FUNCTION

A block of code that do a particular task for you

SYNTAX

def_function_name(perameters): """docstring""" statement1 ststement2 ..... return expression

Create a function using def

Call a function using function_name followed by parathrnsis having paramerters of that particular function

In [3]:
```
def func1():
    print("CSE First Year")

func1()
```

```
CSE First Year
```

Bleow example haaving arguments

In [6]:
```
def func2(a,b):
    if b%2==0:
        print(a+b)
    else:
        print(a-b)

func2(10,8)
func2(12,5)
```

```
18
7
```

In [7]:
```
# If 2nd argument is even then add else subtract
def func2(a,b):
    if b%2==0:
        print(a+b)
    else:
        print(a-b)

func2(10,8)
func2(9,7)
```

```
18
2
```

In [10]:
```
def func2(a,b):
    if b%2==0:
        print(a+b)
    else:
        print(a-b)

c=int(input())
d=int(input())
func2(c,d)
```

```
12
6
18
```

In [30]:
```
def func2(a,b):
    if b%2==0:
        print("Raushan")
    else:
        print(a-b)

print(func2(10,8))
```

```
Raushan
None
```

Types of arguments

1. DEAFULT ARGUMENTS

In [9]:
```
def func3(a,b=7):
    if b%2==0:
        print(a+b)
    else:
        print(a-b)

func3(10)
func3(9,8)
```

```
3
17
```

2. KEYWORD ARGUMENTS

In [11]:
```
# If 2nd argument is even then add else subtract

def func4(a,b):
    if b%2==0:
        print(a+b)
    else:
        print(a-b)

func4(a=10,b=8)
func4(b=12,a=56)
```

```
18
68
```

3. VARIABLE LENGTH ARGUMENTS

a) *args (NOn-keywords Arguments)

b) **kwargs (keywords Arguments)

In [13]:
```
def func5(*argv):
    for i in argv:
        print(i)

func5('This','is','Chitkara','University','Punjab')
```

```
This
is
Chitkara
University
Punjab
```

In [14]:
```
def func5(*argv):
    for i in argv:
        print(i, end=" ")

func5('This','is','Chitkara','University','Punjab')
```

```
This is Chitkara University Punjab
```

In [15]:
```
def func6(**kwargs):
    for i,j in kwargs.items():
        print(i,j)

func6(a='This',b='is',c='Chitkara',d='University',e='Punjab')
```

```
a This
```

```
b is
c Chitkara
d University
e Punjab
```

In [ ]:
```
##### DOSCTEING (Documents string)
##### The first line in the function definition usually. It is optional to use and tells the function of the prog
##### SYNTAX
#####  """DOCSTRING"""
##### Usages:  print(function_name.__doc__)
```

In [16]:
```
def func7(a):
    """ This program is for checking number if odd or even"""
    if a%2!=0:
        print("odd")
    else:
        print("Even")

func7(10)
x=int(input())
func7(x)
```

```
Even
7
odd
```

In [18]:
```
def func7(a):
    """ This program is for checking number if odd or even"""
    if a%2!=0:
        print("odd")
    else:
        print("Even")

func7(10)
x=int(input())
print(func7.__doc__)
```

```
Even
8
 This program is for checking number if odd or even
```

In [22]:
```
def func7(a):
    """ This program is for checking number if odd or even"""
    if a%2!=0:
        print("odd")
    else:
        print("Even")


print(func7.__doc__)
func7(10)
x=int(input())
```

```
 This program is for checking number if odd or even
Even
78
```

Using of Return statement

Code of a number

def cube_number(a): return a**3

print(cube_number(10)) d=cube_number(5) print(d)

In [27]:
```
def cube_number(a):
    return a**3

print(cube_number(5))
d=cube_number(7)
print(d)
```

```
125
343
```

Pass by Refrence

In [31]:
```
a=100
print(type(a))
print(id(a))
```

```
<class 'int'>
140725008151440
```

In [32]:
```
a="Raushan"
print(type(a))
print(id(a))
```

```
<class 'str'>
1584173042672
```

In [34]:
```
b=100
c=b
print(type(b))
print(id(b))
print(type(c))
print(id(c))
```

```
<class 'int'>
140725008151440
<class 'int'>
140725008151440
```

In [35]:
```
b=50
print(id(b))
b=b+10
print(id(b))
```

```
140725008149840
140725008150160
```

In [36]:
```
def func8(arg):
    print ("Received {} value has address {}".format(arg, id(arg)))

a=500
print ("Sent {} value has address {}".format(a, id(a)))
func8(a)
```

```
Sent 500 value has address 1584172111984
Received 500 value has address 1584172111984
```

NOTE

A function is always called by passing a variable by refrence.

Generally, when function defined modifies data that is got from function calling.

The changes will be reflected in original data.

However, this is not always true.

If the passes parameters are immutable object such as int, float, tuple or string

then changes will not be reflected in original data.

Mutable objects like set, dictionary, list etc.

In [37]:
```
def func9(arg):
    print ("Recieved {} value has address {}".format(arg, id(arg)))
    arg=arg+10
```

```
    print ("Changed {} value has address {}".format(arg, id(arg)))

a=100
print ("Sent {} value has address {}".format(a, id(a)))
func9(a)
print ("Value after function call {} has address {}".format(a, id(a)))
```

```
Sent 100 value has address 140725008151440
Recieved 100 value has address 140725008151440
Changed 110 value has address 140725008151760
Value after function call 100 has address 140725008151440
```

In [41]:

```
# Mutable object example

def func10(arg):
    print ("Recieved {} list has address {}".format(arg, id(arg)))
    arg.append(40)
    print ("Changed {} list has address {}".format(arg, id(arg)))

a=[10,20,30]
print ("Sent {} list has address {}".format(a, id(a)))
func10(a)
print ("list after function call {} has address {}".format(a, id(a)))
```

```
Sent [10, 20, 30] list has address 1584173043328
Recieved [10, 20, 30] list has address 1584173043328
Changed [10, 20, 30, 40] list has address 1584173043328
list after function call [10, 20, 30, 40] has address 1584173043328
```

In [42]:

```
# Immutable Example

def func10(arg):
    print ("Recieve {} list has address {}".format(arg, id(arg)))
    arg=(10,20,30)
    print ("Changed {} list has address {}".format(arg, id(arg)))

a=(10,20,30)
print ("Sent {} list has address {}".format(a, id(a)))
func10(a)
print ("list after function call {} has address {}".format(a, id(a)))
```

```
Sent (10, 20, 30) list has address 1584172283392
Recieve (10, 20, 30) list has address 1584172283392
Changed (10, 20, 30) list has address 1584171907008
list after function call (10, 20, 30) has address 1584172283392
```

In [ ]: