**Mutable and Immutable variables**

1. Int, str, float and tuple are mutable.
2. List, dictionary and sets are immutable.

**If Else Conditions**

1. If var1>var2:

 print(“greater”)

 if var1==var2:

 print(“equal”)

 Else:

print(“lesser”) #indentation is must here

1. elif # stands for else if, will come out of if else loop as soon as first true condition occur.
2. L = [1,2,3,4,5]

if 5 in L:

 print(“yes it’s in the list”)

if 7 not in L:

 print(“no, it’s not in the list”)

**FOR Loops**

1. For index in range(5) #range(1,5) , range(0,11,2)

Print(index)

1. List = [“a” , “b” , “c” , “d”]

For item in list:

 Print(item) #all items of the list will be printed, same can be done with a tuple.

1. List1 = [ [“ali” , 21], [“ahmad” ,22], [“hassan” , 23]]

For students in list1:

 Print(students) #data will be displayed in the form of lists

1. For students,marks in list1:

 Print (students,marks) #data will be printed like ali 21

For students,marks in list1:

Print (students, “actually got” , marks, “marks”) # ali actually got 21 marks

1. To iterate a dictionary using for loop

d={"1":"apple" , "2":"orange" , "3":"mango"}
for key,value in d.items():
 print(key,value)

1. How to initialize lists at run time using FOR loop.

list=[]
print("enter values you want to save in list")
for i in range(11):
 n=input()
 list.append(n)
print(list)

7 How to initialize dictionary at run time using FOR loop.

l=[]
for i in range(2):
 k=input("enter key: ")
 l.append(k)
d={}
for i in l:
 print("enter value of" , i , ":")
 k=input()
 d[i]=k
print(d)

**While Loops**

1. i=0

while(i<45):

 print(i) #print(I , end=” “) it will print numbers in one line with a single space.

 i=i+1

**Break and Continue statement**

1. while(i<45):

 print(i)

 if i==44:

 break #it will come out of the loop

 continue #this iteration will be skipped from here and loop will enter into next iteration

 i=i+1

**Operators**

Arithmetic operators

 +, -, \*, /, %, \*\* (power)

Assignment operators

 =, +=, \_=, %=

Comparison operators

 ==, !=, >, >=, <, <=

Logical operators

 And, or, not

 A=true

 B=true

 Print(A and B) # it will return true

Identity operators

 Is, is not

 Print (1 is 3) #it will return false

Membership operators

 In, not in

 List[1,2,3,4,5]

Print(1 in list) # it will print true

**Short hand if else**

1. If (a>b): print (“a is greater than b”) #it is a one liner
2. Print(a is greater than b) if a>b else print(b is greater than a)

**Functions and Docstrings**

1. A=2

B=3

C=sum((a,b))

Print(c) #it will print 5

1. Press ctrl and click on a built-in function name to see its code.
2. Def function1():

 Print(“you are in function 1”)

Function1() # it will print you are in function 1

1. Def function2(a,b):

 Average = (a+b) / 2

 Return average

C=function2(3,5)

Print(c) #it will print average value 4

1. Def function2(a,b):

 “”” this function returns average of two numbers”””

 Average = (a+b) / 2

 Return average

Print(function2.\_\_doc\_\_) # it will print **this function returns average of two numbers(double underscore)**

**Lambda or anonymous functions**

1. In Python, normal functions are defined using the “def” keyword and anonymous functions are defined using the lambda keyword. Hence, anonymous functions are also called lambda functions. We use lambda functions when we require a function for a short period of time. In Python, we generally use it as an argument to a higher-order function (a function that takes in other functions as arguments).

Def add (x,y):

 Return x-y

***Is same as***

M= lambda x,y : x – y

Print(m(5,2))

1. Lambda functions can be used along with built-in functions like filter() and map().
2. The filter() function in Python takes in a function and a list as arguments. This offers an elegant way to filter out all the elements of a sequence “sequence”, for which the function returns True. Here is a small program that returns the odd numbers from an input list:

list1=[2,34,55,67,57,78,11]
odd=list(filter(lambda x: (x%2!=0), list1))
print(odd)

1. The map() function in Python takes in a function and a list as an argument. The function is called with a lambda function and a list and a new list is returned which contains all the lambda modified items returned by that function for each item. Example:

list1=["cat" , "dog" , "panda" , "horse"]
newList=list(map(lambda x: (str.upper(x)), list1))
print(newList)

**Global variables and global keyword**

1. Global variable means that all functions of a program can use it.
2. Variable that is defined within a function is local variable and can only be used in that function.
3. A function cannot change the value of a global variable, but if it is desired then write keyword “global” before the global variable within function definition.
4. When a function tries to change the value of a global variable using keyword global and that global variable doesn’t exist, the function will then create that global variable.
5. In nested functions, inner function cannot change the value of a variable of outer or parent function.

**F-String and string formatting**

1. f-string formatting is the newest method of string formatting in Python and offers conveniences, such as using expressions within strings. Programs commonly need to substitute a variable into a string. Python’s f-strings provide a convenient way for us to do this. Every f-string statement consists of two parts, one is character f, and the next one is a string which we want to format.
2. F = “Aisha”

S = “Tehreem”

A=”This is {} {}”

B=a.format(F,S)

Print(b) # it will print **this is aisha tehreem**

1. A=”This is {1} {0}”

B=a.format(F,S)

Print(b) # it will print **this is tehreem aisha**

1. A = f”this is {A} {B} {11\*1}”

Print(A) # it will print **this is tehreem aisha 11**

Any function or any function of any module can be written inside {}.