In [ ]:
```
d1={"hp":4,"toshiba":7,"macbook":10,"dell":8,"lenovo":9}

key=list(d1.keys())
print(key)

value=list(d1.values())
print(value)

maxi=max(value)

idx=value.index(maxi)
print(idx)

print(key[idx])
```

executed in 8ms, finished 16:39:37 2024-08-21

In [ ]:
```
d1={"hp":4,"toshiba":7,"macbook":10,"dell":8,"lenovo":9}

max_val=0
max_key=""

for key,val in d1.items():
     if val>max_val:
        max_val=val
        max_key=key

print(max_key)
```

executed in 7ms, finished 16:42:34 2024-08-21

Functions:

- It is a block of code used to perform certain operation and helps in code reusability.
- To execute a function , we need to call it.
- You can call the function n no. of times to execute it.
- We create function in python by using a keyword known as def.
  - Syntax:

```
def func_name():
    //Logic
```

- Parts of Function:

1. Function Definition
2. Function Calling

Types of Functions

1. User Defined Functions ---> Functions which we create
2. Inbuilt Functions ----> Functions which are already predefined

In [ ]:
```
def maths():
    a=int(input("Enter a:"))
    b=int(input("Enter b:"))
    print(a+b)
```
executed in 7ms, finished 15:10:03 2024-08-22

In [ ]:
```
maths()
```
executed in 3.87s, finished 15:10:16 2024-08-22

In [ ]:
```
def math(a,b):
    print(a+b)
```
executed in 6ms, finished 15:12:32 2024-08-22

In [ ]:
```
math(10,40)
```
executed in 7ms, finished 15:12:37 2024-08-22

f(x)= x** 2

f(4)= > 16

In [ ]:
```
def f(x):
    print(x**2)
```
executed in 7ms, finished 15:15:21 2024-08-22

In [ ]:
```
f(4)
```
executed in 6ms, finished 15:15:25 2024-08-22

ways to approach Function

1. Function without arguments and without return type
2. Function without arguments and with return type
3. Function with arguments and without return type
4. Function with arguments and with return type

In [ ]:
```
def calci(a,b):
    add=a+b
    sub=a-b
    mul=a*b
    div=a/b

    print( add,mul,sub,div)
```
executed in 6ms, finished 15:24:13 2024-08-22

In [ ]:
```
x=calci(int(input("a:")),int(input("b:")))
```
executed in 2.67s, finished 15:24:16 2024-08-22

51.Write a Python Function to identify the shape using the sides(in m or cm only) entered by user and find the area and perimeter of the identified shape.

Note: Without Using any arguments or parameters.

In [20]:
```
def identify(side1,side2):


    print(side1,side2)

    if "cm" in side1 and "cm" in side2:
        side1=float(side1.split("cm")[0])
        side2=float(side2.split("cm")[0])
        side1=side1*(1/100)
        side2=side2*(1/100)


    elif "cm" in side1 and "m" in side2:
        side1=float(side1.split("cm")[0])
        side2=float(side2.split("m")[0])
        side1=side1*(1/100)


    elif "m" in side1 and "cm" in side2:
        side1=float(side1.split("m")[0])
        side2=float(side2.split("cm")[0])
        side2=side2*(1/100)

    elif "m" in side1 and "m" in side2:
        side1=float(side1.split("m")[0])
        side2=float(side2.split("m")[0])

    else:
        print("Invalid inputs")

    if side1==side2:
        print("Identified shape is Square")
        print("Area of square is {} m2".format(side1*side1))
        print("Perimeter of square is {} m" .format(4*side1))

    elif side1!=side2:
        print("Identified Shape is Rectange")
        print("Area of Rectangle is {} m2".format(side1*side2))
        print("Perimeter of Rectangle is {} m".format(2*(side1+side2)))
    else:
        print("Shape cannot be identified")
```

executed in 17ms, finished 16:04:21 2024-08-22

In [21]:
```
side1=input("Enter side1:")
side2=input("Enter side2:")
identify(side1,side2)
```

executed in 2.63s, finished 16:04:24 2024-08-22

```
Enter side1:10m
Enter side2:10m
10m 10m
Identified shape is Square
Area of square is 100.0 m2
Perimeter of square is 40.0 m
```

In [ ]:
```

```