Introduction to Digital Signal Processing with Python

Ehsan Khodapanah Aghdam

May 14, 2022

Tabriz University

Does Python Worth to Learn It?

History

- Python was originally conceptualized by *Guido van Rossum* in the late 1980s.
- Python 0.9.0 was first released in 1991.
- In 2000 , Python 2.0 was released.
- Python 3.0 was the next version and was released in December of 2008.
- Fun fact: Python is not named after the snake. It s named after the British TV show ”Monty Python: Flying”.

Guido van Rossum

Zen of Python

Python is an interpreted, high level, general purpose programming language. Python has a design philosophy that emphasizes code readability.

- 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.

You can easily show this just by scripting the following code snippets.

```
1 import this
```

Make It Clear !

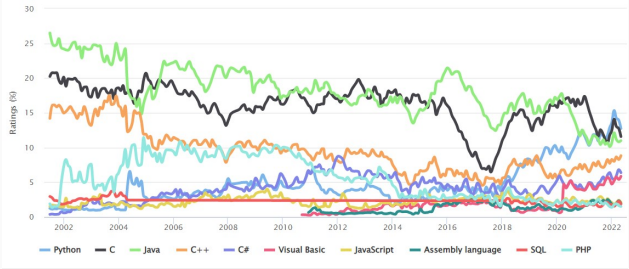
Previously for scientific programming MATLAB or C/C++ used, but for the following reasons Python is considered as a main tool.

Python Pros

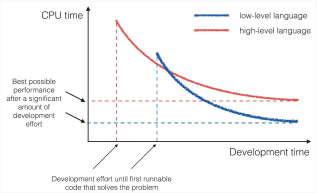
- Easy to learn due to its analogy to human language
- High productivity
- Portable
- Great amount of usable packages [check pypi.org]
- Object Oriented Programming (OOP)

Python Cons

- Low speed
- Inefficient memory consumption
- After learning Python as a first programming language it feels hard to learn other languages



TIOBE index



Comparison between high level and low level languages

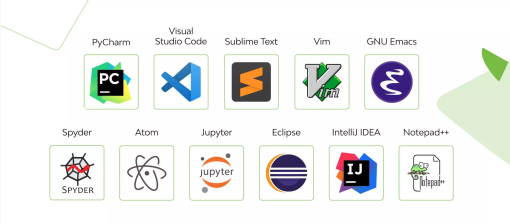
How to learn Python?

- Real PythonTutorials Website
- Python Notes for Professionals book
- Persian Free Python Book
- Bunch of books from distinguished publishers such as: Oreilly, Manning, No Starch Press, Packt , Wiley, Springer and so on that you could easily download it from LibGen freely.
- Solve questions from websites like Quera.org and leetcode.com for getting active mind.

There are a lot of IDEs [1]out there but the followings are our desired:

- PyCharm
- Visual Studio Code
- Jupyter Notebook (used in our class)

We mainly going to cover some important packages that they are vital for our goal in Digital Signal Processing (DSP) like: Numpy, SymPy and Matplotlib



Python IDEs

[1]Integrated Development Environment

Data types in Python

```python
print("Hello World!")
# output: Hello World!
print('"Hello" World!')
# output: "Hello" World!
```

```python
# Boolean
True, False
# Integer
0, 1, 23, int(3.8)
# Float
1.2, 3., .5, float(2)
# Complex
1 + 2j, complex(23)
# None
None
# type function
type(35) # <class 'int'>
type(.5) # <class 'float'>
type(True) # <class 'bool'>
```

Code Snippet 1: Strings

```
"This is a string" str(1000) # '1000'
'ABC' + 'def' # ABCdef
"Hello World!".index('o') # 4
"Hello World!".replace('World', 'Ferdowsi') # 'Hello Ferdowsi!'
```

Code Snippet 2: Tuple (immutable)

```
t = 'a', 'b', 'c'
# better to use parentheses around a tuple:
t = ('a', 'b', 'c')
# tuple with a single element(notice comma here):
t1 = (50,)
t2 = tuple('Hello')
print(t2) # output: ('H', 'e', 'l', 'l', 'o')
print(t2[0]) # output: 'H'
```

Code Snippet 3: Dictionary (mutable)

```python
d = {'a': 1, 'b': 2, 'c': 3}
print( d['a'] )  # 1
print( d['e'] ) # KeyError: 'e'
d['b'] = 200  # modifying value
d['d'] = 4  # adding new key:value
d2 = dict(e=5, f=6)
d.update(d2)
print(d) # {'a': 1, 'b': 200, 'c': 3, 'd': 4, 'e': 5, 'f': 6}
```

Code Snippet 4: List (mutable)

```python
l1 = [1 , 2 , 3]
l2 = list(7, 8)
l1.append(12) # 1 , 2 , 3 , 7 , 8
l2.index(3)  # output: 2
# Slicing
l1 = [1 , 2 , 3 , 4 , 5]
l1[0 : 2 ] # 1 , 2
```

Code Snippet 5: Set (immutable)

```
l = [1, 2, 3, 4, 4, 8, 6, 'e']
s = set(l) # {1, 2, 3, 4, 6, 8, 'e'}
s[1] = 4 # TypeError: 'set' object does not support item assignment
a = {1, 10, 100, 12}
b = {100, 1000, 10}
a.intersection(b) # {10, 100}
a.difference(b) # {1, 12}
a.issubset(b) # False
```

Conditions and Loops

Code Snippet 6: Conditions

```python
x, y = 3, 4

if x > y:
  print(x, '>', y)
elif x == y:
  print(x, 'equals', y)
else:
  print(x, '<', y)

name = 'python'
if name[0] == 'p' and name[-1] == 'n':
  print(name)

l1 = [100, 200, 300]
if 300 in l1:
  print(300, 'is in', l1)

if 'x' not in 'python':
  print('x is not in python!')
```

Code Snippet 7: Loops

```python
names = ['Ali', 'Mehdi', 'Pooya']
# for loop
for name in names:
  print(name)

for idx, name in enumerate(names):
  print(idx, '-', name)

d = {'alpha': 0.7, 'beta': 2.5, 'gamma':
    0.02}

# for loop on dict
for k, v in d.items():
  print(k, ':', v)

# while loop
i = 0
while (i < 5):
  print(i)
  i += 1
```

Functions

Code Snippet 8: Functions

```python
# defining your function
def greeting(name):
  print(f'Hi {name}!')

greeting('John')
# Hi John!

# Keyword arguments
def divide(first, second):
  return first / second

print(divide(second=2, first=5))
 # 2.5
```

Packages and Modules

- Any folder with or without `__init__.py` is a python package.
- Any python file(`*.py`) is a python module.
- After importing a module, it creates a separate name space.

Code Snippet 9: How to import packages and modules

```
import <module_name(s)>
from <module_name> import <name(s)>
from <module_name> import *
import <module_name> as <alt_name>
from <module_name> import <name> as <alt_name>
```

- Executing a Module as a Script Standalone

```
$ python path/to/script_name.py
```

Before diving in the programming please consider that we first cover a brief review on Jupyter Notebook then talk about Python syntax and then our introductory class follow by Numpy, SymPy , Matplotlib and eventually using them in DSP.

Numpy

The foundation for numerical and scientific computation in Python is the NumPy [2] package, and essentially all scientific libraries in Python build on this e.g. Scipy , Pandas , scikit-learn , CV2 , etc. So why Numpy is popular?

- A powerful N-dimensional array object
- Sophisticated (broadcasting) functions
- Tools for integrating with C/C++ and Fortran code
- Useful linear algebra, Fourier transform, and random number capabilities
- For whom immigrate from MATLAB to python this package is very resemblance to MATLAB array functionality [check this link]

[2] https://numpy.org/doc/stable/user/absolute_beginners.html

Why use Numpy instead of Python's List data type?

- Open source numerical library used for working with mathematical functions with multi-dimensional array and matrix data structures.
- Very common in Data Science and Machine Learning
- NumPy arrays use less memory than normal Python lists
- A normal Python list is a group of pointers to separate Python objects
- A NumPy array is designed to be an array of uniform values, without using space for type pointers
- NumPy can also read in information faster than normal Python and has lots of convenient broadcasting operations that can be performed across array dimensions

ND-Array / Array

- The base structure in numpy is ndarray , used to represent vectors, matrices and higher-dimensional arrays.
- Eachndarray has the following attributes:
  - `dtype` = corresponds to data types in C language
  - `ndim` = the number of axes (dimensions) of the array
  - `shape` = dimensions of array
  - `size` = the total number of elements of the array
  - `strides` = number of bytes to step in each direction when traversing the array

```
1 import numpy as np
2 x = np.array([1, 2, 3, 4, 5, 6])
3 print(x) # [1 2 3 4 5 6]
4 print('dytpe', x.dtype) # dytpe int32
5 print('dimension', x.ndim) # dimension 1
6 print('shape:', x.shape, ', size:', x.size) # shape: (6,) , size: 6
7 print('strides', x.strides) # strides (4,)
```

Create Array

Code Snippet 10: There are many ways to create a Numpy array

```
import numpy as np
l = [1, 2, 3, 4, 5, 6]
a = np.array(l)
a = np.array([1, 2, 3], dtype=np.float64)
a = np.array([[0, 1, 2], [3, 4, 5]])
a = np.arange(9)
a = np.arange(3, 9, 0.5)
a = np.zeros((3, 4))
a = np.ones(3)
a = np.eye(3)
# ...
```

Code Snippet 11: Cast and Reshape

```
import numpy as np

a = np.arange(12)  # dtype: int32
b = a.astype(np.float32) # dtype: float64

c = a.reshape(3, 4) # reshape into 3 by 4 array
c = a.reshape(2, -1) # reshape into 2 by 6 array
```

Indexing / Sliding

Code Snippet 12: Indexing and Slicing

```python
import numpy as np

a = np.arange(35)
print(a[2:5])
print(a[:-7])
print(a[0:7:2]) # [start:end:step]

a = a.reshape(5, 7)
print(a[1], a[1, :])
print(a[0:5:2, ::3])
```

Code Snippet 13: Saving And Loading

```python
import numpy as np

a = np.arange(12).reshape(3, 4)
np.savetxt('./my_array.txt', a)

b = np.loadtxt('./my_array.txt')
print(b)

# numpy binary format
np.save('./my_array.npy', a)
b = np.load('./my_array.npy')
```

Operations

Code Snippet 14: Elementwise

```python
import numpy as np

a = np.array([1, 2, 3, 4])
print(a + 1) # [2 3 4 5]
print(a * 5) # [ 5 10 15 20]
print(2 ** a) # [ 2  4  8 16]
print(a + a) # [2 4 6 8]

a = list(a)
print(type(a)) # <class 'list'>
print(a * 3) # [1, 2, 3, 4, 1, 2, 3, 4, 1, 2, 3, 4]
print(a + 1)  # TypeError: only concatenate list to
    list
```

Code Snippet 15: Product

```python
import numpy as np

a = np.arange(12).reshape(3, 4)
np.savetxt('./my_array.txt', a)

b = np.loadtxt('./my_array.txt')
print(b)

# numpy binary format
np.save('./my_array.npy', a)
b = np.load('./my_array.npy')
```

📢 For any Q&A contact with me via 📧: engtekh@gmail.com

📢 For downloadable materials such codes and presentation check my GitHub 🌐: link

📢 Now Let's Code 😜

[1] Robert Johansson.
Numerical Python: Scientific Computing and Data Science Applications with Numpy, SciPy and Matplotlib, Apress, Second Edition. 2018.

[2] José Unpingco
Python for Signal Processing, 2013.

[3] John G. Proakis
Digital Signal Processing Using MATLAB: A Problem Solving Companion, 2017.

[4] NumPy
https://numpy.org/doc/stable/user/tutorials_index.html, **11/4/2021.**

[5] SymPy
https://docs.sympy.org/latest/index.html, **11/20/2021.**

[6] Matplolib
https://matplotlib.org/stable/index.html, **10/10/2021.**

Thanks For Your Attention