Package: rpyodide (via r-universe)

October 29, 2026

Title Call Python (Pyodide) from R in webR, Like 'reticulate'

Version 0.1.0

Description A 'reticulate'-style interface for calling Python from R when R runs in the web browser through 'webR' (R compiled to WebAssembly). Python runs in 'Pyodide' (CPython compiled to WebAssembly) in its own web worker, and R talks to it synchronously through 'webR' worker messaging. Supports running code, importing modules, calling functions, converting vectors, lists, matrices and data frames to and from 'NumPy' and 'pandas', installing packages with 'micropip', and several independent Python sessions.

License MIT + file LICENSE

Encoding UTF-8

Roxygen list(markdown = TRUE)

Imports jsonlite, utils

Suggests spelling, processx, testthat (>= 3.0.0)

Config/testthat/edition 3

Config/roxygen2/version 8.1.1

Language en-US

URL https://github.com/muschellij2/rpyodide

BugReports https://github.com/muschellij2/rpyodide/issues

Repository https://muschellij2.r-universe.dev

Date/Publication 2026-10-29 19:25:21 UTC

RemoteUrl https://github.com/muschellij2/rpyodide

RemoteRef HEAD

RemoteSha 3038e71b89639f943b9609a02ceb58bc3bb2228d

Contents

cpython_start . . . 2
default_pyodide_url . . . 3
import . . . 4
is_py_object . . . 4
is_webr . . . 5
py . . . 5
py_available . . . 6
py_config . . . 6
py_install . . . 7
py_last_error . . . 7
py_list_packages . . . 8
py_module_available . . . 8
py_object . . . 9
py_require . . . 10
py_run_string . . . 11
py_upload . . . 12
pyodide_sessions . . . 12
pyodide_start . . . 13
pyodide_stop . . . 13
r_to_py . . . 14
tuple . . . 15
use_pyodide . . . 15

Index **16**

`cpython_start` *Start a session backed by a local CPython*

Description

Runs rpyodide's Python bridge in a regular Python process instead of Pyodide, talking to it over stdin and stdout. Everything else works the same way, so code written for rpyodide (and rpyodide itself) can be tested outside webR, for example on CRAN or in continuous integration. This needs the processx package and Python 3.8 or later.

Usage

```
cpython_start(name = "cpython", python = find_python(), activate = TRUE)

find_python()
```

Arguments

`name`	Session name.
`python`	Path to the Python executable. By default, the `RPYODIDE_PYTHON` environment variable if it is set, otherwise the first of `python3` and `python` found on the `PATH`.
`activate`	Make this the active session?

Details

Differences from Pyodide sessions: packages come from the Python installation (use pip; `py_install()` and `py_require()` can't install them), files aren't copied because Python shares R's file system, and the Python version is the local one.

Value

The session, invisibly.

Functions

- `find_python()`: The Python executable `cpython_start()` uses by default, or `""` if there is none.

See Also

`pyodide_start()` for Python in webR.

Examples

```
if (nzchar(find_python()) && requireNamespace("processx", quietly = TRUE)) {
  cpython_start("example")
  py_eval("1 + 1", session = "example")
  pyodide_stop("example")
}
```

`default_pyodide_url` *Default Pyodide distribution URL*

Description

Pyodide is loaded from this URL unless `pyodide_url` is given to `pyodide_start()`. Set `options(rpyodide.pyodide_url = ...)` to change it. The URL must end in `/` and contain `pyodide.mjs`.

Usage

```
default_pyodide_url()
```

Value

A character string.

import *Import a Python module*

Description

Imports a module, loading it from the Pyodide distribution first if necessary. Use `$` to get its attributes.

Usage

```
import(module, convert = TRUE, session = NULL)

import_main(convert = TRUE, session = NULL)

import_builtins(convert = TRUE, session = NULL)
```

Arguments

`module`	Module name, e.g. `"numpy"` or `"os.path"`.
`convert`	Convert values returned from this module to R?
`session`	Pyodide session; defaults to the active one.

Value

A Python module object.

Examples

```
## Not run:
np <- import("numpy")
np$linspace(0, 1, 5L)

## End(Not run)
```

is_py_object *Is an object a Python object?*

Description

Is an object a Python object?

Usage

```
is_py_object(x)
```

Arguments

x	An R object.

Value

TRUE if x refers to an object in a Pyodide session.

---

is_webr *Is R running in webR?*

Description

Is R running in webR?

Usage

```
is_webr()
```

Value

TRUE when R is webR (R compiled to WebAssembly).

---

py *The Python __main__ module*

Description

py$name gets a variable from Python's __main__ module in the active session, and py$name <- value sets one.

Usage

```
py
```

Format

An object of class pyodide_main.

py_available *Is Python available?*

Description

Is Python available?

Usage

```
py_available(initialize = FALSE)
```

Arguments

initialize	Start the default session if none is running?

Value

TRUE if a Pyodide session is running (or was started).

py_config *Python configuration*

Description

Python configuration

Usage

```
py_config(session = NULL)
```

Arguments

session	Pyodide session; defaults to the active one.

Value

A list with the Python and Pyodide versions and loaded packages.

py_install *Install Python packages*

Description

Installs packages into a Pyodide session with `micropip`. Packages built for Pyodide (such as numpy, pandas and scikit-learn) come from the Pyodide distribution; others come from PyPI and must be pure Python or have WebAssembly wheels.

Usage

```
py_install(packages, deps = TRUE, session = NULL)
```

Arguments

`packages`	Package names, optionally with version specifiers.
`deps`	Also install the packages' dependencies? Use `FALSE` for packages whose pinned dependencies can't be met in Pyodide, after installing compatible versions of those dependencies yourself.
`session`	Pyodide session; defaults to the active one.

Value

`NULL`, invisibly.

py_last_error *The last Python error*

Description

The last Python error

Usage

```
py_last_error()
```

Value

A list with the Python exception `type`, `message` and `traceback`, or `NULL` if there hasn't been an error.

py_list_packages *List installed Python packages*

Description

List installed Python packages

Usage

```
py_list_packages(session = NULL)
```

Arguments

session	Pyodide session; defaults to the active one.

Value

A data frame of package names, versions and sources.

py_module_available *Is a Python module available?*

Description

Is a Python module available?

Usage

```
py_module_available(module, session = NULL)
```

Arguments

module	Module name.
session	Pyodide session; defaults to the active one.

Value

TRUE if the module can be imported, either because it is installed or because it is in the Pyodide distribution.

py_object *Work with Python objects*

Description

Low-level accessors for Python objects. Usually you use `$`, `[[` and `[` instead: `x$attr`, `x[["attr"]]` and `x[key]` (Python `x[key]`, 0-based).

Usage

```
py_get_attr(x, name)

py_set_attr(x, name, value)

py_has_attr(x, name)

py_list_attributes(x)

py_get_item(x, key)

py_set_item(x, key, value)

py_call(x, ...)

py_len(x)

py_str(x)

py_repr(x)
```

Arguments

`x`	A Python object.
`name`	Attribute name.
`value`	An R value or Python object.
`key`	An item key or index.
`...`	Arguments for the call; named arguments become keyword arguments.

`py_require` *Declare Python requirements*

Description

Like `reticulate::py_require()`: declare the Python packages (and Python version) that your code needs, before or after Python has started. Requirements are kept per session and accumulate across calls.

Usage

```
py_require(
  packages = NULL,
  python_version = NULL,
  deps = TRUE,
  action = c("add", "remove", "set"),
  session = NULL
)
```

Arguments

`packages`	Package requirements, such as `"numpy"` or `"pandas>=2"`. Packages in the Pyodide distribution are used when available; others come from PyPI and must be pure Python or have WebAssembly wheels. Requiring a package that is already listed replaces its version specifier.
`python_version`	Python version constraints, such as `">=3.12"`, `"3.14"` or `">=3.12,<3.15"`.
`deps`	Install the packages' dependencies too? Packages required with `deps = FALSE` are kept in a separate list and installed after the others, without their dependencies. Use this for packages whose pinned dependencies (or `Requires-Python`) can't be met in Pyodide, and require compatible versions of those dependencies normally.
`action`	`"add"` to add requirements, `"remove"` to remove them (by package name) or `"set"` to replace the list.
`session`	Session name. Defaults to the active session, or `"default"` if no session has started.

Details

- Before the session starts, requirements are only recorded. When `pyodide_start()` runs (or Python is first used), they are all installed with a single `micropip` call and the Python version is checked.
- After the session has started, new packages are installed right away. Packages that are already loaded can't be removed, so `action = "remove"` and `action = "set"` must not drop packages from a running session; stop it with `pyodide_stop()` first.

Requirements survive `pyodide_stop()`, so restarting a session installs them again.

Each Pyodide release ships exactly one Python version, so `python_version` can't choose an interpreter. It is a check: if the session's Python doesn't satisfy it, `py_require()` (or `pyodide_start()`) fails. To get a different Python, start the session with a different `pyodide_url`.

Value

With no `packages` or `python_version`, the session's requirements. Otherwise they are returned invisibly.

Examples

```
py_require(c("numpy", "pandas>=2"), session = "example")
py_require(python_version = ">=3.12", session = "example")
py_require(session = "example")
```

`py_run_string` *Run Python code*

Description

`py_run_string()` runs code in the `__main__` module, like `reticulate::py_run_string()`. `py_eval()` evaluates a single expression and returns its value. Pyodide packages that the code imports (for example `numpy`) are loaded automatically. Anything printed by Python appears in R.

Usage

```
py_run_string(code, convert = TRUE, session = NULL)

py_run_file(file, convert = TRUE, session = NULL)

py_eval(code, convert = TRUE, session = NULL)
```

Arguments

`code`	Python code.
`convert`	Convert results to R values where possible? If `FALSE`, Python objects are returned.
`session`	Pyodide session; defaults to the active one.
`file`	A file in webR's virtual file system.

Value

`py_run_string()` and `py_run_file()` return the `__main__` module, invisibly. `py_eval()` returns the value of the expression.

py_upload *Copy a file into Pyodide's file system*

Description

webR and Pyodide have separate virtual file systems. Files whose absolute paths are passed to Python as arguments are copied automatically (see the `rpyodide.mirror_files` option); use `py_upload()` to copy other files, such as Python modules to import.

Usage

```
py_upload(from, to = normalizePath(from, mustWork = TRUE), session = NULL)
```

Arguments

`from`	Path of a file in webR's file system.
`to`	Path in Pyodide's file system. Directories are created as needed.
`session`	Pyodide session; defaults to the active one.

Value

`to`, invisibly.

pyodide_sessions *List Pyodide sessions*

Description

List Pyodide sessions

Usage

```
pyodide_sessions()
```

Value

A data frame with one row per session.

pyodide_start *Start a Pyodide session*

Description

Starts a Python interpreter (Pyodide) in its own web worker. Each session is an independent Python environment with its own packages and globals, so several sessions can act like separate virtual environments. Python functions use the active session unless told otherwise. A `"default"` session is started automatically the first time Python is used.

Usage

```
pyodide_start(
  name = "default",
  packages = character(),
  pyodide_url = default_pyodide_url(),
  lockfile_url = NULL,
  activate = TRUE,
  timeout = 300
)
```

Arguments

`name`	Session name.
`packages`	Packages to require for this session; they are added to its `py_require()` list and installed once Python has started.
`pyodide_url`	URL of the Pyodide distribution.
`lockfile_url`	Optional URL of a `pyodide-lock.json` to pin package versions for this session.
`activate`	Make this the active session?
`timeout`	Seconds to wait for Pyodide to start.

Value

The session, invisibly.

pyodide_stop *Stop a Pyodide session*

Description

Terminates the session's worker (or, for `cpython_start()` sessions, its Python process). Python objects from that session can no longer be used.

Usage

```
pyodide_stop(name = .rpyodide$active)
```

Arguments

name	Session name.

Value

NULL, invisibly.

---

r_to_py *Convert between R and Python*

Description

r_to_py() converts an R value to a Python object; py_to_r() converts a Python object to an R value. Vectors of length one become Python scalars, longer vectors become lists, named vectors and lists become dicts, matrices and arrays become NumPy arrays and data frames become pandas DataFrames. Use I() to send a length-one vector as a list.

Usage

```
r_to_py(x, convert = FALSE, session = NULL)

py_to_r(x)
```

Arguments

x	An R value or Python object.
convert	For r_to_py(), should values returned from the new object be converted to R?
session	Pyodide session; defaults to the active one.

Value

r_to_py() returns a Python object; py_to_r() returns an R value.

tuple *Python tuples and dicts*

Description

Python tuples and dicts

Usage

```
tuple(..., session = NULL)

dict(..., session = NULL)
```

Arguments

`...`	Values for the tuple, or `name = value` pairs for the dict.
`session`	Pyodide session; defaults to the active one.

Value

A Python object.

use_pyodide *Choose the active Pyodide session*

Description

Choose the active Pyodide session

Usage

```
use_pyodide(name, ...)
```

Arguments

`name`	Name of a session started with `pyodide_start()`. If it hasn't been started yet, it is started.
`...`	Passed to `pyodide_start()` when the session is new.

Value

The previously active session name, invisibly.

Index

cpython_start, 2
cpython_start(), *13*

default_pyodide_url, 3
dict *(*tuple*)*, 15

find_python *(*cpython_start*)*, 2

I(), *14*
import, 4
import_builtins *(*import*)*, 4
import_main *(*import*)*, 4
is_py_object, 4
is_webr, 5

py, 5
py_available, 6
py_call *(*py_object*)*, 9
py_config, 6
py_eval *(*py_run_string*)*, 11
py_get_attr *(*py_object*)*, 9
py_get_item *(*py_object*)*, 9
py_has_attr *(*py_object*)*, 9
py_install, 7
py_install(), *3*
py_last_error, 7
py_len *(*py_object*)*, 9
py_list_attributes *(*py_object*)*, 9
py_list_packages, 8
py_module_available, 8
py_object, 9
py_repr *(*py_object*)*, 9
py_require, 10
py_require(), *3*, *13*
py_run_file *(*py_run_string*)*, 11
py_run_string, 11
py_set_attr *(*py_object*)*, 9
py_set_item *(*py_object*)*, 9
py_str *(*py_object*)*, 9
py_to_r *(*r_to_py*)*, 14
py_upload, 12
pyodide_sessions, 12
pyodide_start, 13
pyodide_start(), *3*, *10*, *11*, *15*
pyodide_stop, 13
pyodide_stop(), *10*

r_to_py, 14

tuple, 15

use_pyodide, 15