In [ ]:

```
Two Requirements are There --->
Find the sum of 20 and 30
Find the sum of 200 and 300

Python Program to print sum of two numbers(20,30)
x=20
y=30
z=x+y
z
```

In [ ]:

```
Python Program to print sum of two numbers(200,300)
x=200
y=300
z=x+y
z
```

In [ ]:

```
def add(a,b):
    return a+b
print(add(20,30))
print(add(200,300))
```

What is Function

In [ ]:

```
Function is a block of code that is used to perform a specific task or a logic.
Business logic

Benefit of Functions --> Code Reuseability
```

Example

In [1]:

```
def add(a,b):
    return a+b
print(add(20,30))
print(add(200,300))
```

```
50
500
```

Types of Functions in Python:

In [ ]:

```
Types of Functions in Python:
    1.Userdefined Functions --> functions that are developed by the developer as per business
                                requirements.
    Example:
        add()
        subtract()



    2.Builtin Functions --> Functions that are already defined by the python virtual machine
                            we need not to define those function we will simply call those functions
                            and use that function.

    Example:
        id() --> address of an object.
        type() --> Type of an object
        print() --> display the data on console
        len() -->
        max()-->
        min() -->
```

Example of Builtin Function

In [2]:

```
x=10
print(type(x))
print(id(x))
```

```
<class 'int'>
1493263673936
```

Syntax of User Defined Function

In [ ]:

```
1.For defining any function in Python you need to use def keyword

def function_name(paramter_list):
    return None

While creating a function two keywords are  very important:
    1. def --> def is mandatory for defining any function
    2.return --> return is optional (if you are not giving any return statement in your python
                                     function the PVM will return None
```

Example of User Defined Function

In [3]:

```
def wish():                     #function declearation
    print("GOOD EVENING")       #Function definition
    return "hello"
print(wish())    #Calling function
```

```
GOOD EVENING
hello
```

Parameters

In [ ]:

```
Parameters are the inputs for the functions based and that parameters our function will work
if we are giving any parameters while defining the function then it is very compulsory to give the
values for that parármeters while calling that function.

Note --> whatever arguments you have given while declaring a function and the arguments that
         are passed while calling the function both must be same otherwise you will get an error.

Two types of parameters:
    1.Actual Parameters
    2.Formal Parameter
```

Example

In [4]:

```
a=10
b=20
c=a+b
```

In [5]:

```
def summation(a,b):    #a,b are formal parameters.
    c=a+b
    print(c)
print(summation(100,200))    #100,200 are actual parameters.

#If you are not giving return statement then pvm will return None.
```

```
300
None
```

Return Keyword

In [ ]:

```
Return --> functions can take inputs in form of parameters and execute business logic and return
output for the caller function with return Statement.

Note --> We can return more than one argument from a function but for each function only one return
         statement will be executed. After return statement no any statement will be executed.
```

Example

In [6]:

```
def calculate(x,y):
    add=x+y
    sub=x-y
    mul=x*y
    div=x//y
    mod=x%y
    return add,sub,mul,div,mod

x=calculate(10,20)
print(x)
for i in x:
    print("Calculator details are",i)

Note --> If you are returing more than one element from a function then whole elements
         will be stored in a tuple
```

```
(30, -10, 200, 0, 10)
Calculator details are 30
Calculator details are -10
Calculator details are 200
Calculator details are 0
Calculator details are 10
```

Practice Problems on Function

Python Program to Find the Ascii Value of Each Character of a String.

In [ ]:

```
c = input("Enter a String")
for i in c:
    print(ord(i),end=" ")
```

In [17]:

```
def ascii(c):
    for i in c:
        print(ord(i),end=" ")
    return ""
c1=input("Enter a string")
print(ascii(c1))
```

```
Enter a stringhello
104 101 108 108 111
```

In [18]:

```
l=[]
def ascii(c):
    for i in c:
        l.append(ord(i))
    return l
c1=input("Enter a string")
print(ascii(c1))
```

```
Enter a stringhello
[104, 101, 108, 108, 111]
```

Python Program to Check weather a Given Number is Odd or Even

In [20]:

```
#odd even
def odd_even(number):
    if number%2==0:
        return "It is an even Number"
    else:
        return "It is an odd Number"
number=int(input("Enter a number"))
print(odd_even(number))
```

```
Enter a number19
It is an odd Number
```

Python Program to Check weather a Given Number is Prime or Not.

In [7]:

```
def prime(n):
    if n==1:
        return "1 is neither Prime nor Composite"
    elif n>1:
        for i in range(2,n):
            if n%i==0:
                return "Not a prime Number"
                break
        else:
            return "It is a prime Number"
    else:
        return "It is not a prime number"

print(prime(199))
```

```
It is a prime Number
```

Python Program to Check weather a Given Number is Strong or not.

In [8]:

```
def fact(n):
    fact=1
    for i in range(1,n+1):
        fact=fact*i
    return fact
```

In [9]:

```
def strong(number):
    m=number
    sum=0
    while number!=0:
        rem=number%10 # -->5 -->4 -->1
        sum=sum+fact(rem) #--> 0+120=120--> 120+24=144 -->144+1=145
        number=number//10  #-->14 -->1 -->0
        print("Each case reminder",rem)
        print("Each case sum",sum)
        print("Each case digits",number)
        print("At last number value is",number)
        print("Value of Userinput number",m)
    if(sum==m):
        return("Number is strong")
    else:
        return("Number is not strong")
print(strong(145))
```

```
Each case reminder 5
Each case sum 120
Each case digits 14
At last number value is 14
Value of Userinput number 145
Each case reminder 4
Each case sum 144
Each case digits 1
At last number value is 1
Value of Userinput number 145
Each case reminder 1
Each case sum 145
Each case digits 0
At last number value is 0
Value of Userinput number 145
Number is strong
```

Argument Passing Techniques

In [ ]:

```
Types of paramters:
    Actual --> while calling the function
    Formal --> while declaring the function
```

In [ ]:

```
Actual Parameters are divided into 4 categories(Argument passing techniques):
    1.Positional Argument
    2.Keyword Argument
    3.Default Argument
    4.Variable length argument
```

Positional Argument

In [ ]:

```
1.Positional Argument -->these are the arguments passed to the function with the correct positional
                         order.
```

Example

In [14]:

```
def calculate(y,x,b,a):  #Formal

    return x,y

x=calculate(20,10,100,200)    #Actual Parameter
print(x)
```

```
(10, 20)
```

In [13]:

```
def calculate(x,y,b,a):  #Formal

    return x,y,a,b

x=calculate(20,10,30,50)    #Actual Parameter
print(x)
```

```
(20, 10, 50, 30)
```

Keyword Argument

In [37]:

```
Keyword Argument --> while calling the function you are going to give the keys for each argumnt
Note --> In keyword argument all formal parameters value is given by the keys at the caller function
```

```
(10, 20, 30)
```

Example

In [15]:

```
def calculate(x,y,b):  #Formal

    return x,y,b

x=calculate(b=30,y=20,x=10)    #Actual Parameter
print(x)
```

```
(10, 20, 30)
```

Default Argument

In [ ]:

```
default argument --> sometimes we need to provide few default values for our arguments

Note --> While declaring a function if we are deefining anything then it is treated defult values.
```

Example

In [17]:

```
def wish(a=10,name="Jappan"):
      return ("Hello "+ str(name) + " How are you " ) , a
wish(200,300)
```

Out[17]:

```
('Hello 300 How are you ', 200)
```

Variable Length Arguments

In [ ]:

```
Variable Length Argument --> if we don't know how many argument we need to pass while calling a
                             function then we should use variable length argument.

Two Types of Variable Length Argument :
    1. **args
    2.**Kwargs
```

**args

In [ ]:

```
** args --> If you are passing multiple elements while calling the function in positional order then
            all the elements that are passed while calling a function will be considered as a Tuple.
            and **args will work with any number of Number As internally it is forming a Tuple.
```

Example

In [19]:

```
def sums(a,*n):
    print(type(n))
    return n
print(sums(10,"Python"))
print(sums(10,20))
print(sums(10,20,30))
print(sums(10,20,30,40))
```

```
<class 'tuple'>
('Python',)
<class 'tuple'>
(20,)
<class 'tuple'>
(20, 30)
<class 'tuple'>
(20, 30, 40)
```

**kwargs

In [ ]:

```
**kwargs(dict) -->If you are passing multiple elements while calling the function in form of keyword
                  arguments then all the elements that are passed while calling a function will be
                  considered as a Dictionary and **args will work with any number of Number As
                  internally it is forming a Dictionary.
```

Example

In [22]:

```
def sums(**kwargs):
    print(type(kwargs))
    print(kwargs)
    return kwargs.items()
print(sums(x=10,y=20))
print(sums(x=200,y=3020,z=400))
print(sums(x=10,y=20,z=30))
print(sums(s=10,p=20,z=30,h=40))
```

```
<class 'dict'>
{'x': 10, 'y': 20}
dict_items([('x', 10), ('y', 20)])
<class 'dict'>
{'x': 200, 'y': 3020, 'z': 400}
dict_items([('x', 200), ('y', 3020), ('z', 400)])
<class 'dict'>
{'x': 10, 'y': 20, 'z': 30}
dict_items([('x', 10), ('y', 20), ('z', 30)])
<class 'dict'>
{'s': 10, 'p': 20, 'z': 30, 'h': 40}
dict_items([('s', 10), ('p', 20), ('z', 30), ('h', 40)])
```

SwapCase --> it converts all upper caase to lowrcase and lowercse to uppercase

In [23]:

```
x="Hello World"
x.swapcase()
```

Out[23]:

```
'hELLO wORLD'
```

In [ ]: