Package 'findpython'

May 8, 2026

Type Package

Title Functions to Find an Acceptable Python Binary

Version 1.0.9

URL https://github.com/trevorld/findpython,
https://trevorldavis.com/R/findpython/

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

Description Package designed to find an acceptable python binary.

Suggests reticulate, testthat

License MIT + file LICENSE

Collate 'find_python_cmd.r'

RoxygenNote 7.3.1

Encoding UTF-8

NeedsCompilation no

Author Trevor L. Davis [aut, cre] (ORCID:
<https://orcid.org/0000-0001-6341-4639>),
Paul Gilbert [aut]

Maintainer Trevor L. Davis <trevor.l.davis@gmail.com>

Repository CRAN

Date/Publication 2024-11-19 06:50:06 UTC

Contents

can_find_python_cmd *Determines 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

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.
silent	Passed to try, whether any error messages from find_python_cmd() should be suppressed

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

`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

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

```
try(find_python_cmd())
try(find_python_cmd(minimum_version = "2.6", maximum_version = "2.7"))
try(find_python_cmd(required_modules = c("argparse", "json | simplejson")))
```

`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

Examples

```
try({
  cmd <- find_python_cmd()
  is_python_sufficient(cmd, minimum_version = "3.3", required_modules = "sys")
})
```

Index