



Introduction to python and its application

*Sheeba Rani
TBG Group, ICGEB*

Talk Contents



Brief history

What is python?

Features of python

Python IDE

Anaconda

What makes python so powerful?

Python number data types

Python variable, indexes, strings

Python architecture

Python constructs

Python frameworks

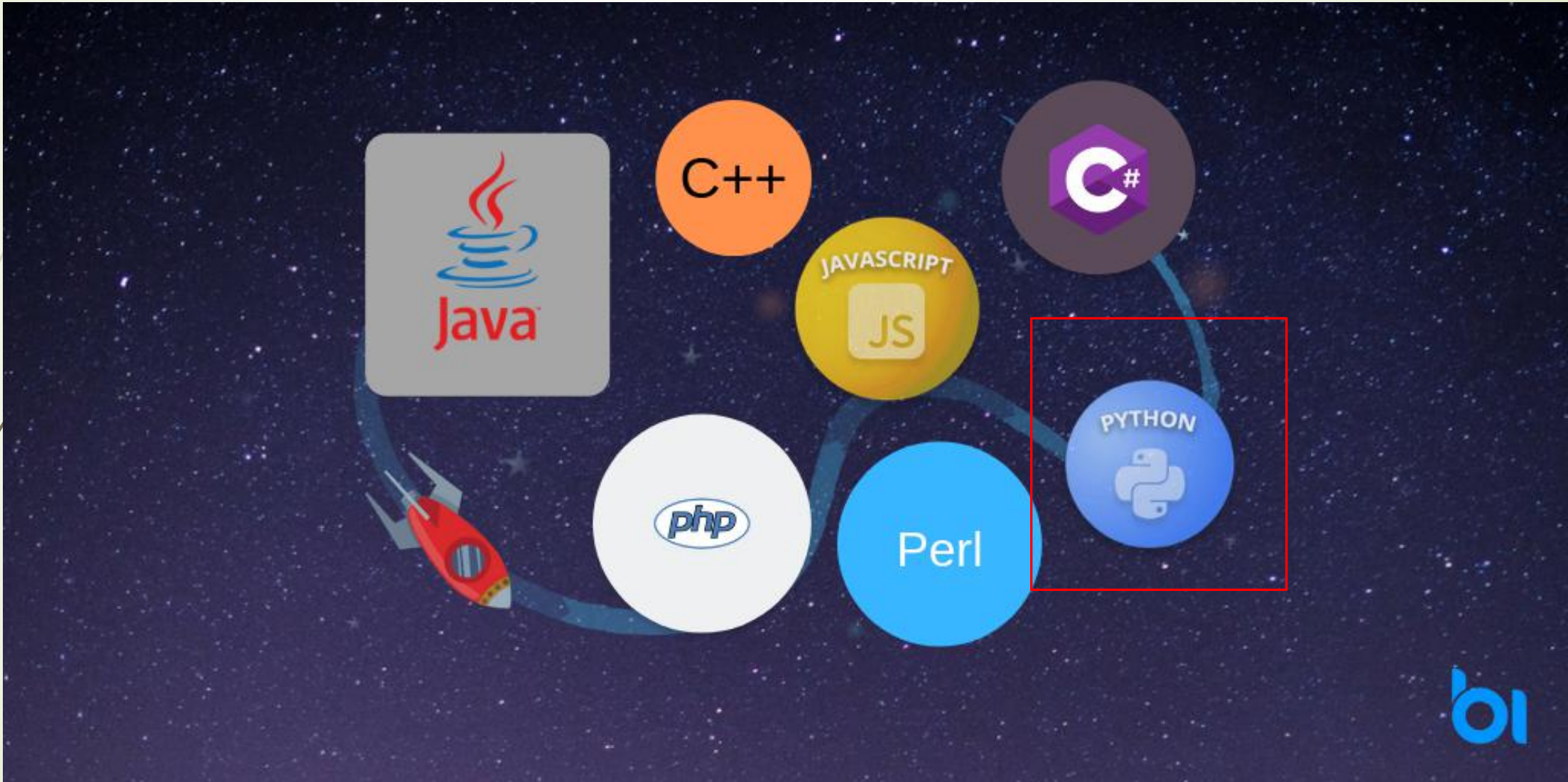
file handling

functions

Loops

conditional statement

Popularly used programming languages



Python and its history

- Python is a widely used general-purpose, high level programming language.
- It was mainly developed for emphasis on code readability, and its syntax allows programmers to express concepts in fewer lines of code.
- **Guido van Rossum** created the Python programming language in the late **1980s**.

Python's Benevolent Dictator For Life

“Python is an experiment in how much freedom programmers need. Too much freedom and nobody can read another's code; too little and expressiveness is endangered.”

- Guido van Rossum



Features of python

Easy

Interpreted

Object-oriented

Free and Open source

Portable

GUI Programming

Large Library

Python IDE

IDE - stands for integrated development environment.

It is a coding platform which gave you the opportunities to write ,test ,& debug your programs in an easier manner.

IDE combines all of the features and tools needed by a software or app developer at one place.



Some popular examples of python IDE

are :-

Python IDLE

Pycharm

Atom

Anaconda

Visual studio

jupyter



Anaconda & Anaconda navigator

Anaconda is a free and open source distribution of the python & R- programming languages.

Anaconda navigator is a desktop graphical user interface included in Anaconda.

Anaconda navigator contains IDE's(Spyder, Jupyter,...) , it comes with Pre-install libraries & plugins.

Numeric Types

Distinct numeric types:

1. integers,
2. floating point numbers,
3. complex numbers.

Python int

Python can hold signed integers

```
b = 10
```

```
b
```

Output 10


Integers can be of any length, it is only limited by the memory available.

Python int number types are of 3 types.

type() function

isinstance() function

Exponential numbers



1) type() function

```
b = 10
```

```
type(b)
```

```
output
```

```
<class 'int'> # this function tells about the type of the numeric data types.
```

This function can only takes 1 argument.



2) isinstance() function

isinstance(b,bool)

output

False # it shows false because b is “int”
number data type.

This function can takes 2 argument

b is first argument.

Bool is second argument.



3) Exponential numbers

exponential number can be written using the letter 'e'.

```
print(3e5)
```

```
300000.0 # Remember that this is power of 10
```

Floating-Point Numbers

The float type in python also called Floating-Point Numbers.
The float number data type are specified with decimal point.

6.5

```
print(type(6.5))  
<class "float">
```

An int cannot store the value of the mathematical constant pi, but a float can.

Complex numbers

Complex numbers are specified as $\langle \text{real part} \rangle + \langle \text{imaginary part} \rangle j$.

It is represented as $a + bj$.

The real part of the number is a , and the imaginary part is b .

Complex numbers are not used much in Python programming.

```
c=4+3j
```

```
c
```

```
output
```

```
4+3j
```



```
In [ ]: # python number data types
```

```
In [34]: #integers  
b = 10
```

```
In [35]: b
```

```
Out[35]: 10
```

```
In [36]: int = 255  
int
```

```
Out[36]: 255
```

```
In [37]: #type()  
print(type(int))  
  
<class 'int'>
```

```
In [43]: #isinstance() function  
isinstance(b,bool)
```

```
Out[43]: False
```

```
In [ ]:
```




```
In [ ]: # python number data types
```

```
In [44]: # floating-point Numbers
6.5
bob = 6.5 #lets store in a variable name bob
bob
```

```
Out[44]: 6.5
```

```
In [ ]: # An int cannot store pi value , but float can do that.
```

```
In [47]: from math import pi
pi
```

```
Out[47]: 3.141592653589793
```

```
In [48]: type(pi)
```

```
Out[48]: float
```

```
In [52]: isinstance(pi,bool)
```

```
Out[52]: False
```

```
In [53]: isinstance(bob,float)
```

```
Out[53]: True
```



```
In [ ]: # python number data types
```

```
In [54]: #Complex numbers  
# represented by a+bj  
c = 1+2j  
c
```

```
Out[54]: (1+2j)
```

```
In [55]: type(c)
```

```
Out[55]: complex
```

```
In [56]: isinstance(c,bool)
```

```
Out[56]: False
```

```
In [57]: 2+2j
```

```
Out[57]: (2+2j)
```

```
In [58]: print(type(2+2j))
```

```
<class 'complex'>
```

```
In [59]: isinstance(2+2j,complex)
```

```
Out[59]: True
```

Number System

Number system	prefix
Binary	0b or 0B
Octal	0o or 0O
Hexadecimal	0x or 0X



1) **Binary**

use the prefix 0b or 0B to write binary number

```
print(0b111)
```

output- **7**

2) **Octal**

use the prefix 0o or 0O

```
print(0o11)
```

output - **9**

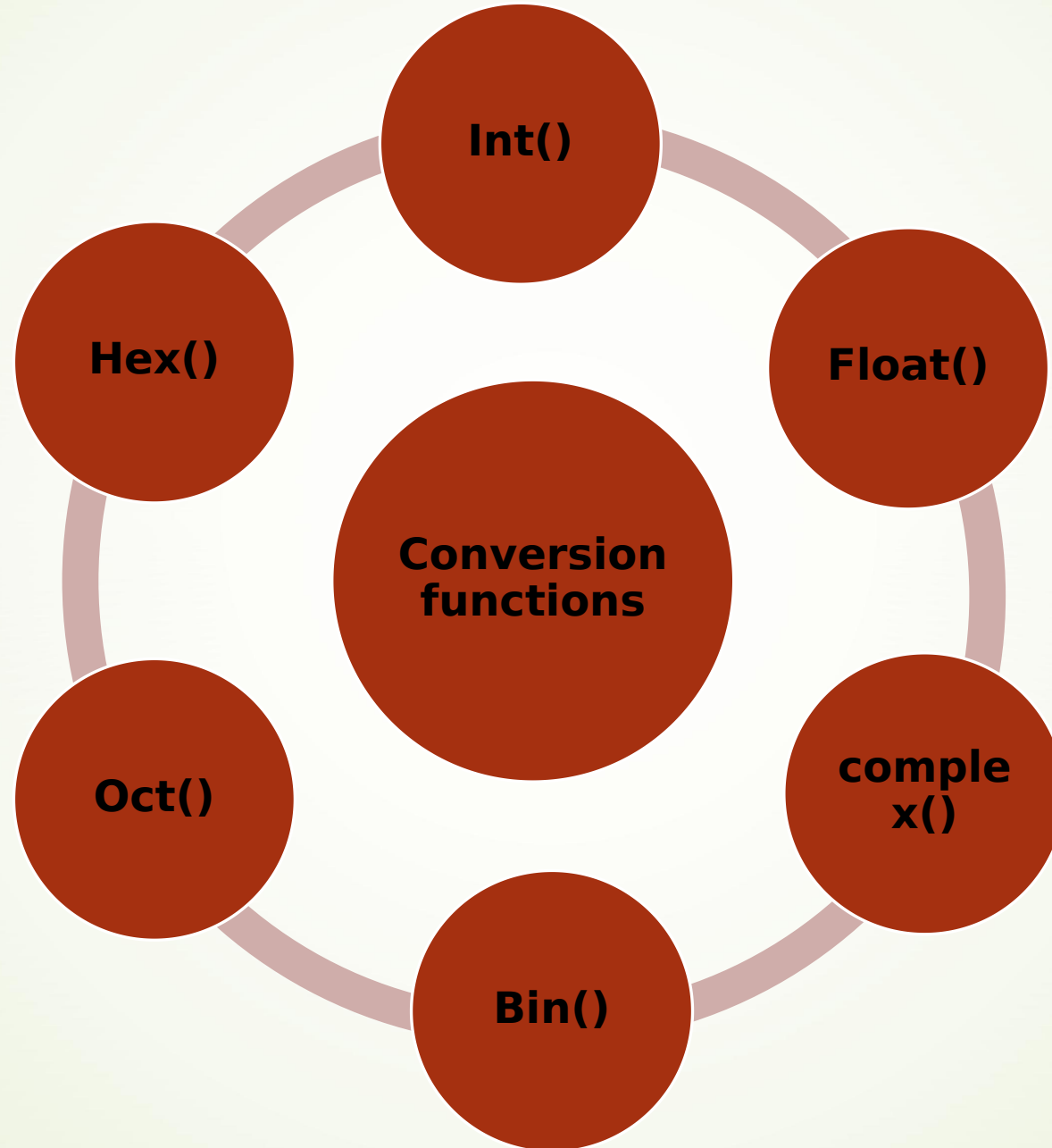
3) **Hexadecimal**

use the prefix 0x or 0X

```
print(0xff)
```

output - **255**

Python Conversion Functions.



Few examples of conversion function.

Example 1: Converting integer to float

```
a = 25  
float(25)  
output - 25.0
```

Example 2: Converting float to integer

```
b = 2.5  
int(b)  
output - 2
```



```
In [ ]: # python conversion function.
```

```
In [1]: #int to float
jhon = 2567
jhon
```

```
Out[1]: 2567
```

```
In [8]: float(jhon)
```

```
Out[8]: 2567.0
```

```
In [9]: 24
```

```
Out[9]: 24
```

```
In [10]: float(24)
```

```
Out[10]: 24.0
```

```
In [2]: # float to int
abc = 5.5
print(int(abc))
```

```
5
```

```
In [3]: 2.4
```

```
Out[3]: 2.4
```

```
In [4]: int(2.4)
```

```
Out[4]: 2
```

```
In [11]: int(8.2)
```

```
Out[11]: 8
```



```
In [ ]: # python conversion function.
```

```
In [12]: # The complex() function converts another numeric type into a complex number.  
complex(5)
```

```
Out[12]: (5+0j)
```

```
In [13]: complex(2.7)
```

```
Out[13]: (2.7+0j)
```

```
In [14]: moon = 24  
complex(moon)
```

```
Out[14]: (24+0j)
```

```
In [15]: f = 678.9  
complex(f)
```

```
Out[15]: (678.9+0j)
```

```
In [ ]:
```




What makes Python so powerful?

Apart from the constructs that Python provides, you can use the PyPI (Python Package Index).

It is a repository of third-party Python modules and you can install it using a program called pip.

Run the following command in Command Prompt:

```
pip install library_name.
```

Python variables

- A Python variable is a reserved memory location to store values.
- Every value in python has a data type.
- Variables can be declared by any name or even alphabets like a, aa, abc, etc.

How to Declare and use a Variable

let see an example. We will declare variable "a" and print it

```
a = 200
```

```
Print a
```

Re-declare a Variable

```
s = 0    # here we initialized variable
```

```
Print(s)
```

```
s = 'workshop2019'    # re-declaring the variable works
```

Indexes

§ Characters in a string are numbered with indexes starting at 0:

Example:

Name = "P. Diddy"

index	0	1	2	3	4	5	6	7
character	P	.		D	i	d	d	y

- Accessing an individual character of a string:
- VariableName [index]
- Example:
- Print name, " starts with" , name[0]
- Output:
- P. Diddy starts with P

Python strings

- In Python everything is object and string are an object too. A sequence of text characters in a program.
- Strings start and end with quotation mark " or apostrophe ' characters.

For example: Name = "ryan"

Age = "19"

pi = "3.14"

Accessing Values in Strings

```
var1 = 'Hello World!'
var2 = "Python Programming"
print "var1[0]: ", var1[0]
print "var2[1:5]: ", var2[1:5]
```

Output

```
var1[0]: H
var2[1:5]: ytho
```

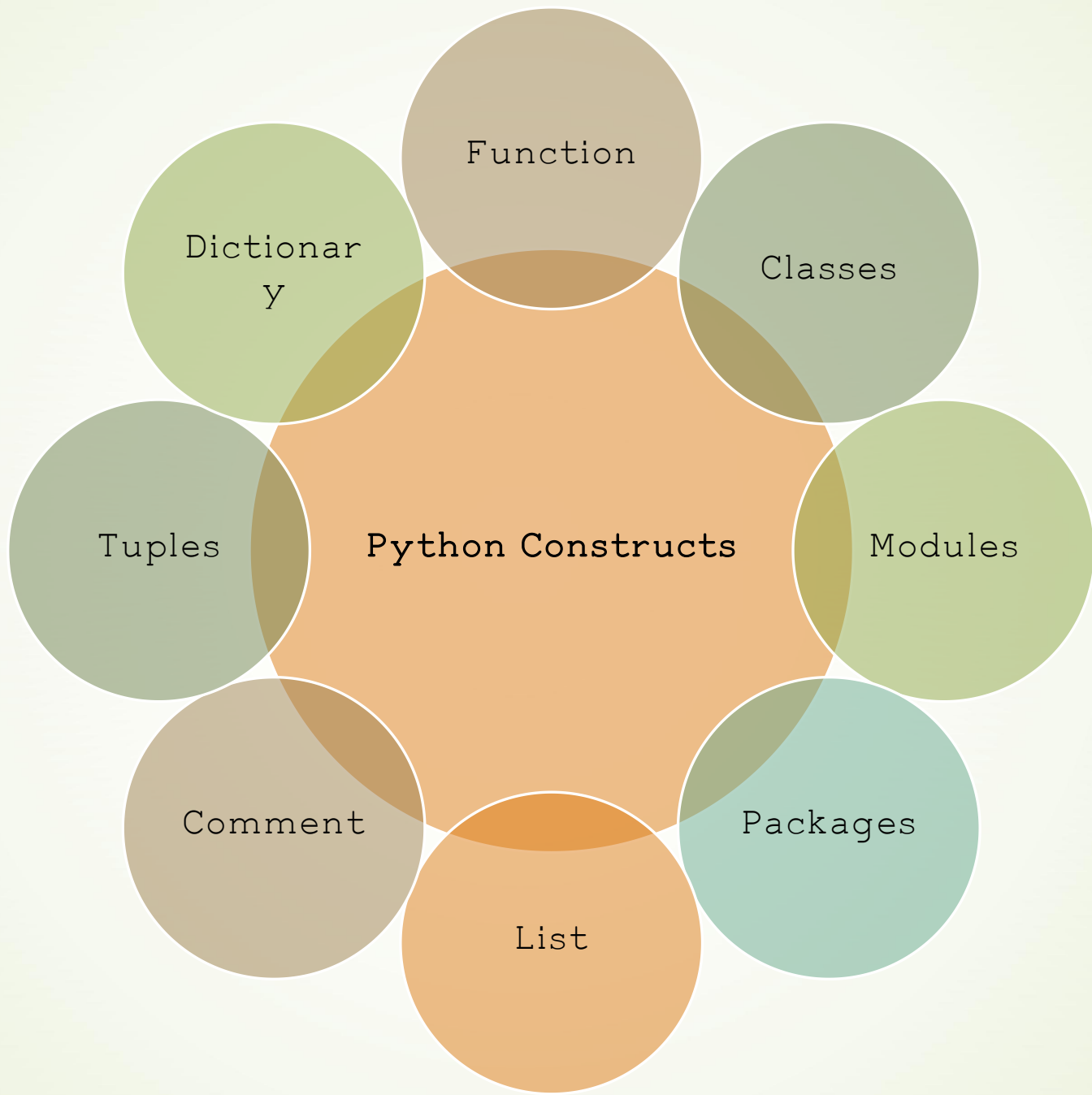


Python Architecture

Parser : It uses the source code to generate an abstract syntax tree.

Compiler: It turns the abstract syntax tree into Python bytecode.

Interpreter: It executes the code line by line in a REPL (Read-Evaluate-Print-Loop) fashion



Python Constructs

Functions: A [function in Python](#) is a collection of statements grouped under a name. You can use it whenever you want to execute all those statements at a time. You can call it wherever you want and as many times as you want in a program. A function may return a value.

Classes - As we discussed earlier, Python is an object-oriented language. It supports classes and objects. A class is an abstract data type.

Modules- A Python module is a collection of related classes and functions.

Packages- [Python package](#) is a collection of related modules. You can either import a package or create your own



List

You can think of a list as a collection of values. Declared in the CSV (Comma-Separated Values) format and delimit using square brackets.

```
life = ['love', 'wisdom', 'anxiety'];  
arity = [1,2,3];
```

A list may also contain elements of different types, and the indexing begins at 0.

```
person = ['firstname', 21];  
print(person[1])
```

Output
21



Tuple

A tuple is like a list, but it is immutable (you cannot change its values).

```
1.pizza = ('base', 'sauce', 'cheese', 'mushroom');
```

```
2.pizza[3] = 'jalapeno'
```

This raises a TypeError.



Dictionary

A dictionary is a collection of key-value pairs. Declare it using curly braces, and commas to separate key-value pairs. Also, separate values from keys using a colon (:).

```
student = {'Name': 'Abc', 'Age': 21}
```

```
print(student['Age'])
```

Output: 21



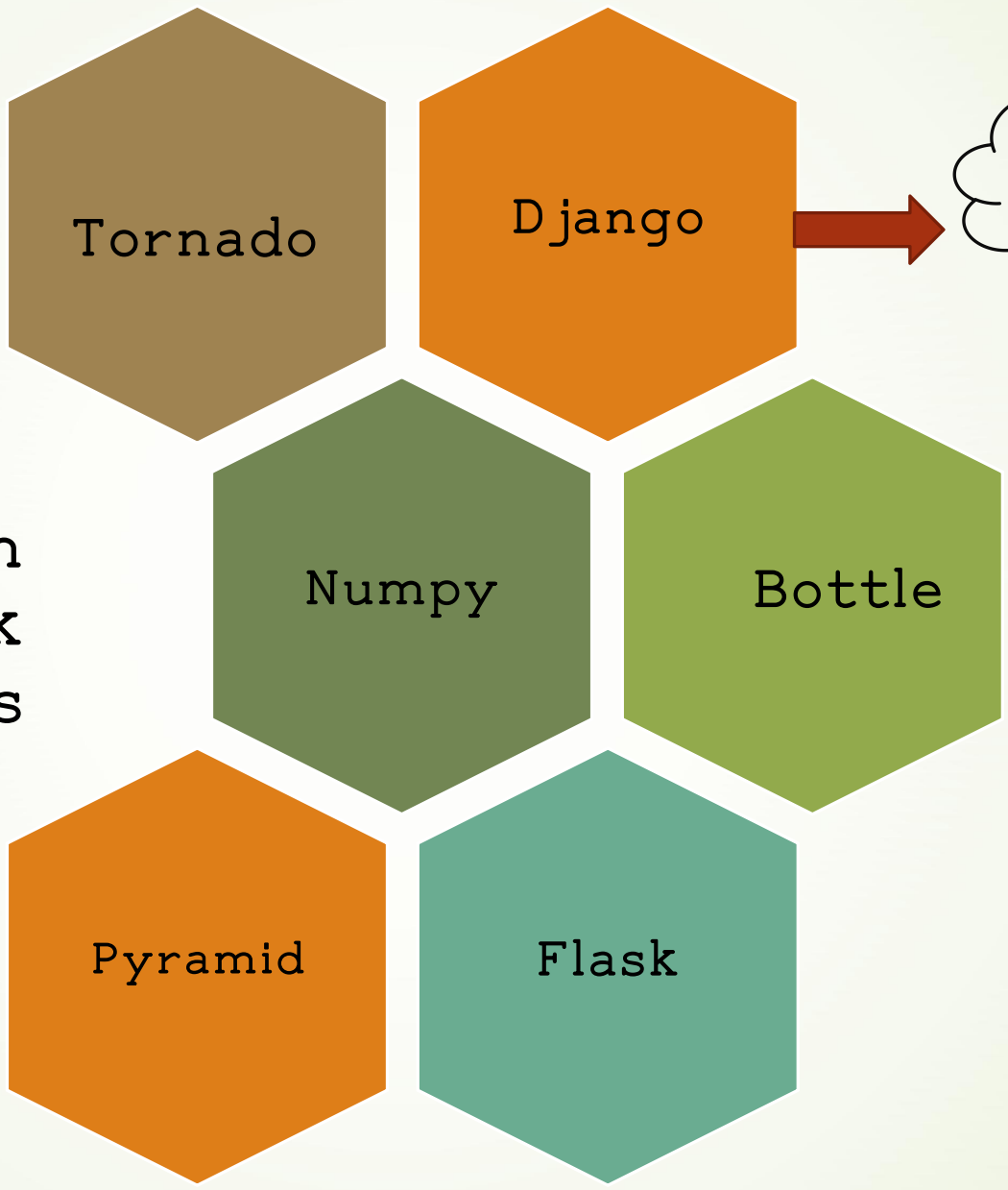
Comments and Docstrings

Declare comments using an octothorpe (#). Also, docstrings are documentation strings that help explain the code.

```
#This is a comment  
"""
```

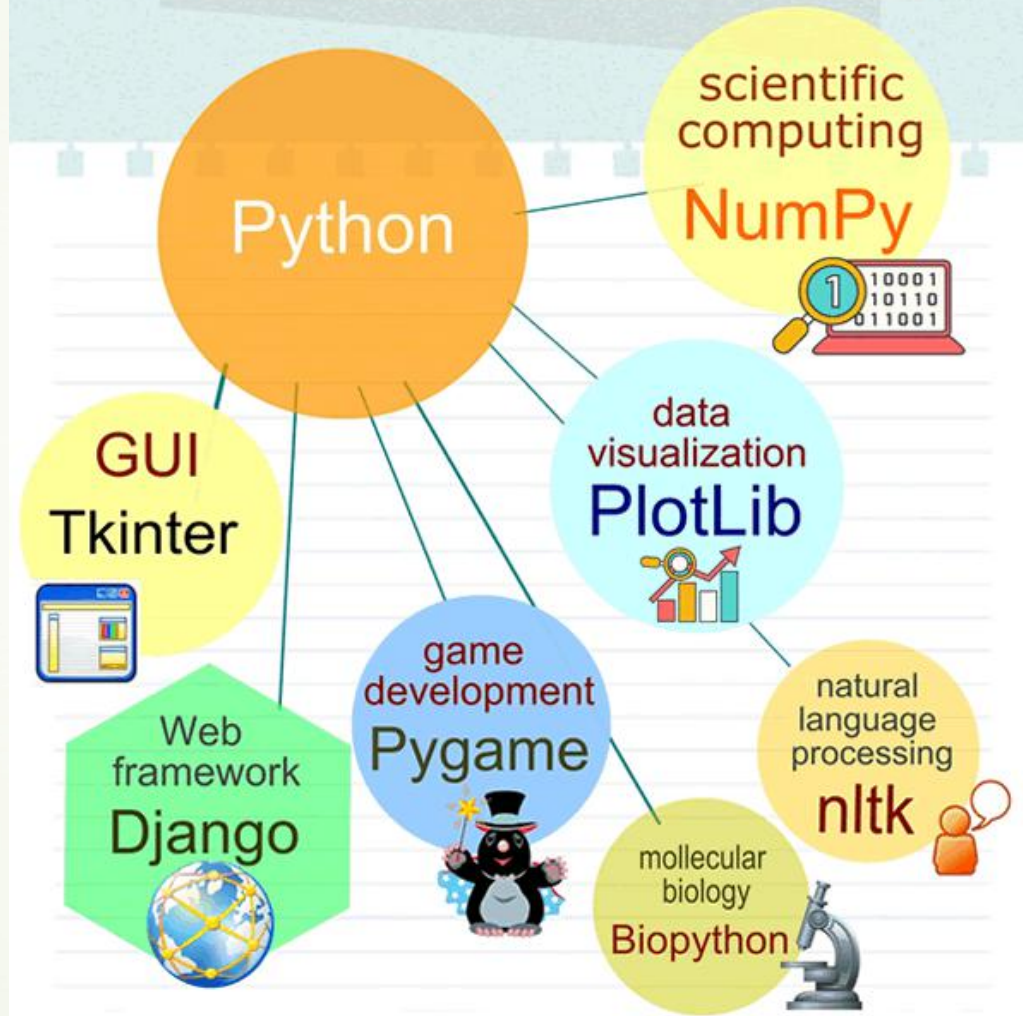
```
This is a docstring  
"""
```

Python
frameworks



Instagram, mozilla

Where Python is used



File Extensions in Python

- ∅ .py -The normal extension for a Python source file
- ∅ .pyc- The compiled bytecode
- ∅ .pyd- A Windows DLL file
- ∅ .pyo- A file created with optimizations
- ∅ .pyw- A Python script for Windows
- ∅ .pyz- A Python script archive
- ∅ .ipynb – jupyter notebook python file extension



Built in functions in python



`len()`

`max()`

`min()`

`sorted()`

`any()`

`all()`

`list()`

`sum()`



```
In [47]: # python list tutorial in detail
```

```
In [48]: colors = ['orange','black','white']# how to create python list
```

```
In [49]: colors
```

```
Out[49]: ['orange', 'black', 'white']
```

```
In [51]: # python list can hold different types of values.  
months = ['january','february','march',1,2,4,8.5]
```

```
In [52]: months
```

```
Out[52]: ['january', 'february', 'march', 1, 2, 4, 8.5]
```

```
In [54]: # a list may contain or have a python list  
days = [['monday'],['tuesday'],['wednesday'],'thursday','friday']
```

```
In [55]: days
```

```
Out[55]: [['monday'], ['tuesday'], ['wednesday'], 'thursday', 'friday']
```

```
In [56]: print(type(days))  
<class 'list'>
```

```
In [57]: print(type(days[0]))  
<class 'list'>
```

```
In [59]: # a list may also contain tuples.  
cities = [ ('delhi','mumbai'),'jammu','pune','chennai','kerala']
```

```
In [60]: cities
```




```
In [59]: # a list may also contain tuples.  
cities = [ ('delhi','mumbai'),'jammu','pune','chennai','kerala']
```

```
In [60]: cities
```

```
Out[60]: [('delhi', 'mumbai'), 'jammu', 'pune', 'chennai', 'kerala']
```

```
In [61]: print(type(cities[0]))
```

```
<class 'tuple'>
```

```
In [63]: abc = [['agra'], ('delhi','mumbai'),'amar','raja'] # list may also contain list & tuple at a same time.
```

```
In [64]: abc
```

```
Out[64]: [['agra'], ('delhi', 'mumbai'), 'amar', 'raja']
```

```
In [65]: type(abc)
```

```
Out[65]: list
```

```
In [69]: type(abc[1])
```

```
Out[69]: tuple
```

```
In [70]: cities
```

```
Out[70]: [('delhi', 'mumbai'), 'jammu', 'pune', 'chennai', 'kerala']
```

```
In [71]: cities[0][0] ="lucknow"
```

```
-----  
TypeError                                 Traceback (most recent call last)  
<ipython-input-71-b561f11a04d1> in <module>  
----> 1 cities[0][0] ="lucknow"  
  
TypeError: 'tuple' object does not support item assignment
```



📄
+
🔍
📄
📄
⬆️
⬆️
▶️ Run
■
↺
▶️
Code
▾
🗨️

TypeError: 'tuple' object does not support item assignment

```
In [72]: # how to access python list?
# to access a python list as a whole, all you need is its name
cities
```

```
Out[72]: [('delhi', 'mumbai'), 'jammu', 'pune', 'chennai', 'kerala']
```

```
In [73]: print(days) # you can put it in a print statement.

[['monday'], ['tuesday'], ['wednesday'], 'thursday', 'friday']
```

```
In [74]: # to access a single element , use its index in square brackets after the list's name
# indexing begins at 0
cities[0]
```

```
Out[74]: ('delhi', 'mumbai')
```

```
In [75]: # an index cannot be a float value
cities[1.0]
```

```
-----
TypeError                                Traceback (most recent call last)
<ipython-input-75-1e31109512e7> in <module>
      1 # an index cannot be a float value
----> 2 cities[1.0]
```

TypeError: list indices must be integers or slices, not float

In []: |

TypeError: list indices must be integers or slices, not float

```
In [14]: #Slicing a Python List.  
#When you want only a part of a Python list, you can use the slicing operator []  
indices=['zero','one','two','three','four','five']  
indices[2:4] #This returns items from index 2 to index 4-1 (i.e., 3)
```

```
Out[14]: ['two', 'three']
```

```
In [15]: indices[:4] #This returns items from the beginning of the list to index 3.
```

```
Out[15]: ['zero', 'one', 'two', 'three']
```

```
In [16]: indices[4:]
```

```
Out[16]: ['four', 'five']
```

```
In [17]: indices[:] #This returns the whole list
```

```
Out[17]: ['zero', 'one', 'two', 'three', 'four', 'five']
```

```
In [18]: #Negative indices- The indices we mention can be negative as well. A negative index means traversal from the end of the list  
indices[:-2] #It returns items from the item at index 1 to two items from the end.
```

```
Out[18]: ['zero', 'one', 'two', 'three']
```

```
In [19]: indices[1:-2] #It returns items from the item at index 1 to two items from the end.
```

```
Out[19]: ['one', 'two', 'three']
```

```
In [20]: #Reassigning a Python List (Mutable)  
colors=['red','green','blue']  
colors
```

```
Out[20]: ['red', 'green', 'blue']
```

```
In [21]: #Reassigning the whole Python List  
colors=['caramel','gold','silver','occur']  
colors
```

```
Out[21]: ['caramel', 'gold', 'silver', 'occur']
```

```
In [22]: #Reassigning a few elements  
colors[2:]=['bronze','silver']  
colors
```

```
Out[22]: ['caramel', 'gold', 'bronze', 'silver']
```

```
In [23]:  
colors=['caramel','gold','silver','occur']  
colors[2:3]=['bronze','silver']  
colors
```

```
Out[23]: ['caramel', 'gold', 'bronze', 'silver', 'occur']
```

```
In [24]: colors[2:2]=['occur']  
colors
```

```
Out[24]: ['caramel', 'gold', 'occur', 'bronze', 'silver', 'occur']
```

```
In [25]: #Reassigning a single element  
colors=['caramel','gold','silver','occur']  
colors[3]='bronze'  
colors
```

```
Out[25]: ['caramel', 'gold', 'silver', 'bronze']
```

```
In [26]: colors[4]='holographic'
```

```
-----  
IndexError                                Traceback (most recent call last)  
<ipython-input-26-e62183d1865d> in <module>  
----> 1 colors[4]='holographic'  
  
IndexError: list assignment index out of range
```

```
In [27]: # How can we Delete a Python List?
```

```
In [27]: # How can we Delete a Python List?
```

```
del colors  
colors
```

```
-----  
NameError                                Traceback (most recent call last)
```

```
<ipython-input-27-48b4ea483126> in <module>
```

```
1 # How can we Delete a Python List?  
2 del colors  
----> 3 colors
```

```
NameError: name 'colors' is not defined
```

```
In [28]: # Deleting a few elements
```

```
colors=['caramel','gold','silver','bronze','holographic']  
del colors[2:4]  
colors
```

```
Out[28]: ['caramel', 'gold', 'holographic']
```

```
In [29]: # Built-in List Functions
```

```
#There are some built-in functions in Python that you can use on python lists  
#len()  
len(days)
```

```
Out[29]: 7
```

```
In [30]: days
```

```
Out[30]: ['Monday', 'Tuesday', 'Wednesday', 4, 5, 6, 7.0]
```

```
In [31]: # max()
```

```
max(days)
```

In [30]: days

Out[30]: ['Monday', 'Tuesday', 'Wednesday', 4, 5, 6, 7.0]

In [31]: # max()
max(days)

```
-----  
TypeError                                 Traceback (most recent call last)  
<ipython-input-31-eb2153a5b228> in <module>  
    1 # max()  
----> 2 max(days)  
  
TypeError: '>' not supported between instances of 'int' and 'str'
```

In [36]: num =[10,12,7,8,0]

In [37]: len(num)# total length of a list

Out[37]: 5

In [38]: max(num)

Out[38]: 12

In [39]: min(num)

Out[39]: 0

In [40]: sum(num)

Out[40]: 37

In [41]: #sorted()
a = [3,1,2] # It returns a sorted version of the list, but does not change the original one.
sorted(a)

Out[41]: [1, 2, 3]



```
In [ ]: days
```

```
In [ ]: # max()  
max(days)
```

```
In [ ]: num = [10,12,7,8,0]
```

```
In [ ]: len(num) # total length of a list
```

```
In [ ]: max(num)
```

```
In [ ]: min(num)
```

```
In [ ]: sum(num)
```

```
In [ ]: #sorted()  
a = [3,1,2] # It returns a sorted version of the list, but does not change the original one.  
sorted(a)
```

```
In [ ]: a
```

```
In [ ]: #If the Python List members are strings, it sorts them according to their ASCII values  
sorted(['hello', 'hell', 'Hello'])
```

```
In [ ]: #any()  
any(['', '', '1']) #It returns True if even one item in the Python List has a True value.
```

```
In [ ]: #all()  
all(['', '', '1']) # It returns True if all items in the list have a True value
```

Python list- built-in Methods

append()

insert()

remove()

pop()

clear()

index()

count()

sort()

reverse()


```
In [ ]: # Built-in methods
```

```
In [17]: a = [23,45,10,34,2,1,25]
```

```
In [18]: a
```

```
Out[18]: [23, 45, 10, 34, 2, 1, 25]
```

```
In [19]: # append()  
a.append(66)
```

```
In [20]: a
```

```
Out[20]: [23, 45, 10, 34, 2, 1, 25, 66]
```

```
In [22]: #insert()  
a.insert(1,11)
```

```
In [23]: a
```

```
Out[23]: [23, 11, 45, 10, 34, 2, 1, 25, 66]
```

```
In [24]: #remove()  
a.remove(23)
```

```
In [25]: a
```

```
Out[25]: [11, 45, 10, 34, 2, 1, 25, 66]
```

```
In [26]: #pop()  
a.pop(0)
```

```
Out[26]: 11
```

```
In [27]: a
```



```
In [34]: a.remove(23)
```

```
In [35]: a
```

```
Out[35]: [11, 45, 10, 34, 2, 1, 25, 66]
```

```
In [36]: a.pop(0)
```

```
Out[36]: 11
```

```
In [37]: a
```

```
Out[37]: [45, 10, 34, 2, 1, 25, 66]
```

```
In [38]: #index()  
a.index(2)
```

```
Out[38]: 3
```

```
In [40]: #count()  
a.count(10)
```

```
Out[40]: 1
```

```
In [41]: #sort()  
a.sort()
```

```
In [42]: a
```

```
Out[42]: [1, 2, 10, 25, 34, 45, 66]
```

```
In [ ]: |
```



File Edit View Insert Cell Kernel Widgets Help

Trusted Python 3



```
In [38]: #index()
a.index(2)
```

```
Out[38]: 3
```

```
In [40]: #count()
a.count(10)
```

```
Out[40]: 1
```

```
In [41]: #sort()
a.sort()
```

```
In [42]: a
```

```
Out[42]: [1, 2, 10, 25, 34, 45, 66]
```

```
In [43]: #reverse()
a.reverse()
```

```
In [44]: a
```

```
Out[44]: [66, 45, 34, 25, 10, 2, 1]
```

```
In [45]: #clear()
a.clear()
```

```
In [46]: a
```

```
Out[46]: []
```

```
In [ ]:
```



```
In [ ]: days
```

```
In [ ]: # max()
max(days)
```

```
In [ ]: num =[10,12,7,8,0]
```

```
In [ ]: len(num)# total length of a list
```

```
In [ ]: max(num)
```

```
In [ ]: min(num)
```

```
In [ ]: sum(num)
```

```
In [ ]: #sorted()
a = [3,1,2] # It returns a sorted version of the list, but does not change the original one.
sorted(a)
```

```
In [ ]: a
```

```
In [ ]: #If the Python list members are strings, it sorts them according to their ASCII values
sorted(['hello','hell','Hello'])
```

```
In [ ]: #any()
any(['', '', '1']) #It returns True if even one item in the Python List has a True value.
```

```
In [ ]: #len()
```



file handling

file is a named location to store information for latter use that are managed by os.

In python file handling is quite simple than in other languages.

To perform file operation such as open a file, read,write & close a file
in python there are in-built functions is present.

modes:

“r” - It's used only for read the file .

“w” - used for write and edit the file.

“a” - to add new data at the end of the file.

“r+” - Special read and write mode, when working with a file we can perform the both function.



```
In [1]: import os
os.chdir('/home/sheeba/Desktop')
os.getcwd()
```

```
Out[1]: '/home/sheeba/Desktop'
```

```
In [3]: # open function :- to open a file in python it takes 2 parameters
# 1 is the filename
# 2 is the mode.

f = open("od1.txt",'r')
# how to fetch data.
# below function is used to fetch the data from file.
print(f.read())
```

```
ABILIFY
ABLYSINOL
ABRAXANE
ACTEMRA
ADAGEN
ARIKAYCE
ARRANON
ARZER
ATNATIV
ATRYN
AVASTIN
AZEDRA ULTRATRACE
BABYBIG
BAT
Bavencio
```



```
In [4]: f1= open ("coco","r")
# how to print only 1st line of file.
print(f1.readline())
```

hello

```
In [5]: # how to print 2nd line of the file.
print(f1.readline())
```

everyone

```
In [6]: # how to print 3rd line of the file.
print(f1.readline())
```

my name is jhon

```
In [7]: # how to write a data in a file.
f2 = open("new","w")
f2.write('mobile')
```

Out[7]: 6

```
In [8]: f2 = open("new","r")
```

```
In [9]: print(f2.read())
```



```
In [7]: # how to write a data in a file.  
f2 = open("new", "w")  
f2.write('mobile')
```

```
Out[7]: 6
```

```
In [8]: f2 = open("new", "r")
```

```
In [9]: print(f2.read())
```

```
mobile
```

```
In [10]: # how to append a file.  
f2 = open("new", "a")
```

```
In [11]: f2.write('charger11\n')
```

```
Out[11]: 10
```

```
In [12]: f2 = open("new", "r")
```

```
In [13]: print(f2.read())
```

```
mobilecharger11
```


how to copy a file content from one file to another

```
Edit View Insert Cell Kernel Widgets Help Trusted python3.7
+ ✂ 📄 📄 ⬆ ⬇ ▶ Run ■ ↺ ▶ Code 🖨

In [15]: # how to copy a file content from one file to another
         oldfile = open("oldfile.txt",'r')

In [16]: newfile = open ('newfile','w')

In [17]: newfile.write(oldfile.read())

Out[17]: 313

In [18]: newfile = open ('newfile','r')

In [19]: print(newfile.read())
Name
Zoledronic Acid
Zaltoprofen
Zafirlukast
Vanillin
Valsartan
Troxipide
Tolcapone
Tazarotene
Tamibarotene
Sofosbuvir (PSI-7977, GS-7977)
Sertaconazole nitrate
Ozagrel hydrochloride
Ospemifene
```



Functions:an essential part of the Python programming language

A function is a block of code.

There is built-in function as well as user- defined function.

Defining any function is one time job and we can use or call that function multiple time whenever needed it saves a lot of time.



```
In [18]: # functions.  
def greet():# define a function  
    print('hello')  
    print('good morning')
```

```
greet()# calling a function
```

```
hello  
good morning
```

```
In [19]: def addition(a,b):  
         c = a+b  
         print( a+b)  
addition(5,5)
```

```
10
```

```
In [20]: addition(115,5)
```

```
120
```

```
In [21]: greet()
```

```
hello  
good morning
```

```
In [22]: addition(1,5)
```

```
6
```



Code



```
hello  
good morning
```

```
In [5]: addition(1,5)
```

```
6
```

```
In [6]: # when function return something
```

```
def addition(a,b):  
    c = a+b  
    return c  
add = addition(5,5)  
print(add)
```

```
10
```

```
In [7]: # loop
```

```
# while loop  
i = 1
```



Loops

Loops are used in programming languages to repeat a specific block of code.

Python provides us the 2 types of looping system are:

1. while loop
2. for loop

While loop in python is used to execute multiple statement or codes repeatedly until the given condition is true.



```
In [7]: # loop
# while loop
i = 1
while i<=5:
    print("bioinformatics")
    i = i+1
```

```
bioinformatics
bioinformatics
bioinformatics
bioinformatics
bioinformatics
```

```
In [8]: # nested loop
i = 1
while i<=5:
    print("sun",end=" ")
    j = 1
    while j<=4:
        print("shine",end=" ")
        j = j+1
    i = i+1
    print()
```

```
sun shine shine shine shine
sun shine shine shine shine
sun shine shine shine shine
sun shine shine shine shine
sun shine shine shine shine
```

```
In [ ]: i = 3
```



```
In [9]: i = 3
while i >= 1:
    print('india')
    i = i - 1
```

```
india
india
india
```

```
In [10]: # for loop
a = [2, 3.5, "string"]
for i in a:
    print(i)
```

```
2
3.5
string
```

```
In [11]: for i in {2, 3, 4.2}:
    print(i)
```

```
2
3
4.2
```

```
In [ ]: x = "string"
for i in x:
    print(i)
```

```
In [ ]: for i in range(6):
    print(i)
```



```
In [13]: for i in range(6):  
         print(i)
```

```
0  
1  
2  
3  
4  
5
```

```
In [14]: for i in range(10,20,1):  
         print(i)
```

```
10  
11  
12  
13  
14  
15  
16  
17  
18  
19
```

```
In [15]: for x in range(20,10,-1):  
         print(x)
```

```
20  
19  
18  
17  
16  
15  
14  
13  
12  
11
```

```
In [16]: for i in range(2,22,2):  
         print(i)
```

```
2
```




11

```
In [16]: for i in range(2,22,2):  
         print(i)
```

```
2  
4  
6  
8  
10  
12  
14  
16  
18  
20
```

```
In [17]: for i in "string":  
         print(i)
```

```
s  
t  
r  
i  
n  
g
```

```
In [18]: for i in range(10,30):# print the numbers which is not divisible by 2.  
         if i%2!=0:           # if condition inside for loop  
             print(i)
```

```
11  
13  
15  
17  
19  
21  
23  
25  
27  
29
```

```
In [ ]: # conditional statement in python.
```



conditional statement in python

(Decision Making)

python programming languages provide the following conditional statement :-

if statement

if-else statement

if-elif-else statement

nested if-else statement



```
In [ ]: # conditional statement in python.
# if condition.
a = 15
b = 100

if b > a:
    print("b is greater than a")
```

```
In [ ]: #elif condition.
a = 2
b = 2
if b > a:
    print("b is greater than a")
elif a == b:
    print("a and b are equal")
```

```
In [ ]: # else condition.
a = 44
b = 6
if b > a:
    print("b is greater than a")
elif a == b:
    print("a and b are equal")
else:
    print("a is greater than b")
```

```
In [ ]: # how to use in function.
def main():
    x,y =5,6
    if (x<y):
        s = "x is smaller than y"
        print(s)
if __name__ == "__main__":
    main()
```

```
In [ ]: def main():
x,y = 7,6
```



```
In [ ]: # else condition.
a = 44
b = 6
if b > a:
    print("b is greater than a")
elif a == b:
    print("a and b are equal")
else:
    print("a is greater than b")
```

```
In [ ]: # how to use in function.
def main():
    x,y =5,6
    if (x<y):
        s = "x is smaller than y"
        print(s)
if __name__ == "__main__":
    main()
```

```
In [ ]: def main():
x,y =7,6
if (x<y):
    s = "x is smaller than y"
else:
    s ="x is larger than y"
print(s)
if __name__ == "__main__":
    main()
```

```
In [ ]: # THANKS FOR YOUR ATTENTION #
```

Bioinformatic Team

THANK YOU

