Python 101

by ZG

What are we going to learn today?

1. Python Environments iPython
2. Philosophy Zen of Python
3. Syntax print “Hello, World!”
4. Operations +, -, /, *, %
5. Control Flow if-else, for-in
6. Data Structures [list] (tuple) {dict : ionary} {set}
7. Functions def my_func(): return 0
8. Exceptions try: … except(Exception as err)
9. Functional Programming Basics lambda, map, reduce, filter
10. Modules and Virtual Environments import my_module

Philosophy of Python

"Simplicity is the ultimate sophistication." Leonardo da Vinci

"Always code as if the guy who ends up maintaining your code will be a violent psychopath who knows where you live." John Woods

Zen of Python

Beautiful is better than ugly.
Explicit is better than implicit.
Simple is better than complex.
Complex is better than complicated.
Flat is better than nested.
Sparse is better than dense.
Readability counts.
Special cases aren't special enough to break the rules.
Although practicality beats purity.
Errors should never pass silently.
Unless explicitly silenced.
In the face of ambiguity, refuse the temptation to guess.
There should be one–and preferably only one–obvious way to do it.
Although that way may not be obvious at first unless you're Dutch.
Now is better than never.
Although never is often better than right now.
If the implementation is hard to explain, it's a bad idea.
If the implementation is easy to explain, it may be a good idea.
Namespaces are one honking great idea–let's do more of those

[type import this to read it in your Python environment]

Readability Guidelines

1. Comment your code meaningfully:

   print 2 + 5 #sum 2 and 5 → bad comment!

   print 2 + 5 #summing means of class A and B → good comment!

2. Use docstrings for your functions:

```
import math
def pythagoras(a,b):
    """
    Calculates the length of the hypotenuse from the given legths of other two sides.

    Example:
        pythagoras(3,4) -> sqrt(3^2,4^2) = 5

    Arguments:
        a (float): one of the sides
        b (float): the other side

    Returns:
        c (float): the lenth of the hypotenuse
    """
    c = math.sqrt(a**2 + b**2)
```

Readability Guidelines

> Don't you hate code that's not properly indented? Making it [indenting] part of the syntax guarantees that all code is properly indented.
>
> Guido Van Rossum

3. Use indentation.. well you have to do it anyway
4. Use consistent naming schema:

module_name, package_name, ClassName, method_name, ExceptionName, function_name, GLOBAL_CONSTANT_NAME, global_var_name, instance_var_name, function_parameter_name, local_var_name.

5. Follow the conventions of the Python community:

   https://google.github.io/styleguide/pyguide.html

Readability Guidelines

6. DRY: Do NOT repeat yourself, which means the same piece of code should not be repeated over and over again.

Readability Guidelines

7. Avoid deep nesting

```
if ( a > b ):
    if ( a < c ):
        if ( a == e ):
            if ( ... ):
                if (...):
```

8. Avoid long lines (max 80 characters) and use parentheses

```
if (width == 0 and height == 0 and
    color == 'red' and emphasis == 'strong'):
```

Readability Guidelines

9. NAME VARIABLES MEANINGFULLY

BAD!

```
a = 32423432
data = "/zgenc/code/data.csv"
```

GOOD!

```
sample_id = 32423432
raw_transactions_file = "/zgenc/code/data.csv"
```

Question

How many of the guidelines do you remember?

Exercise - Learn Style Your Code

1. Go through https://google.github.io/styleguide/pyguide.html and see if you can understand what you read

Python Environments

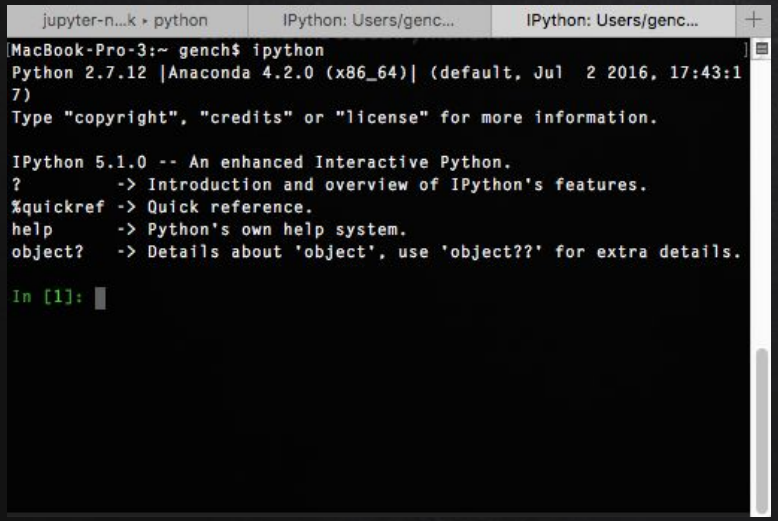
Where we run our Python code..

iPython

shell

command–line based iPython shell

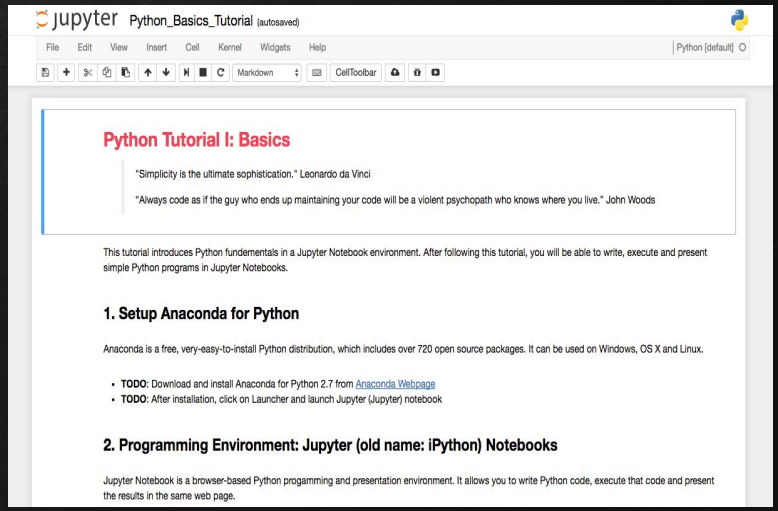
type ipython on the command–line



notebook

browser based iPython environment

type jupyter notebook on the command–line



Exercise – Learn to Ask Help

1. Start ipython shell
2. Run help(abs) and abs? commands on the terminal
3. Run time? and time?? and see the difference
4. Start a notebook
5. Repeat the steps 2 and 3 on the notebook

Python Programming

follow the sections 2–10 in the Python_Basics_Tutorial.html Tutorial to learn the syntax of Jupyter Notebooks and Python Programming.

The Jupyter Notebook version of the tutorial is Python_Basics_Tutorial.ipynb where you can run the Python code.