RegExpGen Documentation

Release 0.0.1

Bartosz Alchimowicz et al.

December 16, 2015

Contents

RegExpGen allows to get a screen shot of a whole webpage and a part of it.

Contents:

CHAPTER 1

Installation

The following libraries are required:

- python 2.7
- python-setuptools

CHAPTER 2

API

Set of functions to generate regular expressions from a pattern similar to printf function.

regexpgen.**auto**(*frmt*, *minV=None*, *maxV=None*)

Generating regular expressions for integer, real, date and time.

Parameters

- **format** – format similar to C printf function (description below)
- **min** – optional minimum value
- **max** – optional maximum value

Returns

regular expression for a given format

Supported formats: see *regexpgen.integer*, *regexpgen.real*, *regexpgen.date*, *regexpgen.time*

Additional information: Because single %d occurs as well in integer format and in date format, the integer function is preferred. To generate single %d for date please use regexpgen.date

Examples of use:

```
>>> import regexpgen
```

```
>>> regexpgen.auto("%Y-%m-%d", "2013-03-15", "2013-04-24")
'^(2013\-03\-(1[5-9]|2[0-9]|3[0-1])|2013\-03\-(0[1-9]|1[0-9]|2[0-9]|3[0-1])|2013\-04\-(0[1-9]|1[
```

```
>>> regexpgen.auto("%0d", -10, 10)
'^(-?([0-9]|10))$'
```

regexpgen.**concatenate**(*concatenationList*)

Concatenating regular expressions of integer, real, time and date.

Parameters **concatenationList** – list of tuples - if tuple has only one element is placed directly into regexp, otherwise appropriate function is called (first parameter of tuple should be string - real, int, date or time)

Supported formats consists syntaxtes from integer, real, date and time formats

Additional information: It is assumed that user knows if the single element should be escaped or not.

Examples of use:

```
>>> import regexpgen
```

```
>>> regexpgen.concatenate([('int', "%d", 100, 105), ('\.',), ('int', "%d", 250, 255)])
'^((0*(10[0-5]))\.(0*(25[0-5])))$'
```

```
>>> regexpgen.concatenate([('date', "%m"), (' ',), ('time', "%M")])
'^(((0[1-9]|1[0-2])) ([0-5][0-9]))$'
```

regexpgen.**date**(*frmt*, *minV=None*, *maxV=None*)

Generating regular expressions for date.

Parameters

- **format** – format similar to C printf function (description below)
- **min** – optional minimum value
- **max** – optional maximum value

Returns regular expression for a given format

Supported formats consists following syntaxes: %d day (01..31) %m month (01..12) %y two last digits of year (00..99) %Y year (four digits)

Additional information: :Leap years are supported :Supported years: 1970 - 2099 :%Y or %y can not be only syntaxes :%d can not come with %Y without %m :%Y and %y can not come together

Examples of use:

```
>>> import regexpgen
```

```
>>> regexpgen.date("%Y-%m-%d", "2013-03-15", "2013-04-24")
'^(2013\-03\-(1[5-9]|2[0-9]|3[0-1])|2013\-03\-(0[1-9]|1[0-9]|2[0-9]|3[0-1])|2013\-04\-(0[1-9]|1[
```

```
>>> regexpgen.date("%d/%m", "15/09")
'^((1[5-9]|2[0-9]|30)\/09|(0[1-9]|1[0-9]|2[0-9]|3[0-1])\/10|(0[1-9]|1[0-9]|2[0-9]|30)\/11|(0[1-9
```

regexpgen.**integer**(*frmt*, *minV=None*, *maxV=None*)

Generating regular expression for integer numbers (-2, -1, 0, 1, 2, 3...).

Parameters

- **frmt** – format similar to C printf function (description below)
- **minV** – optional minimum value
- **maxV** – optional maximum value

Returns regular expression for a given format

Supported formats:

FORMAT = '%d' description: leading zeros are optional, correct examples: -1, -005, 0, 1, 001, 012 incorrect examples: N/A

FORMAT = '%0d' description: leading zeros are forbidden correct examples: -2, -2123, 0, 1, 255 incorrect examples: 001, 012

FORMAT = '%0Xd' description: number written with X characters, in case of number lesser than int('9'*X) it should be leaded with zeros, correct examples for %04d: 0001, 5678 incorrect examples for %04d: 00011, 111

Examples of use:

```
>>> import regexpgen
```

```
>>> regexpgen.integer("%0d", -10, 10)
'^(-?([0-9]|10))$'
```

```
>>> regexpgen.integer("%04d", -10, 10)
'^(-?(000[0-9]|0010))$'
```

regexpgen.**nnint**(*frmt*, *minV=None*, *maxV=None*)

Generating regular expression for a non negative integer numbers.

Parameters

- **frmt** – format similar to C printf function (description below)
- **minV** – optional minimum value
- **maxV** – optional maximum value

Returns regular expression for a given format

Generating regular expressions for non-negative integers (0, 1, 2, 3...).

Supported formats: see *regexpgen.integer*

Examples of use:

```
>>> import regexpgen
```

```
>>> regexpgen.nnint("%0d")
'^([1-9][0-9]*|0)$'
```

```
>>> regexpgen.nnint("%04d", 71, 85)
'^(007[1-9]|008[0-5])$'
```

regexpgen.**real**(*frmt*, *minV=None*, *maxV=None*)

Generating regular expressions for real numbers with accuracy of float() function.

Parameters

- **frmt** – format similar to C printf function (description below)
- **minV** – optional minimum value
- **maxV** – optional maximum value

Returns regular expression for a given format

Supported formats:

FORMAT = '%lf' description: leading zeros are optional correct examples: 0.1, 1.32, 001.21, 012.123

FORMAT = '%0lf' description: leading zeros are forbidden correct examples: 22.1, 1.1 incorrect examples: 001.2, 012.9

FORMAT = '%0.Ylf' description: leading zeros are forbidden, after the comma exactly Y characters are expected correct examples for %0.1lf: 22.1, 1.1 incorrect examples for %0.1lf: 001.2, 012.9

FORMAT = '%.Ylf' description: leading zeros are optional, after the comma exactly Y characters are expected correct examples for %0.1lf: 022.1, 1.1, 001.2, 012.9 incorrect examples for %0.1lf: 3.222, 0.22

FORMAT = '%X.Ylf' description: X is ignored (works like '%.Ylf')

FORMAT = '%0X.Ylf' description: leading zeros are required, number written with at least X characters (including dot),

after the comma exactly Y characters are expected, if number of characters is lesser than X, it should be leaded with zeros

correct examples for %5.1lf: 002.1, 32431.2, 022.9 incorrect examples for %5.2lf: 22.111, 1.1

Examples of use:

```
>>> import regexpgen
```

```
>>> regexpgen.real("%lf", -1.5, 2.5)
'^(-?(0*((0)\.0|(0)(\.(([0-9])[0-9]*))|(1)(\.(([0-3]|4)[0-9]*))|(1)\.50*))|0*((1)\.5|(1)(\.(5|([
```

```
>>> regexpgen.real("%0.1lf", -1.0, 2.0)
'^(-?(((0)\.(([0-9]))|(1)\.0))|((1)\.(([0-9]))|(2)\.0))$'
```

`regexpgen.`**`time`**(*frmt*, *minV=None*, *maxV=None*)

Generating regular expressions for time.

Parameters

- **`format`** – format similar to C printf function (description below)
- **`min`** – optional minimum value
- **`max`** – optional maximum value

Returns regular expression for a given format

Supported formats consists following syntaxes:

%H hours (00..23) %I hours (00..12) %M minutes (00..59) %S seconds (00..59) %p AM or PM %P am or pm

Additional information:

:It is possible to add a time zone to the regexp - it should be added directly to format, eg. "%H:%M:%S +01" :%I must come with %p or %P :%H can not come with %I, %p or %P :%P and %p can not come together :%H can not come with %S without %M :%P and %p can not be only syntaxes

Examples of use:

```
>>> import regexpgen
```

```
>>> regexpgen.time("%H:%M", "12:24", "17:59")
'^(12\:(2[4-9]|[3-4][0-9]|5[0-9])|(1[3-5]|16)\:[0-5][0-9]|17\:[0-5][0-9])$'
```

```
>>> regexpgen.time("%I-%M-%S %P", "10-00-00 PM", None)
'^(10\-00\-[0-5][0-9]\ PM|10\-(0[1-9]|1[0-9]|[2-4][0-9]|5[0-9])\-[0-5][0-9]\ PM|11\-(0[0-9]|1[0-
```

CHAPTER 3

TODO

- mdatetime

r

A

C

D

I

N

R

T