



# Introduction to Digital Signal Processing with Python

---

**Ehsan Khodapanah Aghdam**

September 18, 2022

**Tabriz University**

Does Python Worth to Learn It?

Data types in Python

Conditions and Loops

Functions

Packages and Modules

Numpy

Does Python Worth to Learn It?

---

# Why Python?

## What is Python?

### 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

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
```

# Why Python?

## Make It Clear !

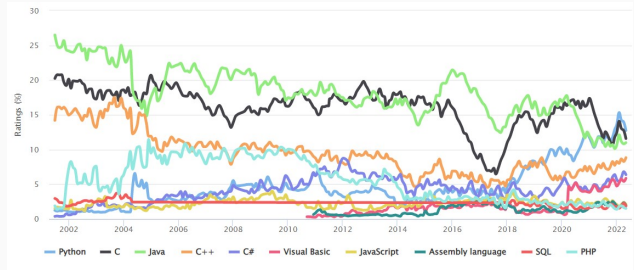
Previously for scientific programming MATLAB or C/C++ used, but for the following reasons Python is considered as a main tool.

### Python Pros

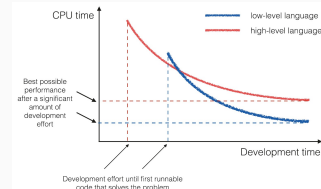
- Easy to learn due to its analogy to human language
- High productivity
- Portable
- Great amount of usable packages [check [pypi.org](https://pypi.org)]
- Object Oriented Programming (OOP)

### Python Cons

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



### TIOBE index

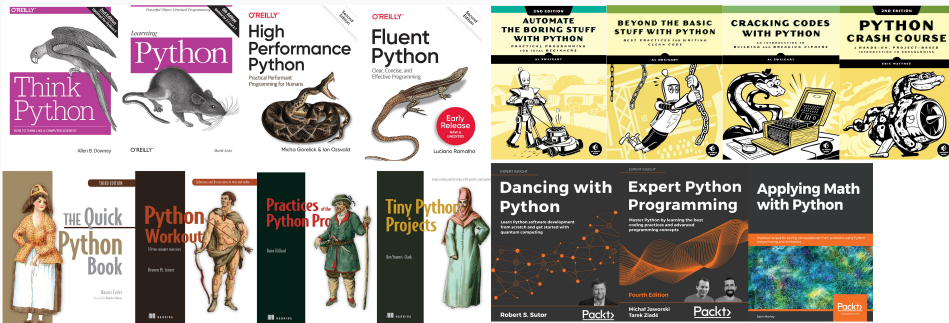


Comparison between high level and low level languages

# References to learn Python

## How to learn Python?

- [Real PythonTutorials Website](#)
- [Python Notes for Professionals book](#)
- [Persian Free Python Book](#)
- Bunch of books from distinguished publishers such as: O'Reilly, 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.

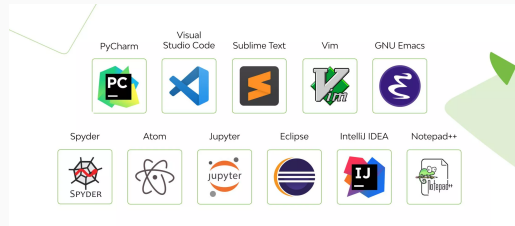


## Where to code?

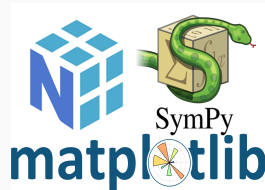
There are a lot of IDEs <sup>1</sup>out there but the followings are our desired:

- 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



---

<sup>1</sup>Integrated Development Environment



## Data types in Python

---

```
1 print("Hello World!")
2 # output: Hello World!
3 print('"Hello" World!')
4 # output: "Hello" World!
```

```
1 # Boolean
2 True, False
3 # Integer
4 0, 1, 23, int(3.8)
5 # Float
6 1.2, 3., .5, float(2)
7 # Complex
8 1 + 2j, complex(23)
9 # None
10 None
11 # type function
12 type(35) # <class 'int'>
13 type(.5) # <class 'float'>
14 type(True) # <class 'bool'>
```

## Code Snippet 1: Strings

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

## Code Snippet 2: Tuple (immutable)

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

## Code Snippet 3: Dictionary (mutable)

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

## Code Snippet 4: List (mutable)

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



## Code Snippet 5: Set (immutable)

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

## Conditions and Loops

---

### Code Snippet 6: Conditions

```
1 x, y = 3, 4
2
3 if x > y:
4     print(x, '>', y)
5 elif x == y:
6     print(x, 'equals', y)
7 else:
8     print(x, '<', y)
9
10 name = 'python'
11 if name[0] == 'p' and name[-1] == 'n':
12     print(name)
13
14 l1 = [100, 200, 300]
15 if 300 in l1:
16     print(300, 'is in', l1)
17
18 if 'x' not in 'python':
19     print('x is not in python!')
```

### Code Snippet 7: Loops

```
1 names = ['Ali', 'Mehdi', 'Pooya']
2 # for loop
3 for name in names:
4     print(name)
5
6 for idx, name in enumerate(names):
7     print(idx, '-', name)
8
9 d = {'alpha': 0.7, 'beta': 2.5, 'gamma':
      0.02}
10
11 # for loop on dict
12 for k, v in d.items():
13     print(k, ': ', v)
14
15 # while loop
16 i = 0
17 while (i < 5):
18     print(i)
19     i += 1
```



## Functions

---



## Code Snippet 8: Functions

```
1 # defining your function
2 def greeting(name):
3     print(f'Hi {name}!')
4
5 greeting('John')
6 # Hi John!
7
8 # Keyword arguments
9 def divide(first, second):
10     return first / second
11
12 print(divide(second=2, first=5))
13 # 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

```
1 import <module_name(s)>
2 from <module_name> import <name(s)>
3 from <module_name> import *
4 import <module_name> as <alt_name>
5 from <module_name> import <name> as <alt_name>
```

- Executing a Module as a Script Standalone

```
1 $ 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 <sup>2</sup> 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](#)]

---

<sup>2</sup>[https://numpy.org/doc/stable/user/absolute\\_beginners.html](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.
- Each ndarray 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('dtype', x.dtype) # dtype 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,)
```

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

```
1 import numpy as np
2 l = [1, 2, 3, 4, 5, 6]
3 a = np.array(l)
4 a = np.array([1, 2, 3], dtype=np.float64)
5 a = np.array([[0, 1, 2], [3, 4, 5]])
6 a = np.arange(9)
7 a = np.arange(3, 9, 0.5)
8 a = np.zeros((3, 4))
9 a = np.ones(3)
10 a = np.eye(3)
11 # ...
```

Code Snippet 11: Cast and Reshape

```
1 import numpy as np
2
3 a = np.arange(12) # dtype: int32
4 b = a.astype(np.float32) # dtype: float64
5
6 c = a.reshape(3, 4) # reshape into 3 by 4
   array
7 c = a.reshape(2, -1) # reshape into 2 by 6
   array
```



Code Snippet 12: Indexing and Slicing

```
1 import numpy as np
2
3 a = np.arange(35)
4 print(a[2:5])
5 print(a[:-7])
6 print(a[0:7:2]) # [start:end:step]
7
8 a = a.reshape(5, 7)
9 print(a[1], a[1, :])
10 print(a[0:5:2, ::3])
```

Code Snippet 13: Saving And Loading

```
1 import numpy as np
2
3 a = np.arange(12).reshape(3, 4)
4 np.savetxt('./my_array.txt', a)
5
6 b = np.loadtxt('./my_array.txt')
7 print(b)
8
9 # numpy binary format
10 np.save('./my_array.npy', a)
11 b = np.load('./my_array.npy')
```

## Operations

Code Snippet 14: Elementwise

```
1 import numpy as np
2
3 a = np.array([1, 2, 3, 4])
4 print(a + 1) # [2 3 4 5]
5 print(a * 5) # [ 5 10 15 20]
6 print(2 ** a) # [ 2  4  8 16]
7 print(a + a) # [2 4 6 8]
8
9 a = list(a)
10 print(type(a)) # <class 'list'>
11 print(a * 3) # [1, 2, 3, 4, 1, 2, 3, 4, 1, 2, 3, 4]
12 print(a + 1) # TypeError: only concatenate list to
    list
```

Code Snippet 15: Product

```
1 import numpy as np
2
3 a = np.arange(12).reshape(3, 4)
4 np.savetxt('./my_array.txt', a)
5
6 b = np.loadtxt('./my_array.txt')
7 print(b)
8
9 # numpy binary format
10 np.save('./my_array.npy', a)
11 b = np.load('./my_array.npy')
```

📣 For any Q&A contact with me via 📧: [engtekh@gmail.com](mailto: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](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] Matplotlib  
<https://matplotlib.org/stable/index.html>, 10/10/2021.



Thanks For Your Attention