Package 'findpython'

July 2, 2014

Type Package

Title Python tools to find an acceptable python binary

Version 1.0.1

Author Trevor L Davis and Paul Gilbert.

Maintainer Trevor L Davis <trevor.l.davis@stanford.edu>

URL https://github.com/trevorld/findpython

BugReports https://github.com/trevorld/findpython/issues

Description Package designed to find an acceptable python binary.

Suggests testthat

License MIT + file LICENSE

Collate 'find_python_cmd.r'

NeedsCompilation no

Repository CRAN

Date/Publication 2014-04-03 20:20:34

R topics documented:

`can_find_python_cmd` *Determins whether or not it can find a suitable python cmd*

Description

`can_find_python_cmd` runs `find_python_cmd` and returns whether it could find a suitable python cmd. If it was successful its output also saves the found command as an attribute.

Usage

```
can_find_python_cmd(minimum_version = NULL,
  maximum_version = NULL, required_modules = NULL,
  error_message = NULL, silent = FALSE)
```

Arguments

`silent`	Passed to `try`, whether any error messages from `find_python_cmd` should be suppressed
`minimum_version`	The minimum version of python it should be. Should be a string with major and minor number separated by a '.'. If left NULL won't impose such a restriction.
`maximum_version`	The maximum version of python it should be. Should be a string with major and minor number separated by a '.'. If left NULL won't impose such a restriction.
`required_modules`	Which modules should be required. Can use a single "|" to represent a single either-or requirement like "json|simplejson". If left NULL won't impose such a restriction.
`error_message`	What error message the user will see if couldn't find a sufficient python binary. If left NULL will print out a default message.

Value

`TRUE` or `FALSE` depending on whether `find_python_cmd` could find an appropriate python binary. If `TRUE` the path to an appropriate python binary is also set as an attribute.

See Also

`find_python_cmd`

Examples

```
did_find_cmd <- can_find_python_cmd()
     python_cmd <- attr(did_find_cmd, "python_cmd")
```

find_python_cmd *Find a suitable python cmd or give error if not possible*

Description

find_python_cmd finds a suitable python cmd or raises an error if not possible

Usage

```
find_python_cmd(minimum_version = NULL,
  maximum_version = NULL, required_modules = NULL,
  error_message = NULL)
```

Arguments

error_message	What error message the user will see if couldn't find a sufficient python binary. If left NULL will print out a default message.
minimum_version	The minimum version of python it should be. Should be a string with major and minor number separated by a '.'. If left NULL won't impose such a restriction.
maximum_version	The maximum version of python it should be. Should be a string with major and minor number separated by a '.'. If left NULL won't impose such a restriction.
required_modules	Which modules should be required. Can use a single "|" to represent a single either-or requirement like "json|simplejson". If left NULL won't impose such a restriction.

Value

The path to an appropriate python binary. If such a path wasn't found then it will throw an error.

See Also

can_find_python_cmd for a wrapper which doesn't throw an error

Examples

```
## Not run:
            find_python_cmd()
            find_python_cmd(minimum_version='2.6', maximum_version='2.7')
            find_python_cmd(required_modules = c('argparse', 'json | simplejson'))

## End(Not run)
```

`is_python_sufficient` *Tests whether the python command is sufficient*

Description

`is_python_sufficient` checks whether a given python binary has all the desired features (minimum and/or maximum version number and/or access to certain modules).

Usage

```
is_python_sufficient(path, minimum_version = NULL,
  maximum_version = NULL, required_modules = NULL)
```

Arguments

`path`	The path to a given python binary. If binary is on system path just the binary name will work.
`minimum_version`	The minimum version of python it should be. Should be a string with major and minor number separated by a '.'. If left NULL won't impose such a restriction.
`maximum_version`	The maximum version of python it should be. Should be a string with major and minor number separated by a '.'. If left NULL won't impose such a restriction.
`required_modules`	Which modules should be required. Can use a single "|" to represent a single either-or requirement like "json|simplejson". If left NULL won't impose such a restriction.

Value

`TRUE` or `FALSE` depending on whether the python binary met all requirements

Index