Session-1: Python basics

Functions

In [1]:
```
# Function performs a particular task
# identification ----> function()
```

In [2]:
```
print('Hello world')
```
```
Hello world
```

Variables

In [3]:
```
# Variables are used to store the values
# Variables are case sensitive and can be any text with a special character. However, we can oly use '_' as the special charcter.
# Variables should always be in the form of text preceding the number and not vice-versa
```

In [4]:
```
a = 12
```

In [5]:
```
a
```
Out[5]:
```
12
```

In [6]:
```
b = 10
```

In [7]:
```
d = a+ b
```

In [8]:
```
d
```
Out[8]:
```
22
```

In [10]:
```
D = 20
```

In [11]:
```
D
```
Out[11]:
```
20
```

In [12]:
```
a9 = 7
```

In [13]:
```
a9
```
Out[13]:
```
7
```

Data types in Python

In [16]:
```
#Integer ------> int -----------> 1,2,3,4,..
#Float ------> float -----------> 1.0, 2.0, 3.5, 4.7....
#Boolean ------> bool-----------> True, False
#String ------> str ------------> "abc",'abc'

# Type function : type() returns the data type of the entered input.
```

In [17]:
```
print(type(1))
```
```
<class 'int'>
```

In [18]:
```
print(type(True))
```
```
<class 'bool'>
```

In [19]:
```
print(type('abc'))
```
```
<class 'str'>
```

In [20]:
```
print(type(1.2))
```
```
<class 'float'>
```

Sequential Data types

In [24]:
```
# tuple ------> ()
# list ------>  []
# set ------>   {}
# dict ------>  {}
```

In [22]:
```
# List
# List can contain any data type as well as another list in itself.
mylist = [23, 23.5, "abc",[1,2], True]
```

In [23]:
```
print(mylist)
```
```
[23, 23.5, 'abc', [1, 2], True]
```

Indexing

```
In [25]: # Indexing can be of 2 type positive and negative indexing.
         # Positive Indexing in Python starts from 0 (left to right) while negative starts from -1 (right to left)
```

```
In [27]: print(mylist)
```

```
[23, 23.5, 'abc', [1, 2], True]
```

```
In [28]: mylist [2]
```

```
Out[28]: 'abc'
```

```
In [30]: mylist [3][1]
```

```
Out[30]: 2
```

Slicing

```
In [31]: # Slicing is displaying the range of values
         # For slicing one needs to put the start value and stop +1 value
         # start: stop +1
         # Here : represents the range
```

```
In [32]: print(mylist[0:3])
```

```
[23, 23.5, 'abc']
```

```
In [33]: print(mylist[0:3], mylist [3])
```

```
[23, 23.5, 'abc'] [1, 2]
```

Mutable and Immutable

```
In [34]: # Mutlable objects can be changed ,while immutable are fixed once assigned.
         # Lists are mutable and Tuple are immutable
```

```
In [35]: print(mylist)
```

```
[23, 23.5, 'abc', [1, 2], True]
```

```
In [36]: mylist[0]= 10
```

```
In [37]: print(mylist)
```

```
[10, 23.5, 'abc', [1, 2], True]
```

```
In [41]: mytuple = (23, 23.5, 'abc', [1, 2], True)
```

```
In [42]: print(mytuple)
```

```
(23, 23.5, 'abc', [1, 2], True)
```

```
In [40]: mytuple[0]= 10    #will display an error
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[40], line 1
----> 1 mytupple[0]= 10

TypeError: 'tuple' object does not support item assignment
```

Sets

```
In [43]: # Sets return only unique values
```

```
In [44]: d = [1,1,1,2,2,2,3,3,3]
```

```
In [46]: set(d)
```

```
Out[46]: {1, 2, 3}
```

```
In [47]: # Conversion of Sets to List
```

```
In [48]:  e = set(d)
```

```
In [49]: f = list(e)
```

```
In [50]: f
```

```
Out[50]: [1, 2, 3]
```

Dictionary

```
In [59]: # Dictionary are in the form of key and value
         mydict = {"name":"Ninad","age":26,"place":"Mumbai"}
```

In [63]:
```
mydict["name"]
```
Out[63]:
```
'Ninad'
```
In [64]:
```
mydict["age"]
```
Out[64]:
```
26
```
In [65]:
```
list2 = [mydict["name"],mydict["age"]]
```
In [66]:
```
list2
```
Out[66]:
```
['Ninad', 26]
```
Append, Insert and Length

In [67]:
```
# Append --------> Appends the value at the end
# Insert --------> Insert the value at the sepcified index
# Length --------> Returns the length (total no of elements) in the list/tuple/set/dict
abc = [1,2,3]
```
In [68]:
```
abc
```
Out[68]:
```
[1, 2, 3]
```
In [71]:
```
len("abc")
```
Out[71]:
```
3
```
In [72]:
```
abc.append(4)
```
In [73]:
```
abc
```
Out[73]:
```
[1, 2, 3, 4]
```
In [74]:
```
abc.insert(1,5)
```
In [75]:
```
abc
```
Out[75]:
```
[1, 5, 2, 3, 4]
```
if elif & else

In [76]:
```
var = 12
```
In [81]:
```
#While writing if, elif and else we always type ':' after writing if, elif and else and also one has to make sure
#to keep the induntation (space)
if var > 13:
    print('Hello!')
elif var == 12:
        print('yes')
else:
        print('bye')
```
```
yes
```
input function

In [82]:
```
#input function will ask the user to input a value
#However, since input function is by default a string data-type we have to make sure to use the desire data type before it
```
In [83]:
```
a = int(input("Enter 1st no: "))
b = int(input("Enter 2nd no: "))
```
```
Enter 1st no: 10
Enter 2nd no: 20
```
In [84]:
```
c = a+b
```
In [85]:
```
c
```
Out[85]:
```
30
```
In [86]:
```
# Question:
# Assign grades as per the criteria
```
In [111…
```
inputMarks= int(input("Enter your marks: "))
```
```
Enter your marks: 101
```
In [112…
```
if inputMarks >= 75 and inputMarks <= 100 :
    print('Grade A')
elif inputMarks >= 50 and inputMarks < 75:
    print('Grade B')
elif inputMarks >= 35 and inputMarks < 50:
    print('Grade C')
elif inputMarks >= 0 and inputMarks < 35:
    print('Fail')
else:
    print('Invalid marks')
```

```
Invalid marks
```

for loop

In [115…

```
for i in range(0,10):
    print(i)
```

```
0
1
2
3
4
5
6
7
8
9
```

In [128…

```
#print the table of 2 using for loop

for i in range(1,11):

    print("2","*",i,"=",2*i)
```

```
2 * 1 = 2
2 * 2 = 4
2 * 3 = 6
2 * 4 = 8
2 * 5 = 10
2 * 6 = 12
2 * 7 = 14
2 * 8 = 16
2 * 9 = 18
2 * 10 = 20
```

In [17]:

```
#Seggreggate the following my_list into different data types

my_list= [23, 'hello','hi',45.9,'bye',12,76,'zero',0,99,[23,'one',12.7],77,11,(6,2,0),11.0]
```

In [130…

```
my_list
```

Out[130]:

```
[23,
 'hello',
 'hi',
 45.9,
 'bye',
 12,
 76,
 'zero',
 0,
 99,
 [23, 'one', 12.7],
 77,
 11,
 (6, 2, 0),
 11.0]
```

In [18]:

```
list_of_integers1=[]
list_of_string1=[]
list_of_float1=[]
list_of_tuple1=[]
list_of_list1=[]

for i in my_list:
    if type(i)==int:
        list_of_integers1.append(i)
    elif type(i)==str:
        list_of_string1.append(i)
    elif type(i)==float:
        list_of_float1.append(i)
    elif type(i)==tuple:
        list_of_tuple1.append(i)
    elif type(i)==list:
        list_of_list1.append(i)

print(list_of_integers1)
print(list_of_string1)
print(list_of_float1)
print(list_of_tuple1)
print(list_of_list1)
```

```
[23, 12, 76, 0, 99, 77, 11]
['hello', 'hi', 'bye', 'zero']
[45.9, 11.0]
[(6, 2, 0)]
[[23, 'one', 12.7]]
```

Defining a Function

In [1]:
```
# Defining a function with no argument.
# Function can be called separtely to reduce the line of codes.

#Single argument
def myfunction():
    a= 10*10
    return a
```

In [2]:
```
myfunction()
```

Out[2]:
```
100
```

In [5]:
```
# Defining a function with a single argument.

#Single argument
def myfunction(no1):
    a= no1*10
    return a
```

In [7]:
```
myfunction(2)
```

Out[7]:
```
20
```

In [8]:
```
# Defining a function with multiple arguments.

#Single argument
def myfunction(no1,no2):
    a= no1*no2
    return a
```

In [9]:
```
myfunction(2,5)
```

Out[9]:
```
10
```

Task: Return a table for the value inputed by user

In [8]:
```
#Return a table for the value inputed by user

def gettable(number1):
    for i in range(1,11):
        print(number1,"*",i,"=",number1*i)
```

In [10]:
```
gettable(3)
```

```
3 * 1 = 3
3 * 2 = 6
3 * 3 = 9
3 * 4 = 12
3 * 5 = 15
3 * 6 = 18
3 * 7 = 21
3 * 8 = 24
3 * 9 = 27
3 * 10 = 30
```

In [14]:
```
#Return a table for the value inputed by user by using append function.

def gettable(number1):
    out=[]
    for i in range(1,11):
        a = number1,"*",i,"=",number1*i
        out.append(a)
    return out
```

In [15]:
```
gettable(3)
```

Out[15]:
```
[(3, '*', 1, '=', 3),
 (3, '*', 2, '=', 6),
 (3, '*', 3, '=', 9),
 (3, '*', 4, '=', 12),
 (3, '*', 5, '=', 15),
 (3, '*', 6, '=', 18),
 (3, '*', 7, '=', 21),
 (3, '*', 8, '=', 24),
 (3, '*', 9, '=', 27),
 (3, '*', 10, '=', 30)]
```

Try and except block

In [21]:
```
# Try and except block is used for eliminating errors

def gettable(number):
    try:
        out=()
        for i in range(1,11):
            a=number,"*",i,"=",i*number
            out.append(a)
        return out
    except:
        print("error in gettable fn!")
```

In [22]:
```
gettable("65")
```

```
error in gettable fn!
```

In [ ]:
```
```