accelerate_python_code

February 21, 2020

1 Welcome to the accelerate python code tutorial

1.1 Build a pdf

This notebook also comes with its own pdf. You can use this pdf as a last-resort tool, when python simply won't work. The very bottom of this notebook contains some instructions for converting this notebook to pdf.

1.2 Setting up the environment

We need some packages that do not come with standard python deployments. Either install these packages via this conda / pip commands yourself, or install the environment based on the .yaml file in this directory.

2 Cython

Cython is aimed to be a superset to the python programming language. Cython is a compiled language with python-like syntax. You can write cython files (.pyx) files in a similar manner than you would write normal python files (.py), compile them and import the generated shared object (.so) as a module into your python code. Doing this allows you to greatly decrease the computational overhead at run time, i.e. make python faster.

2.1 Reading and writing files in jupyter

The cython code we are going to look at won't be that complicated. Instead of opening with your text editor of choice (Hint: There is a correct answer to the question "What is your favorite text editor" and emacs is not it.) let us open the .pyx files direclty in this notebook using magic. With magic I am talking about the stuff you put after the percent characters in cells. This includes line magic `%matplotlib notebook` and cell magic `%%time`.

```
In [37]: %%time
         for i in range(9):
             y = i ** i ** i
```

```
CPU times: user 296 ms, sys: 16 ms, total: 312 ms
Wall time: 310 ms
```

Let's use the %load magic function to open the file helloworld.pyx

```
In [14]: # %load helloworld.pyx
         print(Hello World)
```

As you can probably guess, the quotation marks declaring 'Hello World' as a string are missing. Let's fix that using the %%writefile cell magic.

```
In [31]: %%writefile helloworld.pyx
         def hello_world():
             """This is the docstring which gets transferred into the .so

             Additionally to printing hello world, I will also do some calculations.

             Args:
                 No args lol.

             Returns:
                 None

             """
             for i in range(10):
                 y = i ** i ** i
             print("Hello World")
```

```
Overwriting helloworld.pyx
```

2.2 The setup file

Cython needs a setup.py file which contains some meta-info about the file you are going to compile. It's easiest to look at the setup.py file like it's a python makefile.

```
In [22]: %%writefile setup.py
         from distutils.core import setup
         from Cython.Build import cythonize

         setup(
             ext_modules = cythonize("helloworld.pyx")
         )
```

```
Writing setup.py
```

2.3 Compiling using python

The compiling step consists of executing this line.

```
In [26]: !python setup.py build_ext --inplace
```

```
Traceback (most recent call last):
  File "setup.py", line 2, in <module>
    from Cython.Build import cythonize
ModuleNotFoundError: No module named 'Cython'
```

2.4 Importing the functions

The functions can be imported like any other .py file.

```
In [35]: from helloworld import hello_world
         print(help(hello_world))
```

```
Help on built-in function hello_world in module helloworld:

hello_world(...)
    This is the docstring which gets transferred into the .so

    Additionally to printing hello world, I will also do some calculations.

    Args:
        No args lol.

    Returns:
        None

None
```

```
In [38]: %%timeit
         hello_world()
```

```
Hello World
Hello World
Hello World
Hello World
Hello World
Hello World
Hello World
Hello World
294 ms ś 8.13 ms per loop (mean ś std. dev. of 7 runs, 1 loop each)
```

3 Convert to pdf

To build this pdf you need to have jupyter extensions installed. Either with pip:

```
$ pip install jupyter_contrib_nbextensions
```

or with conda:

```
$ conda install -c conda-forge jupyter_contrib_nbextensions
```

To convert the notebook into a pdf you can execute:

```
$ jupyter nbconvert --to pdf accelerate_python_code.ipynb
```

3.0.1 Errors

These errors might occur when you try to convert this notebook into a pdf:

Permissions to a shared object

If you get an error, because python does not have permissions to some shared object execute this:

```
$ export LD_LIBRARY_PATH=/home/kevin/.conda/envs/work_3/lib:$LD_LIBRARY_PATH
```

Wrong kernelspec

If you get an error, because conda doesn't know the kernel open the notebook file in a texteditor and change the kernel name by hand.

```
 "metadata": {
  "kernelspec": {
   "display_name": "work_3",
   "language": "python",
   "name": "nb-conda-faster_py_env-py"
```

will be changed to:

```
 "metadata": {
  "kernelspec": {
   "display_name": "work_3",
   "language": "python",
   "name": "faster_py_env"
```