Python programming language

Introduction and installation

- **Python** is a general-purpose, dynamic, high-level, and interpreted programming language.
- Guido van Rossum invented python in 1991 at CWI in Netherland.
- Python supports multiple programming patterns, including object-oriented, imperative, and functional or procedural programming styles.

Why learn Python?

Python provides many useful features to the programmer. These features make it most popular and widely used language. We have listed below few-essential feature of Python.

- Easy to use and Learn
- Expressive Language
- Interpreted Language
- Object-Oriented Language
- Open Source Language
- Extensible
- Learn Standard Library
- GUI Programming Support
- Integrated
- Embeddable
- Dynamic Memory Allocation
- Wide Range of Libraries and Frameworks

Where is Python used?

Python is a general-purpose, popular programming language and it is used in almost every technical field. The various areas of Python use are given below.

- Data Science
- Date Mining
- Desktop Applications
- Console-based Applications
- Mobile Applications

- Software Development
- Artificial Intelligence
- Web Applications
- Enterprise Applications
- 3D CAD Applications
- Machine Learning
- Computer Vision or Image Processing Applications.
- Speech Recognitions
- And more

Basic syntax

- >>> **help ( )** … help section is where you can find all python functions, symbols ETC, you also find directly specific topic if you already knew your topic example let us find list… **>>>help ('lists').**
- Running: Before you running your code save it with suing python extension **(.PY)** example: **test.py**
- To run the code right click on the screen then click run
- Debugging: first add break – point to point where you'll trace from then right click on the screen > click debug > click file name
- Python comments start with a **#,** example:
  #this line is comment line one
  #this line is comment line two
  #this line is comment line three
- If you want to write block comments then use this
  " " "
  This comment section starting from those three quotations
  Anything in here will be comment
  To close this comment section you have to write another three quotations
  " " "
- Print ("""
  You can write anything you want here
  ****
  ***
  **
  *
  Python will print without change """)

Simple mathematics

- >>>**2+5** #you can directly add, subtract, divide and multiply numbers
- >>>**5//2** #this will give you only integer answer so 5//2 is 2 instead of 2.5
- >>>**2******3** #this means two power three which is 8 (** =power)
- >>>**10%3** #this will give you the remainder which is 1
- >>> **'moha'+'adan'** #adding two string which the answer will be moha adan
- >>>**3*' moha'** #multiplying strings output… moha moha moha
- >>> **import math** #adding all mathematical functions like pow, sqrt ETC after you add math module you can use all of them example
    - >>>**math.sqrt(25)** #finding square root of 5
    - >>>**x = math.pow(5,2)** #finding 5 power 2, x =25
    - >>>**math.pi** #you can directly find constants pi = 3.1415
- >>> **from math import sqrt,pow** #if you want to access only square root and power from all mathematical function you can just access like this.

- >>>**import math as M** #now you can use M as math example
    - >>>**M.sqrt(25)** #finding square root of 5
    - >>>**x = M.pow(5,2)** #finding 5 power 2, x =25

Operation and number system conversion

Operators

- Arithmetic operators: + , - , * , /
- Assignment operators: = , += , *= ,ETC
- Relational operators: > , <, >=, <=,= =, =!
- Logical operators: AND, OR, NOT
- Unary operations: ++, --, ETC

Number system conversion

- Decimal from 0 to 9 which is base 10 $(\ )_{10}$
- Binary 0,1 which is base 2 $(\ )_2$
- Octal from 0 to 7 which is base 8 $(\ )_8$
- Hexadecimal from 0 to 9 and A to F which is base 16 $(\ )_{16}$

Converting between binary and decimal

- >>>**bin(10)** #finding binary from decimal output … 0b1010
- >>>**0b1010** #finding decimal from binary output … 10

Converting between decimal and octal

- >>>**oct(10)** #finding octal from decimal output … 0o12

- >>>**0o12** #finding decimal from octal output …10

Converting between decimal to hexadecimal

- >>>**hex(10)** #finding hexadecimal from decimal output … 0xa
- >>>**0xa** #finding decimal from hexadecimal output … 10

Bitwise operators (1=true and 0=false)

- Compliment symbol (**~**): reverse in binary
- And symbol (**&**): if both are 1(true) then it's 1 else 0(false)
- Or symbol (|): if one of them is 1 then it's 1 else it's 0
- XOR symbol (^) : if both numbers are different then it's 1 (1:0=1 and 0:1=1) else its 0 (1:1=0 and 0:0=0)
- Left shift symbol (<<) adding zero's to the left side example
    - >>> **10<<2** #the output 40 and this is how
      First: find binary 10 = 1010.0000
      Second: add two 00 from left which is will be 101000.00 = 40
- Right shift symbol (>>) deleting two digits to the right
    - >>> 10>>2 #the output is 2 and this is how
      First: find binary 10 = 1010.00
      Second: move two digits to the right, which it will be 10.1000 = 2

Variables

- >>> **x=2** #in python you can directly declare variables writing data type
- >>> **a='mohamed** #declaring string as well
- >>>**x=5** #if we link from the first line the value of x is 5 now that means every time you have new value to x the value of x will be the last value you gave it.
- >>>**a,b = 1,2** #you can declare more variables in one line a=1 and b=2
- You can also print one word, two words, between from string example
    - >>> **a='mohamed'** #know tha Mohamed = 0 – 6 (counting words)
    - >>>**a[0]** #printing only first word which is **m**
    - >>>**a[3]** #printing the third word which is **a**
    - >>>**a[0:3]** #from 0 to 3 but the ending point isn't counted ...**moh**
    - >>>**a[:3]** #staring from 3 to the end ...**amed**
    - **From backword a= 'mohamed' = from -1 to -7 (counting from back starting -1 not -0)**
    - >>>**a[-1]** #printing only first word which is **d**
    - >>>**len(a)** #finding the length which is 7 (Mohamed = 7 words)
- Finding storage address and variable type
  - >>> **num = 5**
  - >>>**Id(num)** #the output will be the address of Num box like: 186481536

- If the data is same so the address will be same example
    - >>> **a =10**
    - >>> **b= a** #now both variables (a & b) have same data and same address
- You can find the data type of the variables example
    - >>>**type(a)** #finding the data type of variable and the output will <class 'int'>
- Swap two variables – which is very easy in python
    - >>> **a,b = 1,2** #a=1 and b=2 first: first, second: second
    - >>>**a,b = b,a** #so now a=b=2 and b=a=1

List, tuple and set

A. List

- >>>**A = [1,2,3,4,5]** #declaring integer list
- >>>**B=['moha','adam','dau']** #declaring string list
- >>>**C=[9,'september',29.5]** #you can declare integer, float & string in one list
- >>>**a[0]** #printing the value in position zero which is 1
- >>>**D [a,b]** #you can combine two lists (list one + list two)
- >>>**a.append(7)** #adding value 7 to list a so now a =123457
- >>>**a.insert(5,6)** #append = adds to the last and insert you can choose your own position now adding value 6 to fifth place and now a = 1,2,3,4,5,6,7
- >>>**a.remove(1)** #deleting value 1 from list a so now a = 2,3,4,5,6,7
- >>>**a.pop(2)** #you can delete by giving its position number now I am deleting value in position two which is 4 and now a = 2,3,5,6,7
- >>**a.pop( )** #if you don't give position it will delete the last value and now the list is 2,3,5,6
- >>>**a.extend ([0,1,10,11])** #you can add multiple value at a time and now the value of the list is a =0,1,10,11,2,3,5,6
- >>>**min(a)** #finding the minimum number in the list which is 0
- >>>**max(a)** #finding the maximum number in the list which is 11
- >>>**sum(a)** #finding the addition in of the list which is 38
- >>>**a.sort( )** #sorting list number = 0,1,2,3,5,6,10,11
- >>> **new_list = a.copy()** //copying list a into new_list
- >>> **a.reverse()** //it reverses lists

B. Tuple

- Tuple is a list without change …. This means you can't change values in this list, tuple is used fixed list this means when you don't want to change the value of the list.
- >>>**b=('moha',9,2.9)** #tuple uses **( )** , while list uses **[ ]**
- >>> print(**len**(b)) // finding the length of tuple b
- >>>print(b.**count**(9)) // counting how many 9 item in tuple b
- >>>print(b.**index**(9)) //finding the position (index) of 9 which is 1

C. Set

- In the set there is no proper sequence that means the input and the output can be different
- You can't print one value or search because you can't be sure it's position
- Set will not support duplicate values this means if you add two or more same values set will only save one value
- Set is more like normal list you can use pop, remove, and other function
- >>>**c={1,2,2,3,4,4,3,5,6}** # the output will be s = 1,2,3,4,5,6
  tuple uses **( )** , list uses **[ ]** , and set **{ }**

Data types in python

1. **None: null**
2. **Numeric**
   - ✓ Integer: 1, 2, 3…99…999 …
   - ✓ Float: 1.1, 1.2, 1.3,…99.99, 999.999 …
   - ✓ Complex: a+bj (python uses j instead of I, a+bi)
   - ✓ Bool: true and false

3. **Sequence**
   - ✓ List
   - ✓ Tuple
   - ✓ Set
   - ✓ String
   - ✓ Range: ranging value (1-10)

Example of range

- >>>**range(10)** #output …range(0,10) staring from 0 to 10

Converting to list

- >>> **list (range(10))** #output list from 0 to 9 …. 0123456789 the end value 10 is not counted
- >>>**list(range(2,9))** #you can also give staring point …2345678

Printing even numbers between 1 – 10

- >>>**list(range(2,10,2))** #the first 2 is staring point 10 is ending point and the last 2 is division 2 so the output … 2468 (the ending point is not counted)

4. **Dictionary**

- Giving unique keys every value while keys can be anything like number, name, character and this is the formula

  Variable (d) = {'key1':'value1', 'key2':'value2', 'key3':'value3'…}
- **>>>d: {'1':'moha' , '2':'adan' , '3':'daud'}**

- >>> **d.keys( )** #you can find dictionary d’s keys and the output will be like this dict_keys([‘1’, ‘2’, ‘3’])
- You can find dictionary’s data using unique keys example
- >>>**d.get(‘1’)** #finding key 1’s data output … moha

Adding new item

Dictionary_varibale[‘key_here’] = ‘value_here’

- If the key exist and you re-create then it will overwrite

Deleting item

There are several ways to delete item

1. **del** ***Dictionary_varibale[‘key_here’]***

2. ***Dictionary_varibale.pop[‘key_here’]***

3. ***Dictionary_varibale.popitem()*** *// it will delete the last item*

Checking if the key is in the dictionary or now

Try:

Print (dictionary _ variable[‘key _ here ‘]

Except:

Print (‘ Key is not exist in the dictionary ‘)

Copying dictionary

New_dictionary = old_dictionary.copy()

Update dictionary

Example

- >>> a { 1:’mohamed’, 2:’adan’} // dictionary a
- >>> a { 1:’mohamed’, 3:’daud’} // dictionary b
- >>>a.**update**(b) // updating dictionary a … when you are updating dictionary it some value has same key and value then it will take only one but if they has same key but different value then it will update to the last value of that key

User input in python

- >>>a = **input** (‘you can enter anything integer, float and string ‘)
- >>> b =**int** (**input**(‘now it only accepts integer ‘))

Add two numbers and print the result

- >>> x =**int** (**input**(‘Enter the first number: ‘))
- >>> y =**int** (**input**(‘Enter the second number: ‘))
- >>> z: x+y
- >>> print (‘ The answer is ‘ ,z)

Program only printing char

- **>>> a = input (‘Enter anything’) [0]**
- **>>>print(a)** #if you enter everything it will only print the first char, you can also print the second **[1],** and third **[2]** ,ETC you can last print the last word **[-1]**, you can control also like printing **[2:]** from the 2 to the end

Evaluating the output

- Program accepts something like 2+3-1*45/4 and gives answer
- >>> result = **eval** (**input** (‘Enter mathematical expression ‘))
- >>> Print (result)

If, if else, and nested if

- ***One condition == one probability output***

```
    a) If condition:
              Print (‘print statement’)
b) if condition:
          Print (‘print statement one’)
          Print (‘print statemen two’t)
                   ….
          Print (‘print statement N’)
```

  #as long as statement one, two….N uses same space under specific if, they will follow that condition

- ***One condition == two probability output***

```
If condition:
          Statements… probability output one
Else:
          Statements… probability output two
```

- ***N condition == N probability output***

```
If condition1:
          Statements
Elif condtion2:
          Statements
```

```
Elif conditon3:
        Statements
       …
Elif condtionN:
        statements

else:    // if all conditions are not match then else …
    statements;
```

While loop, and nested while loop

Simple while loop

- >>> x=1
- >>> **while** x<=5:
- >>> Print('hello')
- >> x=x+1

Nested while loop

- >>> x=1
- >>> y=1
- >>> **while** x<=5:
- \>>>print(' first while loop')
- >>> x=x+1
- >>>**while** y<=5:
-     \>>>print('second while loop')
-     >>> y=y+1

For loop

Example 1

- >>>x="moha"   #to print every element in x one by one, you can also use to print list elements one by one x=['moha',9,29.9]
- **For** i **in** x:
- Print(i)

Example 2

- >>> **For** i **in** range(11,21,1):   #11 initial point, 21 ending point, 1 increasing+
- >>> print (i)
  // output, 1 2 3 – 20 but every number in new line

Example 3

- >>> **For** i **in** range(20,10,-1): #20 initial point, 10 ending point, 1 decreasing-
- >>> print (i)

  // output, 20,19,18 – 11 but every number in new line

Break, continue and pass

Continue

```
-   >>> For i in range(5):
-       >>> if i ==3:
-           >>> continue
-       >>>print('helle ',i)
```

// output: hello 0, hello 1 hello 2 hello 4 hello 5 … if will only skip when i==3 and continue without jumping out of loop

Break

```
-   >>> For i in range(5):
-       >>> if i ==3:
-           >>> break
-       >>>print('helle ',i)
```

// output: hello 0, hello 1 hello 2 … if i==3 then it will break and jump to out of the function

Pass

```
-   >>>Def fun()
-   >>> pass
```

// let's see you only know you need function call fun() and you don't know what to write inside that function so to keep that function in the IDE just write pass in the function so the IDE will ignore the function without giving you bug

Functions

Syntax: Def function_name():   // declaring function

Function_name()   // calling function by it’s name

Example1

```
-   >>> def add(x,y):
-           >>> c = x+y
-           >>> print(c)
-   >>> Add(5,y)  // passing value while calling
```

Example 2

```
-   >>> def add(x,y):
-           >>> c = x+y
-           >>> return c  // returning value
-   >>> Result = Add(5,y)
-   >>> Print( result)
```

Example 3

```
-   >>> def add_sub(x,y):
-           >>> a = x+y
-           >>> b = x+y
-           >>> return a,b  // returning two value
-   >>> Result1,result2 = Add_sub(5,y)
-   >>> Print( result1)
-   >>> print(result2)
```

Function passing keywords

Def person (name, age)

Person (name=’moha’, age=26)

Function passing my multi arguments

Def person (name, **data)

Person(‘moha’, age=26, city=’mogadishu’, phone =0618622020)

// now this method uses when the name is must include

// **data: this arguments holds = age=26, city=’mogadishu’, phone =0618622020

- If you want to use global variable in function you've to use keyword 'global' to call, example >>> **Global** variable_name
- You can call a function by itself (recursion)
- You can write functions inside function

Modules

- Instead of writing all codes in one file it's better to separate them into different files then link them all together like HTML

This is how it works

- Create two python files first example, main.py and calc.py
- In the calc.py write some lines of code example
- >>>**Def** add (a,b):
-         >>> return a+b
- >>>**Def** sub (a,b):
-         >>> return a-b
- >>>**Def** multiple (a,b):
-         >>> return a*b
- In the second file main.py
- **>>> from calc import *** // linking main.py to calc.py
- >>> a,b = 1,2
- >>> C = add(a,b) // passing two variables to another module or file

Section two

Class

- Class is where you can store multiple functions
- **>>> class class_name:**   // declaring class
- To call the class first you have to create variable that holds the class let's say we have class name called ***Computer*** and inside this class we have function called ***info ( )***, then
- **>>> Variable_A = computer()**   // variable a holds class now
- **>>> Variable_A. *info ( )***   // calling function that is inside class
- Of course you can create variables inside the class then you can call them inside functions example
- >>> **class** student:
-          >>> name = ' Mohamed '
-         >>> **def** info (**cls**):// function info is inside class
-                 >>> print (**cls**.name) // cls = class, accessing variable name here
- You can create class inside class in you want

Inheritance

- >>> **class** father: // declaring class with name father
- >>> **class** son (father): // now class son inherits all class father's activities

Another example

- **Class** A:
- **Class** B: // let's say there is no relationship b/w class A and B
- **Class C** (A,B): // class C, inherits both classes A and B

Linear search

- Let's say we have this list >>> **a = [5,8,4,6,9,2]** and we want to search **n=9**

This is how linear search works … you comparing from the very first element to the last one.

- If element 0 (5) is equal to 9 then you found the target else
- If element 1 (8) is equal to 9 then you found the target else
  ….
- If element 4 (9) is equal to 9 then you found the target

Binary search

- The list of value must be sorted before it's search
- After being sort Point lower value (L) and upper value (U) the find middle value (***Middle value = (lower value + upper value) // 2***)
- Let's say we have >>> **a = [4,7,8,12,45,99]** and we want to search **n=45**
  L= 0 and U=5 , find middle value

This is how binary search works

- Find middle value (0+5)//2=2 // we using position of elements not value
- Midlde = 8 // (//) integer value position 2 =8
- If the value (n=45) is equal to middle value (8) the found target, else
  If the value (45) > middle value, then the middle value will because lower value (L) **a = [4,7,8,12,45,99]** , **L= 2 and U=5 ,** find middle value
- Middle = (2+5)//2 = 3, so middle value = 12
- If the value (n=45) is equal to middle value (12) the found target, else
  If the value (45) > middle value, then the middle value will because lower value (L) **a = [4,7,8,12,45,99]** , **L= 3 and U=5 ,** find middle value
- Middle = (3+5)//2 = 4, so middle value = 45
- So now value(45) == middle value(45)

Code

```
-   Def search (list, n ):
-      L = 0
-      U = len(list)-1
-   While L <= U:
-       Middle = ( L+U )//2
-       If list[middle] = = n
-           Return true
-      Else:
-         If list [middle]<n:
-             L = middle +1
-             U = middle – 1
-    Return false
-   // in the main program
-   a = [ 4,7,8,12,45,99]
-   n = 45
-   if search ( a , n )
-       print (‘found’)
-   Else:
-      Print(‘not found’)
```

Bubble sort

- This algorithm uses swap to sort values

Example:

Let’s say we have **a = [5,3,8,6,7,2]** let’s sort these list

- First compare the first value to second ++
- If 5 > 3 then swap (if it’s true swap else skip)
- **[**3,**5,8,6,7,2]** now compare 5 to 8
- if 5>8 (false so skip)
- **[**3,5,**8,6,7,2]** now compare 8 to 6
- if 8>6
- **[**3,5,6,**8,7,2]** now compare 8 to 7
- if 8>7
- **[**3,5,6,**7,8,2]** now compare 8 to 2
- if 8>2
- **[**3,5,6,7,**8,2]** now compare 8 to 2
- [3,5,6,7,2,**8]** // the goal of the first round was to push the highest value to the end, now re-start the second round from 3 to 2

Selection sort

- Finding the minimum value in the list then putting its position
- Let's sort this list **a = [5,3,8,6,7,2]**
- First let's say the minimum value is 5 so let's compare 5 to list compare 5 to 3, then to 6 … then to 2 the we found the minimum is 2 so swap 5 to 2
- **[2,3,8,6,7,5]** then find minimum value from 3 to 5
- **[2,3,8,6,7,5]** then find minimum value from 8 to 5
- **[2,3,5,6,7,8]** then find minimum value from 6 to 8
- **[2,3,5,6,7,8]** then lastly compare 7 to 8 **[2,3,5,6,7,8]**

Code

```
Def sort (num):
        For i in range(5):
                Minpos = i
                For j in range (i, 6 )
                        If num[j] < num[minpos]"
                                Minpos = j
                Temp = num [i]
                Num[i] = num[minpos]
                Num[minpos] = temp
Num = [5,3,8,6,7,2]
Sort(num)
Print (num)    // sorted list
```

While true, try, and except

- It's very useful to use when you think the user may miss up something example you are asking the user to enter the number and you think the user may enter char or string … to avoid error during run time use this method

```
- >>> while True:
-   >>> try:
-       >>> a = int(input('please enter your favorite number'))
-       >>> print(your favorite number is : ' , a)
-       >>> break
-   >>> except:
```

- >>> print('please make sure to enter your favorite number ')

// this program only accepts number if you enter anything else it will keep asking you

- >>> **except ZerodivisionError:** // use when you don't want a number / zero
- >>> **except valueError:** // when you ask user to enter integer and u don't want the user to enter anything else like float, string ETC

Constructor

- Syntax: **def __init__(selft)** // this is how to declare constructor function of the class

Lambda

- Syntax: Variable = **lambda** x: x*7 // small function and one time use

Zip

- **Zip**: you can use when you want to combine two or more dictionaries and want you want sort dictionaries
- >>> a = **zip** (variable_dicionary_one + variable_dicionary_two) // combining dictionary one and two into a
- >>> **sorted**(**zip** (variable_dicionary.keys() , variable_dicionary.values()))

New line

- >>> print ('Mohamed **\n** Adam ') // it python \n = new line

External Files

Syntax: variable = open (" file_name.extension " , " reason ")

- The reason will be one of these
- A) **" r " =** read, this means I am opening the file for reading purpose
- B) **" w " =** write , when you use "w" in existed file it will overwrite, this means it will delete the previous file and write new things, we can also use "w" to create a file
- C) **" a " =** append, adding something at the end of the file without deleting.
- D) **" r+ " =** read and write

Creating file "w"

- >>> student_file = **open** ('student.txt', **'w'**)
- >>> student_file.**write**(' New student record file \n ')
- >>> student_file.**close**()

Reading files "r"

- >>> student_file = **open** ('student.txt', **'r'**)

- >>> print(student_file.**readable()**) // this function returns True or False, if the file is readable it will return True else False
- print(student_file.**read()**) // this function reads and prints all the data in the file
- print(student_file.**readlines()[1]**) // reading specific line if you only want, like this example I am only reading and printing line index [1]
- >>> student_file.**close**()

Writing files "w"

- Let's say we have student.txt (file) and there record of the student so if you want to overwrite (deleting the previous record and writing new one) use "w" open and then overwrite examples
- >>> student_file = **open** ('student.txt', **'w'**)
- >>> student_file.**write**(' student name is adan \n ')
- >>> student_file.**close**()

Append "a"

- Append is adding something to an existed file without deleting anything , and append will add at the last
- >>> student_file = **open** ('student.txt', **'a'**)
- >>> student_file.**write**(' student age 26 \n ')
- >>> student_file.**close**()

Package (folder)

- Create a package to separate python modules.
- Example: create new python package and name "student" then in the package let's create a new module and name it "info.py" so if you want to access this module in the main use this formula

From package_name.module_name import *

- >>> **from student.info import ***