```
def MyFunction():
    print("Hello i m a function")
```

```
print("Main starts")
print("Function call")
MyFunction()
print("Main ends")
```

```
Main starts
Function call
Hello i m a function
Main ends
```

```
def max(no1,no2):
    '''Takes 2 inputs and prints maximum of 2'''
    if no1>no2:
        print(no1,"is max")
    else:
        print(no2,"is max")
```

```
max()
```

```
max(10,20)
```

```
20 is max
```

```
max(4.5,9.9)
```

```
9.9 is max
```

```
max("amar","zamar")
```

```
zamar is max
```

```
max(True,False)
```

```
True is max
```

```
def max(no1,no2):
    '''Takes 2 inputs and prints maximum of 2'''
    if no1>no2:
        return no1
    else:
        return no2
```

```
#use above code to print max of 4 numbers
max(max(12,4),max(55,1))
```

```
print("Maximum is:",max(int(input("Enter a number")),int(input("Enter other number"))))
```

```
Enter a number12
Enter other number44
Maximum is: 44
```

```
no1=int(input("Enter a number"))
no2=int(input("Enter a number"))
ans=max(no1,no2)
print("Maximum is:",ans)
```

```
Enter a number88
Enter a number9
Maximum is: 88
```

```
def doubler(no):
    print("in function before changes no:",no)
    no*=2
    print("in function after changes no:",no)
```

```
print("In main")
no=10
print("in main before changes no:",no)
doubler(no)
print("in main after changes no:",no)
```

```
In main
in main before changes no: 10
in function before changes no: 10
in function after changes no: 20
in main after changes no: 10
```

```
def doubler(a):
    print("in function before changes no:",a)
    a[0]*=2
    print("in function after changes no:",a)
```

```
print("In main")
no=[10]#it is a list
print("in main before changes no:",no)
doubler(no)
print("in main after changes no:",no)
```

```
In main
in main before changes no: [10]
in function before changes no: [10]
in function after changes no: [20]
in main after changes no: [20]
```

```
def fact(no):
    if no<=1:
        return 1
    else:
```

```
        return no*fact(no-1)

```

```
fact(4)
```

```
24
```

```
#if no1%no2==0 then no2 is gcd else let no1=no2 and no2 be no1%no2
def gcd(no1,no2):
    if no1%no2==0:
        return no2
    else:
        return gcd(no2,no1%no2)
```

```
gcd(18,12)
```

```
6
```

```
#return ----->gives control,data or both back to other script
#can retun more than one element from function
def Ascending(no1,no2):
    '''Takes 2 inputs and prints maximum of 2'''
    if no1>no2:
        return no2,no1
    else:
        return no1,no2


```

```
ans=Ascending(33,2)
print(type(ans),ans)
```

```
<class 'tuple'> (2, 33)
```

```
def Descending(no1,no2,no3):
    '''Takes 2 inputs and prints maximum of 2'''
    if no1>no2 and no1>no3:
        if no2>no3:
            return no1,no2,no3
        else:
            return no1,no3,no2
    elif no2>no1 and no2>no3:
        if no1>no3:
            return no2,no1,no3
        else:
            return no2,no3,no1
    elif no3>no1 and no3>no2:
        if no1>no2:
            return no3,no1,no2
        else:
            return no3,no2,no1
```

```
Descending(5,6,6)

```

```
def intro(name,native):
    print("hi i am",name,"and from",native)
```

```
intro(native="gujrat",name="jayesh")#key value parameter
```

```
hi i am jayesh and from gujrat
```

```
intro("amar","mumbai")
```

```
hi i am amar and from mumbai
```

```
def intro(name="unnamed",native="unknown"):
    print("hi i am",name,"and from",native)
```

```
intro("amar","mumbai")
```

```
hi i am amar and from mumbai
```

```
intro()
```

```
hi i am unnamed and from unknown
```

```
intro("amar")
```

```
hi i am amar and from unknown
```

```
intro(native="kokan")
```

```
hi i am unnamed and from kokan
```

```
def Hobbies(*hlist):
    print(type(hlist))
    print("Total hobbies:",len(hlist))
    for i in hlist:
        print("->",i)
```

```
Hobbies("code","travel","sketch","read")
```

```
<class 'tuple'>
Total hobbies: 4
-> code
-> travel
-> sketch
-> read
```

```
Hobbies("Eat","sleep")
```

```
<class 'tuple'>
Total hobbies: 2
-> Eat
-> sleep
```

```
def total(*elements):
    sum=0
    for i in elements:
        sum+=i
    return sum
```

```
def total(*elements):
    return sum(elements)
```

```
total(11,22,33,44,55)
```

```
165
```

```
def power(x=1,n=1):
    return x**n
```

```
power(2,0)
```

```
1
```

```
rollno=[1,2,3,4,5,6,7]
i=iter(rollno)
```

```
next(i)
```

```
---------------------------------------------------------------------------
StopIteration                             Traceback (most recent call last)
<ipython-input-67-a883b34d6d8a> in <cell line: 1>()
----> 1 next(i)

StopIteration:
```

SEARCH STACK OVERFLOW

```
def fd(amount,roi):
    y=1
    print("at end of a year:")
    yield (amount*(roi/100)*y)
    y+=1
    print("at end of 2nd year:")
    yield (amount*(roi/100)*y)
    y+=1
    print("at end of 3rd year:")
    yield (amount*(roi/100)*y)
```

```
i=fd(10000,7)
```

```
next(i)
```

```
at end of 2nd year:
2100.0000000000005
```

```
def intro(name,native):
    print("hi i am",name,"and from",native)
```

```
hi=intro
```

```
hi("amar","mumbai")

```

```
hi i am amar and from mumbai
```

```
def eng(name):
    print("hi my dear friend",name,"how are you")
def hindi(name):
    print("नमस्ते मेरे प्रिय मित्र",name,"आप कैसे हैं")
def marathi(name):
    print("नमस्कार माझ्या प्रिय मित्रा",name,"तू कसा आहेस")
```

```
def Welcome(name,lang):
    lang(name)
```

```
Welcome("AMAR",hindi)
```

```
#lambda
# basic syantax :
#lambda parameter/s : expression
```

```
d=lambda a: a*2
```

```
d(100)
```

```
200
```

```
tot=lambda a,b: a+b
```

```
tot(10,20)
```

```
30
```

```
#global variable accessed by all
x=10
def test():
```

```
    print("inside test:x=",x)

def test2():
    x=100#local
    print("inside test2------>x=",x)

def adder():
    global x
    x+=1
    print("inside adder++++++x=",x)
print("outside :x=",x)
test()
test2()
adder()
print("outside :x=",x)

```

```
outside :x= 10
inside test:x= 10
inside test2------>x= 100
inside adder++++++x= 11
outside :x= 11
```

```
test()
```

```
100
```

```
def s(data):
    data+="XXXXXX"
    print("in function:"+data)

```

```
x="python"
s(x)
print("in main:",x)
```

```
in function:pythonXXXXXX
in main: python
```