Lab-01-02-Python_fundamentals(ASSIGN01)

October 4, 2021

1 Your first assignment

2 Python_fundamentals.ipynb Overview

- Learn about Jupyter in the online tutorial at https://github.com/aparrish/dmep-python-intro/blob/master/jupyter-notebook-tutorial.ipynb
- Learn about how to edit Notebooks (`Markdown` cells) at https://www.markdowntutorial.com/

Jupyter Notebook gives you a convenient way to experiment with Python code, interspersing your experiments with notes and documentation. You can do all of this without having to muck about on the command line, and the resulting file can be easily published and shared with other people.

We'll be using Jupyter Notebook to introduce Python, show examples and practice with interactive code.

2.1 What's in a Notebook?

An Jupyter Notebook consists of a number of "cells," stacked on the page from top to bottom. Cells can have text or code in them. You can change a cell's type using the "Cell" menu at the top of the page; go to `Cell > Cell Type` and select either `Code` for Python code or `Markdown` for text. (You can also change this for the current cell using the drop-down menu in the toolbar.)

2.2 Text cells

Make a new cell, change its type to `Markdown`, type some stuff and press `Ctrl-Enter`. Jupyter Notebook will "render" the text and display it on the page in rendered format. You can hit `Enter` or click in the cell to edit its contents again. Text in `Markdown` cells is rendered according to a set of conventions called Markdown. Markdown is a simple language for marking up text with basic text formatting information (such as bold, italics, hyperlinks, tables, etc.). Here's a tutorial.

2.3 Code cells

You can also press `Alt-Enter` to render the current cell and create a new cell. New cells will by default be `Code` cells.

[17]:
```
print("This is a code cell.")
print("")
print("Any Python code you type in this cell will be run when you press the␣
 ↪'Run' button,")
```

```
print("or when you press Ctrl-Enter.")
print("")
print("If the code evaluates to something, or if it produces output, that␣
 ↪output will be")
print("shown beneath the cell after you run it.")
```

```
This is a code cell.

Any Python code you type in this cell will be run when you press the 'Run'
button,
or when you press Ctrl-Enter.

If the code evaluates to something, or if it produces output, that output will
be
shown beneath the cell after you run it.
```

[18]:
```
print("If your Python code generates an error, the error will be displayed in␣
 ↪addition")
print("to any output already produced.")

#1 / 0
```

```
If your Python code generates an error, the error will be displayed in addition
to any output already produced.
```

[Tip:] <code>print()</code> is a function. You passed the string <code>'Hello, Python!'</code>

2.4 Kernels

Notebooks are not just for a single programming language. The Kernel specifies the execution environment.

To change Kernel `Kernel > Change kernel`

We are using `Python 3` in this course.

2.4.1 Other useful Kernel operations

`Kernel > Restart & Clear Output`

`Kernel > Restart & run all`

2.5 Built-in modules

Which exact version of Python 3 are we running?

[19]:
```
# Check the Python Version

import sys
print(sys.version)
```

```
3.9.6 | packaged by conda-forge | (default, Jul 11 2021, 03:39:48)
[GCC 9.3.0]
```

[Tip:] <code>sys</code> is a built-in module that contains many system-specific parameters and

2.6 The execution order matters

Any variables you define or modules you import in one code cell will be available in subsequent code cells. Start with this:

[20]:
```
import random
stuff = ["cheddar", "daguerrotype", "elephant", "flea market"]
```

... and in subsequent cells you can do this:

[21]:
```
print(random.choice(stuff))
```

```
flea market
```

[Tip:] What happens if we run <code>print(random.choice(stuff))</code> before <code>import ran

2.7 Keyboard shortcuts

As mentioned above, `Ctrl-Enter` runs the current cell; `Alt-Enter` runs the current cell and then creates a new cell. `Enter` will start editing whichever cell is currently selected. To quit editing a cell, hit `Esc`. If the cursor isn't currently active in any cell (i.e., after you've hit `Esc`), a number of other keyboard shortcuts are available to you:

- `m` converts the selected cell to a Markdown cell
- `b` inserts a new cell below the selected one
- `x` "cuts" the selected cell; `v` pastes a previously cut cell below the selected cell
- `h` brings up a help screen with many more shortcuts.

2.8 Saving your work

Hit `Cmd-S` at any time to save your notebook. Jupyter Notebook also automatically saves occasionally. Make sure to give your notebook a descriptive title by clicking on "Untitled0" at the top of the page and replacing the text accordingly. Notebooks you save will be available on your server whenever you log in again, from wherever you log into the server.

You can "download" your notebook in various formats via `File > Download as`. You can download your notebook as a static HTML file (for, e.g., uploading to a web site), or as a `.ipynb` file, which you can share with other people who have Jupyter Notebook or make available online through, e.g., nbviewer.

2.9 Always add comments alongside your code!

In addition to writing code, note that it's always a good idea to add comments to your code. It will help others understand what you were trying to accomplish (the reason why you wrote a given snippet of code). Not only does this help other people understand your code, it can also serve as a reminder to you when you come back to it weeks or months later.

To write comments in Python, use the number symbol # before writing your comment. When you run your code, Python will ignore everything past the # on a given line.

[22]:
```
#Practice on writing comments

print('Hello, Python!') # This line prints a string
#print('Hi')
```

```
Hello, Python!
```

After executing the cell above, you should notice that This line prints a string did not appear in the output, because it was a comment (and thus ignored by Python).

The # print('Hi') line was also not executed because it was preceded by the number sign (#) as well!

```
The programmer <em>commented out</em> that second line of code. This is very usuful for debugg
```

2.10 What about errors?

Making mistakes is more common than not! Python will tell you that you have made a mistake by giving you an error message. It is important to read error messages carefully to really understand where you made a mistake and how you may go about correcting it. Let's give it a try!

[23]:
```
# running with an error

print("Hello, Python!")
```

```
Hello, Python!
```

The error message tells you:

where the error occurred (more useful in large notebook cells or scripts), and

what kind of error it was (NameError)

Here, Python attempted to run the function frint, but could not determine what frint was since frint is not a built-in function and it has not been previously defined either.

Let's try with a SyntaxError

[24]:
```
# A SyntaxError

print("Hello, Python!")
```

```
Hello, Python!
```

2.11 How are multiple errors caught by the Python interpreter?

Let's have a go with some code

[25]:
```
# Trying to print 3 lines (with an error)
print("The 1st line has been printed")
```

```
print("The 2nd line will cause an error")
print("what about the 3rd line?")
```

```
The 1st line has been printed
The 2nd line will cause an error
what about the 3rd line?
```

Don't forget that Python is an interpreted language. Compiled languages (like C or C++) examine your entire program at compile time, and are able to warn you about a whole class of errors prior to execution. In contrast, Python interprets your script line-by-line (as it executes it). Python will stop executing the entire program as soon as it encounters an error.

```
[Tip:] Programmers can also <em>handle</em> errors appropriately (for instance when they expect
```

3 A final task: submit your first assignment via Teams

In the code cell below, insert the Python code that prints out your full name and student number.

Then compile the whole notebook from scratch (fix all errors first).

Finally generate/export both an HTML and a PDF version of the compiled notebook and submit them as assignment number 1 through Teams.

[26]:
```
# insert the python code in the next line
print("My Name is :", "Idris Attal")
```

```
My Name is : Idris Attal
```

3.1 Where next?

Check the official documentation and tutorials.

[ ]:
```

```