Functions in python are generic type. Functions with parameters receive any type of values and return any type of values. Parameters are not defined with type. Python is dynamically typed language, type of variables are defined based on value assigned.

def max2(a,b):
    if a>b:
        return a
    else:
        return b
 	
res1=max2(10,20)
res2=max2(1.6,1.8)
res3=max2("ABC","abc")
res4=max2({1,2,3,4},{1,2,3,4,5})
print(res1,res2,res3,res4,sep="\n")
	20
	1.8
	abc
	{1, 2, 3, 4, 5}

Python does not support pass value, it supports only pass by reference. In pass by reference whenever function is called by sending object, python does not send object, it send reference/address of object.

In pass by reference, if object is mutable the changes done within function are reflecting to original object.

Example	**Output**

def fun1(a):
    print(a,id(a))
    a[0]=100
    a[1]=200
 	
list1=[10,20,30,40]
fun1(list1)
print(id(list1),list1)
	[10, 20, 30, 40] 2092955203072
	2092955203072 [100, 200, 30, 40]

Example	**Output**
# Pass by reference	Before Sorting [5, 2, 4, 1, 3]

```
def sort_list(a):
   for i in range(len(a)):
      for j in range(len(a)-1):
         if a[j]>a[j+1]:
            a[j],a[j+1]=a[j+1],a[j]

def append_data(a):
   a.append(10)
   a.append(20)



list1=[5,2,4,1,3]
print(f'Before Sorting {list1}')
sort_list(list1)
print(f'After Sorting {list1}')
list2=[]
print(f'Before apped {list2}')
append_data(list2)
print(f'After append {list2}')
```

```
After Sorting [1, 2, 3, 4, 5]
Before apped []
After append [10, 20]
```

Example

```
def count_even(a):
   c=0
   for value in a:
      if value%2==0:
         c=c+1
   return c

def count_odd(a):
   c=0
   for value in a:
      if value%2!=0:
         c=c+1
   return c



list1=[1,2,3,4,5,6,7,8,9,10]
res1=count_even(list1)
```

Output

```
Even count 5
Odd count 5
```

```python
res2=count_odd(list1)
print(f'Even count {res1}')
print(f'Odd count {res2}')
```

Example

```python
def string_upper(s):
    s1=''
    for ch in s:
        if ch>='a' and ch<='z':
            s1=s1+chr(ord(ch)-32)
        else:
            s1=s1+ch
    return s1


def string_lower(s):
    s1=''
    for ch in s:
        if ch>='A' and ch<='Z':
            s1=s1+chr(ord(ch)+32)
        else:
            s1=s1+ch
    return s1


res1=string_upper("abc")
res2=string_lower("ABC")

print(res1)
print(res2)
```

Output

```
ABC
abc
```

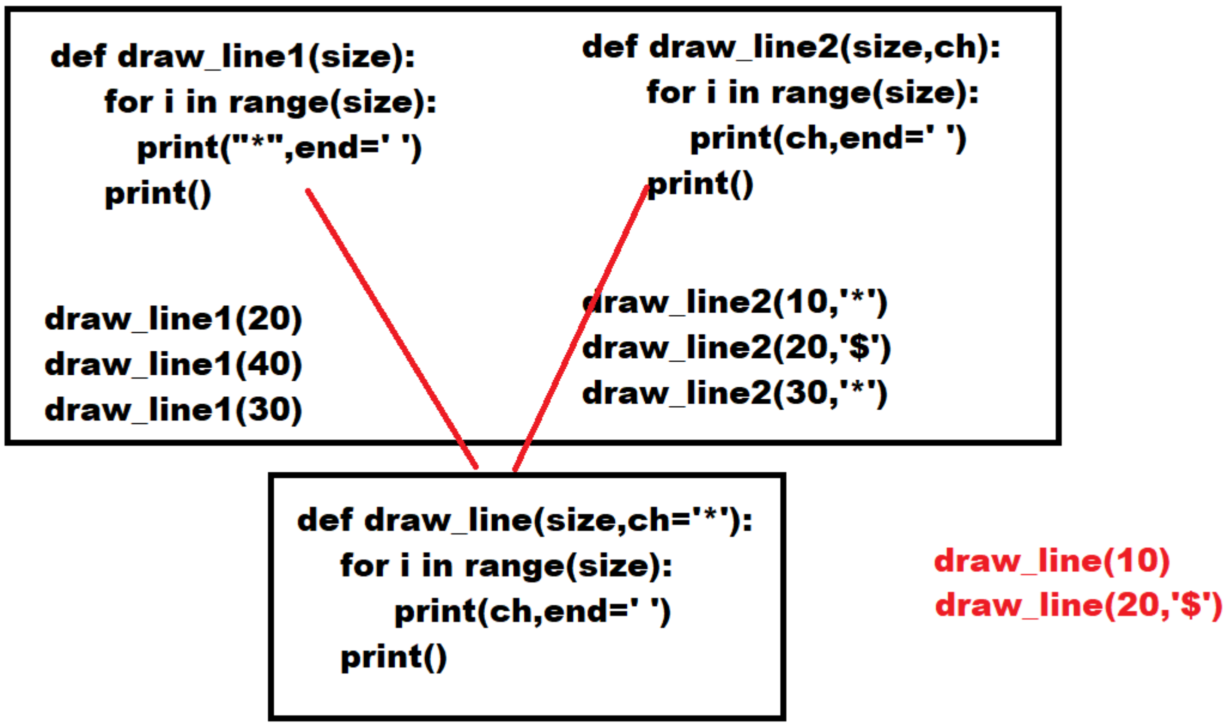
Function with default parameters or arguments or optional arguments

Function with default parameters, does not required values at the time of calling or invoking function. Default parameters are given values at the time of writing or defining function.

Syntax
def <function-name>(req-param,req-param,def-param=value,def-param=value,…):
statement-1
statement-2

Example	**Output**

def fun1(a,b=0):
    print(a,b)
 	
fun1(10,20)
fun1(100)
	10 20
	100 0



```
def draw_line1(size):
    for i in range(size):
      print("*",end=' ')
    print()


draw_line1(20)
draw_line1(40)
draw_line1(30)
```

```
def draw_line2(size,ch):
    for i in range(size):
       print(ch,end=' ')
    print()


draw_line2(10,'*')
draw_line2(20,'$')
draw_line2(30,'*')
```

```
def draw_line(size,ch='*'):
    for i in range(size):
        print(ch,end=' ')
    print()
```

```
draw_line(10)
draw_line(20,'$')
```

Example	**Output**
def draw_line(size=5,ch='*'):	* * * * * * * * * * * * * * * * * * * *

```
    for i in range(size):
        print(ch,end=' ')
    print()


draw_line(20)
draw_line(10,ch='$')
draw_line(30,ch='#')
draw_line()
draw_line(ch='%')
draw_line(size=15)
```

```
$ $ $ $ $ $ $ $ $ $
# # # # # # # # # # # # # # # # # #
# # # # # # # # # # # #
* * * * *
% % % % %
* * * * * * * * * * * * * * *
```

Example

```
def sort_list(a,reverse=False):
    if reverse:
        for i in range(len(a)):
            for j in range(len(a)-1):
                if a[j]<a[j+1]:
                    a[j],a[j+1]=a[j+1],a[j]
    else:
        for i in range(len(a)):
            for j in range(len(a)-1):
                if a[j]>a[j+1]:
                    a[j],a[j+1]=a[j+1],a[j]



list1=[5,2,4,3,1]
print(f'Before Sorting {list1}')
sort_list(list1)
print(f'After sorting in ascending
order {list1}')
sort_list(list1,reverse=True)
print(f'After sorting in descending
order {list1}')
```

Output

```
Before Sorting [5, 2, 4, 3, 1]
After sorting in ascending order
[1, 2, 3, 4, 5]
After sorting in descending order
[5, 4, 3, 2, 1]
```

Example

```
def simple_intrest(amt,t,r=1.5):
    si=(amt*t*r)/100
```

Output

```
900.0
5400.0
```

```
    return si



si1=simple_intrest(5000,12)
print(si1)
si2=simple_intrest(9000,24,r=2.5)
print(si2)
```