Exercise

1. For the given program, find the memory address(es) of the variable *n* for 5 iterations. Note down the observations.

```
#print in triangle
x=int(input("enter the number"))
for n in range(0,x):
    n +=1
    print ("*" *(0+n))

for n in range(-x,0):
    n +=1
    print ("*" *(0-n+1))
```

```
1  x = int(input("Enter the number:" ))
2
3  print("First loop:")
4  for n in range(0,x):
5      print(f"Iteration {n} , n = {n},id(n) = {id(n)}")
6      n += 1
7      print(f"After increment , n = {n},id(n) = {id(n)}")
8      print("*" * (0+n))
9  print("Second loop:")
10 for n in range(-x,0):
11     print(f"Iteration {n} , n = {n},id(n) = {id(n)}")
12     n += 1
13     print(f"After increment , n = {n},id(n) = {id(n)}")
14     print("*" * (0-n+1))
15
16
17 OUTPUT:
18
19 Enter the number:5
20 First loop:
21 Iteration 0 , n = 0,id(n) = 140730440053512
22 After increment , n = 1,id(n) = 140730440053544
23 *
24 Iteration 1 , n = 1,id(n) = 140730440053544
25 After increment , n = 2,id(n) = 140730440053576
26 **
27 Iteration 2 , n = 2,id(n) = 140730440053576
28 After increment , n = 3,id(n) = 140730440053608
29 ***
30 Iteration 3 , n = 3,id(n) = 140730440053608
31 After increment , n = 4,id(n) = 140730440053640
32 ****
33 Iteration 4 , n = 4,id(n) = 140730440053640
34 After increment , n = 5,id(n) = 140730440053672
35 *****
36 Second loop:
37 Iteration -5 , n = -5,id(n) = 140730440053352
38 After increment , n = -4,id(n) = 140730440053384
39 *****
40 Iteration -4 , n = -4,id(n) = 140730440053384
41 After increment , n = -3,id(n) = 140730440053416
42 ****
43 Iteration -3 , n = -3,id(n) = 140730440053416
44 After increment , n = -2,id(n) = 140730440053448
45 ***
46 Iteration -2 , n = -2,id(n) = 140730440053448
47 After increment , n = -1,id(n) = 140730440053480
48 **
49 Iteration -1 , n = -1,id(n) = 140730440053480
50 After increment , n = 0,id(n) = 140730440053512
```

2. Write a Python script to concatenate following dictionaries to create a new one. Find the addresses of the three dictionaries and the concatenated dictionary as well. Note you're your observations. Can you find the address of individual key-value pair?
Sample Dictionary :

```
dic1={1:10, 2:20}
dic2={3:30, 4:40}
dic3={5:50,6:60}
```

Expected Result : {1: 10, 2: 20, 3: 30, 4: 40, 5: 50, 6: 60}

```python
dic1 = {1:10, 2:20}
dic2 = {3:30, 4:40}
dic3 = {5:50, 6:60}

# concatenate dictionaries
dic = {}
dic.update(dic1)
dic.update(dic2)
dic.update(dic3)

# print the concatenated dictionary
print("Concatenated Dictionary:",dic)

# memory address of the dictionaries
print("Memory Addresses:")
print("dic1:",id(dic1))
print("dic2:",id(dic2))
print("dic3:",id(dic3))
print("Concatenated dictionary:",id(dic))

# Adresses of individual key value pairs
print("Addresses of individual key value pairs in the concatenated dictionary:")
for key,value in dic.items():
    print(f"key: {key} , id(key) : {id(key)} | value: {value} , id(value) : {id(value)}")


OUTPUT:

Concatenated Dictionary: {1: 10, 2: 20, 3: 30, 4: 40, 5: 50, 6: 60}
Memory Addresses:
dic1: 2320228961920
dic2: 2320231265856
dic3: 2320231264896
Concatenated dictionary: 2320231267008
Addresses of individual key value pairs in the concatenated dictionary:
key: 1 , id(key) : 140729282884392 | value: 10 , id(value) : 140729282884680
key: 2 , id(key) : 140729282884424 | value: 20 , id(value) : 140729282885000
key: 3 , id(key) : 140729282884456 | value: 30 , id(value) : 140729282885320
key: 4 , id(key) : 140729282884488 | value: 40 , id(value) : 140729282885640
key: 5 , id(key) : 140729282884520 | value: 50 , id(value) : 140729282885960
key: 6 , id(key) : 140729282884552 | value: 60 , id(value) : 140729282886280
```

3. Write down a python script to remove duplicates from the list. Note down the addresses of list before and after the removal of duplicates.

```
l = [1,2,3,4,5,1,2,3,8,9,10]
print(f"Address of list before removal of duplicates: {id(l)}")
# Removing duplicates from list
s = set(l) # As we know set doesn't allow duplicates
l = list(s) # Converting set back to list
print(f"Address of list after removal of duplicates: {id(l)}")

OUTPUT:

Address of list before removal of duplicates: 1715573904000
Address of list after removal of duplicates: 1715575422528
```

4. Write a Python program to count the elements in a list until an element is a tuple. Note down the addresses of the tuple as well as the list.

```
l = [1,2,3,4,(3,2),60,70]
count = 0
for index in l:
    if isinstance(index,tuple): # isinstance() function is used to check the type of variable
        print("Tuple found at index:",index)
        print("Address of tuple:",id(index))
        break
    count += 1

print("Number of element before tuple:",count)
print("Address of list:",id(l))


 OUTPUT:

Tuple found at index: (3, 2)
Address of tuple: 2004220280384
Number of element before tuple: 4
Address of list: 2004220269184
```