Package 'SnakeCharmR'

October 7, 2017

Version 1.0.7

Date 2017-10-05

Title R and Python Integration

Author Alexandre Sieira, forked off of rPython by Carlos J. Gil Bellosta

Maintainer Alexandre Sieira <alexandre.sieira@gmail.com>

Description Run 'Python' code, make function calls, assign and retrieve variables, etc. from R. A fork from 'rPython' which uses 'jsonlite', 'Rcpp' and has several fixes and improvements.

Imports jsonlite (>= 1.3), stringr (>= 1.2.0), Rcpp (>= 0.12.0), utils

License GPL-2

SystemRequirements Python (>= 2.7) and Python headers and libraries (See the README.md file)

OS_type unix

URL https://github.com/asieira/SnakeCharmR

BugReports https://github.com/asieira/SnakeCharmR/issues

Suggests testthat (>= 1.0.0)

LinkingTo Rcpp

RoxygenNote 6.0.1

NeedsCompilation yes

Repository CRAN

Date/Publication 2017-10-06 22:36:32 UTC

R topics documented:

`py.assign` *Assign values to Python variables from R*

Description

`py.assign` assigns values to variables in Python. Objects are serialized as JSON strings on R with `jsonlite::toJSON`, are transferred to Python and are converted back to a Python value using `json.loads`.

Usage

```
py.assign(var.name, value,
  json.opt.args = getOption("SnakeCharmR.json.opt.args", list(auto_unbox =
  TRUE, null = "null")))
```

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
`json.opt.args`	explicit arguments to pass to `jsonlite::toJSON` when serializing the value

Examples

```
py.assign("a", 1:4)
py.get("a")
# [1] 1 2 3 4

py.assign("b", list(one = 1, foo = "bar"))
str(py.get("b"))
# List of 2
#  $ foo: chr "bar"
#  $ one: int 1

py.exec("import math")
py.get("math.pi")
# [1] 3.141593

# by default jsonlite::toJSON is called on the value to be assigned
# with auto_unbox = TRUE, so vectors of length 1 are simplified to
# scalar values in Python
py.assign("c", "foo bar")
py.exec("crepr = repr(c)")
py.get("crepr")
# [1] "u'foo bar'"

# if we explicitly set auto_unbox to FALSE, a vector of length 1
# becomes a Python list of length 1
py.assign("d", "foo bar", json.opt.args = list(auto_unbox = FALSE))
```

```
py.exec("drepr = repr(d)")
py.get("drepr")
# [1] "[u'foo bar']"
```

`py.call` *Calls Python functions and methods from R*

Description

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, mainly which data types can be converted to/from JSON in R (using `jsonlite`) and Python (using the `json` module).

Usage

```
py.call(fname, ..., json.opt.args = getOption("SnakeCharmR.json.opt.args",
  list(auto_unbox = TRUE, null = "null")),
  json.opt.ret = getOption("SnakeCharmR.json.opt.ret", list()))
```

Arguments

`fname`	name of function/method to call
`...`	R objects to pass as arguments to the Python function or method
`json.opt.args`	explicit arguments to pass to `jsonlite::toJSON` when serializing the function argument values
`json.opt.ret`	explicit arguments to pass to `jsonlite::fromJSON` when deserializing the function return value

Value

an R object containing the function or method call return value after serialization to JSON in the Python environment and deserialization from JSON in the R environment

Examples

```
py.call("len", 1:3)
# [1] 3

py.call("repr", 1:3)
# [1] "[1, 2, 3]"

a <- 1:4
b <- 5:8
py.exec("def concat(a,b): return a+b")
py.call("concat", a, b)
# [1] 1 2 3 4 5 6 7 8
```

```
py.assign("a",  "hola hola")
py.call("a.split", " ")
# [1] "hola" "hola"

# by default jsonlite::toJSON is called on the function call arguments
# with auto_unbox = TRUE, so vectors of length 1 are simplified to
# scalar values in Python
py.call("repr", "foo bar")
# [1] "u'foo bar'"

# if we explicitly set auto_unbox to FALSE, a vector of length 1
# becomes a Python list of length 1
py.call("repr", "foo bar", json.opt.args = list(auto_unbox = FALSE))
# [1] "[u'foo bar']"
```

`py.exec` *Executes arbitrary Python code from R*

Description

This function runs Python code that is provided in a character vector.

Usage

```
py.exec(code, stopOnException = TRUE)
```

Arguments

`code`	a character vector containing Python code, typically a single line with indentation and EOL characters as required by Python syntax
`stopOnException`	if `TRUE` then `stop` will be called if a Python exception occurs, otherwise only a warning will be flagged

Details

The character vector containing the code to execute may consist 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.

Examples

```
a <- 1:4
b <- 5:8
py.exec(c("def concat(a,b):", "\treturn a+b"))
py.call("concat", a, b)
# [1] 1 2 3 4 5 6 7 8

## Not run:
```

```
py.exec("raise Exception('Stop the presses!')")
# Error in py.exec("raise Exception('Stop the presses!')") (from py.exec.R#48) :
#   Traceback (most recent call last):
#   File "<string>", line 2, in <module>
# Exception: Stop the presses!

## End(Not run)

py.exec("raise Exception('Houston, we have a problem!')", stopOnException = FALSE)
# Warning message:
# In py.exec("raise Exception('Houston, we have a problem!')", stopOnException = FALSE) :
#   Traceback (most recent call last):
#   File "<string>", line 2, in <module>
# Exception: Houston, we have a problem!
```

`py.get` *Get values from Python variables to R*

Description

`py.get` get the value of Python and returns it to the R environment. Objects are serialized as JSON strings on Python with `json.dumps`, are transferred to R and are converted back to an R value using `jsonlite::fromJSON`.

Usage

```
py.get(var.name, json.opt.ret = getOption("SnakeCharmR.json.opt.ret", list()))
```

Arguments

`var.name`	a character string containing a valid Python variable name
`json.opt.ret`	explicit arguments to pass to `jsonlite::fromJSON` when deserializing the value

Value

an R object containing the variable value after serialization to JSON in the Python environment and deserialization from JSON in the R environment

Examples

```
py.assign("a", 1:4)
py.get("a")
# [1] 1 2 3 4

py.assign("b", list(one = 1, foo = "bar"))
str(py.get("b"))
# List of 2
#  $ foo: chr "bar"
#  $ one: int 1
```

```
py.exec("import math")
py.get("math.pi")
# [1] 3.141593

## Not run:
py.rm("notset")
py.get("notset")
# Error in py.get("notset") (from py.get.R#60) : Traceback (most recent call last):
#   File "<string>", line 2, in <module>
# NameError: name 'notset' is not defined

## End(Not run)
```

`py.load` *Reads and executes Python code from a file*

Description

This function runs Python code contained in a file. This is basically a convenience function that is the equivalent of `py.exec(readLines(file))`.

Usage

```
py.load(file, stopOnException = TRUE)
```

Arguments

`file`	a connection or file name that will be passed to `readLines` to read the Python code in the file
`stopOnException`	logical value indicating whether to check or not to call `stop` if a Python exception occurs

Details

For better maintainability, it might be worth investigating concentrating more complex Python code that needs to be called from R into proper packages that can be installed using `pip` and loaded with `py.exec("import package_name")`.

Value

if `stopOnException` is `FALSE`, invisibly returns a string representation of any raised Python exceptions or NULL if none occur.

py.rm *Remove a Python variable from R*

Description

This function uses the `del` Python command to remove a variable and reclaim its memory. Any exceptions, such as the one that would happen if the variable did not exist, will be caught and ignored.

Usage

```
py.rm(var.name, stopOnException = FALSE)
```

Arguments

`var.name`	a character string containing a valid Python variable name
`stopOnException`	if `TRUE` then `stop` will be called if a Python exception occurs, typically because the variable doesn't exist, otherwise only a warning will be flagged

Examples

```
py.assign("a", "foo bar")
py.get("a")
# [1] "foo bar"
py.rm("a")
## Not run:
py.rm("a")
# Warning message:
# In py.rm("a") : Traceback (most recent call last):
#   File "<string>", line 2, in <module>
# NameError: name 'a' is not defined

## End(Not run)
```

SnakeCharmR *R and Python Integration*

Description

A fork from rPython which uses jsonlite and has several fixes and improvements

Author(s)

Alexandre Sieira

Index