FUNCTION (METHOD)

 # -- What is function? full Explaination.

Function is nothing but group of statements which only executes when we call the function.

If a group of statements is repeatedly required then it is not recommanded to write these statement everytime seprately. we have to define these statements as a single unit and we can call that unit any number of times based on our requirement without rewriting.

This unit is nothing but function.

The main advantage of the function is code reusability

note: -

1) code reusablity
2) functions returns value as a result
3) we can pass value to the functions called parameter.

Types of function: -

1. **pre-define/ built-in function**
   - len()
   - print()
   - int()
   - etc...
2. **user-defined function**
   - code()
   - add()
   - fact()

syntax: -

```
def function-name:    # def - define
    code statement
```

pre-define function: -

---

len()

Ex: -

```
print("this is a print function")


print(len("something"))
```

#result:

9

user-defined function: - we create to fulfill our business requirement.

direct without using function

 # 8,4 -> add, sub, mul, div

```
a=8
b=4
print("Addition:",8+4)
print("Sunbtraction:",8-4)
print("Multiplication:",8*4)
print("Division:",8/4)

a=7
b=13
print("Addition:",7+13)
print("Sunbtraction:",7-13)
print("Multiplication:",7*13)
print("Division:",7/13)
```


8983226463
Cbangp31@gmail.com

using function

```
def function_name(parameter,parameter,....,n)
    "body"
    "body"
    "body"
    "body"
    return value
```

Ex: 1

```
def calculator(a,b)
    print("Addition:",a+b)
    print("Sunbtraction:",a-b)
    print("Multiplication:",a*b)
    print("Division:",a/b)

#Function calling
calculator(8,4)
calculator(10.20)
calculator(18,4.6)
calculator(10.9.20.1)
```

Ex: 2

```
def greeting():
    ##print("Good Morning")
    2+2

print(greeting())

# Output Show:  none
```

Ex: 3

```
def greeting():
    print("Good Morning")


msg=greeting()
print(msg)
```

Ex: 4

```
def greeting(name)
    print("Good Evening", name)


greeting("CodeBlock")

# Output Show:  Good Evening CodeBlock
```

Types Of Functions Arguments:

---

1. position:

-> order should be maintain

-> no. of parameter = n0. of argument

Ex. 1:

```
def greeting(fname,lname)
    print("Good Evening", fname, lname)

greeting("Code","Block")
```

Note:- change the parameter value

2. keyword:

-> order is not inportant

-> no. of parameter = n0. of argument

-> keyowrd is set the passing value order of argument

-> keyword argument should follow positional arguments

Ex. 1:

```
def greeting(fname,lname)
    print("Good Evening", fname, lname)

greeting(fname="Code",lname="Block")
```

3. default argument:

-> order is important

-> no. of parameter may not be equal no. of argument

Ex. 1:

```
def greet(fname="guest"):
    print("good evening",fname)

greet()
#output show:   good evening guest
```

Ex. 2:

```
def greet(fname="guest"):
    print("good evening",fname)

greet("Roy")
#output show:   good evening Roy
```

4. Variable Length Argument

```
def test(*args):
    print(args)
    print(len(args))
    print(type(args))

test("india","Shri Lanka","US")
#output show:
    ("india","Shri Lanka","US")
    3
    <class 'Tuple'>
```

5. Using Keyword Argument: -

➔ Don't Identify the no. of Argument than used kw argument function.

```
def test(**kwargs):          # kw-keyword modules
    print(kwargs)
    print(len(kwargs))
    print(type(kwargs))

test country="india",fname="Viraj",lname="Vishal")

#output show:
    [country:"india",fname:"Viraj",lname:"Vishal"]
    3
    <class 'Dict'>
```