node-calls-python

1.9.1 • Public • Published 14 days ago

Readme

Code Beta

1 Dependency

7 Dependents

33 Versions

NODE-CALLS-PYTHON

node-calls-python - call Python from Node.js directly in-process without spawning processes

Suitable for running your ML or deep learning models from Node directly

Motivation

Current solutions spawn a new process whenever you want to run Python code in Node.js and communicate via IPC using sockets, stdin/stdout, etc. But creating new processes every time you want to run Python code could be a major overhead and can lead to significant performance penalties. If the execution time of your Python code is less than creating a new process, you will see significant performance problems because your Node.js code will keep creating new processes instead of executing your Python code. Suppose you have a few NumPy calls in Python: do you want to create a new process for that? I guess your answer is no. In this case, running the Python code in-process is a much better solution because using the embedded Python interpreter is much faster than creating new processes and does not require any IPC to pass the data around. The data can stay in memory and requires only some conversions between Python and Node types (using the N-API and Python C API).

Installation

```
npm install node-calls-python
```

Installation FAQ

Sometimes you have to install prerequisites to make it work.

Linux: install node, npm, node-gyp, python3, python3-dev, g++ and make

Install Node

```
sudo apt install curl
curl -sL https://deb.nodesource.com/setup_13.x | sudo -E bash -
sudo apt install nodejs
```

Install Python

```
sudo apt install python3
sudo apt install python3-dev
```

Install Node-gyp

```
sudo apt install make
sudo apt install g++
sudo npm install -g node-gyp
```

Windows: install NodeJS and Python

Install Node-gyp if missing

```
npm install --global --production windows-build-tools
npm install -g node-gyp
```

Mac: install XCode from AppStore, Node.js and Python

```
npm install node-calls-python
```

If you see installation problems on Mac with ARM (E.g. using M1 Pro), try to specify 'arch' and/or 'target_arch' parameters for npm

```
npm install --arch=arm64 --target_arch=arm64 node-calls-python
```

Examples

Calling a simple python function

Let's say you have the following python code in **test.py**

```
import numpy as np

def multiple(a, b):
    return np.multiply(a, b).tolist()
```

Then to call this function directly you can do this in Node

```
const nodecallspython = require("node-calls-python");

const py = nodecallspython.interpreter;
```

```
py.import("path/to/test.py").then(async function(pymodule) {
    const result = await py.call(pymodule, "multiple", [1, 2,
    console.log(result);
});
```

Or to call this function by using the synchronous version

```
const nodecallspython = require("node-calls-python");

const py = nodecallspython.interpreter;

py.import("path/to/test.py").then(async function(pymodule) {
    const result = py.callSync(pymodule, "multiple", [1, 2, 3,
    console.log(result);
});
```

Creating python objects

Let's say you have the following python code in **test.py**

```
import numpy as np

class Calculator:
    vector = []

    def __init__(self, vector):
        self.vector = vector

    def multiply(self, scalar, vector):
        return np.add(np.multiply(scalar, self.vector), vecto
```

Then to instance the class directly in Node

```
const nodecallspython = require("node-calls-python");

const py = nodecallspython.interpreter;
```

```
py.import("path/to/test.py").then(async function(pymodule) {
    const pyobj = await py.create(pymodule, "Calculator", [1.4
    const result = await py.call(pyobj, "multiply", 2, [10.4,
});
```

Or to instance the class synchronously and directly in Node

```
const nodecallspython = require("node-calls-python");

const py = nodecallspython.interpreter;

py.import("path/to/test.py").then(async function(pymodule) {
    const pyobj = py.createSync(pymodule, "Calculator", [1.4,
    const result = await py.callSync(pyobj, "multiply", 2, [10
});
```

Running python code

```
const nodecallspython = require("node-calls-python");

const py = nodecallspython.interpreter;

py.import("path/to/test.py").then(async function(pymodule) {
    await py.exec(pymodule, "run_my_code(1, 2, 3)"); // exec v
    const result = await py.eval(pymodule, "run_my_code(1, 2,
    console.log(result);
});
```

Running python code synchronously

```
const nodecallspython = require("node-calls-python");

const py = nodecallspython.interpreter;

const pymodule = py.importSync("path/to/test.py");
```

```
await py.execSync(pymodule, "run_my_code(1, 2, 3)"); // exec v
const result = py.evalSync(pymodule, "run_my_code(1, 2, 3)");
console.log(result);
```

Reimporting a python module

You have to set **allowReimport** paramter to **true** when calling **import/importSync**.

```
const nodecallspython = require("node-calls-python");

const py = nodecallspython.interpreter;

let pymodule = py.importSync("path/to/test.py");
pymodule = py.importSync("path/to/test.py", true);
```

Development Mode

During development you may want to update your python code running inside Node without restarting your Node process. To achieve this you can reimport your python modules. All your python modules will be reimported where the filename of your python module matches the string parameter: `path/to/your/python/code` .

```
const nodecallspython = require("node-calls-python");

const py = nodecallspython.interpreter;

py.reimport('path/to/your/python/code');
```

Another option is to run ***node-calls-python*** in development mode. In this case once you have updated your python code under `path/to/your/python/code` the runtime will automatically reimport the changed modules.

```
const nodecallspython = require("node-calls-python");

const py = nodecallspython.interpreter;

py.developmentMode('path/to/your/python/code');
```

Passing kwargs

Javascript has no similar concept to kwargs of Python. Therefore a little hack is needed here. If you pass an object with **__kwargs** property set to **true** as a parameter to **call/callSync/create/createSync** the object will be mapped to kwargs.

```
const nodecallspython = require("node-calls-python");

const py = nodecallspython.interpreter;

let pymodule = py.importSync("path/to/test.py");
py.callSync(pymodule, "your_function", arg1, arg2, {"name1": \


def your_function(arg1, arg2, **kwargs):
    print(kwargs)
```

Doing some ML with Python and Node

Let's say you have the following python code in **logreg.py**

```
from sklearn.datasets import load_iris, load_digits
from sklearn.linear_model import LogisticRegression

class LogReg:
    logreg = None

    def __init__(self, dataset):
        if (dataset == "iris"):
            X, y = load_iris(return_X_y=True)
        else:
            X, y = load_digits(return_X_y=True)

        self.logreg = LogisticRegression(random_state=42, sol\
        self.logreg.fit(X, y)
```

```
    def predict(self, X):
        return self.logreg.predict_proba(X).tolist()
```

Then you can do this in Node

```
const nodecallspython = require("node-calls-python");

const py = nodecallspython.interpreter;

py.import("logreg.py")).then(async function(pymodule) { // imp
    const logreg = await py.create(pymodule, "LogReg", "iris")

    const predict = await py.call(logreg, "predict", [[1.4, 5.
    console.log(predict);
});
```

Using as ES Module

You can import ***node-calls-python*** as an ***ES module***.

```
import { interpreter as py } from 'node-calls-python';

let pymodule = py.importSync(pyfile);
```

Using in Next.js

If you see the following error when importing in Next.js `Module not found: Can't resolve './build/Release/nodecallspython'`

You have add the following code to your next.config.mjs because currently Next.js cannot bundle native node addons properly. For more details please see **serverComponentsExternalPackages in Next.js**

```
/** @type {import('next').NextConfig} */
const nextConfig = {
  experimental: {
    serverComponentsExternalPackages: [
      'node-calls-python'
```

```
      ]
    }
};

export default nextConfig;
```

Using Python venv

You have to add the proper import path so that python could use your installed packages from your venv.

If you have created a venv by `python -m venv your-venv` your installed python packages can be found under `your-venv/Lib/site-packages` . So you have to use `addImportPath` before importing any module to pick up the python packages from your venv.

```
const nodecallspython = require("node-calls-python");

const py = nodecallspython.interpreter;

py.addImportPath(your-venv/Lib/site-packages)
```

Working Around Linking Errors on Linux

If you get an error like this while trying to call Python code `ImportError: /usr/local/lib/python3.7/dist-packages/cpython-37m-arm-linux-gnueabihf.so: undefined symbol: PyExc_RuntimeError`

You can fix it by passing the name of your libpython shared library to fixlink

```
const nodecallspython = require("node-calls-python");

const py = nodecallspython.interpreter;
py.fixlink('libpython3.7m.so');
```

See more examples here

Supported data mapping

From Node to Python

- undefined to None
- null to None
- boolean to boolean
- number to double or long (as appropriate)
- int32 to long
- uint32 to long
- int64 to long
- string to unicode (string)
- array to list
- object to dictionary

From Python to Node

- None to undefined
- boolean to boolean
- double to number
- long to int64
- unicode (string) to string
- list to array
- tuple to array
- set to array
- dictionary to object
- numpy.array to array (this has limited support, will convert every

Keywords

python c++ v8 node nodejs node-js napi

Install

```
npm i node-calls-python
```

Repository

github.com/hmenyus/node-calls-python

Homepage

github.com/hmenyus/node-calls-python#readme

Weekly Downloads

2,916

Version

1.9.1

License

MIT

Unpacked Size

93.8 kB

Total Files

25

Issues

0

Pull Requests

0

Last publish

14 days ago

Collaborators

Try on RunKit

Report malware

Support

Help

Advisories

Status

Contact npm

Company

About

Blog

Press

Terms & Policies

Policies

Terms of Use

Code of Conduct

Privacy