cmd-intro

June 16, 2015

1 Python and the command line

So far we have used Python exclusively from the iPython notebooks. This is great and for many applications, such as plotting or documenting a project, the notebooks are a wonderful solution. However, some of you may inherit some code from a colleague that was not written in a notebook, but is saved in an executable text file.

In this session we will briefly introduce you to the basics of

- **How to import python files into the notebook**
- **How to create a Python file outside of an iPython notebook**
- **How to navigate the command line**
- **Execute a simple script using command line arguments**

1.1 Importing files into Python

We talked about importing modules already, so lets get started by just adding one module to our iPython notebook

```
In [7]: # Operating System (OS) related functionality
        import os
        # System-specific parameters and functions
        import sys
```

We can use the sys module to retun a list that stores all the directories in which Python will try to find modules. These are just paths, like any other folder on your computer! Beware that this will look very different on each computer.

```
In [8]: # Print some information about our Python version
        print(sys.version)
        print('----------')
        # Print all our system paths
        print(sys.path)
```

```
2.7.10 (default, Jun 10 2015, 19:42:47)
[GCC 4.2.1 Compatible Apple LLVM 6.1.0 (clang-602.0.53)]
----------
['', '/usr/local/lib/python2.7/site-packages/setuptools-17.1.1-py2.7.egg', '/usr/local/lib/python2.7/si
```

```
In [9]: #Pick one directory
        python_path = sys.path[6]

        #Print a portion of the files present in this directory
        print(python_path)
        print(os.listdir(python_path)[20:40])
```

```
/usr/local/Cellar/python/2.7.10/Frameworks/Python.framework/Versions/2.7/lib/python2.7/plat-mac
['bundlebuilder.py', 'bundlebuilder.pyo', 'Carbon', 'cfmfile.py', 'cfmfile.pyo', 'dialogs.rsrc', 'dialog
```

1.1.1 Why should we care about all theses paths?

You can see that Python is able to import **files** from specific **directories**. This is useful, because we can just save our own files containing certain functionality in a file and then importing it just like all the system modules!

```
In [10]: # Print the current working directory (very useful!)
         print(os.getcwd())
```

```
/Users/fabian/Code/ucl-python-course/notebooks/day2
```

You can see, that I (Fabian) am currently working in the above directory. Very organised, of course!

1.2 Creating a new file using the iPython editor

So how do we make a new file in this directory and load it into our current session?

1. Navigate back to the iPython home screen
2. Select "New" on the top right and create a new textfile
3. Now edit the textfile and add two functions

```
def greet_course():
    print("Hello course!")

def greet_person(name):
    print("Hello %s!" %name)
```

```
In [11]: print(os.listdir(os.getcwd()))
```

```
['.DS_Store', '.ipynb_checkpoints', 'alice.txt', 'AliceInWonderland.txt', 'AliceInWonderland.txt.txt', '
```

1.2.1 Import our new mini greeter module

We can now import our new greeter file and use it just as any other module

```
In [12]: import greeter
```

```
In [13]: greeter.greet_person("Fabian")
```

```
Hello Fabian!
```

```
In [14]: greeter.greet_course()
```

```
Hello course!
```

1.3 Our first command line tool - a book downloader

In scientific applications you often run into situations where you have no graphical user interface, but only a command line interface. This is the case for the UCL/Computer Science clusters. You also often get software from other people that only runs in the command line.

Now that we know how to import a text file (or script) into our notebook, let us write a small command line application to download a book from the Gutenberg project! We will use the same tool to make it available as a module

1. Go back to your iPyhton notebook home
2. Create a new file called gutenberg.py
3. Copy and **save** the following code:

```
# A tiny downloader for files in the Project Gutenberg

import urllib2
import sys

def download(url, bookname):
    '''
    Takes a valid URL and downloads the content.
    Then creates a new file in the same folder using the
    provided bookname.
    '''

    print("Downloading %s" %url)
    book = urllib2.urlopen(url).read()
    with open(bookname, "wb") as outfile:
       outfile.write(book)
    print("Writting file %s" %bookname)
    print("Done!")

def main():
    pass

if __name__ == "__main__":
    main()
```

```
In [15]: import gutenberg
```

NOTE: If you get weird error messages just restart the Kernel! (In the top menu select Kernel –> Restart)

```
In [16]: # The URL for the Alice in Wonderland book
         URL = "http://www.gutenberg.org/cache/epub/28885/pg28885.txt"
```

```
In [17]: gutenberg.download(URL, "AliceInWonderland.txt")
```

```
Downloading http://www.gutenberg.org/cache/epub/28885/pg28885.txt
Writting file AliceInWonderland.txt
Done!
```

1.3.1 Now add the command line functionality:

```
def main():
    print("All command line arguments: ", sys.argv)

    if len(sys.argv) == 3:
        url = sys.argv[1]
        bookname = sys.argv[2]
        download(url, bookname)
    else:
        print("Supply a url and bookname!")
```

Now Move to the command line

1.3.2 Here some help on how to navigate:

Windows:

1. Go to Start and search for "cmd"
2. In the command promp use the following commands:

dir - list folders
cd - moves between folders
rename - rename a file

OSX/Linux:

1. Open a Terminal window
2. In the terminal use the follwing commands

ls - list files and folders
cd - move between folders
mv - rename a file

You want to navigate to the current working directory (output will differ for everyone):

```
In [20]: print(os.getcwd())
```

```
/Users/fabian/Code/ucl-python-course/notebooks/day2
```

Once inside the directory type:

```
python gutenberg.py
```

```
In [ ]:
```