Default arguments or Optional arguments

Default arguments are given values at the time writing function definition. If the values for these arguments are not given at the time of invoking function, python virtual machine assigns default values.

Syntax:

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

Example:

```
# function with default arguments
def fun1(x=100,y=200):
    print(x,y)


fun1()
fun1(50)
fun1(10,20)
fun1(y=400)
```

Output:

```
100 200
50 200
10 20
100 400
```

Example:

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

si1=simple_interest(10000,12)
si2=simple_interest(5000,24,2.0)
print(f'simple interest {si1:.2f}')
print(f'simple interest {si2:.2f}')
```

Output:

```
simple interest 1800.00
```

simple interest 2400.00

Example:

```
def sort(l,reverse=False):
    if reverse==False:
        for i in range(len(l)):
            for j in range(len(l)-1):
                if l[j]>l[j+1]:
                    l[j],l[j+1]=l[j+1],l[j]
    elif reverse==True:
        for i in range(len(l)):
            for j in range(len(l)-1):
                if l[j]<l[j+1]:
                    l[j],l[j+1]=l[j+1],l[j]



list1=[3,8,4,7,1,9,2,7,3]
sort(list1)
print(list1)
sort(list1,reverse=True)
print(list1)
```

Output:

```
[1, 2, 3, 3, 4, 7, 7, 8, 9]
[9, 8, 7, 7, 4, 3, 3, 2, 1]
```

Example:

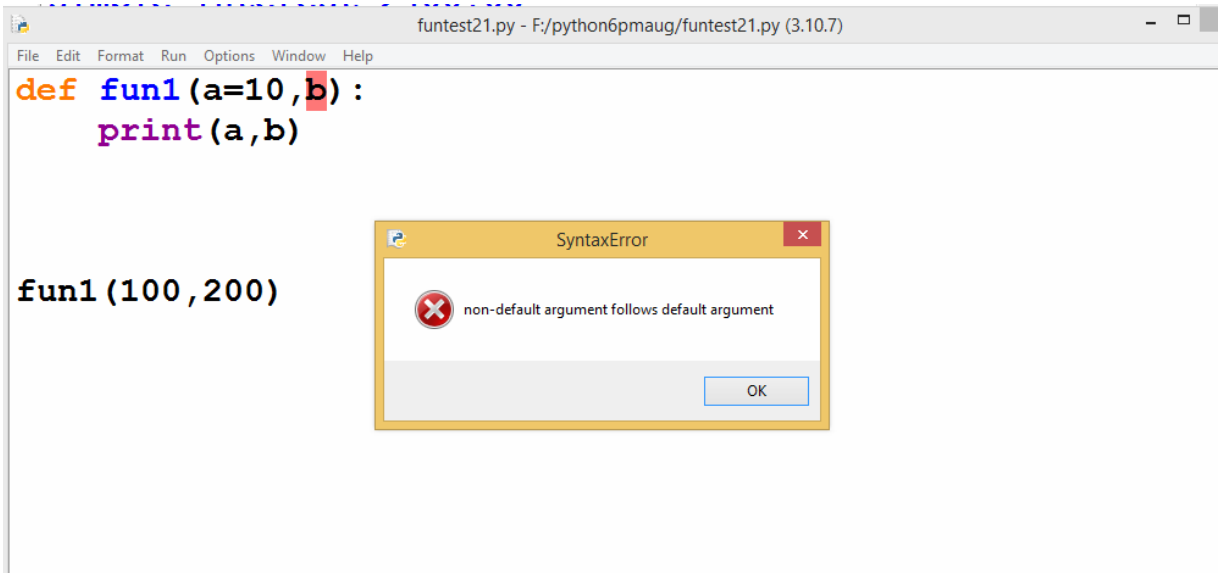
```
def string_conversion(s,ctype="upper"):
    if ctype=="upper":
        return s.upper()
    elif ctype=="lower":
        return s.lower()
    elif ctype=="title":
        return s.title()
    elif ctype=="capitalize":
        return s.capitalize()
    else:
        return s
```

```
def main():
    str1=string_conversion("python")
    str2=string_conversion("PYTHON",ctype="lower")
    str3=string_conversion("python language",ctype="title")
    str4=string_conversion("python",ctype="capitalize")
    print(str1,str2,str3,str4)
main()
```

Output:

PYTHON python Python Language Python

Order of defining default and non-default arguments



When function is defined with required arguments/non default and default argument. First define non default arguments and default arguments.

Example:

```
def draw_line(ch="*",l=30):
    print(ch*l)



draw_line()
draw_line('$')
```

```
draw_line(l=20)
draw_line('#',20)
```

Output:

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

Variable length arguments

Variable length argument receives 0 or more values.
Variable length argument is type of tuple
Variable length argument is defined with prefix *
A Function can is defined with one variable length argument.

```
def maximum(a,b):
    if a>b:
      return a
    else:
      return b


c=maximum(10,20)
print(c)
```

```
def maximum3(a,b,c):
    if a>b and a>c:
       return a
    elif b>a and b>c:
       return b
    else:
       return c


  d=maximum3(10,20,30)
  print(d)
```

```
def maximum4(a,b,c,d):
    ....
    .....
```

Syntax:

```
def function-name(req-arg,*var-arg,def-arg=value):
        statement-1
        statement-2
```

Example:

```
def fun1(*a):
   print(a)
   print(type(a))

fun1()
fun1(10,20,30,40,50)
fun1(1,"naresh","python")
```

Output:

```
()
```

```
<class 'tuple'>
(10, 20, 30, 40, 50)
<class 'tuple'>
(1, 'naresh', 'python')
<class 'tuple'>
```

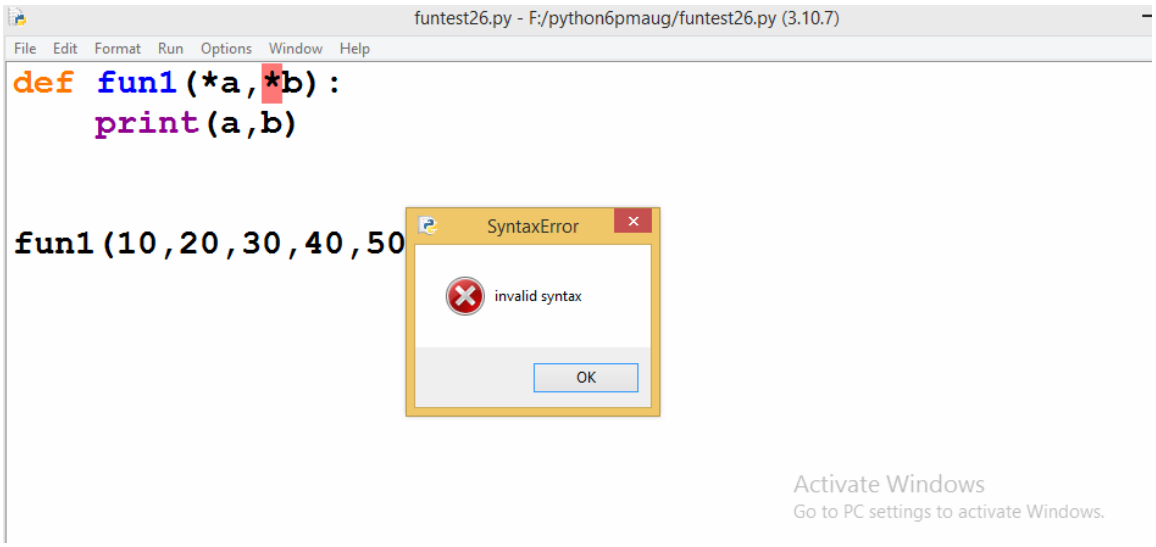
Example:

```
def maximum(*n):
    m=0
    for value in n:
        if value>m:
            m=value
    return m


res1=maximum()
res2=maximum(10,20)
res3=maximum(1,2,3,4,5,6,7,10,4,8,9)
print(res1,res2,res3,sep="\n")
```

Output:

```
======== RESTART: F:/python6pmaug/funtest25.py ========
0
20
10
```



Example:

```
def fun1(*a,b):
    print(a,b)

def fun2(b,*a):
    print(b,a)

def fun3(a,*b,c=100):
    print(a,b,c)

fun1(10,20,30,40,50,b=60)
fun2(10,20,30,40,50)
fun3(10)
fun3(10,20,30,40)
fun3(10,20,30,40,50,c=200)
```

Output:

```
(10, 20, 30, 40, 50) 60
10 (20, 30, 40, 50)
10 () 100
10 (20, 30, 40) 100
10 (20, 30, 40, 50) 200
```