Functions

Defining and Calling a Function:-

In programming, A function is a block of code organized by a set of rules to accomplish a specific task.

A function is a block of code which only runs when it is called.

You can pass data, known as parameters into a function, you can return data as a result.

Types of Functions:-

(i) Pre-defined (or) Built-in functions.

(ii) User-defined functions.

Pre-defined (or) Built-In Functions:-

The functions which come installed along with python software are called pre-defined (or) Built-in functions.

ex:- id(), type(), input(), print() etc...

User-defined Functions:-

The functions which are defined by the user as per the requirements are called user-defined functions.

A user can define a function in four ways:-

(i) Function without argument and without return value.

(ii) Function with argument and with return value.

(iii) Function without argument and with return value.

(iv) Function with argument and without return value.

Creating and Using the function:-

(i) Defining a Function
(ii) Calling a Function.

Defining Function:-

def keyword, name for the function, parentheses, parameters (optional), colon (:), body, return (optional)

Syntax:-

```
def func_name(par):
    """ docstring """
    stat1
    stat2
     :
    return [expr].
```

ex:- ~~def~~

```
def display ():
    print ("CPDS CPDS")
```

Calling a Function:-

Function calling is also important along with function definition.

ex:-

```
def display():
        print ("CPDS")
    display()
```

o/p ⇒ CPDS

Function without argument and without return value:-

```
def sum():
    a,b = 20,30
    print("sum =", (a+b))
sum()
```

o/p ⇒

```
sum = 50
```

Function with argument and with return value:-

```
def sum (a,b) :
        return (a+b)
 result = sum (20,30)
 print ("sum =", result)
```

o/p ⇒

```
  sum = 50.
```

Function without argument and with return value:-

```
 def sum() :
        a,b = 20,30
        return (a+b)
  result = sum()
  print ("Sum =", result)
```

o/p ⇒

```
  sum=50.
```

Function with argument and without return value:-

```
def sum (a,b) :
       result = a+b
       print ("sum =", result)

 sum (20,30)
```

o/p ⇒

```
  sum = 50.
```

Returning multiple values from a function:-

```
def m1 (a,b) :
      c=a+b
      d = a-b
      return c,d
```

```
x,y = m1(10,5)
print("Sum =", x)
print("Sub =", y)
```

o/p ⇒

```
sum = 15
sub = 5.
```

A function can call another function inside it. is called "nesting of function".

Functions as First class objects:-

In python, functions are considered as first class objects.

Assign a function to variables:-

```
def add():
    print(" Assigned")

sum=add
sum()
```

o/p⇒ Assigned

Pass function as a parameter to another function:-

```
def display(x):
    print (" Display Function")

def message():
    print("Message Function")

display(message())
```

o/p ⇒

```
Message Function
Display Function.
```

Function inside Another function :-

```
def first():
    print ("outer Function")
    def second():
        print ("Inner Function")

    second ()
first ()
```

o/p ⇒

```
outer Function
Inner Function.
```

Function can return another function :-

```
def first():
    def second():
        print (" This func is return to outer func")
    return second
x = first ()
x()
```

o/p ⇒

```
This func is return to outer func.
```

Actual and Formal Parameters (or Arguments) :-

Actual Parameter	Formal Parameter.
(i) The args in the function calling.	(i) The args in the function definition.
(ii) Data type not required.	(ii) Data type required.

ex :-

```
def   sum (a,b) :
      c = a+b
      print(c)
```

Here, a, b are Actual parameters (x, y)

ex : x=10 ; y=15
sum (x,y)

Here a,b are formal parameters.

Types of Arguments:-

A parameter is the variable listed inside the parentheses in the function definition.

An argument is the value that is sent to the function when it is called.

Keyword Arguments:-

If user wants to change order of arguments from function call to function definition, then user has to define keyword arguments.

ex:-
```
def man(name, age, height, weight):
        print (name, age, height, weight)
    man (height = 5.8, name = "Hari", weight = 65, age = 21)
o/p ⇒ Hari 21 5.8 65
```
Keyword args.

Default Arguments:-

A default argument assumes a default value if a value is not supplied as an argument while calling the function.

ex:-
```
def person (age=21, weight =65, height, name):
        print(name, age, height, weight)
    person (height = 5.8, name = "Hari")

o/p ⇒ Hari 21 5.8 65
```

Variable Length Arguments:-

We can define the function with a flexible number of arguments which are called variable length arguments. Use (*) symbol before keyword inside the parentheses.

```
ex: def add (*args):
        total = 0
        for a in args:
            total += a
        print(total)
    add(3,5)
    add(3,4,5,1,2)
  o/p → 8
        15.
```

Positional Arguments:-

Arguments that need to be included in the proper position (or order.

(or)

Argument that has a given position in the list of arguments passed into your function.

```
ex:- def abc(a,b,c=2):
         return a+b+c
     x = abc (1,2)
     print (x)
     y = abc (2,b=3)
     print(y)
     z = abc (a=2,b=4)
     print(z)
  o/p ⇒ 5
        7
        8
```

Recursive Function:-

A function that calls itself until it doesn't.

A recursive function always has a condition that stops calling itself.

eg:-
```
def factorial(x):
    if x==1:
        return 1
    else:
        return (x* factorial(x-1))
num=3
print("Factorial of",num,"is", factorial(num))
```
o/p ⇒ Factorial of 3 is 6.

Part-2 :- Exceptions

Errors in Python:-

We can make certain mistakes while writing a program that lead to errors.

These errors can be classified into 3 classes.

(i) Syntax Errors

(ii) Logical Errors

(iii) Run time Errors (Exceptions).

Python Syntax Errors:-

Error caused by not following the syntax of the language is called syntax error or parsing error.

eg:- if a<3

o/p ⇒ syntax Error: invalid syntax.

Python Logical Errors:-

Logical errors are the most difficult to fix. They occur the program runs without crashing but produces incorrect result. You won't get an error Message.

Python Runtime Errors (Exceptions):-

Errors that occur at runtime are called exceptions or runtime errors.

Exception:-

An unwanted | unexpected event that disturbs the normal flow of the program.

Python has many buit-in exceptions. is given below,

Zero Division Error:-
Occurs when a number is divided by zero.
ex:-
```
m=5
n=0
print(m/n)
```

o/p

Name Error:-
It occurs when a name is not found. It may be local (or) global.
ex:-
```
n=int(input("Enter any number ="))
m=sqrt(n)
print(m).
```

Indentation Error:-
It occurs, when incorrect indentation is given.
ex:-
```
n=int(input("Enter any number"))
if n>0:
print("+ve number")
```

IO Error:-
It occurs when input, output operation fails.

EOF Error:-
It occurs when the end of the file is reached.

Exception handling statements in python:-

Try-except statement:-
try block is placed where the exception is raised.
The try block must be followed with the except statement.

Syntax:-
```
try:
    # block of code
except Exception1:
    #block of code
except Exception2:
    #block of code.
```

ex:-
```
try:
    a= int(input())
    b= int(input())
    c=a/b
except:
    print("Can't divide
          with zero")
```

o/p →
```
10
0
Can't divide with zero.
```

Try-except else statement:-

Syntax:-
```
try:
    #block of code
except Exception1:
    #block of code
else:
    # block of code (if no exception occur).
```

ex:-
```
try:
    a=int(input())
    b=int(input())
    c=a/b
    print("a/b= %.d" %c)
except Exception:
    print("Can't divide by zero")
    print(Exception)
else:
    print("Hi I am else block")
```

o/p →
```
10
0
Can't divide by zero
<class 'Exception'>
```

Except statement with no exception:-

ex:-
```
try:
      a=1
      b=0
      c=a/b
      print (c)
   except :
      print ("can't divide by zero")
   else :
      print (" Else block")
```
o/p => can't divide by zero

The except statement using with exception variable:-

ex:-
```
try:
      a=int(input())
      b=int(input())
      c=a/b
      print("a/b =%d" %c)
   except Exception as e:
        print ("can't divide by zero")
        print (e)
   else:
      print ("Elese block)
```
o/p =>
```
10
0
can't divide by zero
Division by zero.
```

Declarating multiple Exceptions:-

Syntax:-
```
try:
    #Block of code
except (<Except1>, <Except2>,...,<Excep n>):
    # block of code
else:
    #block of code
```

ex:-
```
try:
    a = 10/0
except (ArithmeticError, IO Error):
    print ("Arithmetic Exception")
else:
    print (" Done")
```

o/p ⇒ Arithmetic Exception.

Try -Finally Block:-

Syntax:-
```
try:
    #block of code
finally:
    #block of code (this always be executed)
```

ex:-
```
try:
    fileptr = open ("file 2.txt", "r")
    try:
        fileptr.write ("Hi I am good")
    finally:
        fileptr.close ()
        print ("file closed")
except:
    print ("Error")
```

o/p ⇒ file closed
Error.

Types of Exceptions:-

There are 2 types of exceptions in Python.

(i) Built-in Exceptions.

(ii) User-Defined Exceptions.

Built-in Exceptions:-

These are the standard exceptions in python.

ex:- Zero Division Error, Name Error, Indentation Error, IO Error, EOF Error..

User-Defined Exceptions:-

The exceptions defined by a user are called user-defined exceptions.

Assert Statement:-

The assert statement is used to continue the execution. If the assert condition is False then it raises Assertion Error.

syntax:-

```
assert condition [, Error Message]
```

ex:-
```
x=10
  assert x > 0
  print(" x is +ve number")
```

o/p →
```
x is a positive number.
```

User defined Exceptions:-

~~ex:- class NegativeValueError(Error):~~

ex:-
```
class Error(Exception):
    class NegativeValueError(Error):
```

```
class ValueTooSmallError(Error):
class ValueTooLargeError(Error):
num=11
while True:
        try:
            n=int(input())
            if n<0:
                raise NegativeValueError
            elif n<num:
                raise ValueTooSmallError
            elif n>num:
                raise ValueTooLargeError.
            break
        except NegativeValueError:
                print("Negative number, try again")
        except ValueTooSmallError:
                print(" Too small, try again")
        except ValueTooLargeError:
                print("Too large" try again")
        print("Correct value entered")
```

o/p ⇒ -1

```
Negative number, try again
3
Too small, try again
100
Too large, try again
11
Correct value Entered.
```

✓ ① Different Arithmetic Operations

```
def arithmetic (a,b):
    print (" Addition:", a+b)
    print ("Substraction :", a-b)
    print ("Multiplication:", a*b)
    print ("Division :", a/b)
    print ("Exponential :", a**b)
x = float (input (" Enter 1st number:"))
y = float (input (" Enter 2nd number :"))
arithmetic (x,y)
```

② printing stars

```
def space (s):
    if (s==0):
        return
    print (" ", end =" ")
    space (s-1)
def star (a):
    if (a==0):
        return
    print ("*", end = " ")
    star(a-1)
def pattern (n,num):
    if (n==0):
        return
    space (n-1)
    star (num-n+1)
    print (" ")
    pattern (n-1,num)

n=5
pattern (n,n)
```

3. List Operations.

```
print ("Do you want to create a list (y/n)")
ans = input ("Enter y or n:")
if (ans=="y"):
    print ("Creating List")
    l=list()
    choice =1
    while choice!=4:
        print ("1. Append\n2. Remove\n3. Display\n
                4. Exit ")
        choice =int (input (" Enter your choice:"))
        if (choice ==1):
            n=int (input ("Enter the number to
                                append:"))
            l.append(n)
        elif choice ==2:
            m=int (input ("Enter the number to remove:"))
            l.remove(n)
        elif choice ==3:
            print ("List elements are:", l)
        elif choice ==4:
            print ("Exiting")
        else:
            print ("Invalid choice")
else:
    print ("Not Creating List")
```

(4) Counting the number of occurances

```
text = input ("Enter the text :")
d={}
for char in text:
    if(char!="!@#$%^&*()<>?|{}[];;.,1234567890-=":
        char=char.lower()
        if char in d:
            d[char]+=1
        else:
            d[char]=1
print(d)
```

---

(5) Tuples in Python.

```
tup = (1,2,3,4,5,6,"gskd","chinna","ibj")
print ("Elements of tuple are :", tup)
choice =1
print (" Indexing   \n1. First Element \n 2. Last Element \n
        slicing     \n 3. Elements from 2nd to 5th \n
        4. Elements from 5th to last \n 5. Exiting")
while choice != 5:
    choice = int(input("Enter the choice :"))
    if choice ==1:
        print ("First element is:", tup[0])
    elif choice ==2:
        print ("Last Element is:", tup[-1])
    elif choice ==3:
        print ("Elements from 2nd to 5th is:", tup[1:5
    elif choice==4:
        print ("Elements from 5th to last is:", tup [4:])
    elif choice ==5:
        print ("Exiting")
    else:
        print ("Invalid choice ")
```

⑥. Finding the sum and Average of Tuple of Numbers.

```
tup= eval (input ("Enter the tuple of numbers"))
l= len(tup)
sum=0
avg=0
for i in tup:
      sum=sum+i.
print("The sum of the tuple of numbers is:", sum)
print("The avg of the tuple of numbers is:", sum/l)
```

⑦ Dictionary Operations in Python.

```
d={1:"Gskd", 2:"Chinna", 3:"Vijay", 4:"ibj"}
print ("The dictionary is:", d)
choice =1
print ("1. First Student Name \n 2. Last Student Name \n.
   .3. Keys \n. 4. Values \n 5. Add another Student Name \n
   6. Remove any student from the dictionary \n
   7. Exiting")
while choice !=7:
   choice =int (input ("Enter the choice:"))
    if choice ==1:
         print ("First Student Name:", d[1])
    elif choice ==2:
         print ("Last Student Name:", d[4])
    elif choice ==3:
         key = d.keys ()
         print ("Keys:", key)
    elif choice ==4:
         value = d.values ()
         print ("Values:", value)
    elif choice==5:
        name=input("Enter Student Name:")
        roll=int(input("Enter Student Roll Number:"))
        d[roll]=name
        print("The new dictionary is:",d)
    elif choice==6:
        r=int(input("Which student do you want
              to remove \n Please Enter his Roll Number:
        remove=d.pop(r)
        print("The new dictionary is:",d)
    elif choice==7:
        print("Exiting")
    else:
        print("Invalid Choice")
```

⑧. To accept Student name and Marks from the key Board.

```
n=int(input("Enter how many student details do you
              want to enter:"))
d={}
for i in range(n):
    name=input("Enter the student name:")
    marks=int(input("Enter the student marks:"))
    d[name]=marks
print("The dictionary is:",d)
choice=1
while choice!=2:
    if choice==1:
        name=input("Enter the student name to get
                    his marks:")
        mark=d.get(name)
        print("Marks:",mark)
        m=input("Do you want to know another
                        student name(y/n):")
        if(m=="y"):
                p=input("Enter the another student
                                        name:")
                q = d.get(p)
                print("Marks:",q)
                break
            else:
              break
    elif choice==2:
         print("Exiting")
    else:
        print("Invalid choice")
```

---

(9) Arrays and Array Indexing:

```
import numpy as np
 a=np.array(input("Enter the elements of array:"))
print("The array is:",a)
 l=eval(input("Enter the list:"))
print("the list is:"l)
 choice =1
print("\n Indexing \n 1.First Element \n 2.Last Element \n
        slicing  \n 3.2nd to Last \n 4.Exiting")
while choice!=4:
       choice =int(input("Enter the choice:"))
        if choice==1:
             print("First Element is:",l[0])
        elif choice==2:
             print("Last Element is:",l[-1])
```

```
elif choice==3:
    print("2nd to Last:",l[1:])
elif choice==4:
    print("Exiting")
else:
    print("Invalid choice")
```