```
#functions
```

```
#program for finding the factorial of a number
#facorial of n= n*(n-1)*(n-2)*...........*1
n=5
mul=1
for i in range(1,n+1):
  mul=mul*i
print(mul)
```

```
120
```

```
#5!=5*4*3*2*1=120
```

```
n=6
mul=1
for i in range(1,n+1):
  mul=mul*i
print(mul)
```

```
720
```

```
n=9                         #1
mul=1                       #2
for i in range(1,n+1):      #2
  mul=mul*i                 #2
print(mul)                  #2
```

```
362880
```

```
#defining the function
#1. your code depends on which value? --- n --> function parameter
#2. find out which portion of the code remains the same --> function body  (core logic of a code)
```

```
'''
def function_name ([parameters]):
  functionbody
'''
```

```
'\ndef function_name ([parameters]):\n  functionbody\n'
```

```
#function definition--> def is the keyword for defining the function
def factorial(n):
  mul=1                       #2
  for i in range(1,n+1):      #2
    mul=mul*i                 #2
  print(mul)
```

```
#function calling----> function_name([parameters])
```

```
factorial(9)
```

```
     362880
```

```
#2 types of the functions: in-built function and user-defined function
#in-built functions: print, input, range, strcpy, strlen, strcat
print("hii")
```

```
     hii
```

```
#add, sub, evenodd, factorial
```

```
#No argument, No return statement
def printing():
  print("Welcome to DAIICT")
```

```
printing()
```

```
     Welcome to DAIICT
```

```
printing()
```

```
     Welcome to DAIICT
```

```
printing()
```

```
     Welcome to DAIICT
```

```
#having argument but no return statement
def printing(student_name):
  print("Welcome",student_name, "to DAIICT")
```

```
printing("Robert")
```

```
     Welcome Robert to DAIICT
```

```
printing("Nishith")
```

```
     Welcome Nishith to DAIICT
```

```
printing("Aryan")
```

```
     Welcome Aryan to DAIICT
```

```
#having argument and return statement
def to_print(student_name):
  a="welcome "+student_name+" to DAIICT"    #scope of a is within the function--> a is a local varia
  return a
```

```
to_print("Robert")
#print(a)
```

```
'welcome Robert to DAIICT'
```

```
to_print("Aryan")
```

```
'welcome Aryan to DAIICT'
```

```
#addition of two numbers
def addition(a,b):
  c=a+b
  return c
```

```
addition(5,2)   #print(addition(5,2))
```

```
7
```

```
#no argument with return statement
```

```
def pi():
  return 3.14
```

```
r=5
area=pi()*r*r
print(pi())
print(area)
```

```
3.14
78.5
```

```
import random
def dice():
  dice_list=[1,2,3,4,5,6]
  randon_num=random.choice(dice_list)
  return(randon_num)
```

```
num=dice()
print(num)
```

```
5
```

```
#functions are having positional argument dependencies
```

```
def student_list(name,age,sem):
  print("the student named ",name," of age ",age, "is studying in semester ",sem)
```

```
student_list("robert",sem=2,age=25)
```

```
the student named  robert  of age  25 is studying in semester  2
```

```
#Default Arguments
```

```
def student_list(name,age,sem=1): #default argument
  print("the student named ",name," of age ",age, "is studying in semester ",sem)

student_list("robert",25)
```

```
the student named  robert  of age  25 is studying in semester  1
```

```
student_list("aryan",30)
```

```
the student named  aryan  of age  30 is studying in semester  1
```

```
student_list("aryan",30,3)
```

```
the student named  aryan  of age  30 is studying in semester  3
```

```
#global vs local variable
```

```
def swap(x,y):
  x,y=y,x        #x and y are swapped within this function, so the values are changed within this fu
  print("inside the function, the value of x and y are respectively ",x,y)
  return x,y
```

```
x,y=3,5
print("before calling the swap function, the values of x and y are respectively ",x,y)
swap(x,y)
print("after calling the swap function, the values of x and y are respectively ",x,y)
x=x+1
y=y+1
print("incremented values are ",x,y)
```

```
before calling the swap function, the values of x and y are respectively  3 5
inside the function, the value of x and y are respectively  5 3
after calling the swap function, the values of x and y are respectively  3 5
incremented values are  4 6
```

```
x,y=3,5
print("before calling the swap function, the values of x and y are respectively ",x,y)
x,y=swap(x,y)
print("after calling the swap function, the values of x and y are respectively ",x,y)
x=x+1
y=y+1
print("incremented values are ",x,y)
```

```
before calling the swap function, the values of x and y are respectively  3 5
inside the function, the value of x and y are respectively  5 3
after calling the swap function, the values of x and y are respectively  5 3
incremented values are  6 4
```

```
def listoperation(y):
  for i in range(len(y)):
    y[i]=y[i]+1
  print("within function", y)
```

```
#call by reference
```

```
x=[1,2,3,4,5]
print("before but outside function", x)
listoperation(x)
print("after but outside the function", x)
```

```
before but outside function [1, 2, 3, 4, 5]
within function [2, 3, 4, 5, 6]
after but outside the function [2, 3, 4, 5, 6]
```

```
#lambda function
#function_name=lambda <[parameters]>:function_body
```

```
def mul(arg1,arg2):
  return (arg1*arg2)
```

```
print(mul(5,3))
```

```
15
```

```
mul=lambda arg1,arg2,arg3: arg1*arg2*arg3
```

```
print(mul(5,3,2))
```

```
30
```