Python Programs on functions

Note:
Don’t copy paste these programs to check the output, instead type the program and execute.

What is the output ?

```
a=['1','2','3']
print(list(map(len,a)))

a=[1,2,3]
print(list(map(len,a)))

a=[[1],[2],[3]]
print(list(map(len,a)))

k=30
print(len(k))
k="30"
print(len(k))
```

What is the output ?

```
def double(y):
  y = 2 * y

def changeit(lst):
  lst[0] = "amar"
  lst[1] = "akbar"

y = 5
double(y)
print(y)

mylst = ['106', 'students', ‘anthony’]
changeit(mylst)
print (mylst)
```

What is the output ?

```
def display(data):
    print(data)
    del data[0]
def show(data):
    print(data)

a=[5,6,7,8]
display(a)
show(a)
```

What is the output ?

```
def display(data):
    print(data)
    del data
def show(data):
    print(data)

a=5678
display(a)
show(a)
```

What is the output ?

```
def extendList(val, list=[]):
    list.append(val)
    return list

list1 = extendList(10)
print(list1)
```

What is the output ?

```
def extendList(val,listdata=[]):
    listdata.append(val)
    return(listdata)

list1 = extendList(10)
print(list1)

list2 = extendList(123)
print(list2)

k=[]
list3 = extendList(2222,k) # k is a different and new list
print(list3)
```

Listdata is a function argument, bound between function calls, and hence values remain,
It is pass by ref, any mod to dummy , will change the original

What is the output ?

```
def extendList(val, list=[]):
    list.append(val)
    return list

list1 = extendList(10)
list2 = extendList(123,[])
print(list1,list2)
```

What is the output ?

```
def extendList(val, listdata=[]):
    listdata.append(val)
    return listdata

list1 = extendList(10)
list2 = extendList(123,[])
list3 = extendList('a')

print(list1,list2,list3)
```

What is the output ?

```
import functools

n = '1729'

print(functools.reduce(lambda x,y:x+y,n))  #string concat


print(functools.reduce(lambda x,y:int(x)+int(y),n)) #addition


print(functools.reduce(int.__add__ , map(int, n)))
```

What is the output ?

```
def foo(x, a = []) :
   a.append(x)
   print(a)

foo(10) #[10]
foo(20) #[10, 20]

z = [30, 40]
foo(50, z) #[30, 40, 50]
foo(60) # [10, 20, 60]
foo(100,z)
```

What is the output ?

```
a=[1,2,3,4]
res=map(lambda x:x*x,a)
print(list(res))
print(list(res)) #you can walk thr map object only once
```

What is the output ?

```
def extendList(val,listdata=[]):
    listdata.append(val)
    return listdata

list1 = extendList(10)
list2 = extendList(123,list1)
print(list1,list2)
```

What is the output ?

```
def check(data):
    data.append(10000)



a=[1,2,3]
b=[4,5,6]
check(a)
check(b)
print(a)
print(b)
```

What is the output ?

```
def foo(x, a = []) :
    a.append(x)
    print(a)

foo(10)
foo(20)
z = [30, 40]
foo(50, z)
foo(60)
y=[90,100]
foo(y)
foo(200)
```

What is the output ?

```
a=[1,2,3]
b=a
a[0]=1000  //modification through the list
print(b)
print(a)
```

```
a=[1,2,3]
b=a
a=[7,8,9]
print(b)
print(a)
```

What is the output ?

```
a=[1,2,3]
b=a
a=[1000]
#a refers to different list, b does not change
print(b)
print(a)
```

What is the output ?

```
def fun(a):
    a[0]=1000


x=[1,2,3]
fun(x)
print(x)
```

What is the output ?

```
def fun(a):
    a=[7,8,9]

x=[1,2,3]
fun(x)
print(x)
```

What is the output ?

```
def fun(a):
    a.append([7,8,9])

x=[1,2,3]
fun(x)
print(x)
```

What is the output ?
Shallow copy:pass by reference

```
def change(list):

    list.extend([13,21,34])

 fib = [0,1,1,2,3,5,8]
print("before",fib)
change(fib) //sending actual or original
print ("after",fib)
```

What is the output ?
Deep copy:pass by value

```
def change(list):

    list.extend([13,21,34])



fib = [0,1,1,2,3,5,8]
print("before",fib)
change(fib[::-1]) //make a copy and send
print ("after",fib)
```

What is the output ?

```
a=[1,2,3]
data=map(lambda x:x*x,a)
print(list(data))
print(list(data))

data=filter(lambda x:x%2==0,a)
print(list(data))
print(list(data))
```

What is the output ?

```
def f1():
    data=10
    def f2():
        data=data+20 //error
        print(data)
    f2()
f1()
```

What is the output ?

```
def f1():
    data=10
    print(data)
    def f2():
        data=30
        data=data+20
        print(data)
    f2()
f1()
```

What is the output ?

```
def f1():
    data=10
    def f2():
        nonlocal data
        data=data+20 #no error, uses outer variable
        print(data)
    f2()
f1()
```

What is the output ?

```
my_dict = {'x':500, 'y':5874, 'z': 560}


key_max = max(my_dict.keys(), key=(lambda k: my_dict[k]))

key_min = min(my_dict.keys(), key=(lambda k: my_dict[k]))


print('Maximum Value: ',my_dict[key_max])

print('Minimum Value: ',my_dict[key_min])

print(‘key and Maximum Value: ',key_max,my_dict[key_max])
```

What is the output ?

```
a = 1
def f1():

    a = 5 #global variable is changed to 5
    print (a) #will print 5
    def f2():
        global a
        a=20
    f2()

f1()

print (a)
```

What is the output ?

```
a = 1
def f1():

    a = 5 #global variable is changed to 5
    print (a) #will print 5
    def f2():
          global a
          a=20



f1()
print (a)
```

What is the output ?

```
a = 1
def f1():
    global a
    a = 5 #global variable is changed to 5
    print (a) #will print 5
    def f2():
        global a
        a=20
    f2()

f1()

print (a)
```

What is the output ?

```
def extendList(val):
    listdata=[]
    listdata.append(val)
    return(listdata)

list1 = extendList(10)
print(list1)

list2 = extendList(123)
print(list2)
```

Listdata is a local copy, not bound between function calls
Listdata is available within extendlist