Function required or required positional arguments
Required arguments required values at the time invoking or calling function. If values of required arguments are not given python translator generate TypeError.

Syntax:
```
def <function-name>(arg-name,arg-name,arg-name,…):
        statement-1
        statement-2
```

Example:
```
def add(a,b):
    print(f'Sum of {a} and {b} is {a+b}')


add(100,200)
add(10,20)
```

Output:
```
Sum of 100 and 200 is 300
Sum of 10 and 20 is 30
```

return keyword
A function not only receives values, it also returns value. A function returns value using “return” keyword. After returning value to calling place, it terminates execution of function. Return keyword returns one object.

```
return <expression>/<variable-name>/<object-name>
```

Example:
```
def add(a,b):
    c=a+b
    return c


res1=add(100,200)
res2=res1/2
print(f'Sum is {res1}')
print(f'Avg is {res2:.2f}')
```

Output:
```
Sum is 300
Avg is 150.00
```

Example:
```
def fun1():
    return 10,20,30,40,50



a=fun1()
print(a)
```

Output:
```
(10, 20, 30, 40, 50)
```

Example:
```
def simple_interest(amt,t,rate):
    si=(amt*t*rate)/100
    return si


res1=simple_interest(5000,12,1.5)
res2=simple_interest(9000,24,2.0)
res3=simple_interest(rate=1.8,amt=9000,t=20)
print(f'Simple Interest {res1:.2f}')
print(f'Simple Interest {res2:.2f}')
print(f'Simple Interest {res3:.2f}')
res4=max(10,20)
print(f'Maximum is {res4}')
```

Output:
```
Simple Interest 900.00
Simple Interest 4320.00
Simple Interest 3240.00
Maximum is 20
```

Example:
```
def add(n1,n2):
    n3=n1+n2
    return n3


res1=add(100,200)
res2=add(1.5,2.5)
res3=add(1+2j,1+3j)
res4=add("Python","Language")
print(res1,res2,res3,res4)
```

Output:
```
300 4.0 (2+5j) PythonLanguage
```

Example:
```
# Program with type hint
def add(num1:int,num2:int):
    num3=num1+num2
    return num3

res1=add(100,200)
print(res1)
res2=add(1.5,2.5)
print(res2)
```

Output:
```
300
4.0
```

Example:
```
def isEven(num):
    return num%2==0

def isPrime(num):
    c=0
    for i in range(1,num+1):
        if num%i==0:
```

```
        c=c+1
    return c==2

def str_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


res1=isEven(4)
res2=isEven(9)
res3=isPrime(7)
res4=isPrime(10)
res5=str_upper("python")
print(res1,res2,res3,res4,res5)
```

Output:

```
True False True False PYTHON
```

Python does not support function overloading

Defining more than one function with same name with different number of arguments is called function overloading.

```
def fun1():
    print("inside fun1 without arguments")
def fun1(a):
    print("inside fun1 with argument")
def fun1(a,b):
    print("inside fun1 with 2 arguments")


fun1()
fun1(100)
fun1(100,200)
```

Output:

```
Traceback (most recent call last):
  File "C:\Users\nit\PycharmProjects\pythonProject1\funtest20.py", line 9, in
<module>
    fun1()
TypeError: fun1() missing 2 required positional arguments: 'a' and 'b'
```

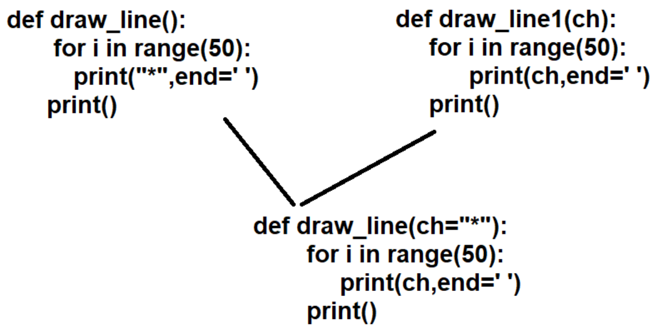
When more than one function written with same name, old function replaced with new function, old function is not available.

Default arguments or optional arguments

Default arguments are given values at the time writing function. If values for these arguments are not given at the time invoking or calling function, PVM assign default value. If values are given at time of invoking function PVM assign given values.

Syntax:

```
def <function-name>(arg-name,arg-name,arg-name=value,arg-
name=value,…):
        statement-1
        statement-2
```



```
def draw_line():
    for i in range(50):
      print("*",end=' ')
    print()
```

```
def draw_line1(ch):
    for i in range(50):
        print(ch,end=' ')
    print()
```

```
def draw_line(ch="*"):
    for i in range(50):
        print(ch,end=' ')
    print()
```

Example:

```
def draw_line(ch='*'):
    for i in range(20):
        print(ch,end='')
    print()
```

```
draw_line()
draw_line("$")
```

Output:

```
********************
$$$$$$$$$$$$$$$$$$$$
```