An Introduction to Python for Scientists

Hands-On Tutorial

Ahmed Attia

Statistical and Applied Mathematical Science Institute (SAMSI)
19 TW Alexander Dr, Durham, NC 27703
attia@ {samsi.info || vt.edu}

Department of Mathematics, North Carolina State University,
amttia2@ncsu.edu

SAMSI
SAMSI/NCSU Undergraduate Workshop; May 16, 2017

Outline

Getting Started with Python

Basic numerical, string, and boolean operations

Iterables, and basic data structures

Branching, and Looping

Functions

Python Modules and Packages

Scientific Packages: Numpy & Scipy

File I/O

Plotting: matplotlib

OOP: Python Classes

Getting Started with Python

Basic numerical, string, and boolean operations

Iterables, and basic data structures

Branching, and Looping

Functions

Python Modules and Packages

Scientific Packages: Numpy & Scipy

File I/O

Plotting: matplotlib

OOP: Python Classes

Getting Started with **Python**

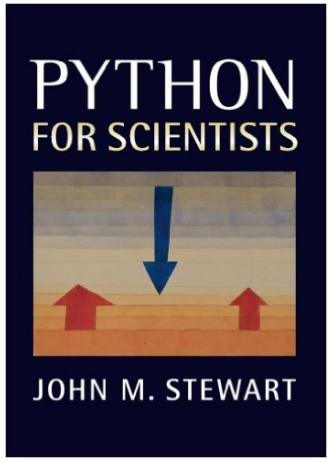
- According to (Python.org),

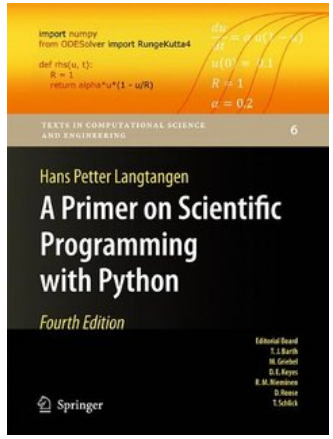
  "Python is an interpreted, object-oriented, high-level programming language with dynamic semantics.
  Its high-level built in data structures, combined with dynamic typing and dynamic binding, make it very attractive for Rapid Application Development, as well as for use as a scripting or glue language to connect existing components together."

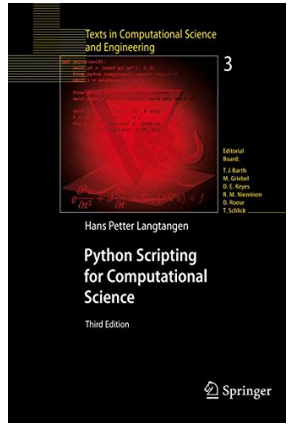
- **Why Python**:
  1. `https://www.continuum.io/why-python`
  2. `https://www.python.org/doc/essays/comparisons/`
  3. `http://www.bestprogramminglanguagefor.me/why-learn-python`
  4. `https://www.codefellows.org/blog/5-reasons-why-python-is-powerful-enough-for-google/`

Learn **Python**: where should I start?

- **Textbooks (for Scientists)**:
  1. Python for Scientists
  2. A Primer on Scientific Programming with Python
  3. Python Scripting for Computational Science







- **Phone/Tablet Apps**: Learn Python
  - **Android:** `https://play.google.com/store/apps/details?id=com.sololearn.python&hl=en`
  - **IPhone/IPad:** `https://itunes.apple.com/us/app/learn-python-pro/id953972812?mt=8`
- **Python online Documentation**: https://docs.python.org/2/
- **Practice, Practice, Practice,**

Basic Concepts

- **Python** script files have extension (.py), and compiled code files (executables) have extension (.pyc).
- **Python** scripts can be written using any plain text editor.
- Avoid using **SOLID Tabs**.
  Make sure Tabs are converted to **SPACES** (check your editor settings).
- **INDENTATION** is Extremely important. (For now, make sure all script line start at the beginning of the line.)
- **Python** is **CASE-SENSITIVE**.

Running **Python** code

- **IPython** : is an interactive command shell originally developed for **Python**, that supports tab completion, and history amongst other features. *This makes working with* **Python** *very similar to working with* MATLAB .

- **To run a Python script**:
  1. from a terminal using the command: `$ python [script-name.py]`
  2. from IPython: `$ run [script-name.py]`

- **Task:**
  1. *Slides & Code* $\longrightarrow$
     `http://people.cs.vt.edu/~attia/Files/Classes/PythonIntro/SAMSI_NCSU_May_17/`
  2. *write a "Hello World" program in:*
     - I *a python script and run it,*
     - II *in IPython shell,*

Terminal output; `print`

- Now, given the last task, you know that to print anything to the screen, we will need to use the "print" function (or statement)
- Python's ‘‘`print`’’ function can be used to print a *"string representation"* of an ‘‘`object`’’ , e.g. *string, number, etc.*:

  `print(<object>)` or `print <object>`

- The most obvious object to be printed to a screen is a *String*; one of the main datatypes in **Python** is `str` . (Next)

- **IF** you like to complicate things, check:

  `print(str([object]))`, vs. `print([object].__str__())`

- **Task:** *open the script lesson0_terminal_output.py*

Data types, and variable assignment I

- **Python** supports a variety of data types, including:
  1. Strings: `str`
  2. Numerical types: `int, long, float, complex`
  3. Boolean: `bool`
  4. None/Null: `None`
- To check/view the data type of an object, you can use: `type([object name])`

- **Python**, by default, allocates memory dynamically, and does not require data type declaration; *unlike Fortran, C, etc*.
- Variables are created with dynamic binding; assignment is done using assignment operator ‘‘=’’.

```
variable-name = variable value
```

- Multiple assignment (with dynamic binding) is allowed , e.g. `x = y = 3`.

Data types, and variable assignment II

- **Variable naming rules**:
  1. Variables names must start with a letter or an underscore,
  2. the remainder of the name may consist of letters, numbers, or underscores,
  3. variable name is case sensitive.
  4. variable name (identifier) length can be one character or more, and should be meaningful.

- Examples:

Correct	Incorrect
x, i, $x\_$, $radius$, $\_radius$, $\_\_radius$, par_1, $peremeter0\_2$, etc.	$1x$, $radi\ us$, $radi - us$ $x\$$, $s\_\#$, etc.

- Variable names starting with underscore will play vital role when we learn how to import data from **Python** modules.

- **Task:** *open the script lesson1_datatypes.py*

Numerical arithmetics

- **Python** can apply one or more numerical arithmetic operations, to integers, real numbers, or both.
  1. Addition, subtraction: [+, −]
  2. Multiplication, division, quotient, remainder(modulo): [∗, /, //, %]
  3. Unitary: [+, −]
  4. Exponentiation: [**]

  Priority (precedence) is from "lowest" to "highest". Override, and arrange operations using round braces ( ).

- **Question:** What is the result of the following operation:

$$-3 * *2 + 24//12\%4/3 - 5 * (2/3) * (1/4.)$$

  Try by hand, then use IPython, and check your answer!

String operations

- You should know by now, that strings are defined, using single, double, or triple-double quotes ("docstring").
- Strings are saved into variables of "`str`" data type. Try the following:

```
x = 'this is a string'
print(type(x))
```

- Simple string operations:
  - You can use addition `'+'` to add two strings,
  - you can replicate a string using `'*'` (multiply a string by an INTEGER)
- A string can be **formatted** easily using escape characters, e.g. \n, \r, \t. You can also use %[datatype] to insert something in a string,

- **Task:**
  What is the result of the following expression?

```
" Lesson%d \n is \t pointless%s " % (0, ':  '*2)
```

Boolean (logical) operations

- Logical operations include:

```
>, < , ==, >=, <=, !=
in, not in, is, is not
not, and, or
```

- You can also use double expressions such as: `a < x < b`.
- The priority of all these is less than arithmetics, and is ranked from highest to lowest as follows:
  1. `>, < , ==, >=, <=, !=, in, not in, is, is not`
  2. `not`
  3. `and`
  4. `or`
- For full operator precedence, check: `https://docs.python.org/2/reference/expressions.html`

Basic numerical, string, and boolean operations

Task:

1. *Open the script lesson2_numbers_arithmetics.py*
2. *Open the script lesson2_string_operations.py*
3. *Open the script lesson2_boolean_expressions.py*

Iterables, and basic data structures

- An "*iterable*" is a Pythonic term, refers to any object capable of returning its members one at a time.
- Basic examples include:
  1. lists,
  2. tuples,
  3. dictionaries,
  4. sets,
  5. strings.

- **Task:**
  To learn how to create, access, or modify these objects:
  1. *Open the script lesson3_lists.py*
  2. *Open the script lesson3_tuples.py*
  3. *Open the script lesson3_dictionaries.py*

Getting Started with Python

Basic numerical, string, and boolean operations

Iterables, and basic data structures

Branching, and Looping

Functions

Python Modules and Packages

Scientific Packages: Numpy & Scipy

File I/O

Plotting: matplotlib

OOP: Python Classes

Branching, and Looping: if, for, while

- We will learn this lesson directly from the code examples.

Task:

1. *Open the script lesson4_if_statement.py*
2. *Open the script lesson4_loops.py*

Functions

- We will learn this lesson directly from the code examples.

- **Task:**
  Open the script lesson5_functions.py

Using **Python** Modules

- The components of a **Python** module, or a (.py) file can be imported into the current working space, using the '`import`' statement. Examples (discussed further in class)
  1. `import [module name]`
  2. `import module1, module2`
  3. `import module1 as alias`
  4. `from [module name] import [variable, function, class, submodule] <as alias>`
- Once a module is imported, its contents can be accessed using the [.] operator
- The following statement imports everything(ish) from a module:

  `from [module name] import *`
- Popular packages for scientific computing: math, numpy, scipy, matplotlib, statsmodel, re, etc.
- Check this: `https://pythontips.com/2013/07/30/20-python-libraries-you-cant-live-without/`

- **Task:** *Open the script lesson6_modules.py*

Numpy

- According to (`http://www.numpy.org/`):
  NumPy is the fundamental package for scientific computing with **Python**.

- **Numpy** contains, *among other things*:
  1. a powerful N-dimensional array object
  2. sophisticated "*broadcasting*" functions
  3. tools for integrating C/C++ and Fortran code
  4. useful linear algebra, Fourier transform, and random number capabilities

- **Task:** *Open the script lesson7_numpy_intro.py*

Scipy: sparsity and more!

- Once you become familiar with Numpy, you will realize that some linear algebra and statistical tools are not provided.
- Amongst others, Numpy does not provide sparse linear algebra functionalities. This is the time to start learning/using **Scipy**
- According to (`https://docs.scipy.org/doc/scipy/reference/`), SciPy (pronounced Sigh Pie) open-source software for mathematics, science, and engineering.

- **Task:** *Open the script lesson8_scipy_intro.py*

Files, input, and output operations

- You can open a file, to read 'r', write 'w', modify 'r+', or append 'a' .

```
file_id = open([file name or path], mode='r')
<Do stuff with the file>
file_id.method()
<Do stuff with the file>
file_id.close()
```

- A better approach:

```
with open([file name or path]) as file_id:
    Do stuff with the file>
    file_id.method()
    <Do stuff with the file>
```

- Matlab (`.mat`) files:

```
import scipy.io as file_io
file_contents = file_io.loadmat([matlab mat file name or
path])
```

- **Task:** *Open the script lesson9_fileIO.py*

Plotting: matplotlib

- According to (`http://matplotlib.org/`):

  Matplotlib is a **Python** 2D plotting library which produces publication quality figures in a variety of hardcopy formats and interactive environments across platforms. Matplotlib can be used in **Python** scripts, the **Python** and IPython shell, the jupyter notebook, web application servers, and four graphical user interface toolkits.

- **Task:** *Open the script lesson10_matplotlib.py*

Getting Started with Python

Basic numerical, string, and boolean operations

Iterables, and basic data structures

Branching, and Looping

Functions

Python Modules and Packages

Scientific Packages: Numpy & Scipy

File I/O

Plotting: matplotlib

OOP: Python Classes

OOP: **Python** Classes

Task:

Open the script lesson11_classes_intro.py