python-form Documentation

Release 0.2.3

Takahiro Ueda

Aug 28, 2019

Contents:

CHAPTER 1

API Documentation

1.1 form module

Provide routines for communicating with FORM.

Example

```
>>> import form
>>> with form.open() as f:
...     f.write('''
...         AutoDeclare Vector p;
...         Local F = g_(0,p1,...,p4);
...         trace4,0;
...         .sort
...     ''')
...     print(f.read('F'))
4*p1.p2*p3.p4-4*p1.p3*p2.p4+4*p1.p4*p2.p3
```

`form.`**`open`**(*args=None*, *keep_log=False*)

Open a connection to FORM and return a link object.

Open a connection to a new FORM process and return a *link object*. The opened connection should be closed by *close()* of the returned object, which is automatically done by use of the "with" statement:

```
>>> import form
>>> with form.open() as formlink:
...     pass  # use formlink ...
```

The optional argument `args` is for the FORM command, a string or a sequence of strings. For example '/path/to/form' or ['tform', '-w4']. By default, the value of the environment variable `$FORM` is used if set, otherwise 'form' will be used.

The other argument `keep_log` indicates whether the log from FORM is kept and used as detailed information when an error occurs. If the value is >= 2, it specifies the maximum number of lines for the scrollback. The default value is False.

Note: In the current implementation, `keep_log=True` may cause a dead lock when the listing of the input is enabled and very long input is sent to FORM.

---

class form.**FormLink**(*args=None*, *keep_log=False*)
Connection to a FORM process.

close()
Close the connection to FORM.

Close the connection to the FORM process established by *open()*. Do nothing if the connection is already closed. The user should call this method after use of a link object, which is usually guaranteed by use of the "with" statement.

closed
Return True if the connection is closed.

flush()
Flush the channel to FORM.

Flush the communication channel to FORM. Because *write()* is buffered and *read()* is a blocking operation, this method is used for asynchronous execution of FORM scripts.

head
Return the first line of the FORM output.

kill()
Kill the FORM process and close the connection.

open(*args=None*, *keep_log=False*)
Open a connection to FORM.

Open a connection to a new FORM process. The opened connection should be closed by *close()*, which can be guaranteed by use of the "with" statement:

```
>>> import form
>>> with form.open() as formlink:
...     pass  # use formlink ...
```

If this method is called for a link object that has an established connection to a FORM process, then the existing connection will be closed and a new connection will be created.

The optional argument `args` is for the FORM command, a string or a sequence of strings. For example '/path/to/form' or ['tform', '-w4']. By default, the value of the environment variable `$FORM` is used if set, otherwise 'form' will be used.

The other argument `keep_log` indicates whether the log from FORM is kept and used as detailed information when an error occurs. If the value is >= 2, it specifies the maximum number of lines for the scrollback. The default value is False.

Note: In the current implementation, `keep_log=True` may cause a dead lock when the listing of the input is enabled and very long input is sent to FORM.

---

read(**names*)
Read results from FORM.

Wait for a response of FORM to obtain the results specified by the given names and return a corresponding string or (nested) list of strings. Objects to be read from FORM are expressions, $-variables and preprocessor variables.

name	meaning
"F"	expression F
"$x"	$-variable $x
"$x[]"	factorized $-variable $x
"'A'"	preprocessor variable A

Note that the communication for the reading is performed within the preprocessor of FORM (i.e., at compile-time), so one may need to write ".sort" to get the correct result.

If non-string objects are passed, they are considered as sequences, and the return value becomes a list corresponding to the arguments. If a sequence is passed as the argument to this method, it is guaranteed that the return value is always a list:

```
>>> import form
>>> f = form.open()
>>> f.write('''
...     S a1,...,a3;
...     L F1 = a1;
...     L F2 = a2;
...     L F3 = a3;
...     .sort
... ''')
```

```
>>> f.read(['F1'])
['a1']
>>> f.read(['F1', 'F2'])
['a1', 'a2']
>>> f.read(['F1', 'F2', 'F3'])
['a1', 'a2', 'a3']
```

A more complicated example, which returns a nested list, is

```
>>> f.read('F1', ['F2', 'F3'])
['a1', ['a2', 'a3']]
```

```
>>> f.close()
```

Note: Currently nested inputs are not supported for static typing. You need `# type: ignore` or other hacks.

write(*script*)

Send a script to FORM.

Write the given script to the communication channel to FORM. It could be buffered and so FORM may not execute the sent script until *flush()* or *read()* is called.

exception form.**FormError**

Bases: exceptions.RuntimeError

FORM stopped by an error.

This exception is raised when *read()* finds the FORM process stopped by some error.

CHAPTER 2

Indices and tables

- genindex
- modindex
- search

Python Module Index

f

Index