Introduction to Python

Phạm Quang Nhật Minh
AImesoft JSC
minhpham0902@gmail.com

December 24, 2022

Agenda

- Python setup
- Python Introduction

Why Python?

- Python is the most used language for data science and machine learning and top popular language in 2019

Language Ranking: IEEE Spectrum

Rank	Language	Type	Score
1	Python	web, desktop, embedded	100.0
2	Java	web, mobile, desktop	96.3
3	C	mobile, desktop, embedded	94.4
4	C++	mobile, desktop, embedded	87.5
5	R	desktop	81.5
6	JavaScript	web	79.4
7	C#	web, mobile, desktop, embedded	74.5
8	Matlab	desktop	70.6
9	Swift	mobile, desktop	69.1
10	Go	web, desktop	68.0

Why Python?

- “the popularity of the language in the fast-growing field of machine learning,…”

  NumPy, Pandas, Scikit-learn, Pytorch, Tensorflow and Keras, NLTK

- Easy to use, easy to learn

  With human language like syntax

```
if a not None:
    print(“a is not None”)
```

- Let you quickly prototype and test your ideas!

Python Setup

- Most OS other than Windows already have Python installed by default

  Type python in Terminal

- We use Python 3.x in the class

- Anaconda is recommended to setup Python environment

  https://www.anaconda.com/distribution

Installing Packages

- Use **pip**
- E.g.,

  pip install nltk

  pip install numpy

- Or **conda**

Edit Python code

- Sublime Text

  https://www.sublimetext.com/3

- IDEs:

  PyCharm (heavy): https://www.jetbrains.com/pycharm

  Visual Studio Code: https://code.visualstudio.com

Google Colab

- If you do not have a good computer or you struggle to setup Python on your machine, then
  - Use Google Colab: https://colab.research.google.com
- Why Google Colab?
  - Free cloud service based on Jupyter Notebook
  - Colab provides GPU and it's **totally free**.
- You need a Google account in order to use Google Colab
  - If you do not have it yet, please register one

Jupyter Notebook

- An interactive environment that lets you write and execute code in Python and other languages
- You can write code and document in the same place

Getting Started

The document you are reading is a Jupyter notebook, hosted in Colaboratory. It is not a static page, but an interactive environment that lets you write and execute code in Python and other languages.

For example, here is a **code cell** with a short Python script that computes a value, stores it in a variable, and prints the result:

Text cell

```
[ ]  1 seconds_in_a_day = 24 * 60 * 60
     2 seconds_in_a_day
```

Code cell

```
86400
```

Output

- Documents are written with Markdown language

Practice Exercise 1

- Create a Google account if you do not have one
- Open link: https://colab.research.google.com/notebooks/intro.ipynb

  Open Overview of Colaboratory notebook
- Create a new notebook
- Try to write something with Markdown syntax
- Add a code cell and execute

Language Introduction

- Python is a dynamic, interpreted (bytecode-compiled) language

  No compiling process like C/C++

  You lose the compile-time type checking of the source code.

- There are no type declarations of variables, parameters, functions, or methods in source code

```
>> a = 6
>> b = a + 2
```

Python Interactive mode


```
$ python         ## Run the Python interpreter
Python 2.7.9 (default, Dec 30 2014, 03:41:42)
[GCC 4.1.2 20080704 (Red Hat 4.1.2-55)] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> a = 6        ## set a variable in this interpreter session
>>> a            ## entering an expression prints its value
6
>>> a + 2
8
>>> a = 'hi'     ## 'a' can hold a string just as well
>>> a
'hi'
>>> len(a)       ## call the len() function on a string
2
>>> a + len(a)  ## try something that doesn't work
Traceback (most recent call last):
  File "", line 1, in
TypeError: cannot concatenate 'str' and 'int' objects
>>> a + str(len(a))  ## probably what you really wanted
'hi2'
>>> foo          ## try something else that doesn't work
Traceback (most recent call last):
  File "", line 1, in
NameError: name 'foo' is not defined
>>> ^D           ## type CTRL-d to exit (CTRL-z in Windows/DOS terminal)
```

Python source code

- Use .py extension
- Following code will input command line argument and print out a greeting message

```
# import modules used here -- sys is a very standard one
import sys

# Gather our code in a main() function
def main():
    print('Hello there', sys.argv[1])
    # Command line args are in sys.argv[1], sys.argv[2] ...
    # sys.argv[0] is the script name itself and can be ignored

# Standard boilerplate to call the main() function to begin
# the program.
if __name__ == '__main__':
    main()
```

Run a Python source code

- Open Terminal, and type

  $ python hello.py John

  Hello there John

Practice Exercise 2

- Install Python environment in your computer
- Open your text editor, copy & paste code in previous slide, saved the file as hello.py
- Open terminal, and run

```
$ python hello.py John
```

User-defined Functions


```
# Defines a "repeat" function that takes 2 arguments.
def repeat(s, exclaim):
    """
    Returns the string 's' repeated 3 times.
    If exclaim is true, add exclamation marks.
    """

    result = s + s + s # can also use "s * 3" which is faster (Why?)
    if exclaim:
        result = result + '!!!'
    return result
```

Code that call the repeat function

```
def main():
    print repeat('Yay', False)      ## YayYayYay
    print repeat('Woo Hoo', True)   ## Woo HooWoo HooWoo Hoo!!!
```

Indentation

- The whitespace indentation of a piece of code affects its meaning
- A logical block of statements should all have the same indentation

✓
```
if a > 2:
    b = a + 2
    c = a * 2
```

✗
```
if a > 2:
    b = a + 2
      c = a * 2
```

Code check at Runtime

- No Runtime error in this case

```
def main():
    name = 'John'
    if name == 'Guido':
        print(repeeeet(name) + '!!!')
    else:
        print(repeat(name))
```

Variable Names

- Use meaningful names to your variables

  e.g, `name` for a single name, `names` for a list of name

  Important in python to avoid type errors

- Should not override built-in names as variable

  str, list, dict, set,..

More about modules and namespaces


```
# utils.py module

def test(got, expected):
    if got == expected:
        print(“Ok”)
    else:
        print(“Not Ok”)

def foo(name):
    print(“Hello:”, name)
```

```
# client code a.py

import utils

utils.test(2, 3)

utils.foo(“John”)
```

```
# client code b.py

from utils import test, foo

test(2, 3)

foo(“John”)
```

Control structures

- If statement

```
import random

a = random.randint(1,101)

print("a=%d" % a)

if a >= 20:
    print("Greater than or equal to 20")
else:
    print("Less than 20")
```

Loop

- For loop

```
for i in range(10):
    print("step = %d" % i)
```

- While loop

```
i = 0
while i < 10:
    print("step = %d" % i)
    i += 1
```

Online help, help(), and dir()

- Ways to get helps

  Do a Google search: “python list”, “python string lowercase”

  The official Python docs site — docs.python.org

  Use the help() and dir() functions

```
>> help(len)
>> a = [1, 2, 3]
>> dir(a)  # show list of attributes of an object
```

For further

- Google Python class

  https://developers.google.com/edu/python/introduction