essere eseguite ogni ora in modalità 'live logging' o l'esecuzione oraria di un attività di cron.

`services` è un elenco di 'servizi' per caricare i dati. Ognuno qui elencato deve avere un file di configurazione `pywws/services/`. See *pywws.toservice* per ulteriori dettagli.

`text` and `plot` sono elenchi di modelli di testo e grafici che devono essere elaborati ed, opzionalmente, caricati sul tuo sito Web.

Cambiato nella versione 13.06_r1015: Aggiunto il flag `'T'`. Precedentemente i modelli Twitter sono stati indicati separatamente in voci `twitter` in `[hourly]` e di altre sezioni. La sintassi precedente funziona ancora, ma è obsoleta.

12 hourly: le attività da fare ogni 12 ore

```
[12 hourly]
services = []
text = []
plot = []
```

Questa sezione specifica le attività che devono essere eseguite ogni 12 ore in modalità 'live logging' o l'esecuzione di un attività oraria di cron. Per cose che non cambiano molto spesso, come i grafici mensili.

`services` è un elenco di 'servizi' per caricare i dati. Ognuno qui elencato deve avere un file di configurazione `pywws/services/`. See *pywws.toservice* per ulteriori dettagli.

`text` and `plot` sono elenchi di modelli di testo e grafici che devono essere elaborati ed, opzionalmente, caricati sul tuo sito Web.

daily: le attività da fare ogni 24 ore

```
[daily]
services = []
text = ['feed_daily.xml']
plot = ['2008.png.xml', '2009.png.xml', '2010.png.xml', '28days.png.xml']
```

Questa sezione specifica le attività che devono essere eseguite ogni giorno in modalità 'live logging' o l'esecuzione di un jun attività oraria di cron. Per le cose che non cambiano molto spesso, ad esempio i grafici mensilmente o annualmente.

`services` è un elenco di 'servizi' per caricare i dati. Ognuno qui elencato deve avere un file di configurazione `pywws/services/`. See *pywws.toservice* per ulteriori dettagli.

`text` and `plot` sono elenchi di modelli di testo e grafici che devono essere elaborati ed, opzionalmente, caricati sul tuo sito Web.

ftp: configurazione di caricamento su un sito web

```
[ftp]
local site = False
secure = False
site = ftp.your_isp.co.uk
user = username
password = userpassword
directory = public_html/weather/data/
```

Queste voci forniscono i dettagli del sito web (o delle directory locali) in cui processare i file di testo e le immagini grafiche che devono essere trasferite.

`local site` Specifica se il file deve essere copiato in una directory locale o inviato a un sito remoto. È possibile impostare questa opzione se si esegue il server web sulla stessa macchina di pywws.

`secure` Specifica se per trasferire i file utilizzando il protocollo SFTP (secure FTP) al posto del più comune FTP. Il sito web provider dovrebbe essere in grado di dirvi se è possibile utilizzare SFTP.

`site` è l'indirizzo web del sito FTP dove trasferire i file.

`user` e `password` sono i dati di login del sito FTP. Il provider del sito web dovrebbe avere comunicato a voi.

`directory` specifica il percorso sul sito FTP (o il file system locale) dove devono essere memorizzati i file. Nota che si potrebbe avere la necessità di sperimentare un po' - potrebbe essere necessario un carattere '/' all'inizio del percorso.

twitter: Configurazione della pubblicazione su Twitter

```
[twitter]
secret = longstringofrandomcharacters
key = evenlongerstringofrandomcharacters
latitude = 51.365
longitude = -0.251
```

`secret` and `key` sono i dati di autenticazione forniti da Twitter. Per averli, usa il programma *pywws.TwitterAuth*.

`latitude` and `longitude` sono i dati di posizione facoltativi. Se si include la vostra stazione meteo tweet avrà informazioni sulla posizione in modo che gli utenti possano vedere dov'è la vostra stazione meteo. Potrebbe anche consentire alle persone di trovare la vostra stazione meteo tweet se si cerca per località.

underground, metoffice, temperaturnu ecc: Configurazione della pubblicazione su 'servizi'

```
[underground]
station = IXYZABA5
password = secret
```

Queste sezioni contengono informazioni quali password e ID stazione necessari per caricare i dati in servizi meteorologici. I nomi dei dati dipendono dal servizio. L'esempio illustrato è per Weather Underground.

`station` Il PWS ID (identificativo della stazione) assegnato alla stazione meteo da Weather Underground.

`password` è la tua password di Weather Underground.

status.ini - formato del file di stato

Questo file è stato scritto da pywws e non dovrebbe (solitamente) essere modificato. Le seguenti sezioni che sono attualmente in uso:

- fixed: valori copiati dalla stazione meteorologica "fixed block".
- clock: informazioni di sincronizzazione.
- last update: data e ora del completamento delle attività più recenti.

fixed: valori copiati dalla stazione meteorologica "fixed block".

```
[fixed]
fixed block = {...}
```

`fixed block` tutti i dati memorizzati nei primi 256 byte di memoria della stazione. Questo include i valori massimi e minimi, impostazioni di soglia di allarme, unità di visualizzazione e così via.

clock: informazioni di sincronizzazione

```
[clock]
station = 1360322930.02
sensor = 1360322743.69
```

Questi valori sono i tempi misurati quando l'orologio della stazione registra alcuni dati e quando i sensori esterni trasmettono un nuovo set di dati. Essi sono utilizzati per cercare di impedire che l'interfaccia USB si blocca se il computer accede alla stazione meteo al tempo stesso di uno di questi eventi, è un problema comune a molte stazioni compatibile con EasyWeather. I tempi sono misurati ogni 24 ore per consentire la deriva negli orologi.

last update: data e ora del completamento delle attività più recenti

```
[last update]
hourly = 2013-05-30 19:04:15
logged = 2013-05-30 19:04:15
daily = 2013-05-30 09:04:15
openweathermap = 2013-05-30 18:59:15
underground = 2013-05-30 18:58:34
metoffice = 2013-05-30 18:59:15
12 hourly = 2013-05-30 09:04:15
```

Questi record data & ora sono dell'ultimo completamento riuscito delle varie attività. Essi sono utilizzati per consentire alle attività infruttuose (per esempio mancanza di rete prevenzione upload) ad essere riprovata dopo pochi minuti.

Commenti o domande? Si prega di iscriversi per al mailing list pywws http://groups.google.com/group/pywws e facci sapere.

Indice di umidità (Humidex)

Autore della sezione: Rodney Persky

Premessa

Utilizzare la tua stazione meteo può essere divertente e la segnalazione giornaliera nei siti dei vari dati meteo possono essere molto utili per i vostri vicini per controllare il tempo. Tuttavia, ad un certo punto si potrebbe voler sapere quali effetti ha tempo sul tuo corpo, e se c'è un modo di dire quando è bene o no lavorare all'aperto.

Qui si inserisce in un mondo tutto di calcoli basati sul trasferimento di energia attraverso pareti, e la resistenza da loro offerta. Essa può essere una grande avventura della conoscenza, e consente di risparmiare molto denaro, energia che si muove intorno.

Introduzione

Humidex è uno strumento per determinare in che modo il corpo di un individuo reagirà alla combinazione di vento, umidità e temperatura. Sullo sfondo del quale c'è l'equilibrio termico tra la vita e la pelle, e' gratuito per ISO 7243 "Ambienti caldi - Valutazione dello stress termico sul lavoro l'uomo". Alcune note importanti,

- Questi indici sono basati su un certo numero di ipotesi che possono risultare in sopra o sottovalutazione del tuo stato interno del corpo
- Una stazione meteo personale potrebbe non mostrare le corrette condizioni, e possono avere una sovra o sottostima dell'umidità, vento, temperatura
- Scelte di abbigliamento personale, effetto della stanchezza e la capacità del fisico di respingere il calore, un basso indice di umidità non significa che si può indossare qualsiasi cosa
- Un individuale idoneità effettuerà la propria risposta del fisico alla temperatura che cambia, e l'esperienza sarà di aiuto nel sapere quando smettere di lavorare
- La durata delle attività che possono essere eseguite richiede conoscenze sull'intensità, che non può essere rappresentata da questo indice

Ipotesi

Ci sono un certo numero di ipotesi che sono state fatte per fare questo lavoro che influenzerà direttamente la sua utilizzabilità. Queste ipotesi tuttavia non sono state messe a disposizione da Environment Canada, che sono gli sviluppatori originali dell'Humidex utilizzato nel PYWWS funzione cadhumidex. Tuttavia è abbastanza sicuro nel dire che sarebbe sono state eseguite alcune ipotesi:

- Tipo di abbigliamento, spessore
- Zona di pelle esposta all'aria libera
- Esposizione al sole

Tuttavia, ci sono un certo numero di ipotesi che pywws deve fare nel calcolo Humidex:

- Le letture di temperatura, vento e umidità sono corrette

Ci sono anche ipotesi circa il tipo di fisico individuale e di 'acclimatazione'

- Un individuale idoneità effettuerà la risposta del fisico alla temperatura che cambia
- L'esperienza sarà di aiuto nel sapere quando smettere di lavorare

Importanti riferimenti

Corso di preparazione per l'estate - http://www.ec.gc.ca/meteo-weather/default.asp?lang=En&n=86C0425B-1

Come utilizzare

La funzione descrittivamente è denominata `cadhumidex` e ha i parametri di temperatura e di umidità, essenzialmente la funzione opera come una conversione e può essere utilizzata in maniera diretta

```
<ycalc>cadhumidex(data['temp_out'],data['hum_out'])</ycalc>
```

Mettendo insieme, ho aggiunto i colori che seguono i colori base di avvertimento e le diverse staffe per produrre un grafico decente:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<graph>
  <title>Humidity Index, Bands indicate apparent discomfort in standard on-site␣
↪working conditions</title>
  <size>1820, 1024</size>
  <duration>hours=48</duration>
  <xtics>2</xtics>
  <xformat>%H%M</xformat>
  <dateformat></dateformat>
  <plot>
    <yrange>29, 55</yrange>
    <y2range>29, 55</y2range>
    <ylabel></ylabel>
    <y2label>Humidex</y2label>
    <source>raw</source>
    <subplot>
      <title>Humidex</title>
      <ycalc>cadhumidex(data['temp_out'],data['hum_out'])</ycalc>
      <colour>4</colour>
      <axes>x1y2</axes>
    </subplot>
```

```
    <subplot>
      <title>HI > 54, Heat Stroke Probable</title>
      <ycalc>54</ycalc>
      <axes>x1y2</axes>
      <colour>1</colour>
    </subplot>
    <subplot>
      <title>HI > 45, Dangerous</title>
      <ycalc>45</ycalc>
      <axes>x1y2</axes>
      <colour>8</colour>
    </subplot>
    <subplot>
      <title>HI > 40, Intense</title>
      <ycalc>40</ycalc>
      <axes>x1y2</axes>
      <colour>6</colour>
    </subplot>
    <subplot>
      <title>HI > 35, Evident</title>
      <ycalc>35</ycalc>
      <axes>x1y2</axes>
      <colour>2</colour>
    </subplot>
    <subplot>
      <title>HI > 30, Noticeable</title>
      <ycalc>30</ycalc>
      <axes>x1y2</axes>
      <colour>3</colour>
    </subplot>
  </plot>
</graph>
```

Non hai eseguito l'aggiornamento più recente?

Se non si esegue l'aggiornamento più recente o non vuoi, allora questo può essere implementato utilizzando una <ycalc> come segue:

```
<ycalc>data['temp_out']+0.555*(6.112*10**(7.5*data['temp_out']/(237.7+data['temp_out
↪']))*data['hum_out']/100-10)</ycalc>
```

Programmi Python e moduli

Contenuto:

pywws.Hourly	Ottenere dati meteo, elaborarli, preparare grafici i file di testo e caricare su un sito web.
pywws.LiveLog	Ottenere dati meteo, memorizzarli ed elaborarli.
pywws.Reprocess	Rigenerare i dati di riepilogo orari e giornalieri
pywws.TwitterAuth	Autorizzare pywws per inviare al vostro account Twitter
pywws.SetWeatherStation	Impostare alcuni parametri della stazione meteo
pywws.TestWeatherStation	Testare la connessione alla stazione meteo.

Continua alla pagina successiva

Tabella 4.1 – continua dalla pagina precedente

pywws.USBQualityTest	Verificare la qualità della connessione USB della stazione meteo
pywws.EWtoPy	Convertire i dati EasyWeather.dat nel formato pywws
pywws.Tasks	Procedura per eseguire attività comuni quali disegnare grafici o caricare i file.
pywws.LogData	Salva i dati della stazione meteo nel file
pywws.Process	Generare tracciati orari, giornalieri & riepiloghi mensili dei dati meteorologici stazione
pywws.calib	Calibra i dati grezzi stazione meteo
pywws.Plot	Tracciare grafici di dati meteo secondo una direttiva XML
pywws.WindRose	Traccia una "rosa dei venti"
pywws.Template	Creare il file dati di testo basato sul modello
pywws.Forecast	Previsione del meteo utilizzando i dati della stazione meteo
pywws.ZambrettiCore	
pywws.Upload	Carica i file su un server web tramite ftp o copiarli in una directory locale
pywws.ToTwitter	Postare messaggi su Twitter
pywws.toservice	Posta gli aggiornamenti delle condizioni meteo a servizi come Weather Underground
pywws.YoWindow	Genera il file YoWindow XML
pywws.WeatherStation	Ottenere i dati da stazioni meteo compatibili con WH1080/WH3080.
pywws.device_ctypes_hidapi	Interfaccia di basso livello USB della stazione meteo, utilizzando ctypes per accedere a hidapi.
pywws.device_cython_hidapi	Interfaccia di basso livello USB della stazione meteo, tramite cython-hidapi.
pywws.device_pyusb1	Interfaccia di basso livello USB della stazione meteo tramite PyUSB.
pywws.device_pyusb	Interfaccia di basso livello USB della stazione meteo tramite PyUSB.
pywws.DataStore	DataStore.py - memorizza le letture in file di facile accesso
pywws.TimeZone	Provide a couple of `datetime.tzinfo` objects representing local time and UTC.
pywws.Localisation	Localisation.py - fornisce traduzioni di stringhe in lingua locale
pywws.calib	Calibra i dati grezzi stazione meteo
pywws.conversions	conversions.py - un insieme di funzioni per convertire unità native pywws
pywws.Logger	Codice comune per la registrazione di informazioni ed errori.

pywws.Hourly

Ottenere dati meteo, elaborarli, preparare grafici i file di testo e caricare su un sito web.

In genere sono eseguiti ogni ora da cron.

```
usage: python -m pywws.Hourly [options] data_dir
options are:
 -h or --help     display this help
 -v or --verbose  increase amount of reassuring messages
data_dir is the root directory of the weather data (e.g. $(HOME)/weather/data)
```

Questo script fa poco più che chiamare altri moduli in sequenza per ottenere i dati dalla stazione meteo, elabora, disegna alcuni grafici, genera alcuni file di testo e carica i risultati su un sito web.

Per ulteriori informazioni sull'utilizzo di `Hourly.py`, vedi *Come impostare 'hourly' per la registrazione oraria con pywws*.

Funzioni

`ApplicationLogger`(verbose[, logfile])	
`Hourly`(data_dir)	
`main`([argv])	

Classi

`DataLogger`(params, status, raw_data)	

pywws.Hourly.**Hourly**(*data_dir*)

pywws.Hourly.**main**(*argv=None*)

pywws.LiveLog

Ottenere dati meteo, memorizzarli ed elaborarli.

Eseguire questo continuamente, avendo impostato quali sono i compiti da fare.

```
usage: python -m pywws.LiveLog [options] data_dir
options are:
 -h      or --help      display this help
 -l file or --log file  write log information to file
 -v      or --verbose   increase amount of reassuring messages
data_dir is the root directory of the weather data (e.g. /data/weather)
```

Per ulteriori informazioni sull'utilizzo di `LiveLog.py`, see *Come impostare una registrazione 'live' con pywws*.

Funzioni

`ApplicationLogger`(verbose[, logfile])	
`LiveLog`(data_dir)	
`main`([argv])	

Classi

`DataLogger`(params, status, raw_data)	
`datetime`(year, month, day[, hour[, minute[, ...)	The year, month and day arguments are required.
`timedelta`	Differenza tra due valori datetime.

pywws.LiveLog.**LiveLog**(*data_dir*)

pywws.LiveLog.**main**(*argv=None*)

pywws.Reprocess

Rigenera i dati di riepilogo orari e giornalieri

```
usage: python -m pywws.Reprocess [options] data_dir
options are:
 -h | --help     display this help
 -v | --verbose  increase number of informative messages
data_dir is the root directory of the weather data
```

Introduzione

Il programma ricrea i dati di riepilogo orari, giornalieri e mensili che sono creati dal programma *pywws.Process*. Esso deve essere eseguito ogni volta che si esegue l'aggiornamento a una versione più recente di pywws.

Dettagli API

Funzioni

ApplicationLogger(verbose[, logfile])	
Reprocess(data_dir)	
main([argv])	

pywws.Reprocess.**Reprocess**(*data_dir*)

pywws.Reprocess.**main**(*argv=None*)

pywws.TwitterAuth

Autorizzare pywws per inviare al vostro account Twitter

```
usage: python -m pywws.TwitterAuth [options] data_dir
options are:
 -h or --help       display this help
data_dir is the root directory of the weather data
```

Questo programma autorizza *pywws.ToTwitter* per postare su un account Twitter. È necessario creare un account prima di eseguire TwitterAuth. Si apre una finestra del browser web (o un URL da copiare nel il browser web) che consente di accedere al tuo account Twitter. Se il login ha successo il browser visualizzerà un numero di 7 cifre, numero che poi copia in TwitterAuth.

Vedere *Come configurare pywws per pubblicare messaggi su Twitter* per ulteriori dettagli su come usare Twitter con pywws.

Funzioni

TwitterAuth(params)	
main([argv])	

Classi

Twitter	

pywws.TwitterAuth.**TwitterAuth**(*params*)

pywws.TwitterAuth.**main**(*argv=None*)

pywws.SetWeatherStation

Imposta alcuni parametri della stazione meteo

```
usage: python -m pywws.SetWeatherStation [options]
options are:
 -h    | --help           display this help
 -c    | --clock          set weather station clock to computer time
                          (unlikely to work)
 -p f  | --pressure f     set relative pressure to f hPa
 -r n  | --read_period n  set logging interval to n minutes
 -v    | --verbose        increase error message verbosity
 -z    | --zero_memory    clear the weather station logged reading count
```

Funzioni

ApplicationLogger(verbose[, logfile])	
bcd_encode(value)	
main([argv])	

Classi

datetime(year, month, day[, hour[, minute[, ...)	The year, month and day arguments are required.
timedelta	Differenza tra due valori datetime.

pywws.SetWeatherStation.**bcd_encode**(*value*)

pywws.SetWeatherStation.**main**(*argv=None*)

pywws.TestWeatherStation

Testare la connessione alla stazione meteo.

Si tratta di una semplice utilità per verificare la comunicazione con la stazione meteo. Se questo non funziona, allora c'è un problema che va risolto prima di altri programmi. Problemi che potrebbero includere la non corretta

installazione delle librerie USB, o un problema di autorizzazioni. Il problema più improbabile è che hai dimenticato di collegare la stazione meteo al computer !

```
usage: python -m pywws.TestWeatherStation [options]
options are:
      --help           display this help
 -c | --change         display any changes in "fixed block" data
 -d | --decode         display meaningful values instead of raw data
 -h | --history count  display the last "count" readings
 -l | --live           display 'live' data
 -m | --logged         display 'logged' data
 -u | --unknown        display unknown fixed block values
 -v | --verbose        increase amount of reassuring messages
                       (repeat for even more messages e.g. -vvv)
```

Funzioni

`ApplicationLogger`(verbose[, logfile])	
main([argv])	
raw_dump(pos, data)	

pywws.TestWeatherStation.**raw_dump**(*pos*, *data*)

pywws.TestWeatherStation.**main**(*argv=None*)

pywws.USBQualityTest

Verificare la qualità della connessione USB della stazione meteo

```
usage: python -m pywws.USBQualityTest [options]
options are:
 -h | --help           display this help
 -v | --verbose        increase amount of reassuring messages
                       (repeat for even more messages e.g. -vvv)
```

Il collegamento USB per la mia stazione meteo non è affidabile al 100%. La lettura dei dati della stazione dal computer è occasionalmente corrotto, forse per interferenza. Ho cercato di risolvere questo problema mettendo ferrite intorno al cavo USB e trasferire possibili sorgenti di interferenza, come dischi rigidi esterni. Tutti senza successo finora.

Questo programma mette alla prova la connessione USB per gli errori per leggere in modo continuo tutta la memoria della stazione meteo (ad eccezione di quelle parti che possono variare) alla ricerca di errori. Ogni blocco di 32-byte viene letto due volte, e se i due valori differiscono viene visualizzato un messaggio di avviso. Sono visualizzati anche il numero di blocchi letti, e il numero di errori riscontrati.

In genere sono uno o due errori per ora, quindi il test deve essere eseguito per diverse ore per produrre una misura utile. Nota che un altro software che permette di accedere alla stazione meteo (come *pywws.Hourly* or *pywws.LiveLog*) non deve essere eseguito mentre il test è in corso.

Se si esegue questo test e non trovate gli errori, per favore fatemelo sapere. C'è qualcosa di miracoloso nella configurazione e mi piacerebbe sapere cosa è!

Funzioni

ApplicationLogger(verbose[, logfile])	
main([argv])	

pywws.USBQualityTest.**main**(*argv=None*)

pywws.EWtoPy

Converte i dati EasyWeather.dat in formato pywws

```
usage: python -m pywws.EWtoPy [options] EasyWeather_file data_dir
options are:
 -h or --help    display this help
EasyWeather_file is the input data file, e.g. EasyWeather.dat
data_dir is the root directory of the weather data
```

Introduzione

Questo programma converte i dati dal formato utilizzato dal programma EasyWeather fornito con la stazione meteo nel formato utilizzato da pywws. È utile se hai usato EasyWeather per un po' prima di scoprire pywws.

Il file `EasyWeather.dat` è utilizzato solo per recuperare i dati prima dell'installazione di pywws. Siccome la tua stazione meteo ha la propria memoria, è necessario eseguire *pywws.LogData* prima di *pywws.EWtoPy* per ridurre al minimo l'utilizzo del file EasyWeather.dat.

pywws.EWtoPy converte i timestamp di EasyWeather.dat da ora locale in UTC. Questo può causare problemi quando l'ora legale finisce, con l'ora solare sembra tornare indietro un'ora. Il programma tenta di rilevare questo e correggere il timestamp errato, ma non ho potuto testare questo su una varietà di fusi orari.

Dettagli API

Funzioni

main([argv])	

Classi

datetime(year, month, day[, hour[, minute[, ...)	The year, month and day arguments are required.
timedelta	Differenza tra due valori datetime.

pywws.EWtoPy.**main**(*argv=None*)

pywws.Tasks

Routine per eseguire attività comuni quali disegnare grafici o caricare i file.

Classi

Calib(params, stored_data)	Classe di taratura che implementa il default o la calibrazione da parte dell'utente.
RegularTasks(params, status, raw_data, ...)	
ToService(params, status, calib_data, ...)	Carica dati meteo a servizi meteorologici come Weather Underground.
datetime(year, month, day[, hour[, minute[, ...)	The year, month and day arguments are required.
deque	deque([iterable[, maxlen]]) –> deque object
timedelta	Differenza tra due valori datetime.

class pywws.Tasks.**RegularTasks**(*params*, *status*, *raw_data*, *calib_data*, *hourly_data*, *daily_data*, *monthly_data*, *asynch=False*)

stop_thread()

has_live_tasks()

do_live(*data*)

do_tasks()

do_twitter(*template*, *data=None*)

do_plot(*template*)

do_template(*template*, *data=None*)

pywws.LogData

Salva i dati della stazione meteo nel file

```
usage: python -m pywws.LogData [options] data_dir
options are:
 -h   | --help     display this help
 -c   | --clear    clear weather station's memory full indicator
 -s n | --sync n   set quality of synchronisation to weather station (0 or 1)
 -v   | --verbose  increase number of informative messages
data_dir is the root directory of the weather data
```

Questo programma / modulo riceve i dati dalla memoria della stazione meteo e li memorizza in un file. Ogni volta che viene eseguito recupera tutti i dati che sono più recenti rispetto agli ultimi dati memorizzati, quindi ha bisogno di essere eseguito solo ogni ora circa. Siccome la stazione meteorologica tipicamente memorizza le letture di due settimane (a seconda dell'intervallo di registrazione), LogData.py potrebbe essere eseguito molto raramente, se non hai bisogno di dati aggiornati in tempo reale.

Se non c'è nessuna informazione data o ora nei dati raw grezzi nella Stazione Meteo , quindi LogData.py crea un timestamp per ogni lettura. Esso utilizza l'orologio del computer, piuttosto che l'orologio della stazione meteo che non può essere letta con precisione dal computer. Un computer in rete dovrebbe avere il suo orologio impostato con precisione da ntp.

La sincronizzazione con la stazione meteo è realizzata in attesa di un cambiamento nei dati attuali. Ci sono due livelli di sincronizzazione, impostate nel file weather.ini la sezione [config] l'opzione logdata sync =. Il livello 0 è più veloce, ma è poco preciso scarta circa dodici secondi. Livello 1 attende dalla stazione meteo che memorizza un nuovo record di dati e ottiene il timestamp accurato di un paio di secondi. Si noti che questo potrebbe richiedere molto tempo, se l'intervallo di registrazione supera i cinque minuti consigliati.

Funzioni

ApplicationLogger(verbose[, logfile])	
main([argv])	

Classi

DataLogger(params, status, raw_data)	
datetime(year, month, day[, hour[, minute[, ...)	The year, month and day arguments are required.
timedelta	Differenza tra due valori datetime.

class pywws.LogData.**DataLogger**(*params*, *status*, *raw_data*)

check_fixed_block()

catchup(*last_date*, *last_ptr*)

log_data(*sync=None*, *clear=False*)

live_data(*logged_only=False*)

pywws.LogData.**main**(*argv=None*)

pywws.Process

Generare tracciati orari, giornalieri & riepiloghi mensili dei dati meteorologici stazione

```
usage: python -m pywws.Process [options] data_dir
options are:
 -h or --help     display this help
 -v or --verbose  increase number of informative messages
data_dir is the root directory of the weather data
```

Questo modulo memorizza i dati grezzi della stazione meteo(tipicamente campionati ogni cinque o dieci minuti) e genera i dati orari, giornalieri e mensili di riepilogo, che è utile per la creazione di tabelle e grafici

Prima di calcolare i riepiloghi dei dati, i dati grezzi sono "calibrated" utilizzando una funzione programmabile dall'utente. Vedere *pywws.calib* per i dettagli

Il dato hourly deriva da tutti i record in un'ora, per esempio dalle 18.00 alle 18:59:59 è l'indice dell'ultimo record completo in quell'ora

La cartella dati giornalieri (daily) riassume il tempo in un periodo di 24 ore in genere fino alle 21:00 o alle 09:00, ora locale (non DST), anche se la mezzanotte è un altra popolare convenzione. È inoltre indicizzata dall' ultimo record nel periodo. Durante le ore diurne e notturne, utilizzato per calcolare le temperature massima e minima, si presume che l'inizio 09:00 o 21:00 dell'ora solare, o 10.00 e 22.00 quando L'ORA LEGALE è in vigore, indipendentemente dal giorno della meteorologia.

Regola la giornata meteorologica con le vostre preferenze, che viene utilizzata dalla stazione meteo ufficiale locale, modificare la linea "day end hour" nel vostro file weather.ini, eseguire quindi Reprocess.py per rigenerare la sintesi.

I dati di riepilogo mensile viene calcolato dal riepilogo giornaliero dei dati. Se la giornata meteorologica non finisce a mezzanotte, ogni mese può iniziare e terminare fino a 12 ore prima o dopo la mezzanotte.

I dati di velocità media del vento è l'ora (o giorno) e la raffica massima velocità durante l'ora (o giorno) è registrato. La direzione predominante del vento è calcolato utilizzando il vettore medio aritmetico.

Le precipitazioni sono state convertite dal raw "total since last reset" rappresenta un più utile totale dell'ultima ora, giorno o mese.

Funzioni

ApplicationLogger(verbose[, logfile])	
Process(params, raw_data, calib_data, ...)	Genera i dati riepilogativi della stazione meteorologica.
calibrate_data(logger, params, raw_data, ...)	'Calibrare', corregge i dati grezzi (raw), utilizzando una funzione fornita dall'utente
generate_daily(logger, day_end_hour, ...)	Genera i riepiloghi orari da dati calibrati.
generate_hourly(logger, calib_data, ...)	Generare i riepiloghi orari (hourly) da dati calibrati.
generate_monthly(logger, rain_day_threshold, ...)	Genera i riepiloghi mensili dai dati giornalieri.
main([argv])	

Classi

Average()	Calcola la media di più valori di dati.
Calib(params, stored_data)	Classe di taratura che implementa il default o la calibrazione da parte dell'utente.
DayAcc(daytime)	ìAccumula' dati meteo per produrre un riepilogo giornaliero.
HourAcc(last_rain)	'Accumula' i dati meteorologici per produrre un riepilogo ogni ora.
Maximum()	Calcola il valore massimo e data ora di più valori di dati.
Minimum()	Calcola il valore minimo e data ora di più valori di dati.
MonthAcc(rain_day_threshold)	Accumula giornalmente i dati meteo di produrre riepilogo mensile.
date	date(year, month, day) –> date object
datetime(year, month, day[, hour[, minute[, ...)	The year, month and day arguments are required.
deque	deque([iterable[, maxlen]]) –> deque object
timedelta	Differenza tra due valori datetime.

class pywws.Process.**Average**
Calcola la media di più valori di dati.

add(*value*)

result()

class pywws.Process.**Minimum**
Calcola il valore minimo e data ora di più valori di dati.

add(*value*, *time*)

result()

class pywws.Process.**Maximum**
Calcola il valore massimo e data ora di più valori di dati.

add(*value*, *time*)

result()

class pywws.Process.**HourAcc**(*last_rain*)
'Accumula' i dati meteorologici per produrre un riepilogo ogni ora.

Calcola la velocità media del vento e la raffica di vento massima, trova la direzione del vento dominante e calcola la piovosità totale.

reset()

add_raw(*data*)

result()

class pywws.Process.**DayAcc**(*daytime*)
ìAccumula' dati meteo per produrre un riepilogo giornaliero.

Calcola la velocità media del vento, la raffica massima del vento e max temperatura diurne & min notturna, trova la direzione del vento dominante e calcola la piovosità totale.

Il giorno si suppone sia dalle 09:00-21:00 e notturno dalle 21:00-09:00 durante l'ora solare, (10:00-22:00 e 22:00-10:00 durante l'ora legale), a prescindere dall'impostazione "day end hour".

reset()

add_raw(*data*)

add_hourly(*data*)

result()

class pywws.Process.**MonthAcc**(*rain_day_threshold*)
Accumula giornalmente i dati meteo di produrre riepilogo mensile.

Calcola le temperature max diurne min notturna.

reset()

add_daily(*data*)

result()

pywws.Process.**calibrate_data**(*logger*, *params*, *raw_data*, *calib_data*)
'Calibrare', corregge i dati grezzi (raw), utilizzando una funzione fornita dall'utente

pywws.Process.**generate_hourly**(*logger*, *calib_data*, *hourly_data*, *process_from*)
Generare i riepiloghi orari (hourly) da dati calibrati.

pywws.Process.**generate_daily**(*logger*, *day_end_hour*, *daytime*, *calib_data*, *hourly_data*, *daily_data*, *process_from*)
Genera i riepiloghi orari da dati calibrati.

pywws.Process.**generate_monthly**(*logger*, *rain_day_threshold*, *day_end_hour*, *time_offset*, *daily_data*, *monthly_data*, *process_from*)
Genera i riepiloghi mensili dai dati giornalieri.

pywws.Process.**Process**(*params*, *raw_data*, *calib_data*, *hourly_data*, *daily_data*, *monthly_data*)
Genera i dati riepilogativi della stazione meteorologica.

La fine del giorno meteorologico (tipicamente 21:00 o 09:00 ora solare) è impostata nel file delle preferenze `weather.ini`. Il valore predefinito è 21:00 (22:00 durante l'ora legale), seguendo la storica convenzione per letture della stazione meteo.

pywws.Process.**main**(*argv=None*)

pywws.calib

Calibra i dati grezzi stazione meteo

Questo modulo permette l'adattamento dei dati grezzi(raw) della stazione meteo come parte del passo 'processing' (vedi: doc:*pywws.Process*). Ad esempio, se abbiamo montato un imbuto a doppia zona di raccolta del pluviometro, è possibile scrivere una routine di calibrazione per raddoppiare il valore di pioggia.

La calibrazione di default fa due cose:

1. Generare la pressione atmosferica relativa.
2. Rimuovere i valori della direzione del vento non validi.

Qualsiasi calibrazione si scriva deve anche fare di questi.

Scrivendo il vostro modulo di calibrazione

In primo luogo, decidere dove si desidera salvare il vostro modulo. Come i modelli di testo e grafici, è meglio tenerlo separato dal codice pywws, quindi esso non è influenzato da aggiornamenti pywws. Vi suggerisco di creare una directory `modules` nello stesso luogo come la directory `templates`.

You could start by copying one of the example calibration modules, or you can create a plain text file in your `modules` directory, e.g. `calib.py` and copy the following text into it:

```
class Calib(object):
    def __init__(self, params, stored_data):
        self.pressure_offset = eval(params.get('config', 'pressure offset'))

    def calib(self, raw):
        result = dict(raw)
        # sanitise data
        if result['wind_dir'] is not None and result['wind_dir'] >= 16:
            result['wind_dir'] = None
        # calculate relative pressure
        result['rel_pressure'] = raw['abs_pressure'] + self.pressure_offset
        return result
```

The *Calib* class has two methods. `Calib.__init__()` is the constructor and is a good place to set any constants you need. It is passed a reference to the raw data storage which can be useful for advanced tasks such as spike removal. `Calib.calib()` generates a single set of 'calibrated' data from a single set of 'raw' data. There are a few rules to follow when writing this method:

- Assicurati di includere la riga `result = dict(raw)`, che copia tutti i dati grezzi(raw) per il risultato valido, dall'inizio.
- Non modificare i dati grezzi(raw).
- Assicurarsi di impostare `result['rel_pressure']`.
- Non dimenticate `return` situato alla fine del file.

Quando hai finito di scrivere il tuo modulo di calibrazione è possibile ottenere da pywws che la utilizzi mettere la sua posizione nel vostro file `weather.ini` Va in sezione `[paths]`, come mostrato nell'esempio di seguito:

```
[paths]
work = /tmp/weather
templates = /home/jim/weather/templates/
```

```
graph_templates = /home/jim/weather/graph_templates/
user_calib = /home/jim/weather/modules/usercalib
```

Si noti che il valore `user_calib` non comprende l'estenzione `.py` alla fine del nome del file.

Classi

Calib(params, stored_data)	Classe di taratura che implementa il default o la calibrazione da parte dell'utente.
DefaultCalib(params, stored_data)	Classe di calibrazione predefinita.

class pywws.calib.**DefaultCalib** (*params*, *stored_data*)
Classe di calibrazione predefinita.

Questa classe definisce la pressione relativa, utilizzando il parametro di pressione originariamente letto dalla stazione meteo, e 'sanitises' la direzione del vento. Questo è il minimo richiesto per la 'calibrazione'.

calib (*raw*)

class pywws.calib.**Calib** (*params*, *stored_data*)
Classe di taratura che implementa il default o la calibrazione da parte dell'utente.

Altri moduli pywws utilizzano questo metodo per creare un oggetto di taratura. Il costruttore crea un oggetto predefinito calibrazione o un oggetto di calibrazione dell'utente, secondo il valore di `user_calib` nella sezione `[paths]` dei parametri `params`. Allora adotta la calibrazione oggetto `calib()` metodo come propri.

calibrator = None

pywws.Plot

Tracciare i grafici dei dati meteo secondo un file di comandi XML

```
usage: python -m pywws.Plot [options] data_dir temp_dir xml_file output_file
options are:
 -h or --help    display this help
data_dir is the root directory of the weather data
temp_dir is a workspace for temporary files e.g. /tmp
xml_file is the name of the source file that describes the plot
output_file is the name of the image file to be created e.g. 24hrs.png
```

Introduzione

Come Template.py questo è uno dei più difficili moduli software da utilizzare della stazione meteo. Traccia un grafico o set di grafici di dati meteo. Quasi tutto ciò che riguarda il grafico è controllato da un file XML. Mi riferisco a questi file come modelli, ma non sono modelli nello stesso modo come Template.py viene utilizzato per creare file di testo.

Prima di scrivere i propri file di template grafici, potrebbe essere utile dare un'occhiata ad alcuni degli esempi nella directory example_graph_templates. Se (come me) non si ha familiarità con XML, vi suggerisco di leggere il W3 Schools XML tutorial.

Sintassi di file grafico XML

Qui è il più semplice modello grafico. Esso riporta la temperatura esterna per le ultime 24 ore.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<graph>
  <plot>
    <subplot>
      <title>Temperature (°C)</title>
      <ycalc>data['temp_out']</ycalc>
    </subplot>
  </plot>
</graph>
```

In questo esempio, l'elemento radice è un elemento grafico, che ha un elemento secondario. Il tracciato secondario contiene un elemento title e un elemento ycalc. Per tracciare più dati sullo stesso sistema di assi (per esempio punto di rugiada e temperatura), possiamo aggiungere più elementi secondari. Per tracciare più di una serie di assi (per esempio la velocità del vento è misurata in unità diverse dalla temperatura) nello stesso file possiamo aggiungere ulteriori elementi del grafico.

La gerarchia degli elementi è mostrata di seguito.

```
graph
    plot
        subplot
            xcalc
            ycalc
            axes
            style
            colour
            title
        bmargin
        yrange
        y2range
        ytics
        y2tics
        ylabel
        ylabelangle
        y2label
        y2labelangle
        grid
        source
        boxwidth
        title
        command
    start
    stop
    duration
    layout
    size
    fileformat
    terminal
    lmargin
    rmargin
    xformat
    xlabel
    dateformat
```

```
    xtics
    title
```

graph

Questo è l'elemento principale del file grafico XML. Essa non deve essere chiamato "graph", ma ci deve essere esattamente un elemento radice.

plot

Ogni grafico deve contenere almeno un elemento grafico. Un grafico è disegnato per ogni elemento grafico, ma tutti condividono lo stesso asse X.

start

Questo elemento imposta la data e ora di inizio dell'asse X. È utilizzato nel costruttore di un oggetto Python data e ora. Ad esempio, per avviare il grafico a mezzogiorno (Ora locale) del giorno di Natale 2008: `<start>year=2008, month=12, day=25, hour=12</start>`. Il valore predefinito è (stop - duration).

stop

Questo elemento imposta la data e ora della fine dell'asse X. È utilizzato nel costruttore di un oggetto Python data e ora. Ad esempio, per terminare il grafico a 10 am (ora locale) del giorno di capodanno: `<stop>year=2009, month=1, day=1, hour=10</stop>`. Il valore predefinito è (start + duration), a meno che l'inizio non è definito in questo caso la data l'ultima lettura oraria della stazione meteo viene utilizzata.

duration

Questo elemento imposta la misura dell'asse X del grafico, a meno che non sono definite sia di inizio che fine . È utilizzato nel costruttore di un oggetto Python timedelta . Ad esempio, per tracciare una settimana: `<duration>weeks=1</duration>`. Il valore predefinito è ore=24.

layout

Controlla il layout dei grafici. Predefinita è una singola colonna. L'elemento layout specifica le righe e colonne. Ad esempio: `<layout>4, 2</layout>` verrà utilizzata una griglia di 4 righe e 2 colonne.

size

Imposta le dimensioni dell' immagine contenente il grafico. Valore predefinito (in un layout a colonna singola) è una larghezza di 600 pixel e l'altezza di 200 pixel per ogni tracciato, così un grafico con i quattro elementi della trama sarebbe 600 x 800 pixel. Qualsiasi elemento deve includersi sia in larghezza che in altezza. Ad esempio: `<size>800, 600</size>` produrrà un'immagine di 800 pixel di larghezza e 600 pixel di altezza.

fileformat

Imposta il formato del del file contenente l'immagine del grafico. Predefinito è png. Dovrebbe fare qualsiasi stringa riconosciuta dalla vostra installazione di gnuplot. Ad esempio: `<fileformat>gif</fileformat>` produrrà un'immagine GIF.

terminal

'terminal' permette un controllo completo delle impostazioni di gnuplot . È possibile utilizzare questa opzione se si traccia con un insolito formato immagine. Qualsiasi stringa riconosciuta dall'installazione di gnuplot in 'terminal' viene eseguita. Ad esempio: `<terminal>svg enhanced font "arial,9" size 600,800 dynamic rounded</terminal>`. Questa impostazione sovrascrive le dimensioni del carattere e la dimensione del grafico.

lmargin

Imposta il margine sinistro dei tracciati, cioè la distanza dall'asse sinistro dal bordo sinistro della zona di immagine. Secondo la documentazione di gnuplot le unità di lmargin sono le larghezze di carattere. Il valore di default è 5, che dovrebbe apparire su OK nella maggior parte dei casi.

rmargin

Imposta il margine destro dei tracciati, cioè la distanza tra l'asse di destra e il bordo destro dell'area immagine. Secondo la documentazione di gnuplot le unità di rmargin sono le larghezze di carattere. Il valore predefinito è -1, che imposta la definizione automatica.

xformat

Imposta il formato delle etichette xtic data / ora. Il valore è una stringa in formato strftime. Dipende dalla durata grafico: se 24 ore o meno è "%H%M", da 24 ore a 7 giorni è "%a %d" e 7 giorni o più è "%Y/%m/%d".

xlabel

Imposta l'etichetta dell'asse X. Il valore è una stringa in formato strftime. Dipende dalla durata grafico: Se 24 ore o meno è "Time (%Z)", da 24 ore a 7 giorni è "Day" e 7 giorni o più è "Date".

dateformat

Imposta il formato delle etichette di data ad ogni estremità dell'asse X. Il valore è una stringa in formato strftime. L'impostazione predefinita è "%Y/%m/%d". L'etichetta a destra del grafico è elaborato solo se diversa dalla sinistra. Per non avere etichette, impostare un formato vuoto: `<dateformat></dateformat>`

xtics

Imposta la spaziatura dei "tic" i segni sull'asse X. Il valore è un numero intero di ore. L'impostazione predefinita è di consentire a gnuplot di impostare un intervallo appropriato.

title

Imposta il titolo del grafico. Una singola linea di testo, per esempio: `<title>Today's weather</title>`. Questo titolo appare al vertice del grafico, fuori di qualsiasi area del tracciato.

subplot

Ogni elemento grafico deve contenere almeno un elemento sottotrama. Una traccia a parte è disegnata per ogni elemento sottotrama, ma tutti condividono gli stessi assi X e Y.

bmargin

Imposta il margine inferiore, ossia la distanza tra l'asse X e il bordo inferiore del grafico (o la trama successiva). L'impostazione predefinita è quella di lasciare a gnuplot di regolare automaticamente, che funziona bene per la maggior parte dei casi, ma si potrebbe desiderare di mettere a punto il valore per soddisfare la vostra installazione.

Il valore ammesso è un qualsiasi numero reale non negativo. Sulla mia configurazione 0.9 è un buon valore, impostare la seguente: `<bmargin>0.9</bmargin>`.

yrange

Imposta i limiti inferiore e superiore dell' asse Y (sinistra). Il valore compreso da gnuplot, è tipicamente una coppia di numeri. L'impostazione predefinita è quella di consentire a gnuplot di impostare i valori appropriati, ed è improbabile che è quello che volete. Ad esempio, per tracciare il tipico grafico UK temperature senza nessun valore fuori del grafico: `<yrange>-10, 30</yrange>`. Nota che le virgole sono convertiti in due punti, in modo che '' <yrange> -10:30 </ YRange> '' è equivalente.

È possibile utilizzare un asterisco per lasciare a gnuplot scegliere un valore adeguato. Ad esempio, per avere la scala automatica del valore superiore mentre fissa il valore inferiore a zero, usare `<yrange>0:*</yrange>`.

y2range

Imposta i limiti inferiore e superiore dell' asse Y di destra. Predefinito per il lato dell'asse Y destro è essere uguale all'asse di sinistra, ma per impostare un diverso range è utile in un grafico a doppio tracciato.

ytics

Controlla il "tic" segni sulla asse Y di sinistra. Il valore può essere tutto ciò che è compreso da gnuplot. Ad esempio, per impostare la spaziatura di 45 punti uso `<y2tics>('N' 0, 'E' 90, 'S' 180, 'W' 270, 'N' 360)</y2tics>`.

y2tics

Controlla il "tic" dei segni sul proprio asse. Il formato è lo stesso di quello per ytics. Comportamento predefinito è di copiare i segni di sinistra, ma senza etichette.

ylabel

Aggiunge un'etichetta a (sinistra) asse Y. Per esempio, quando la temperatura: `<ylabel>°C</ylabel>`. Se si utilizza ylabel probabilmente si desidera regolare lmargin.

ylabelangle

Regola l'angolo (lato sinistro) etichetta asse Y, se dalla versione di gnuplot è supportato. Ad esempio, per scrivere l'etichetta orizzontalmente: `<ylabelangle>90</ylabelangle>`.

y2label

Aggiunge un'etichetta a destra asse Y. Per esempio, quando riportando l'umidità: `<y2label>%</y2label>`. Questo è usato soprattutto per la stampa a doppi assi grafici. Se si utilizza y2label sarà probabilmente necessario regolare rmargin.

y2labelangle

Regola l'angolo della etichetta a destra asse Y, se è supportato dalla versione di gnuplot. Ad esempio, per scrivere l'etichetta orizzontalmente: `<y2labelangle>90</y2labelangle>`.

grid

Aggiunge una griglia di osservazione. Nella maggior parte dei casi la griglia predefinita di gnuplot è adatta, quindi non è necessario: `<grid></grid>`. Il controllo della griglia è possibile utilizzando una qualsiasi delle opzioni usate da gnuplot. Ad esempio, per le linee della griglia orizzontali è: `<grid>ytics</grid>`.

source

Selezionare i dati meteo da tracciare. I valori consentiti sono `<source>raw</source>`, `<source>hourly</source>`, `<source>daily</source>` e `<source>monthly</source>`. L' impostazione predefinita è `raw`. Nota che le diverse fonti hanno diversi dizionari di dati, tale scelta ha effetto su ycalc.

boxwidth

Imposta la larghezza del "boxes" con i grafici a barre. Si tratta di un valore espressione intero producendo un numero di secondi. Il valore predefinito dipende dalla fonte: raw è 240, hourly è 2800 e ogni daily è 2.800 * 24.

title

Imposta il titolo della trama. Una singola riga di testo, ad esempio: `<title>Temperatura (°C)</title>`. Questo titolo viene visualizzato all'interno dell'area del tracciato, sopra qualsiasi titolo del tracciato.

command

Esegue qualsiasi comando di gnuplot, subito prima del principale comando "plot". Questa opzione consente agli utenti avanzati di avere maggiore controllo sull'aspetto grafico. Il valore è qualsiasi comando valido gnuplot, iniziando in genere con il set di comandi. Ad esempio: `<command>set key tmargin center horizontal width 1 noreverse enhanced autotitles box linetype -1 linewidth 1</command>`. (Non mi chiedete che cosa fa questo esempio — io non sono un utente avanzato)

xcalc

Controlla il posizionamento dei valori sull'asse X dei dati tracciati. Il valore predefinito dei dati ['idx'] è corretto per la maggior parte dei casi, ma ci sono alcune eccezioni. Ad esempio, durante la stampa dei grafici a barre di pioggia oraria, è bello centrare le barre sui 30 minuti dell'ora: `<xcalc>data['idx'].replace(minute=30, second=0)</xcalc>`.

ycalc

Seleziona i dati da stampare. Può essere utilizzata una qualsiasi linea di espressione Python che restituisce un singolo valore a virgola mobile . Nella sua forma più semplice questo seleziona solo un valore dalla "data" del dizionario, per esempio: `<ycalc>data['temp_out']</ycalc>` traccia la temperatura esterna. Espressioni più complesse sono possibili, e alcune funzioni di supporto sono fornite. Ad esempio: `<ycalc>dew_point(data['temp_out'], data['hum_out'])</ycalc>` traccia la temperatura esterna del punto di rugiada, e `<ycalc>wind_mph(data['wind_ave'])</ycalc>` traccia la velocità media del vento in miglia all'ora.

Tracciati cumulativi sono anche possibili. Il risultato di ogni calcolo ycalc sono conservati e messi a disposizione per il successivo calcolo dalla variabile last_ycalc. Questo può essere utilizzato con qualsiasi tipo di dati, ma è più utile con le precipitazioni:`<ycalc>data['rain'] + last_ycalc</ycalc>`.

axes

Seleziona quale asse Y i dati è in funzione. Per impostazione predefinita è sull' asse sinistro, ma l'asse di destra può essere scelto con: `<axes>x1y2</axes>`. Questo può essere utilizzato in combinazione con y2 per tracciare due grandezze indipendenti su un grafico, ad esempio la temperatura e l'umidità.

style

Imposta lo stile della linea del grafico. Impostazione predefinita è una linea continua, liscia, spessore 1. Per selezionare un grafico a barre: `<style>box</style>`. Per selezionare i punti senza una linea di collegamento: `<style>+</style>` o `<style>x</style>`. Per selezionare uno spessore di linea 3 (per esempio) usate: `<style>line 3</style>`. Lo spessore dei punti può essere impostato in un modo simile. Per un controllo completo (per utenti avanzati) un pieno stile gnuplot può essere impostato: `<style>smooth unique lc 5 lw 3</style>`.

colour

Imposta il colore del subtracciato a linea o barre. Qualsiasi valore intero è accettato. La mappatura dei colori in numeri è impostato da gnuplot. Valore di default è il colore precedente più uno.

title

Imposta il titolo del secondario. Una singola riga di testo, ad esempio: `<title>Temperature (°C)</title>`. Questo titolo viene visualizzato nell'area di stampa, accanto a un breve tratto di linea del colore utilizzato per il tracciato.

Dettagli API

Funzioni

`ApplicationLogger`(verbose[, logfile])	
`apparent_temp`(temp, rh, wind)	Calcola la temperatura apparente (sensazione reale), con formula da
`cadhumidex`(temp, humidity)	Calcolo Indice di umidità come per gli standard Canadian Weather Standards
`dew_point`(temp, hum)	Calcola il punto di rugiada, usando la formula da http://en.wikipedia.org/wiki/Dew_point.
`illuminance_wm2`(lux)	Conversione approssimativa di illuminazione in lux a radiazione solare in W/m2
`main`([argv])	
`pressure_inhg`(hPa)	Converte la pressione da ettopascal/millibar in pollici di mercurio
`pressure_trend_text`(trend)	Converte in una stringa, la tendenza barometrica sono usati per la UK met office.
`rain_inch`(mm)	Converte le precipitazioni da millimetri a pollici
`temp_f`(c)	Converte la temperatura da gradi Celsius a Fahrenheit
`usaheatindex`(temp, humidity, dew)	Calcolare l'indice di calore secondo Standards USA National Weather Service
`wind_bft`(ms)	Converte i metri al secondo del vento in scala Beaufort
`wind_chill`(temp, wind)	Calcola il vento gelido, usando la formula di
`wind_kmph`(ms)	Converte il vento da metri al secondo a chilometri orari
`wind_kn`(ms)	Converte il vento da metri al secondo a nodi
`wind_mph`(ms)	Converte il vento da metri al secondo a miglia all'ora
`winddir_degrees`(pts)	Convertire la direzione del vento a 0..15 in gradi
`winddir_text`(pts)	Convertire direzione del vento da 0 .. 15 per i punti della bussola

Classi

`BasePlotter`(params, status, raw_data, ...)	
`GraphPlotter`(params, status, raw_data, ...)	
`Record`	
`datetime`(year, month, day[, hour[, minute[, ...)	The year, month and day arguments are required.
`timedelta`	Differenza tra due valori datetime.

class `pywws.Plot.`**`BasePlotter`**(*params*, *status*, *raw_data*, *hourly_data*, *daily_data*, *monthly_data*, *work_dir*)

`DoPlot`(*input_file*, *output_file*)

GetChildren(*node*, *name*)

GetValue(*node*, *name*, *default*)

class pywws.Plot.**Record**

class pywws.Plot.**GraphPlotter**(*params*, *status*, *raw_data*, *hourly_data*, *daily_data*, *monthly_data*, *work_dir*)

GetPlotList()

GetDefaultRows()

GetDefaultPlotSize()

GetPreamble()

PlotData(*plot_no*, *plot*, *source*)

pywws.Plot.**main**(*argv=None*)

pywws.WindRose

Traccia una "rosa dei venti"

```
usage: python -m pywws.WindRose [options] data_dir temp_dir xml_file output_file
options are:
 -h or --help    display this help
data_dir is the root directory of the weather data
temp_dir is a workspace for temporary files e.g. /tmp
xml_file is the name of the source file that describes the plot
output_file is the name of the image file to be created e.g. 24hrs.png
```

Introduzione

Questa routine traccia uno o più "wind roses" (vedi Wikipedia per una descrizione). Come *pywws.Plot* quasi tutto è controllato da un modello di file XML "recipe" / template.

Prima di scrivere i file di modello, potrebbe essere utile a guardare alcuni esempi nella directory example_graph_templates. Se (come ero) non si ha dimestichezza con XML, vi suggerisco di leggere la W3 Schools XML tutorial.

Sintassi di file grafico XML

Questo è il modello più semplice rosa dei venti. Esso disegna un grafico del vento nel corso delle ultime 24 ore.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<graph>
  <windrose>
    <ycalc>data['wind_ave']</ycalc>
  </windrose>
</graph>
```

In questo esempio, l'elemento radice grafico ha una rosa dei venti che non contiene più di un elemento ycalc.

La gerarchia degli elementi è mostrata di seguito.

```
graph
    windrose
        xcalc
        ycalc
        threshold
        colour
        yrange
        points
        source
        title
    start
    stop
    duration
    layout
    size
    fileformat
    lmargin, rmargin, tmargin, bmargin
    title
```

graph

Questo è l'elemento principale del file grafico XML. Essa non deve essere chiamato "graph", ma ci deve essere esattamente un elemento radice.

windrose

Una traccia separata è disegnata per ogni elemento della rosa dei venti, ma tutti condividono lo stesso periodo di tempo.

start

Questo elemento imposta la data & ora della rosa dei venti. È usato nel costruttore dell'oggetto datetime di Python. Ad esempio, per iniziare a mezzogiorno (ora locale) il giorno di Natale 2008: `<start>year=2008, month=12, day=25, hour=12</start>`. Il valore predefinito è (stop - durata).

stop

Questo elemento imposta la data di fine della rosa dei venti. È usato nel costruttore dell'oggetto datetime di Python. Ad esempio, per finire alle 10:00 (ora locale) il giorno di Capodanno 2009: `<stop>year=2009, month=1, day=1, hour=10</stop>`. Il valore predefinito è (start + durata), a meno che l'inizio non è definito in questo caso viene utilizzato il timestamp dell'ultima lettura oraria della stazione meteo.

duration

Questo elemento imposta la durata della rosa dei venti, a meno che non siano definiti start e stop. È usato nel costruttore dell'oggetto timedelta Python. Ad esempio, traccia una settimana: `<duration>weeks=1</duration>`. Il valore predefinito è ore 24.

layout

Controlla il layout delle tracce. Predefinita è una griglia che è più largo che alto. L'elemento layout specifica di righe e colonne. Ad esempio: `<layout>4, 2</layout>` verrà utilizzata una griglia di 4 righe e 2 colonne.

size

Imposta le dimensioni di ingombro del file immagine contenente il grafico. Predefinito è un'altezza di 600 pixel e una larghezza che varia a seconda del layout. Qualsiasi elemento di dimensioni deve includere sia la larghezza che l'altezza. Ad esempio: `<size>800, 600</size>` produrrà un'immagine di 800 pixel di larghezza e 600 pixel di altezza.

fileformat

Imposta il formato dell'immagine del file contenente le tracce. Predefinito è png. Qualsiasi stringa riconosciuta dalla vostra installazione di gnuplot dovrebbe funzionare. Ad esempio: `<fileformat>gif</fileformat>` produrrà un'immagine GIF.

lmargin, rmargin, tmargin, bmargin

Esclude il calcolo automaticamente a sinistra, a destra, margine superiore o inferiore. Ogni numero reale positivo, per esempio `<lmargin>1.3</lmargin>`. Alcune sperimentazioni possono essere necessarie per trovare i migliori valori.

title

Imposta il titolo assoluto. Una singola riga di testo, ad esempio: `<title>Il Meteo oggi</title>`. Questo titolo viene visualizzato nella parte superiore, al di fuori di qualsiasi area del grafico.

xcalc

Seleziona i dati inclusi nella rosa dei venti. Il valore deve essere un'espressione logica valida di Python. Ad esempio, per tracciare una rosa dei venti solo per il pomeriggio: `<xcalc>data['idx'].hour &gt;= 12</xcalc>`. Questo permette l'aggregazione dei dati del vento pomeridiano per diversi giorni. Ricordo che dati sono indicizzati in UTC, quindi è necessario utilizzare un'espressione che tiene conto del tuo fuso orario. Il valore predefinito è 'True'.

ycalc

Seleziona i dati da tracciare. Può essere utilizzato una qualsiasi linea espressione Python che restituisce un valore a virgola mobile. Nella sua forma più semplice questo seleziona un valore dal dizionario "data", per esempio: `<ycalc>data['wind_ave']</ycalc>`. Per la conversione in mph: `<ycalc>data['wind_ave'] * 3.6 / 1.609344</ycalc>`. È improbabile che desidera utilizzare altro qui che 'wind_ave'.

threshold

Fissa le soglie per ogni colore dei petali di rosa. Valori predefiniti sono basati su Wikipedia. I valori devono essere una lista correttamente ordinata di numeri reali, ad esempio: `<threshold>0.5, 3.5, 7.5, 12.5, 18.5, 24.5, 31.5</threshold>` si avvicina alla scala di Beaufort, se ycalc è stato impostato per convertire la velocità del vento in km/h.

colour

Imposta la soglia dei colori nei segmenti del petalo. Ogni sequenza di valori interi è accettata. La mappatura dei colori ai numeri è impostato da gnuplot. Valore di default è 0, 1, 2, 3, ecc.

yrange

Imposta il limite superiore degli assi. La rosa rappresenta la percentuale del tempo che il vento proveniva da una particolare direzione. Per esempio, se vivi in un posto con un vento molto stabile è possibile consentire a percentuali più elevate del normale: `<yrange>91</yrange>`. Ridimensionamento automatico è inoltre possibile, utilizzando un asterisco: `<yrange>*</yrange>`

points

Imposta il testo dei punti cardinali. Le impostazioni predefinite sono 'N', 'S', 'E' & 'W'. Per i grafici in un'altra lingua è possibile sovrascrivere questo, ad esempio: `<points>'No', 'Zu', 'Oo', 'We'</points>`. (Il modo migliore per farlo è di creare il file della lingua, vedi Localisation.py.)

source

Selezionare i dati meteo da tracciare. I valori consentiti sono `<source>raw</source>`, `<source>hourly</source>`, `<source>daily</source>` e `<source>monthly</source>`. L' impostazione predefinita è `raw`. Nota che le diverse fonti hanno diversi dizionari di dati, tale scelta ha effetto su ycalc.

title

Imposta il titolo della grafico. Una singola riga di testo, ad esempio: `<title>Venti mattutini</title>`. Questo titolo viene visualizzato all'interno dell'area di stampa, al di sopra della soglia chiave del colore.

Dettagli API

Funzioni

`apparent_temp`(temp, rh, wind)	Calcola la temperatura apparente (sensazione reale), con formula da
`cadhumidex`(temp, humidity)	Calcolo Indice di umidità come per gli standard Canadian Weather Standards

Continua alla pagina successiva

Tabella 4.23 – continua dalla pagina precedente

dew_point(temp, hum)	Calcola il punto di rugiada, usando la formula da http://en.wikipedia.org/wiki/Dew_point.
illuminance_wm2(lux)	Conversione approssimativa di illuminazione in lux a radiazione solare in W/m2
main([argv])	
pressure_inhg(hPa)	Converte la pressione da ettopascal/millibar in pollici di mercurio
pressure_trend_text(trend)	Converte in una stringa, la tendenza barometrica sono usati per la UK met office.
rain_inch(mm)	Converte le precipitazioni da millimetri a pollici
temp_f(c)	Converte la temperatura da gradi Celsius a Fahrenheit
usaheatindex(temp, humidity, dew)	Calcolare l'indice di calore secondo Standards USA National Weather Service
wind_bft(ms)	Converte i metri al secondo del vento in scala Beaufort
wind_chill(temp, wind)	Calcola il vento gelido, usando la formula di
wind_kmph(ms)	Converte il vento da metri al secondo a chilometri orari
wind_kn(ms)	Converte il vento da metri al secondo a nodi
wind_mph(ms)	Converte il vento da metri al secondo a miglia all'ora
winddir_degrees(pts)	Convertire la direzione del vento a 0..15 in gradi
winddir_text(pts)	Convertire direzione del vento da 0 .. 15 per i punti della bussola

Classi

BasePlotter(params, status, raw_data, ...)	
RosePlotter(params, status, raw_data, ...)	
datetime(year, month, day[, hour[, minute[, ...)	The year, month and day arguments are required.
timedelta	Differenza tra due valori datetime.

class pywws.WindRose.**RosePlotter**(*params*, *status*, *raw_data*, *hourly_data*, *daily_data*, *monthly_data*, *work_dir*)

GetPlotList()

GetDefaultRows()

GetDefaultPlotSize()

GetPreamble()

PlotData(*plot_no*, *plot*, *source*)

pywws.WindRose.**main**(*argv=None*)

pywws.Template

Creare file di dati di testo basato su un modello

```
usage: python -m pywws.Template [options] data_dir template_file output_file
options are:
 --help    display this help
data_dir is the root directory of the weather data
```

```
template_file is the template text source file
output_file is the name of the text file to be created
```

Introduzione

Questo è probabilmente il modulo più difficile da utilizzare dell'insieme di software della stazione meteo. Genera file di testo basati su un file "template" dai dati raw, hourly, daily & monthly della stazione meteo. L'elaborazione del modello va al di là della semplice sostituzione di valori da includere in cicli, salti in avanti o indietro dei dati, l'elaborazione dei dati e la sostituzione di valori mancanti.

Un file di modello può essere qualsiasi tipo di file di testo (testo normale, xml, html, ecc.) in cui sono stati aggiunti "processing instructions". Queste istruzioni di elaborazione sono delimitate da caratteri di hash ('#'). Essi non vengono copiati all'uscita, ma accade qualcos'altro: o viene inserito un valore di dati o viene effettuato un numero limitato di altre azioni.

Prima di scrivere il proprio file di template, potrebbe essere utile esaminare alcuni esempi nella directory example_templates.

Istruzioni di elaborazione

Note that if the closing '#' of a processing instruction is the last character on a line then the following line break is not outputted. This makes templates easier to edit as you can have a separate line for each processing instruction and still produce output with no line breaks. If you want to output a line break after a processing instruction, put a blank line immediately after it.

##

output di un singolo carattere '#'.

#! testo di commento#

Il commento, non genera nessun output. `testo di commento` può essere qualsiasi testo senza un'interruzione di riga.

#monthly#

ritorna i dati "monthly" di riepilogo. L'indice viene reimpostato sul valore più recente.

#daily#

ritorna i dati "daily" di riepilogo. L'indice viene reimpostato sul valore più recente.

#hourly#

ritorna i dati "hourly" di riepilogo. L'indice viene reimpostato sul valore più recente.

#raw#

ritorna i dati "raw" di riepilogo. L'indice viene reimpostato sul valore più recente.

Cambiato nella versione 11.09: This now selects "calibrated" data. The directive name remains unchanged for backwards compatibility.

#timezone name#

Converte tutti i valori data e ora del fuso orario `name` prima dell'output. I valori consentiti per nome sono `utc` o `local`.

#locale expr#

switch use of 'locale' on or off, according to `expr`. When locale is on floating point numbers may use a comma as the decimal separator instead of a point, depending on your localisation settings. Use `"True"` or `"False"` for expr.

#roundtime expr#

Interruttore arrotondamento tempo on/off, secondo `expr`. Quando l'arrotondamento del tempo è attivo, 30 secondi vengono aggiunti a ciascun valore di tempo. Questo è utile se si stampa solo in ore e minuti, ad esempio con un formato " %H: %M", e che i valori di tempo come ad esempio 10:23:58 di apparire come "10:24 ". Usare `"True"` o `"False"` per `expr`.

#jump count#

Salta i valori `count`. L'indice dei dati è regolata da `count` ore o giorni. Con valori negativi salta indietro nel tempo.

È una buona idea inserire salti all'interno di un loop alla fine, poco prima dell'istruzione `#endloop#`. Il ciclo può poi terminare in modo pulito se ha esaurito i dati.

#goto date-time#

va alla `date-time`'. L'indice di dati è impostato al record immediatamente dopo `date-time`.Questo può essere in UTC o il fuso orario locale, secondo l'impostazione di `timezone`, E deve corrispondere esattamente al formato data ISO, per esempio `"2010-11-01 12:00:00"` è mezzogiorno a partire dal 1° novembre 2010.

Parti di `date-time` possono essere sostituite con strftime % stile caratteri di formattazione per specificare l'attuale indice del ciclo. Ad esempio, `"%Y-%m-01 12:00:00"` è mezzogiorno a partire dal giorno 1 del mese.

#loop count#

avvia un ciclo che si ripete `count` volte. `count` deve essere uno o più.

#endloop#

finisce un ciclo avviato da `#loop count#`. L'elaborazione del modello torna alla riga contenente le istruzioni `#loop count#`. Non provare a nido loop.

#key fmt_string no_value_string conversion#

ricava un valore di dati. *key'* è la chiave dei dati, es. *temp_out'* per la temperatura esterna. `fmt_string` è un printf come stringa di formattazione (in realtà l'operatore % di Python's) tranne che per i valori data e ora, quando si ha data e ora usare il metodo `strftime()`. `no_value_string` risulta al posto di `fmt_string` quando il valore dei dati è assente, ad esempio, se la stazione ha perso il contatto con il sensore esterno. `conversion` è un espressione Python per convertire i dati, ad esempio per convertire la velocità da m/s a mph è possibile utilizzare `"x * 3.6 / 1.609344"`, o la funzione più conveniente prevista `"wind_mph(x)"`.

Tutti questi valori tra virgolette " contengono spazi e altri caratteri potenzialmente difficili. Tutti tranne `key` sono facoltativi, ma si noti che se si desidera specificare una conversione, inoltre, è necessario specificare `fmt_string` e `no_value_string`.

#calc expression fmt_string no_value_string conversion#

restituisce un valore calcolato da uno o più elementi di dati. `expression` è una espressione valida in Python, ad es. `"dew_point(data['temp_out'], data['hum_out'])"` per calcolare il punto di rugiada all'aperto. `fmt_string`, `no_value_string` e `conversion` sono come descritti sopra. Si noti che, probabilmente, è più efficiente incorporare qualsiasi conversione nell'espressione stessa.

Esempio

Ecco un esempio di base e utilizzo avanzato del modello. E' parte della 6hrs.txt del file di esempio del modello, che genera una tabella HTML di 7 letture orarie (che deve coprire 6 ore).

```
#hourly#
#jump -6#
#loop 7#
  <tr>
    <td>#idx "%Y/%m/%d" "" "[None, x][x.hour == 0 or loop_count == 7]"#</td>
    <td>#idx "%H%M %Z"#</td>
    <td>#temp_out "%.1f °C"#</td>
    <td>#hum_out "%d%%"#</td>
    <td>#wind_dir "%s" "-" "winddir_text(x)"#</td>
    <td>#wind_ave "%.0f mph" "" "wind_mph(x)"#</td>
    <td>#wind_gust "%.0f mph" "" "wind_mph(x)"#</td>
    <td>#rain "%0.1f mm"#</td>
    <td>#rel_pressure "%.0f hPa"#, #pressure_trend "%s" "" "pressure_trend_text(x)"#</
↪td>
  </tr>
#jump 1#
#endloop#
```

Le prime tre righe di questo frammento: seleziona i dati orari, salta indietro di 6 ore, avvia un loop con un conteggio di 7. un salto in avanti di un'ora appare poco prima della fine del segmento ripetuto. A quest'ultimo salto (di un'ora) accade ogni volta nel ciclo, una sequenza di 7 letture dei dati sarà il risultato. L'ultima riga indica la fine del ciclo - tutto tra le linee `#loop 7#` e `#endloop#` è ripetuta 7 volte.

Le istruzioni `#temp_out ...#`, `#hum_out ...#`, `#rain ...#` e `#rel_pressure ...#` mostrano dati di base. Ognuno di essi con un `fmt_string` per formattare i dati in modo appropriato. Le istruzioni `#wind_ave ...#` e `#wind_gust ...#` mostra come utilizzare un'espressione di conversione per convertire m/s a mph.

Le istruzioni `#wind_dir ...#` e `#pressure_trend ...#` mostrano l'uso della funzione incorporata `wind_dir_text` e la funzione `pressure_trend_text` convertono i valori numerici in testo.

Finalmente arriviamo ai valori di datetime. L'istruzione `#idx "%H%M"#` emette semplicemente l'ora (in formato HHMM) dell'indice dei dati meteo. L'istruzione‘‘#idx “%Y/%m/%d” “” “[None, x][x.hour == 0 or loop_count == 7]”#‘‘ è un po' più complicato. Esso emette la data, ma solo sulla prima riga o se la data è cambiata. Ciò è reso possibile dall'indicizzazione dell'array `[None, x]` con un'espressione booleana che è vera quando `loop_count` è 7 (Cioè il primo passo del ciclo) o `x.hour` è pari a zero (Vale a dire che questa è la prima ora del giorno).

Dettagli API

Funzioni

`ApplicationLogger`(verbose[, logfile])	
`Zambretti`(params, hourly_data)	
`ZambrettiCode`(params, hourly_data)	
`apparent_temp`(temp, rh, wind)	Calcola la temperatura apparente (sensazione reale), con formula da
`cadhumidex`(temp, humidity)	Calcolo Indice di umidità come per gli standard Canadian Weather Standards
`dew_point`(temp, hum)	Calcola il punto di rugiada, usando la formula da http://en.wikipedia.org/wiki/Dew_point.
`illuminance_wm2`(lux)	Conversione approssimativa di illuminazione in lux a radiazione solare in W/m2
`main`([argv])	
`pressure_inhg`(hPa)	Converte la pressione da ettopascal/millibar in pollici di mercurio
`pressure_trend_text`(trend)	Converte in una stringa, la tendenza barometrica sono usati per la UK met office.
`rain_inch`(mm)	Converte le precipitazioni da millimetri a pollici
`temp_f`(c)	Converte la temperatura da gradi Celsius a Fahrenheit
`usaheatindex`(temp, humidity, dew)	Calcolare l'indice di calore secondo Standards USA National Weather Service
`wind_bft`(ms)	Converte i metri al secondo del vento in scala Beaufort
`wind_chill`(temp, wind)	Calcola il vento gelido, usando la formula di
`wind_kmph`(ms)	Converte il vento da metri al secondo a chilometri orari
`wind_kn`(ms)	Converte il vento da metri al secondo a nodi
`wind_mph`(ms)	Converte il vento da metri al secondo a miglia all'ora
`winddir_degrees`(pts)	Convertire la direzione del vento a 0..15 in gradi
`winddir_text`(pts)	Convertire direzione del vento da 0 .. 15 per i punti della bussola

Classi

`Template`(params, status, calib_data, ...[, ...])	
`datetime`(year, month, day[, hour[, minute[, ...)	The year, month and day arguments are required.
`timedelta`	Differenza tra due valori datetime.

class `pywws.Template.`**`Template`**(*params*, *status*, *calib_data*, *hourly_data*, *daily_data*, *monthly_data*, *use_locale=True*)

`process`(*live_data*, *template_file*)

make_text(*template_file*, *live_data=None*)

make_file(*template_file*, *output_file*, *live_data=None*)

pywws.Template.**main**(*argv=None*)

pywws.Forecast

Previsione meteo utilizzando dati recenti

```
usage: python -m pywws.Forecast [options] data_dir
options are:
 -h | --help  display this help
data_dir is the root directory of the weather data
```

Funzioni

Zambretti(params, hourly_data)	
ZambrettiCode(params, hourly_data)	
main([argv])	

Classi

datetime(year, month, day[, hour[, minute[, ...)	The year, month and day arguments are required.
timedelta	Differenza tra due valori datetime.

pywws.Forecast.**ZambrettiCode**(*params*, *hourly_data*)

pywws.Forecast.**Zambretti**(*params*, *hourly_data*)

pywws.Forecast.**main**(*argv=None*)

pywws.ZambrettiCore

Funzioni

ZambrettiCode(pressure, month, wind, trend)	Semplice implementazione dell'algoritmo previsioni del tempo di Zambretti.
ZambrettiText(letter)	
main([argv])	

pywws.ZambrettiCore.**ZambrettiCode**(*pressure*, *month*, *wind*, *trend*, *north=True*, *baro_top=1050.0*, *baro_bottom=950.0*)

Semplice implementazione dell'algoritmo previsioni del tempo di Zambretti. Ispirato dall'algoritmo Java di beteljuice.com, e convertito in Python da honeysucklecottage.me.uk, ulteriori informazioni da http://www.meteormetrics.com/zambretti.htm

pywws.ZambrettiCore.**ZambrettiText**(*letter*)

pywws.ZambrettiCore.**main**(*argv=None*)

pywws.Upload

Carica i file su un server web tramite ftp o li copia in una directory locale

```
usage: python -m pywws.Upload [options] data_dir file [file...]
options are:
 -h or --help    display this help
data_dir is the root directory of the weather data
file is a file to be uploaded
```

I dettagli login e ftp vengono letti dal file weather.ini in data_dir.

Introduzione

Questo modulo che carica il file (in genere) di un sito Web *tramite* ftp/sftp o copia i file in una directory locale (ad esempio se si esegue pywws sul proprio server web). Dettagli della destinazione upload sono memorizzati nel file `weather.ini` nella directory dati. L'unico modo per impostare questi dettagli è modificare il file. Eseguire *pywws.Upload* una volta per impostare i valori predefiniti, che poi possono cambiare. Ecco cosa è probabile trovare quando si modifica la `weather.ini`:

```
[ftp]
secure = False
directory = public_html/weather/data/
local site = False
password = secret
site = ftp.username.your_isp.co.uk
user = username
```

Queste sono, spero, abbastanza ovvie. Il `local site` consente di passare dal caricamento in un sito remoto alla copia di un sito locale. Se si imposta `local site = True` quindi è possibile eliminare le linee `secure`, `site`, `user` e `password`.

`directory``è il nome di una directory in cui verranno messi tutti i file caricati. Ciò dipenderà dalla struttura del sito web e l'ordinamento dell'host che si utilizza. Il provider di hosting dovrebbe essere in grado di dirvi quali dettagli ``site` and `user` utilizzare. Si dovrebbe già aver scelto una `password`.

L'opzione `secure` consente di passare da ftp normale a sftp (ftp sopra ssh). Alcuni fornitori di hosting offrono questo come un meccanismo di caricamento più sicuro, quindi si dovrebbe probabilmente usare se disponibile.

Dettagli API

Funzioni

`ApplicationLogger`(verbose[, logfile])	
main([argv])	

Classi

Upload(params)	

class pywws.Upload.**Upload**(*params*)

connect()

upload_file(*file*)

disconnect()

upload(*files*)

pywws.Upload.**main**(*argv=None*)

pywws.ToTwitter

Postare messaggi su Twitter

```
usage: python -m pywws.ToTwitter [options] data_dir file
options are:
 -h | --help  display this help
data_dir is the root directory of the weather data
file is the text file to be uploaded
```

Questo modulo invia un breve messaggio a Twitter. Prima di pubblicare su Twitter è necessario configurare un account e quindi autorizzare pywws eseguendo il programma `TwitterAuth`. Vedere *Come configurare pywws per pubblicare messaggi su Twitter* per le istruzioni dettagliate.

Funzioni

ApplicationLogger(verbose[, logfile])	
main([argv])	

Classi

ToTwitter(params)	
pct	alias per Twitter

class pywws.ToTwitter.**ToTwitter**(*params*)

Upload(*tweet*)

UploadFile(*file*)

pywws.ToTwitter.**main**(*argv=None*)

pywws.toservice

Posta gli aggiornamenti delle condizioni meteo a servizi come Weather Underground

```
usage: python -m pywws.toservice [options] data_dir service_name
options are:
 -h or --help     display this help
```

```
 -c or --catchup  upload all data since last upload
 -v or --verbose  increase amount of reassuring messages
data_dir is the root directory of the weather data
service_name is the service to upload to, e.g. underground
```

Introduzione

Diverse organizzazioni consentono alle stazioni meteorologiche di caricare i dati utilizzando un semplice HTTP 'POST' o 'GET' richiesta, con i dati codificati come una sequenza di coppie `key=value` separate dal carattere &.

Questo modulo permette a pywws di caricare letture a queste organizzazioni. E 'altamente personalizzabile utilizzando i file di configurazione. Ogni 'servizio' richiede un file di configurazione e di due modelli in `pywws/services` (Che non dovrebbe essere necessario modificare dall'utente) e una sezione in `weather.ini` contenente i dati specifici dell'utente, quali l'ID del sito e la password.

Attualmente ci sono sei servizi per i quali i file di configurazione sono stati scritti.

ente	nome del servizio	file di configurazione
UK Met Office	`metoffice`	`../../pywws/services/metoffice.ini`
Open Weather Map	`openweathermap`	`../../pywws/services/openweathermap.ini`
PWS Weather	`pwsweather`	`../../pywws/services/pwsweather.ini`
Stacja Pogody	`stacjapogodywawpl`	`../../pywws/services/stacjapogodywawpl.ini`
temperatur.nu	`temperaturnu`	`../../pywws/services/temperaturnu.ini`
Weather Underground	`underground`	`../../pywws/services/underground.ini`
wetter.com	`wetterarchivde`	`../../pywws/services/wetterarchivde.ini`

Configurazione

Se non lo hai già fatto, visitate il sito web dell'organizzazione e creare un account per la vostra stazione meteo. Prendere nota di qualsiasi ID del sito e la password che vi vengono dati.

Interrompere qualsiasi software Pywws che è in esecuzione, quindi esegui `toservice` per creare una sezione in `weather.ini`:

```
python -m pywws.toservice data_dir service_name
```

`service_name` è un nome (singola parola) del servizio, ad esempio `metoffice`, `data_dir` è la directory dei dati meteo, come al solito.

Modificare `weather.ini` e trovare la sezione corrispondente al nome del servizio, ad esempio `[underground]`. Copiare i dettagli del tuo sito in questa sezione, ad esempio:

```
[underground]
password = secret
station = ABCDEFG1A
```

Ora si può verificare la configurazione:

```
python -m pywws.toservice -vvv data_dir service_name
```

Questo dovrebbe mostrare la stringa di dati che viene caricata. Qualsiasi inconveniente dovrebbe generare un messaggio di errore.

Carica i dati vecchi

Ora potete caricare il vostri ultimi 7 giorni di dati, se il servizio lo supporta. Modifica il tuo file `status.ini` e cancella la riga relativa nella sezione `last update`, quindi eseguire di nuovo `toservice` con l'opzione 'catchup':

```
python -m pywws.toservice -cvv data_dir service_name
```

Questo può durare 20 minuti o più, a seconda della quantità di dati.

Aggiungere servizi di upload ad intervalli regolari

Modificare nuovamente il file `weather.ini` e aggiungere un elenco dei servizi per nella sezione `[live]`, `[logged]`, `[hourly]`, `[12 hourly]` o ''[daily]', a seconda di quante volte si desidera inviare i dati. Ad esempio:

```
[live]
twitter = []
plot = []
text = []
services = ['underground_rf']

[logged]
twitter = []
plot = []
text = []
services = ['metoffice', 'stacjapogodywawpl']

[hourly]
twitter = []
plot = []
text = []
services = ['underground']
```

Si noti che la sezione `[live]` viene utilizzata solo quando si esegue *LiveLog*. È una buona idea di ripetere qualsiasi servizio selezionato in `[live]` nella sezione `[logged]` o `[hourly]` nel caso in cui si passa all' esecuzione di *Hourly*.

Riavviare il programma pywws (*Hourly* o *LiveLog*) e visitare il sito web appropriato per vedere gli aggiornamenti per la tua stazione meteo.

API

Funzioni

`ApplicationLogger`(verbose[, logfile])	
main([argv])	

Classi

`SafeConfigParser`([defaults, dict_type, ...])	

Continua alla pagina successiva

Tabella 4.35 – continua dalla pagina precedente

ToService(params, status, calib_data, ...)	Carica dati meteo a servizi meteorologici come Weather Underground.
datetime(year, month, day[, hour[, minute[, ...)	The year, month and day arguments are required.
timedelta	Differenza tra due valori datetime.

class pywws.toservice.**ToService**(*params*, *status*, *calib_data*, *service_name*)

Carica dati meteo a servizi meteorologici come Weather Underground.

Parametri

- **params** (*pywws.DataStore.params*) – Configurazione pywws
- **status** (*pywws.DataStore.status*) – Stato dell' archivio pywws.
- **calib_data** (*pywws.DataStore.calib_store*) – 'calibrated' data.
- **service_name** (*string*) – nome del servizio per caricare in.

encode_data(*data*)

Codificare un record di dati meteo.

Il parametro data contiene i dati da codificare. Dovrebbe essere un record di dati 'calibrato', memorizzato in *pywws.DataStore.calib_store*.

Parametri **data** (*dict*) – il record di dati meteo.

Ritorna Url dati codificati

Tipo di ritorno string

send_data(*coded_data*)

Carica un set di dati meteo.

Il parametro coded_data contiene i dati da caricare. Dovrebbe essere una stringa url codificata.

Parametri **coded_data** – I dati da caricare.

Ritorna esito positivo

Tipo di ritorno bool

next_data(*start*, *live_data*)

Riceve i record di dati meteo da caricare .

Questo metodo restituisce i record meteo più recenti oppure tutti i record di un oggetto datetime, secondo il valore di start.

Parametri

- **start** (*datetime*) – data-ora del primo record da leggere, o None per ottenere solo i dati più recenti.
- **live_data** (*dict*) – Il corrente record di dati 'live' , o None.

Ritorna restituisce i record di dati meteo.

Tipo di ritorno dict

set_status(*timestamp*)

catchup_start()

Ottiene data e ora del primo 'catchup' record per inviare.

Tipo di ritorno datetime

Upload(*catchup=True*, *live_data=None*)
Caricare uno o più record di dati meteorologici.

Questo metodo carica o il più recente record di dati meteo, o tutti i record dopo l'ultimo caricamento (fino a 7 giorni), secondo il valore di `catchup`.

Imposta il valore di configurazione `last update` il timestamp del record più recente correttamente caricato.

Parametri **catchup** (*bool*) – carica tutti i dati dall'ultimo caricamento.

Ritorna esito positivo

Tipo di ritorno bool

pywws.toservice.**main**(*argv=None*)

pywws.YoWindow

Generare file XML YoWindow

```
usage: python -m pywws.YoWindow [options] data_dir output_file
options are:
 -h or --help     display this help
 -v or --verbose  increase amount of reassuring messages
data_dir is the root directory of the weather data
output_file is the YoWindow XML file to be written
```

Funzioni

ApplicationLogger(verbose[, logfile])	
apparent_temp(temp, rh, wind)	Calcola la temperatura apparente (sensazione reale), con formula da
main([argv])	

Classi

YoWindow(calib_data)	Classe YoWindow per scrivere il file XML
datetime(year, month, day[, hour[, minute[, ...)	The year, month and day arguments are required.
timedelta	Differenza tra due valori datetime.

class pywws.YoWindow.**YoWindow**(*calib_data*)
Classe YoWindow per scrivere il file XML. Specifiche per il file per vedere http://yowindow.com/doc/yowindow_pws_format.xml

write_file(*file_name*, *data=None*)

pywws.YoWindow.**main**(*argv=None*)

pywws.WeatherStation

Ottenere i dati da stazioni meteo compatibili con WH1080/WH3080.

Derivato da wwsr.c by Michael Pendec (michael.pendec@gmail.com), wwsrdump.c by Svend Skafte (svend@skafte.net), modificato da Dave Wells, e altre fonti.

Introduzione

Questo è il modulo che in realtà comunica con l'unità di base della stazione meteo. Non ho molta comprensione dell' USB, quindi copiato molto dal programma in C wwsr di Michael Pendec's

La memoria della stazione meteo è divisa in due parti: "fixed block" di 256 byte e un buffer circolare di 65280 byte. Siccome ogni stringa prende 16 byte la stazione può immagazzinare 4080 letture, o 14 giorni di letture di intervallo di 5 minuti. (Le stazioni tipo 3080 memorizzano 20 byte per la stringa, quindi memorizzano un massimo di 3264). Siccome lettura dei dati è in blocchi di 32 byte, ma ogni lettura meteo è 16 o 20 byte, una piccola cache viene utilizzata per ridurre il traffico USB. Il comportamento di memorizzazione nella cache può essere esclusa con il parametro `unbuffered` in `get_data` e `get_raw_data`.

La decodifica dei dati è controllato da dizionari statici `reading_format`, `lo_fix_format` e `fixed_format`. Le chiavi sono i nomi di elementi di dati e i valori possono essere di tipo `(offset, type, multiplier)` o un altro dizionario. Così, per esempio, la voce del dizionario reading_format `'rain' : (13, 'us', 0.3)` significa che il valore di pioggia è un breve short senza segno (due byte), 13 byte dall'inizio del blocco e deve essere moltiplicato per 0,3 per ottenere un valore utile.

L'uso di dizionari annidati nel dizionario `fixed_format` permette utili sottoinsiemi di dati per essere decodificati. Ad esempio, per decodificare l'intero blocco `get_fixed_block` viene chiamato senza parametri

```
ws = WeatherStation.weather_station()
print ws.get_fixed_block()
```

Per ottenere la temperatura esterna minima memorizzata, è chiamato `get_fixed_block` con una sequenza di comandi:

```
ws = WeatherStation.weather_station()
print ws.get_fixed_block(['min', 'temp_out', 'val'])
```

Spesso non non c'è nessun obbligo di leggere e decodificare l'intero blocco, siccome i primi 64 byte contengono i dati più utili: l'intervallo dei valori memorizzati, l'indirizzo del buffer dove è memorizzata la lettura corrente e l'ora della data corrente. Il metodo `get_lo_fix_block` fornisce facile accesso a questi dati.

Per ulteriori esempi di utilizzo del modulo WeatherStation, consultare il programma TestWeatherStation.

Dettagli API

Funzioni

decode_status(status)	

Classi

CUSBDrive()	Interfaccia di basso livello della stazione meteo tramite USB.
USBDevice(vendor_id, product_id)	Basso livello di accesso al dispositivo USB tramite la libreria hidapi.

Continua alla pagina successiva

Tabella 4.39 – continua dalla pagina precedente

`datetime`(year, month, day[, hour[, minute[, ...)	The year, month and day arguments are required.
`weather_station`([ws_type, params, status, avoid])	Classe che rappresenta la stazione meteorologica nel programma utente.

pywws.WeatherStation.**decode_status**(*status*)

class pywws.WeatherStation.**CUSBDrive**

Interfaccia di basso livello della stazione meteo tramite USB.

Liberamente ispirato su una classe C++ ottenuta da http://site.ambientweatherstore.com/easyweather/ws_1080_2080_protocol.zip.Non so la provenienza di questo, ma sembra che siano venuti dal produttore.

EndMark = 32

ReadCommand = 161

WriteCommand = 160

WriteCommandWord = 162

read_block(*address*)

Legge 32 byte dalla stazione meteo.

Se la lettura non riesce per qualche motivo, `None` viene restituito.

Parametri **address** (*int*) – Indirizzo di lettura da.

Ritorna I dati della stazione meteo.

Tipo di ritorno list(int)

write_byte(*address*, *data*)

Scrive un singolo byte sulla stazione meteo.

Parametri

- **address** (*int*) – Indirizzo di scrittura su.
- **data** (*int*) – il valore da scrivere.

Ritorna eseguito con successo.

Tipo di ritorno bool

class pywws.WeatherStation.**weather_station**(*ws_type='1080'*, *params=None*, *status=None*, *avoid=3.0*)

Classe che rappresenta la stazione meteorologica nel programma utente.

Connettersi alla stazione metereologica e preparare per la lettura dei dati.

min_pause = 0.5

margin = 0.9

live_data(*logged_only=False*)

inc_ptr(*ptr*)

Ottenere il puntatore ai dati successivi del buffer circolare.

dec_ptr(*ptr*)

Ottenere il puntatore ai dati precedenti nel buffer circolare.

get_raw_data(*ptr*, *unbuffered=False*)

Ottiene i dati grezzi (raw) dal buffer circolare.

Se 'unbuffered' è false, potrebbe essere restituito un valore memorizzato nella cache, che è stato ottenuto in precedenza.

get_data(*ptr*, *unbuffered=False*)
Ottiene i dati decodificati dal buffer circolare.

Se 'unbuffered' è false, potrebbe essere restituito un valore memorizzato nella cache, che è stato ottenuto in precedenza.

current_pos()
Ottiene la posizione del buffer circolare dove sono scritti dati correnti.

get_raw_fixed_block(*unbuffered=False*)
Ottiene il grezzo (raw) "fixed block" di impostazioni e dati di min/max.

get_fixed_block(*keys=[]*, *unbuffered=False*)
Ottiene decodificato "fixed block" di impostazioni e dati di min/max.

Per selezionare un sottoinsieme di tutto il blocco tasti.

write_data(*data*)
Scrive una serie di byte singoli sulla stazione meteo. Dati devono essere una matrice di coppie (ptr, value).

reading_format = {'3080': {'status': (15, 'pb', None), 'hum_out': (4, 'ub', None), 'wind_gust': (10, 'wg', 0.1), 'uv': (19

lo_fix_format = {'alarm_1': (21, 'bf', ('bit0', 'time', 'wind_dir', 'bit3', 'hum_in_lo', 'hum_in_hi', 'hum_out_lo', 'hum

fixed_format = {'alarm_1': (21, 'bf', ('bit0', 'time', 'wind_dir', 'bit3', 'hum_in_lo', 'hum_in_hi', 'hum_out_lo', 'hum_

data_start = 256

reading_len = {'3080': 20, '1080': 16}

pywws.device_ctypes_hidapi

Interfaccia di basso livello USB della stazione meteo, utilizzando ctypes per accedere a hidapi.

Introduzione

Questo modulo gestisce la comunicazione di livello basso con la stazione meteo tramite la libreria ctypes e hidapi. Un modo alternativo utilizza la libreria, *pywws.device_cython_hidapi*, e *pywws.device_pyusb*. La scelta di quale modo utilizzare dipende da quali librerie sono disponibili per il tuo computer.

Gli utenti delle recenti versioni di Mac OS non hanno altra scelta. Il sistema operativo rende molto difficile accedere direttamente, ai dispositivi HID (come la stazione meteorologica) così la libreria `hidapi` deve essere usata.

Gli utenti di OpenWRT e simili piattaforme Linux embedded probabilmente non sono in grado di installare `ctypes` o `cython-hidapi`, così sono costretti ad usare `libusb` e la sua interfaccia Python `PyUSB`.

Installazione

Alcuni di questi software potrebbero essere già installati sulla vostra macchina, quindi controllare prima di scaricare i sorgenti e compilarli da te.

1. Installazione hidapi

   Creare una copia locale del repository git, passare alla nuova directory e quindi seguire le istruzioni in `README.txt`:

```
git clone https://github.com/signal11/hidapi.git
cd hidapi
more README.txt
```

2. Installare ctypes.

   Questo dovrebbe essere disponibile come pacchetto per il sistema operativo. Ad esempio:

```
sudo zypper install python-ctypes
```

Collaudo

Eseguire `TestWeatherStation.py` con un livello di dettaglio maggiore così segnala quale modulo USB di accesso al dispositivo viene usato:

```
python TestWeatherStation.py -vv
18:10:27:pywws.WeatherStation.CUSBDrive:using pywws.device_ctypes_hidapi
0000 55 aa ff ff ff ff ff ff ff ff ff ff ff ff ff ff 05 20 01 51 11 00 00 00 81 00 00␣
↪07 01 00 d0 56
0020 61 1c 61 1c 00 00 00 00 00 00 00 12 02 14 18 09 41 23 c8 00 32 80 47 2d 2c 01 2c␣
↪81 5e 01 1e 80
0040 a0 00 c8 80 a0 28 80 25 a0 28 80 25 03 36 00 05 6b 00 00 0a 00 f4 01 18 00 00 00␣
↪00 00 00 00 00
0060 00 00 54 1c 63 0a 2f 01 71 00 7a 01 59 80 7a 01 59 80 e4 00 f5 ff 69 54 00 00 fe␣
↪ff 00 00 b3 01
0080 0c 02 d0 ff d3 ff 5a 24 d2 24 dc 17 00 11 09 06 15 40 10 03 07 22 18 10 08 11 08␣
↪30 11 03 07 12
00a0 36 08 07 24 17 17 11 02 28 10 10 09 06 30 14 29 12 02 11 06 57 09 06 30 14 29 12␣
↪02 11 06 57 08
00c0 08 31 14 30 12 02 14 18 04 12 02 01 10 12 11 09 13 17 19 11 08 21 16 53 11 09 13␣
↪17 19 12 01 18
00e0 07 17 10 02 22 11 06 11 11 06 13 12 11 11 06 13 12 11 11 10 11 38 11 11 10 11 38␣
↪10 02 22 14 43
```

API

Funzioni

find_library(name)	

Classi

USBDevice(vendor_id, product_id)	Basso livello di accesso al dispositivo USB tramite la libreria hidapi.

class pywws.device_ctypes_hidapi.**USBDevice**(*vendor_id*, *product_id*)

Basso livello di accesso al dispositivo USB tramite la libreria hidapi.

Parametri

- **idVendor** (*int*) – the USB "vendor ID" number, per esempio 0x1941.

- **idProduct** (*int*) – the USB "product ID" number, per esempio 0x8021.

read_data(*size*)
Riceve i dati dalla stazione meteo.

Se la lettura non riesce per qualche motivo, `IOError` l'eccezione è restituita.

Parametri **size** (*int*) – il numero di byte da leggere.

Ritorna i dati ricevuti.

Tipo di ritorno list(int)

write_data(*buf*)
Inviare dati al dispositivo.

Parametri **buf** (*list(int)*) – i dati da inviare.

Ritorna eseguito con successo.

Tipo di ritorno bool

pywws.device_cython_hidapi

Interfaccia di basso livello USB della stazione meteo, tramite cython-hidapi.

Introduzione

Questo modulo gestisce la comunicazione di livello basso con la stazione meteo tramite la libreria cython-hidapi. Un modulo alternativo, *pywws.device_pyusb*, utilizza la libreria PyUSB. La scelta di quale modulo utilizzare dipende da quali librerie sono disponibili per il tuo computer.

Gli utenti delle recenti versioni di Mac OS non hanno altra scelta. Il sistema operativo rende molto difficile accedere direttamente, ai dispositivi HID (come la stazione meteorologica) così la libreria `hidapi` deve essere usata. `cython-hidapi` è un'interfaccia Python per quella libreria.

Gli utenti di OpenWRT e simili piattaforme Linux embedded probabilmente non sono in grado di installare `cython-hidapi`, così sono costretti ad usare `libusb` e la sua interfaccia Python `PyUSB`.

Installazione

Alcuni di questi software potrebbero essere già installati sulla vostra macchina, quindi controllare prima di scaricare i sorgenti e compilarli da te.

1. Installazione hidapi

   Creare una copia locale del repository git, passare alla nuova directory e quindi seguire le istruzioni in `README.txt`:

   ```
   git clone https://github.com/signal11/hidapi.git
   cd hidapi
   more README.txt
   ```

2. Installare cython.

   Questo dovrebbe essere disponibile come pacchetto per il sistema operativo. Ad esempio:

   ```
   sudo apt-get install cython
   ```

3. Installare cython-hidapi.

   Anche questo va scaricato e installato:

```
git clone https://github.com/gbishop/cython-hidapi.git
cd cython-hidapi
python setup.py build
sudo python setup.py install
```

   Sostituire `setup.py` con `setup-mac.py` o `setup-windows.py` se si utilizza Mac OS o Windows.

Collaudo

Eseguire `TestWeatherStation.py` con un livello di dettaglio maggiore così segnala quale modulo USB di accesso al dispositivo viene usato:

```
python TestWeatherStation.py -vv
18:10:27:pywws.WeatherStation.CUSBDrive:using pywws.device_cython_hidapi
0000 55 aa ff ff ff ff ff ff ff ff ff ff ff ff ff ff 05 20 01 51 11 00 00 00 81 00 00␣
↪07 01 00 d0 56
0020 61 1c 61 1c 00 00 00 00 00 00 00 12 02 14 18 09 41 23 c8 00 32 80 47 2d 2c 01 2c␣
↪81 5e 01 1e 80
0040 a0 00 c8 80 a0 28 80 25 a0 28 80 25 03 36 00 05 6b 00 00 0a 00 f4 01 18 00 00 00␣
↪00 00 00 00 00
0060 00 00 54 1c 63 0a 2f 01 71 00 7a 01 59 80 7a 01 59 80 e4 00 f5 ff 69 54 00 00 fe␣
↪ff 00 00 b3 01
0080 0c 02 d0 ff d3 ff 5a 24 d2 24 dc 17 00 11 09 06 15 40 10 03 07 22 18 10 08 11 08␣
↪30 11 03 07 12
00a0 36 08 07 24 17 17 11 02 28 10 10 09 06 30 14 29 12 02 11 06 57 09 06 30 14 29 12␣
↪02 11 06 57 08
00c0 08 31 14 30 12 02 14 18 04 12 02 01 10 12 11 09 13 17 19 11 08 21 16 53 11 09 13␣
↪17 19 12 01 18
00e0 07 17 10 02 22 11 06 11 11 06 13 12 11 11 06 13 12 11 11 10 11 38 11 11 10 11 38␣
↪10 02 22 14 43
```

API

Classi

USBDevice(idVendor, idProduct)	Basso livello di accesso al dispositivo USB tramite libreria cython-hidapi.

class pywws.device_cython_hidapi.**USBDevice**(*idVendor*, *idProduct*)

Basso livello di accesso al dispositivo USB tramite libreria cython-hidapi.

Parametri

- **idVendor** (*int*) – the USB "vendor ID" number, per esempio 0x1941.
- **idProduct** (*int*) – the USB "product ID" number, per esempio 0x8021.

read_data(*size*)

Riceve i dati dalla stazione meteo.

Se la lettura non riesce per qualche motivo, `IOError` l'eccezione è restituita.

Parametri **size** (*int*) – il numero di byte da leggere.

Ritorna i dati ricevuti.

Tipo di ritorno list(int)

write_data(*buf*)
Inviare dati al dispositivo.

Parametri **buf** (*list(int)*) – i dati da inviare.

Ritorna eseguito con successo.

Tipo di ritorno bool

pywws.device_pyusb1

Interfaccia di basso livello USB della stazione meteo tramite PyUSB.

Introduzione

Questo modulo gestisce la comunicazione di livello basso con la stazione meteo tramite la libreria PyUSB (versione 1.0). Un modulo alternativo utilizza la libreria, *pywws.device_ctypes_hidapi*, e *pywws.device_cython_hidapi*. La scelta di quale modulo utilizzare dipende da quali librerie sono disponibili per il computer.

Gli utenti delle recenti versioni di Mac OS non hanno altra scelta. Il sistema operativo rende molto difficile accedere direttamente, ai dispositivi HID (come la stazione meteorologica) così la libreria `hidapi` deve essere usata.

Installazione

Alcuni di questi software potrebbero essere già installati sulla vostra macchina, quindi controllare prima di scaricare i sorgenti e compilarli da te.

1. Installazione di libusb e PyUSB.

   Questi dovrebbero essere disponibili come pacchetti per sistema operativo, ma i loro nomi possono variare. Ad esempio, su Ubuntu Linux:

   ```
   sudo apt-get install python-usb
   ```

   Su alcuni sistemi embedded di linux:

   ```
   ipkg install libusb py25-usb
   ```

Collaudo

Eseguire `TestWeatherStation.py` con un livello di dettaglio maggiore così segnala quale modulo USB di accesso al dispositivo viene usato:

```
python TestWeatherStation.py -vv
18:28:09:pywws.WeatherStation.CUSBDrive:using pywws.device_pyusb1
0000 55 aa ff ff ff ff ff ff ff ff ff ff ff ff ff ff 05 20 01 41 11 00 00 00 81 00 00␣
↪0f 05 00 e0 51
0020 03 27 ce 27 00 00 00 00 00 00 00 00 12 02 14 18 27 41 23 c8 00 00 00 46 2d 2c 01 64␣
↪80 c8 00 00 00
```

```
0040 64 00 64 80 a0 28 80 25 a0 28 80 25 03 36 00 05 6b 00 00 0a 00 f4 01 12 00 00 00␣
↪00 00 00 00 00
0060 00 00 49 0a 63 12 05 01 7f 00 36 01 60 80 36 01 60 80 bc 00 7b 80 95 28 12 26 6c␣
↪28 25 26 c8 01
0080 1d 02 d8 00 de 00 ff 00 ff 00 ff 00 00 11 10 06 01 29 12 02 01 19 32 11 09 09 05␣
↪18 12 01 22 13
00a0 14 11 11 04 15 04 11 12 17 05 12 11 09 02 15 26 12 02 11 07 05 11 09 02 15 26 12␣
↪02 11 07 05 11
00c0 09 10 09 12 12 02 02 12 38 12 02 07 19 00 11 12 16 03 27 12 02 03 11 00 11 12 16␣
↪03 27 11 12 26
00e0 21 32 11 12 26 21 32 12 02 06 19 57 12 02 06 19 57 12 02 06 19 57 12 02 06 19 57␣
↪12 02 06 19 57
```

API

Classi

USBDevice(idVendor, idProduct)	Basso livello del dispositivo di accesso USB tramite libreria PyUSB 1.0.

class pywws.device_pyusb1.**USBDevice**(*idVendor*, *idProduct*)

Basso livello del dispositivo di accesso USB tramite libreria PyUSB 1.0.

Parametri

- **idVendor** (*int*) – the USB "vendor ID" number, per esempio 0x1941.
- **idProduct** (*int*) – the USB "product ID" number, per esempio 0x8021.

read_data(*size*)

Riceve i dati dalla stazione meteo.

Se la lettura non riesce per qualche motivo, IOError l'eccezione è restituita.

Parametri **size** (*int*) – il numero di byte da leggere.

Ritorna i dati ricevuti.

Tipo di ritorno list(int)

write_data(*buf*)

Inviare dati al dispositivo.

Se la lettura non riesce per qualche motivo, IOError l'eccezione è restituita.

Parametri **buf** (*list(int)*) – i dati da inviare.

Ritorna eseguito con successo.

Tipo di ritorno bool

pywws.device_pyusb

Interfaccia di basso livello USB della stazione meteo tramite PyUSB.

Introduzione

Questo modulo gestisce la comunicazione di livello basso con la stazione meteo tramite la libreria PyUSB. Un modulo alternativo utilizza la libreria, *pywws.device_cython_hidapi*. La scelta di quale modulo utilizzare dipende da quali librerie sono disponibili per il computer.

Gli utenti delle recenti versioni di Mac OS non hanno altra scelta. Il sistema operativo rende molto difficile accedere direttamente, ai dispositivi HID (come la stazione meteorologica) così la libreria `hidapi` deve essere usata. `cython-hidapi` è un'interfaccia Python per quella libreria.

Gli utenti di OpenWRT e simili piattaforme Linux embedded probabilmente non sono in grado di installare `cython-hidapi`, così sono costretti ad usare `libusb` e la sua interfaccia Python `PyUSB`.

Installazione

Alcuni di questi software potrebbero essere già installati sulla vostra macchina, quindi controllare prima di scaricare i sorgenti e compilarli da te.

1. Installazione di libusb e PyUSB.

   Questi dovrebbero essere disponibili come pacchetti per sistema operativo, ma i loro nomi possono variare. Ad esempio, su Ubuntu Linux:

   ```
   sudo apt-get install libusb-0.1 python-usb
   ```

   Su alcuni sistemi embedded di linux:

   ```
   ipkg install libusb py25-usb
   ```

Collaudo

Eseguire `TestWeatherStation.py` con un livello di dettaglio maggiore così segnala quale modulo USB di accesso al dispositivo viene usato:

```
python TestWeatherStation.py -vv
18:28:09:pywws.WeatherStation.CUSBDrive:using pywws.device_pyusb
0000 55 aa ff ff ff ff ff ff ff ff ff ff ff ff ff 05 20 01 41 11 00 00 00 81 00 00␣
↪0f 05 00 e0 51
0020 03 27 ce 27 00 00 00 00 00 00 00 12 02 14 18 27 41 23 c8 00 00 00 46 2d 2c 01 64␣
↪80 c8 00 00 00
0040 64 00 64 80 a0 28 80 25 a0 28 80 25 03 36 00 05 6b 00 00 0a 00 f4 01 12 00 00 00␣
↪00 00 00 00 00
0060 00 00 49 0a 63 12 05 01 7f 00 36 01 60 80 36 01 60 80 bc 00 7b 80 95 28 12 26 6c␣
↪28 25 26 c8 01
0080 1d 02 d8 00 de 00 ff 00 ff 00 ff 00 00 11 10 06 01 29 12 02 01 19 32 11 09 09 05␣
↪18 12 01 22 13
00a0 14 11 11 04 15 04 11 12 17 05 12 11 09 02 15 26 12 02 11 07 05 11 09 02 15 26 12␣
↪02 11 07 05 11
00c0 09 10 09 12 12 02 02 12 38 12 02 07 19 00 11 12 16 03 27 12 02 03 11 00 11 12 16␣
↪03 27 11 12 26
00e0 21 32 11 12 26 21 32 12 02 06 19 57 12 02 06 19 57 12 02 06 19 57 12 02 06 19 57␣
↪12 02 06 19 57
```

API

Classi

USBDevice(idVendor, idProduct)	Basso livello di accesso al dispositivo USB tramite libreria PyUSB.

class pywws.device_pyusb.**USBDevice**(*idVendor*, *idProduct*)
Basso livello di accesso al dispositivo USB tramite libreria PyUSB.

Parametri

- **idVendor** (*int*) – the USB "vendor ID" number, per esempio 0x1941.
- **idProduct** (*int*) – the USB "product ID" number, per esempio 0x8021.

read_data(*size*)
Riceve i dati dalla stazione meteo.

Se la lettura non riesce per qualche motivo, IOError l'eccezione è restituita.

Parametri **size** (*int*) – il numero di byte da leggere.

Ritorna i dati ricevuti.

Tipo di ritorno list(int)

write_data(*buf*)
Inviare dati al dispositivo.

Se la lettura non riesce per qualche motivo, IOError l'eccezione è restituita.

Parametri **buf** (*list(int)*) – i dati da inviare.

Ritorna eseguito con successo.

Tipo di ritorno bool

pywws.DataStore

DataStore.py - memorizza le letture in file di facile accesso

Introduzione

Questo modulo è il cuore del mio software per la stazione meteo. Esso memorizza i dati su disco, ma senza il sovraccarico di un complesso database. L'ho progettato per funzionare su una macchina con poca memoria come il mio router Asus. Per ridurre al minimo l'utilizzo della memoria, carica solo l'equivalente di un giorno di dati alla volta in memoria.

Da un punto di vista "user", i dati vengono eseguiti come un incrocio tra un elenco e un dizionario. Ogni record di dati è indicizzato da un oggetto datetime.datetime (comportamento di dizionario), ma i record vengono archiviati in ordine ed è possibile accedervi come fogli (comportamento di elenco).

Ad esempio, per accedere ai dati orari per il giorno di Natale 2009, uno potrebbe effettuare le seguenti operazioni:

```
from datetime import datetime
from pywws import DataStore
hourly = DataStore.hourly_store('weather_data')
for data in hourly[datetime(2009, 12, 25):datetime(2009, 12, 26)]:
    print data['idx'], data['temp_out']
```

Alcuni esempi di accesso dati:

```
# get value nearest 9:30 on Christmas day 2008
data[data.nearest(datetime(2008, 12, 25, 9, 30))]
# get entire array, equivalent to data[:]
data[datetime.min:datetime.max]
# get last 12 hours worth of data
data[datetime.utcnow() - timedelta(hours=12):]
```

Si noti che l'indice `datetime.datetime` è in formato UTC. Potrebbe essere necessario applicare un offset per convertire in ora locale.

Il modulo fornisce cinque classi per archiviare dati diversi. *data_store* prende i dati "raw" dalla stazione meteo; *calib_store*, *hourly_store*, *daily_store* e *monthly_store* memorizza i dati elaborati (vedi *pywws.Process*).Tutti e tre sono derivati dallo stessa classe `core_store`, differiscono solo per le chiavi e i tipi di dati memorizzati in ogni record.

Dettagli API

Funzioni

safestrptime(date_string[, format])	

Classi

ParamStore(root_dir, file_name)	
RawConfigParser([defaults, dict_type, ...])	
calib_store(root_dir)	Memorizza i dati 'calibrati' della Stazione Meteo.
core_store(root_dir)	
daily_store(root_dir)	Memorizza i dati giornalieri di riepilogo della Stazione Meteo.
data_store(root_dir)	Memorizza i dati grezzi Stazione Meteo.
date	date(year, month, day) –> date object
datetime(year, month, day[, hour[, minute[, ...)	The year, month and day arguments are required.
hourly_store(root_dir)	Memorizza i dati orari di riepilogo della Stazione Meteo.
monthly_store(root_dir)	Memorizza i dati mensili di riepilogo della Stazione Meteo.
params(root_dir)	I parametri sono memorizzati in un file "weather.ini" in root_dir.
status(root_dir)	Lo stato viene memorizzato in un file "status.ini" in root_dir.
timedelta	Differenza tra due valori datetime.

pywws.DataStore.**safestrptime**(*date_string*, *format=None*)

class pywws.DataStore.**ParamStore**(*root_dir*, *file_name*)

flush()

get(*section*, *option*, *default=None*)
Ottiene un dato di parametro e restituisce una stringa.

Se si specifica default e la sezione o l'opzione non sono definite nel file, sono creati e impostati su default, che è poi il valore restituito.

get_datetime(*section*, *option*, *default=None*)

set(*section*, *option*, *value*)
Impostare l'opzione nella sezione valore stringa.

unset(*section*, *option*)
Rimuovere l'opzione dalla sezione.

class pywws.DataStore.**params**(*root_dir*)
I parametri sono memorizzati in un file "weather.ini" in root_dir.

class pywws.DataStore.**status**(*root_dir*)
Lo stato viene memorizzato in un file "status.ini" in root_dir.

class pywws.DataStore.**core_store**(*root_dir*)

before(*idx*)
Ritorno data ora dell'ultimo record di dati esistenti cui data ora è < idx.

Potrebbe anche non essere nello stesso anno! Se non esiste nessun record, restituisce None.

after(*idx*)
Restuisce data ora del record di dati esistente più antico cui data ora è >= idx.

Potrebbe anche non essere nello stesso anno! Se non esiste nessun record, restituisce None.

nearest(*idx*)
Restuisce data ora del record cui data ora è più vicina di idx.

flush()

class pywws.DataStore.**data_store**(*root_dir*)
Memorizza i dati grezzi Stazione Meteo.

key_list = ['idx', 'delay', 'hum_in', 'temp_in', 'hum_out', 'temp_out', 'abs_pressure', 'wind_ave', 'wind_gust', 'wind_

conv = {'status': <type 'int'>, 'wind_ave': <type 'float'>, 'rain': <type 'float'>, 'hum_in': <type 'int'>, 'temp_out': <typ

class pywws.DataStore.**calib_store**(*root_dir*)
Memorizza i dati 'calibrati' della Stazione Meteo.

key_list = ['idx', 'delay', 'hum_in', 'temp_in', 'hum_out', 'temp_out', 'abs_pressure', 'rel_pressure', 'wind_ave', 'win

conv = {'status': <type 'int'>, 'wind_ave': <type 'float'>, 'rain': <type 'float'>, 'rel_pressure': <type 'float'>, 'hum_in':

class pywws.DataStore.**hourly_store**(*root_dir*)
Memorizza i dati orari di riepilogo della Stazione Meteo.

key_list = ['idx', 'hum_in', 'temp_in', 'hum_out', 'temp_out', 'abs_pressure', 'rel_pressure', 'pressure_trend', 'wind_

conv = {'pressure_trend': <type 'float'>, 'wind_ave': <type 'float'>, 'rain': <type 'float'>, 'rel_pressure': <type 'float'>,

class pywws.DataStore.**daily_store**(*root_dir*)
Memorizza i dati giornalieri di riepilogo della Stazione Meteo.

key_list = ['idx', 'start', 'hum_out_ave', 'hum_out_min', 'hum_out_min_t', 'hum_out_max', 'hum_out_max_t', 'temp

conv = {'temp_in_min': <type 'float'>, 'temp_in_max': <type 'float'>, 'uv_ave': <type 'float'>, 'temp_out_max': <type '

class pywws.DataStore.**monthly_store**(*root_dir*)

Memorizza i dati mensili di riepilogo della Stazione Meteo.

key_list = ['idx', 'start', 'hum_out_ave', 'hum_out_min', 'hum_out_min_t', 'hum_out_max', 'hum_out_max_t', 'temp

conv = {'uv_ave': <type 'float'>, 'illuminance_max_hi_t': <function safestrptime>, 'uv_max_lo_t': <function safestrptim

pywws.TimeZone

datetime.tzinfo sono paio di oggetti che rappresentano l'ora locale e UTC

Introduzione

Questo modulo fornisce due datetime.tzinfo gli oggetti che rappresentano l'ora UTC e l'ora locale. Questi sono utilizzati per convertire timestamp da e verso l'ora UTC e l'ora locale. Il software della stazione meteo archivia i dati con data e ora UTC, per evitare problemi con l'ora legale, ma i programmi modelli template e i grafici usano l'ora locale.

Il modulo viene copiato direttamente dalla documentazione datetime.tzinfo.

Dettagli API

Classi

LocalTimezone	Ora locale
UTC	
datetime(year, month, day[, hour[, minute[, ...)	The year, month and day arguments are required.
timedelta	Differenza tra due valori datetime.
tzinfo	Classe base astratta per informazioni oggetti di fuso orario.

class pywws.TimeZone.**UTC**

utcoffset(*dt*)

tzname(*dt*)

dst(*dt*)

class pywws.TimeZone.**LocalTimezone**

Ora locale

utcoffset(*dt*)

dst(*dt*)

tzname(*dt*)

pywws.Localisation

Localisation.py - effetua le traduzioni di stringhe in lingua locale

```
usage: python -m pywws.Localisation [options]
options are:
 -h       or  --help       display this help
 -t code  or  --test code  test use of a language code
```

Introduzione

Alcuni dei moduli pywws, come WindRose.py, può utilizzare automaticamente la lingua locale per cose come la direzione del vento. Il modulo Localisation.py, per lo più copiato da esempi nella documentazione di Python, permette questo

Localizzazione di pywws è fatta in due parti - tradurre le stringhe come 'rising very rapidly', e cambiando le impostazioni internazionali che controlla le diciture come i nomi dei mesi e la rappresentazione dei numeri (e.g. '23,2' cambiando in '23.2'). Su alcuni computer potrebbe non essere possibile impostare le impostazioni internazionali, ma è possibile utilizzare le stringhe tradotte.

Usare un linguaggio diverso

Il linguaggio utilizzato da pywws e definito nel file `weather.ini` alla sezione `[config]`. Questo può essere un codice di due lettere della lingua, ad esempio `it` (Italiano), o si può specificare una variante nazionale, come `fr_CA` (Canadian French). Potrebbe anche includere un set di caratteri, ad esempio `de_DE.UTF-8`.

La scelta della lingua è a carico del sistema operativo, così Localisation.py può essere eseguito come programma autonomo per testare i codici di lingua. Un buon punto di partenza potrebbe essere la variabile di ambiente di sistema ' LANG ', ad esempio:

```
jim@brains:~/Documents/weather/pywws/code$ echo $LANG
en_GB.UTF-8
jim@brains:~/Documents/weather/pywws/code$ python -m pywws.Localisation -t en_GB.UTF-8
Locale changed from (None, None) to ('en_GB', 'UTF8')
Translation set OK
Locale
  decimal point: 23.2
  date & time: Friday, 14 October (14/10/11 13:02:00)
Translations
  'NNW' => 'NNW'
  'rising very rapidly' => 'rising very rapidly'
  'Rain at times, very unsettled' => 'Rain at times, very unsettled'
jim@brains:~/Documents/weather/pywws/code$
```

Nella maggior parte dei casi è richiesto un codice di non più di due lettere:

```
jim@brains:~/Documents/weather/pywws/code$ python -m pywws.Localisation -t fr
Locale changed from (None, None) to ('fr_FR', 'UTF8')
Translation set OK
Locale
  decimal point: 23,2
  date & time: vendredi, 14 octobre (14/10/2011 13:04:44)
Translations
  'NNW' => 'NNO'
  'rising very rapidly' => 'en hausse très rapide'
  'Rain at times, very unsettled' => 'Quelques précipitations, très perturbé'
jim@brains:~/Documents/weather/pywws/code$
```

Se impostate una lingua non supportata, pywws usa per default l'Inglese:

```
jim@brains:~/Documents/weather/pywws/code$ python -m pywws.Localisation -t ja
Failed to set locale: ja
No translation file found for: ja
Locale
  decimal point: 23.2
  date & time: Friday, 14 October (10/14/11 13:08:49)
Translations
  'NNW' => 'NNW'
  'rising very rapidly' => 'rising very rapidly'
  'Rain at times, very unsettled' => 'Rain at times, very unsettled'
jim@brains:~/Documents/weather/pywws/code$
```

Dopo aver trovato un codice di lingua adatto che funziona, è possibile configurare pywws per usarlo modificando il file `weather.ini`:

```
[config]
language = fr
```

Creazione di una nuova traduzione

Se non c'è nessun file di traduzione per la lingua preferita, allora avete bisogno di crearne uno. Vedi *Come utilizzare pywws in un altra lingua* per istruzioni dettagliate.

Funzioni

SetApplicationLanguage(params)	Impostare le impostazioni internazionali e la traduzione di un programma pywws.
SetLocale(lang)	Impostare le impostazioni locali utilizzate dal programma.
SetTranslation(lang)	Impostare la traduzione utilizzata da (alcuni) moduli pywws.
main([argv])	

pywws.Localisation.**SetLocale**(*lang*)

Impostare le impostazioni locali utilizzate dal programma.

Questo riguarda l'intera applicazione, cambiando il modo in cui le date, valute e numeri sono rappresentati. Essa non deve essere chiamata da una routine di libreria che può essere utilizzata in un altro programma.

Il parametro `lang` può essere qualsiasi stringa che è riconosciuto da `locale.setlocale()`, per esempio `it`, `it_IT` or `it_IT.UTF-8`.

Parametri **lang** (*string*) – codice della lingua.

Ritorna eseguito con successo.

Tipo di ritorno bool

pywws.Localisation.**SetTranslation**(*lang*)

Impostare la traduzione utilizzata da (alcuni) moduli pywws.

Questo imposta l'oggetto di traduzione `Localisation.translation` per utilizzare una particolare lingua.

Il parametro `lang` può essere qualsiasi stringa nel formato `it`, `it_IT` o `it_IT.UTF-8`. Tutto ciò dopo un carattere `.` viene ignorato. Nel caso di una stringa come `it_IT`, la routine cercherà un file di lingua `it_IT` prima di cercare un file `it`.

Parametri **lang** (*string*) – codice della lingua.

Ritorna eseguito con successo.

Tipo di ritorno bool

pywws.Localisation.**SetApplicationLanguage**(*params*)

Impostare le impostazioni internazionali e la traduzione di un programma pywws.

Questa funzione legge la lingua dal file di configurazione, quindi chiama *SetLocale()* e *SetTranslation()*.

Parametri **params** (*object*) – a *pywws.DataStore.params* object.

pywws.Localisation.**main**(*argv=None*)

pywws.conversions

conversions.py - è un insieme di funzioni per convertire l'unità nativa di pywws (gradi centigradi, mm, m/s, hPa) ad altre unità popolari

Funzioni

apparent_temp(temp, rh, wind)	Calcola la temperatura apparente (sensazione reale), con formula da
cadhumidex(temp, humidity)	Calcolo Indice di umidità come per gli standard Canadian Weather Standards
dew_point(temp, hum)	Calcola il punto di rugiada, usando la formula da http://en.wikipedia.org/wiki/Dew_point.
illuminance_wm2(lux)	Conversione approssimativa di illuminazione in lux a radiazione solare in W/m2
pressure_inhg(hPa)	Converte la pressione da ettopascal/millibar in pollici di mercurio
pressure_trend_text(trend)	Converte in una stringa, la tendenza barometrica sono usati per la UK met office.
rain_inch(mm)	Converte le precipitazioni da millimetri a pollici
temp_f(c)	Converte la temperatura da gradi Celsius a Fahrenheit
usaheatindex(temp, humidity, dew)	Calcolare l'indice di calore secondo Standards USA National Weather Service
wind_bft(ms)	Converte i metri al secondo del vento in scala Beaufort
wind_chill(temp, wind)	Calcola il vento gelido, usando la formula di
wind_kmph(ms)	Converte il vento da metri al secondo a chilometri orari
wind_kn(ms)	Converte il vento da metri al secondo a nodi
wind_mph(ms)	Converte il vento da metri al secondo a miglia all'ora
winddir_degrees(pts)	Convertire la direzione del vento a 0..15 in gradi
winddir_text(pts)	Convertire direzione del vento da 0 .. 15 per i punti della bussola

pywws.conversions.**illuminance_wm2**(*lux*)

Conversione approssimativa di illuminazione in lux a radiazione solare in W/m2

pywws.conversions.**pressure_inhg**(*hPa*)
Converte la pressione da ettopascal/millibar in pollici di mercurio

pywws.conversions.**pressure_trend_text**(*trend*)
Converte in una stringa, la tendenza barometrica sono usati per la UK met office.

pywws.conversions.**rain_inch**(*mm*)
Converte le precipitazioni da millimetri a pollici

pywws.conversions.**temp_f**(*c*)
Converte la temperatura da gradi Celsius a Fahrenheit

pywws.conversions.**winddir_degrees**(*pts*)
Convertire la direzione del vento a 0..15 in gradi

pywws.conversions.**winddir_text**(*pts*)
Convertire direzione del vento da 0 .. 15 per i punti della bussola

pywws.conversions.**wind_kmph**(*ms*)
Converte il vento da metri al secondo a chilometri orari

pywws.conversions.**wind_mph**(*ms*)
Converte il vento da metri al secondo a miglia all'ora

pywws.conversions.**wind_kn**(*ms*)
Converte il vento da metri al secondo a nodi

pywws.conversions.**wind_bft**(*ms*)
Converte i metri al secondo del vento in scala Beaufort

pywws.conversions.**dew_point**(*temp*, *hum*)
Calcola il punto di rugiada, usando la formula da http://en.wikipedia.org/wiki/Dew_point.

pywws.conversions.**cadhumidex**(*temp*, *humidity*)
Calcolo Indice di umidità come per gli standard Canadian Weather Standards

pywws.conversions.**usaheatindex**(*temp*, *humidity*, *dew*)
Calcolare l'indice di calore secondo Standards USA National Weather Service

See http://en.wikipedia.org/wiki/Heat_index, formula 1. The formula is not valid for T < 26.7C, Dew Point < 12C, or RH < 40%

pywws.conversions.**wind_chill**(*temp*, *wind*)
Calcola il vento gelido, usando la formula da http://en.wikipedia.org/wiki/wind_chill

pywws.conversions.**apparent_temp**(*temp*, *rh*, *wind*)
Calcola la temperatura apparente (temperatura percepita), con formula da http://www.bom.gov.au/info/thermal_stress/

pywws.Logger

Codice comune per la registrazione di informazioni ed errori.

Funzioni

ApplicationLogger(verbose[, logfile])	

pywws.Logger.**ApplicationLogger**(*verbose*, *logfile=None*)

Indici e tabelle

- genindex
- modindex
- search

CAPITOLO 5

Ringraziamenti

Non sarei stato in grado di ottenere tutte le informazioni dalla stazione meteo senza avere accesso ai sorgenti di Michael Pendec's programma "wwsr". Sono anche grata alla Dave Wells per la decodifica del weather station's "fixed block" data.

Infine, un grande ringraziamento a tutti gli utenti pywws che hanno aiutato con domande e suggerimenti e soprattutto a coloro che hanno tradotto pywws e la relativa documentazione in altre lingue.

CAPITOLO 6

Licenze

pywws - Python software per Stazione Meteo USB senza filo.
http://github.com/jim-easterbrook/pywws
Copyright (C) 2008-13 Jim Easterbrook jim@jim-easterbrook.me.uk

Questo programma è software libero; può essere redistribuito e/o modificarlo secondo i termini della GNU General Public License come pubblicata dalla Free Software Foundation; versione 2 della licenza, o (a tua scelta) qualsiasi versione successiva.

Questo programma è distribuito nella speranza che sia utile, ma senza alcuna garanzia; senza neppure la garanzia implicita di commerciabilità o idoneità per uno scopo particolare. Vedi la GNU General Public License per maggiori dettagli.

Dovresti aver ricevuto una copia del GNU General Public License insieme a questo programma; in caso contrario, scrivete a Free Software Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA

Indice del modulo Python

p

Indice

H

I

K

L

M

N

P

R

S

T

U

W

Y

Z