Package ‘rPython’

August 3, 2014

Version 0.0-5

Date 2014-08-02

Title Package allowing R to call Python

Author Carlos J. Gil Bellosta

Maintainer Carlos J. Gil Bellosta <cgb@datanalytics.com>

Description This package permits calls to Python from R

Depends RJSONIO (>= 0.7-3)

License GPL-2

SystemRequirements Python (>= 2.7) and Python headers and libraries (See the INSTALL file)

OS_type unix

URL http://rpython.r-forge.r-project.org/

NeedsCompilation yes

Repository CRAN

Date/Publication 2014-08-03 08:04:47

R topics documented:

python.assign 2
python.call 3
python.exec 4
python.load 5

Index **7**

`python.assign` *Assign and get variables in Python from R*

Description

Functions that assign and get Python variables from R.

Usage

```
python.assign( var.name, value )
python.get( var.name )
```

Arguments

`var.name`	a character string containing a valid python variable name
`value`	an R object whose equivalent wants to be assigned to the variable in python

Details

These functions can assign values to variables in Python as well as get their values back to R. Objects are serialized as json strings while being transferred between R and Python.

Value

Function `python.get` returns a R version of the Python variable `py.var`.

References

http://code.google.com/p/simplejson

Examples

```
a <- 1:4
python.assign( "a", a )
python.exec( "b = len( a )" )
python.get( "b" )

python.exec( "import math" )
python.get( "math.pi" )
```

`python.call` *python.call*

Description

Calls Python functions and methods from R

Usage

```
python.call( py.foo, ..., simplify = TRUE, as.is = FALSE )
python.method.call( py.object, py.method, ... )
```

Arguments

`py.foo`	rame of a Python function
`py.object`	name of a Python object
`py.method`	name of a method of such object
`...`	R objects to pass as arguments to the Python function or method
`simplify`	logical value indicating whether simplification of output should be simplified
`as.is`	logical value indicating whether length 1 vectors in R should be passed as atomic variables in Python as opposed to length 1 vectors. Note that, e.g., strings such as "hello" in R are vectors of length 1 in R, i.e., `"hello"` is the same as `c("hello")`. But Python functions operating on arrays will want to receive the array `["hello"]` rather than the literal string `"hello"`. This argument provides little granularity: it affects either all or none of the arguments of the function. Finer control can be obtained using the `I()` function as shown in the examples section below.

Details

This function runs a Python function taking as arguments R objects and returning an R object. Some limitations exist as to the nature of the objects that can be passed between R and Python. As of this writing, atomic arguments and vectors are supported.

The user has to be careful to indicate named parameters as required according to Python conventions.

Value

An R representation of the object returned by the call to the Python function.

Examples

```
python.call( "len", 1:3 )
a <- 1:4
b <- 5:8
python.exec( "def concat(a,b): return a+b" )
python.call( "concat", a, b)

python.assign( "a",  "hola hola" )
python.method.call( "a", "split", " " )

## simplification of arguments
a <- 1
b <- 5:8

## Not run:
python.call("concat", a, b)
## End(Not run)

# using function I()
python.call("concat", I(a), b)

# setting as.is = TRUE
python.call("concat", a, b, as.is = TRUE)
```

`python.exec` *python.exec*

Description

Executes Python code contained in an R character vector.

Usage

```
python.exec( python.code, get.exception = TRUE )
```

Arguments

`python.code`	a character vector containing Python code, typically a single line with indentation and EOL characters as required by Python syntax
`get.exception`	logical value indicating whether to check or not for exceptions in Python

Details

This function runs Python code. It needs to be provided by the caller in a character vector.

The vector may consists of a single string with EOL and indentation characters embedded.

Alternatively, it can be a character vector, each entry containing one or more lines of Python code.

The `get.exception` option allows the user to disregard Python exceptions in cases where safe calls to avoid the overhead of checking for them.

Value

None. If the code produces some output, it is up to the caller to go and fetch if from Python.

Examples

```
a <- 1:4
b <- 5:8
python.exec( c( "def concat(a,b):", "\treturn a+b" ) )
python.call( "concat", a, b)
```

`python.load` *python.load*

Description

Executes Python code.

Usage

```
python.load( file, get.exception = TRUE )
```

Arguments

`file`	a file containing python code to be executed
`get.exception`	logical value indicating whether to check or not for exceptions in Python

Details

This function runs Python code contained in a file. Typically, this file would contain functions to be called via `python.call` or other functions in this package.

The `get.exception` option allows the user to disregard Python exceptions in cases where safe calls to avoid the overhead of checking for them.

Value

None. If the code produces some output, it is up to the caller to go and fetch if from Python using function `python.get`.

Examples

```
a <- 1:4
b <- 5:8

# this file contains the definition of function concat
python.load( system.file( "concat.py", package = "rPython" ) )
python.call( "concat", a, b)
```

Index

∗Topic **manip**
  python.assign, 2
  python.call, 3
  python.exec, 4
  python.load, 5

python.assign, 2
python.call, 3, *5*
python.exec, 4
python.get, *5*
python.get *(*python.assign*)*, 2
python.load, 5
python.method.call *(*python.call*)*, 3