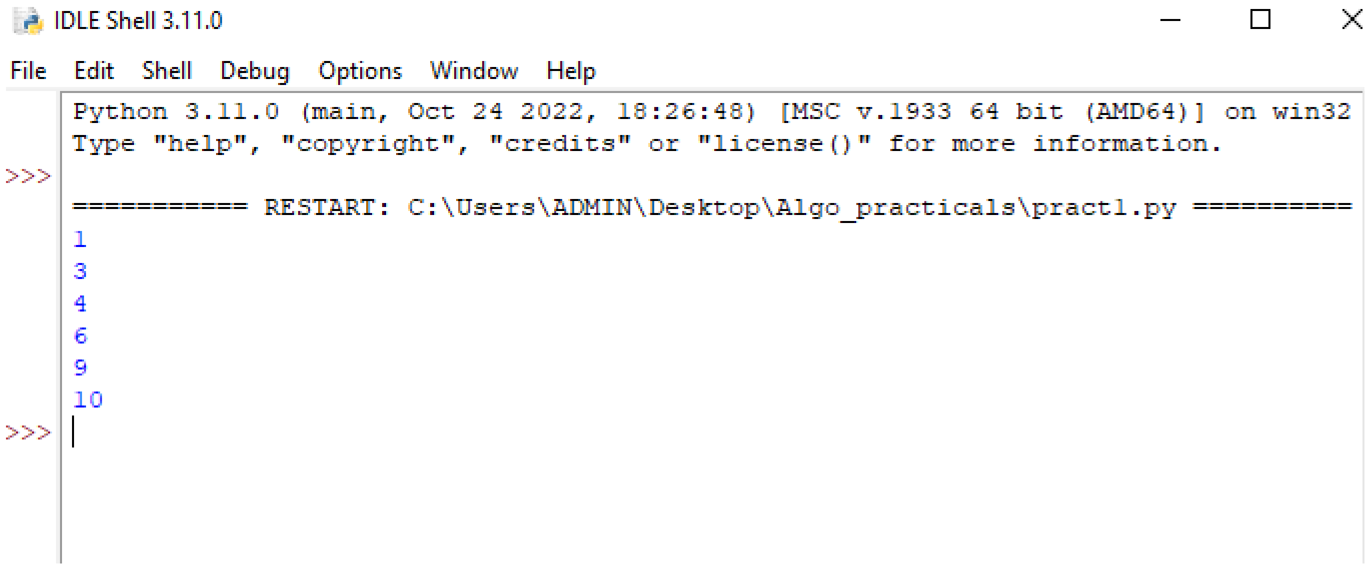
Anjalee Pagare
Roll no:525

Q1) Write a Program for Randomized Selection Algorithm

```
from random import randrange

def partition(x, pivot_index = 0):

    i = 0

    if pivot_index !=0: x[0],x[pivot_index] = x[pivot_index],x[0]

    for j in range(len(x)-1):

        if x[j+1] < x[0]:

            x[j+1],x[i+1] = x[i+1],x[j+1]

            i += 1

    x[0],x[i] = x[i],x[0]

    return x,i

def RSelect(x,k):

    if len(x) == 1:

        return x[0]

    else:

        xpart = partition(x,randrange(len(x)))

        x = xpart[0] # partitioned array

        j = xpart[1] # pivot index

        if j == k:

            return x[j]

        elif j > k:

            return RSelect(x[:j],k)

        else:

            k = k - j - 1
        return RSelect(x[(j+1):], k)
x = [10,1,3,6,4,9]
for i in range(len(x)):
    print (RSelect(x,i))
```

- **Output**



```
Python 3.11.0 (main, Oct 24 2022, 18:26:48) [MSC v.1933 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
=========== RESTART: C:\Users\ADMIN\Desktop\Algo_practicals\pract1.py ==========
1
3
4
6
9
10
>>>
```

Q.2) Write a Program for Heap Sort Algorithm

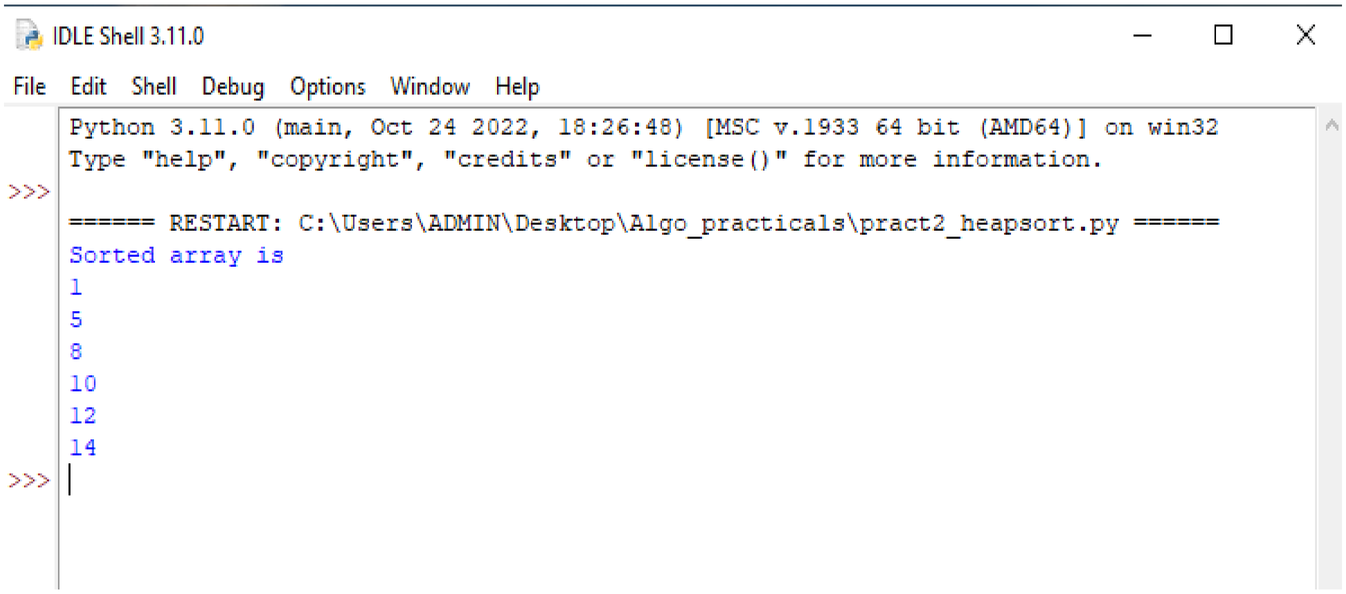
```
#Python program for implementation of heap Sort
# To heapify subtree rooted at index i.
# n is size of heap
def heapify(arr, n, i):
    largest = i  # Initialize largest as root
    l = 2 * i + 1     # left = 2*i + 1
    r = 2 * i + 2     # right = 2*i + 2
    # See if left child of root exists and is
    # greater than root
    if l < n and arr[i] < arr[l]:
```

Anjalee Pagare
Roll no:525

```
        largest = l

    # See if right child of root exists and is

    # greater than root

    if r < n and arr[largest] < arr[r]:

        largest = r


    # Change root, if needed

    if largest != i:

        arr[i],arr[largest] = arr[largest],arr[i]  # swap


        # Heapify the root.

        heapify(arr, n, largest)

# The main function to sort an array of given size

def heapSort(arr):

    n = len(arr)


    # Build a maxheap.

    for i in range(n, -1, -1):

        heapify(arr, n, i)

    # One by one extract elements

    for i in range(n-1, 0, -1):

        arr[i], arr[0] = arr[0], arr[i]   # swap

        heapify(arr, i, 0)
```

```
# Driver code to test above

arr = [ 12, 10, 14, 5, 8, 1]

heapSort(arr)

n = len(arr)

print ("Sorted array is")

for i in range(n):

    print ("%d" %arr[i]),
```

- **Output**



```
Python 3.11.0 (main, Oct 24 2022, 18:26:48) [MSC v.1933 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
====== RESTART: C:\Users\ADMIN\Desktop\Algo_practicals\pract2_heapsort.py ======
Sorted array is
1
5
8
10
12
14
>>>
```

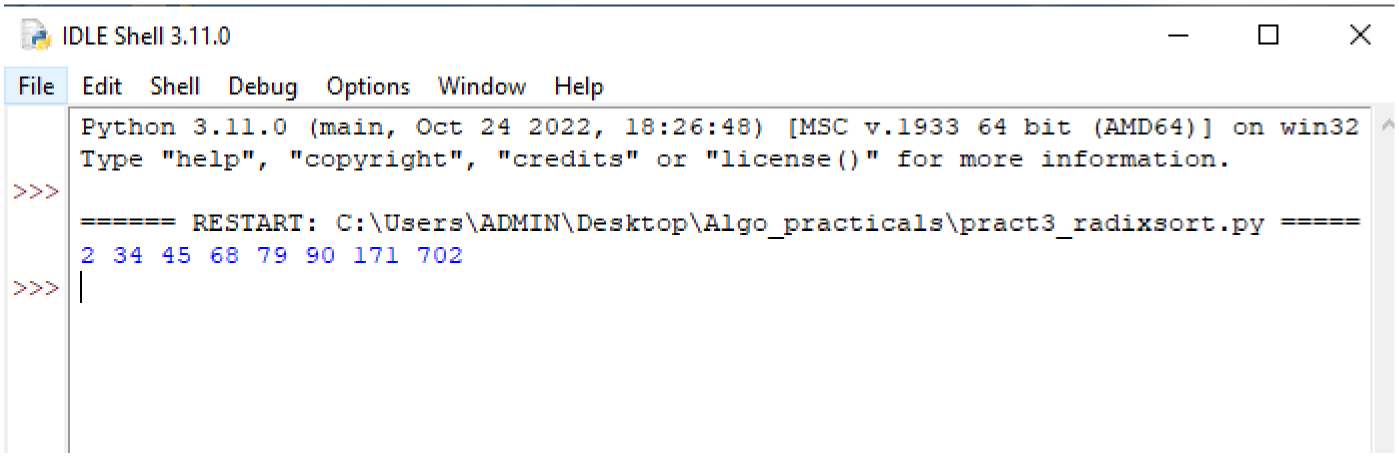
3) Write a Program to perform Radix Sort Algorithm

```
def countingSort(arr, exp1):

    n = len(arr)

    # The output array elements that will have sorted arr

    output = [0] * (n)
```

```
# initialize count array as 0

count = [0] * (10)

# Store count of occurrences in count[]

for i in range(0, n):

    index = arr[i] // exp1

    count[index % 10] += 1

# Change count[i] so that count[i] now contains actual

# position of this digit in output array

for i in range(1, 10):

    count[i] += count[i - 1]

# Build the output array

i = n - 1

while i >= 0:

    index = arr[i] // exp1

    output[count[index % 10] - 1] = arr[i]

    count[index % 10] -= 1

    i -= 1

# Copying the output array to arr[],
    # so that arr now contains sorted numbers

    i = 0

    for i in range(0, len(arr)):

        arr[i] = output[i]

# Method to do Radix Sort

def radixSort(arr):

    # Find the maximum number to know number of digits

    max1 = max(arr)

    # Do counting sort for every digit. Note that instead

    # of passing digit number, exp is passed. exp is 10^i

    # where i is current digit number

    exp = 1

    while max1 / exp >= 1:

        countingSort(arr, exp)

        exp *= 10

# Driver code

arr = [171, 45, 79, 90, 702, 34, 2, 68]

# Function Call
```

```
radixSort(arr)

for i in range(len(arr)):

    print(arr[i],end=" ")
```

- **Output**



```
Python 3.11.0 (main, Oct 24 2022, 18:26:48) [MSC v.1933 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
====== RESTART: C:\Users\ADMIN\Desktop\Algo_practicals\pract3_radixsort.py =====
2 34 45 68 79 90 171 702
>>>
```

4) Write a Program to Perform Bucket Sort Algorithm

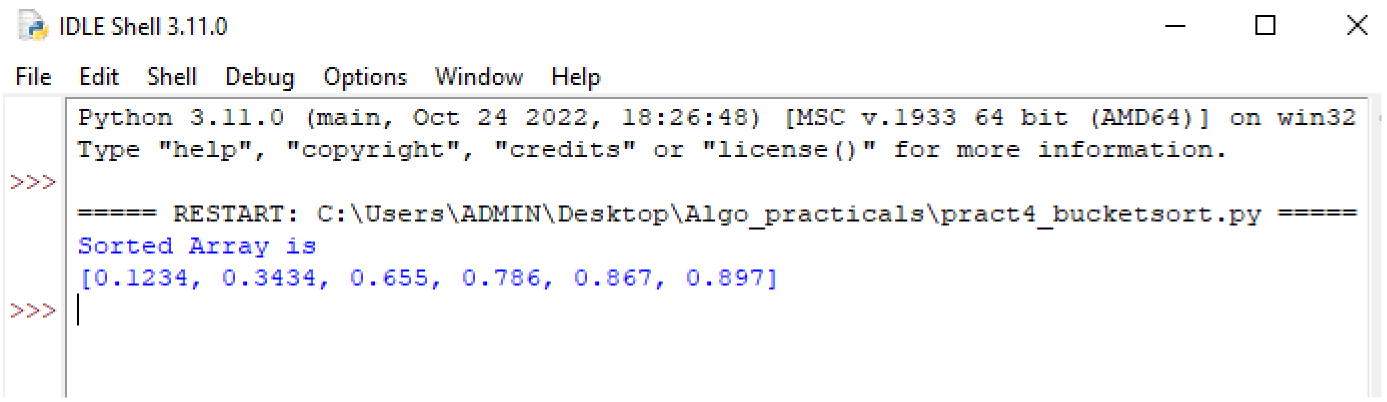
```
def insertionSort(b):

 for i in range(1, len(b)):

        up = b[i]

        j = i - 1

        while j >=0 and b[j] > up:

            b[j + 1] = b[j]

            j -= 1

        b[j + 1] = up

    return b

def bucketSort(x):

    arr = []
```

Anjalee Pagare
Roll no:525

```
    slot_num = 10 # 10 means 10 slots, each

                  # slot's size is 0.1

    for i in range(slot_num):

        arr.append([])

    # Put array elements in different buckets

    for j in x:

        index_b = int(slot_num * j)

        arr[index_b].append(j)

    # Sort individual buckets

    for i in range(slot_num):

        arr[i] = insertionSort(arr[i])


    # concatenate the result

    k = 0

    for i in range(slot_num):

        for j in range(len(arr[i])):

            x[k] = arr[i][j]

            k += 1

    return x

# Driver Code

x = [0.867, 0.655, 0.786,

     0.1234, 0.897, 0.3434]

print("Sorted Array is")

print(bucketSort(x))
```

Anjalee Pagare
Roll no:525

- **Output**

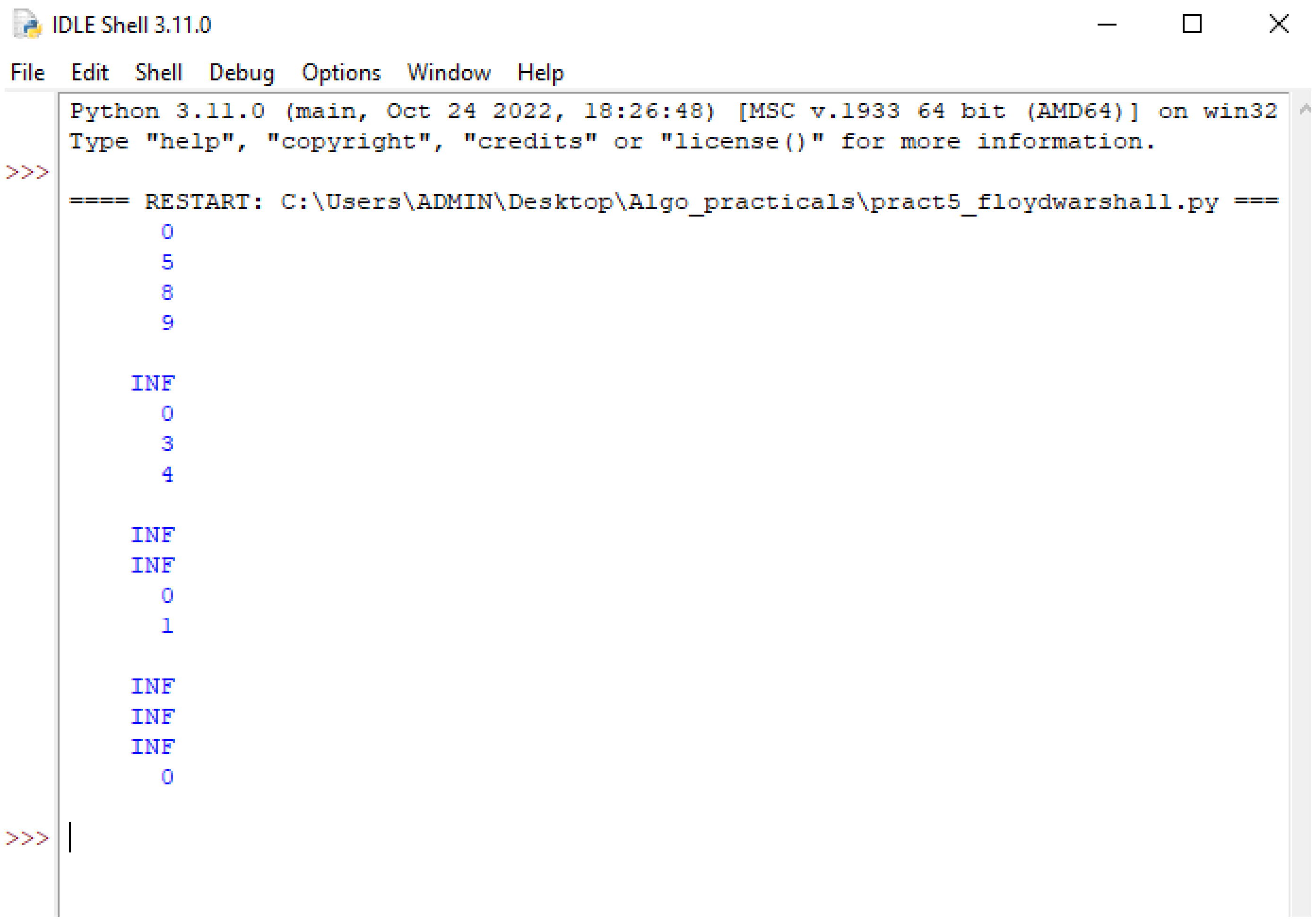


```
Python 3.11.0 (main, Oct 24 2022, 18:26:48) [MSC v.1933 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: C:\Users\ADMIN\Desktop\Algo_practicals\pract4_bucketsort.py =====
Sorted Array is
[0.1234, 0.3434, 0.655, 0.786, 0.867, 0.897]
>>>
```

5) Write a Program to Perform Folyd-Warshall algorithm

```
# Floyd-Warshall Algorithm
v = 4
INF = 99999
def floydWarshall(graph):
    dist = list(map(lambda i: list(map(lambda j:j, i)) ,graph))
    for k in range(v):
        for i in range(v):
            for j in range(v):
                dist[i][j] = min(dist[i][j] , dist[i][k]+dist[k][j])
    printSolution(dist)
def printSolution(dist):
    for i in range(v):
        for j in range(v):
            if(dist[i][j] == INF):
                print('%7s' %("INF"),)
            else:
                print('%7d\t' %(dist[i][j]),)
            if j == v-1:
                print(" ")
graph = [[0,5,INF,10],
        [INF,0,3,INF],
        [INF, INF, 0, 1],
        [INF, INF, INF, 0]
        ]
floydWarshall(graph);
```

- **Output**



```
Python 3.11.0 (main, Oct 24 2022, 18:26:48) [MSC v.1933 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
==== RESTART: C:\Users\ADMIN\Desktop\Algo_practicals\pract5_floydwarshall.py ===
      0
      5
      8
      9

    INF
      0
      3
      4

    INF
    INF
      0
      1

    INF
    INF
    INF
      0

>>> 
```

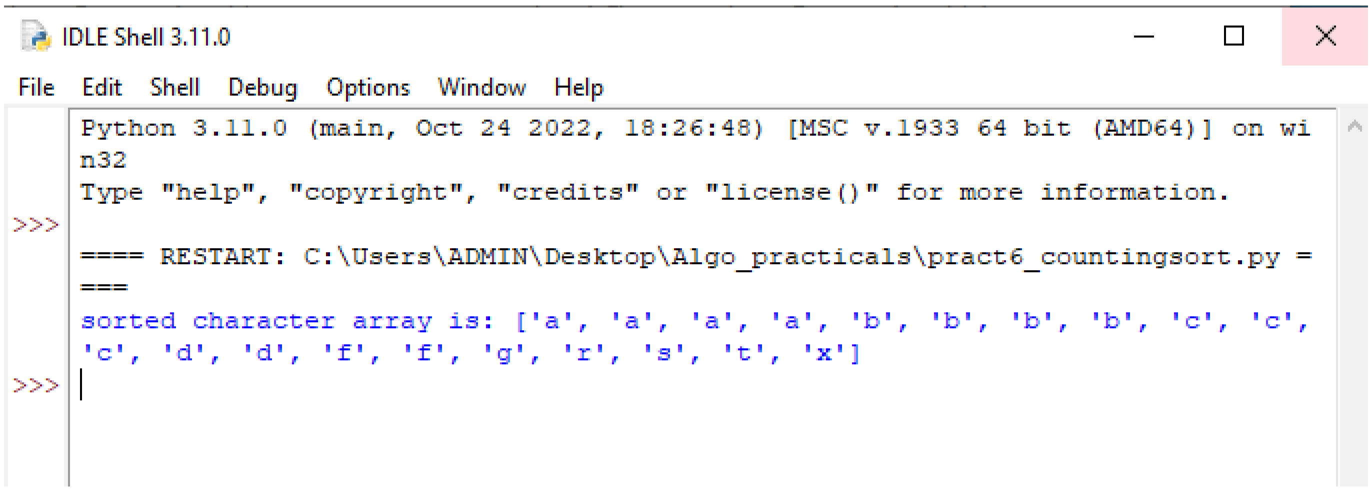
Anjalee Pagare
Roll no:525

6) Write a Program for Counting Sort Algorithm

```
 def countSort(arr):
output = [ 0 for i in range(256)]
count = [0 for i in range(256)]
ans = ["" for _ in arr]
for i in arr:
    count[ord(i)] += 1
for i in range(256):
    count[i] += count[i - 1]
for i in range(len(arr)):
    output[count[ord(arr[i])] - 1] = arr[i]
    count[ord(arr[i])] -= 1
for i in range(len(arr)):
    ans[i] = output[i]
return ans
arr = "abbcdffgrtxcaaacdbbs"
ans = countSort(arr)
print('sorted character array is:', ans)
```

- **Output**

Anjalee Pagare
Roll no:525



```
Python 3.11.0 (main, Oct 24 2022, 18:26:48) [MSC v.1933 64 bit (AMD64)] on wi
n32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
==== RESTART: C:\Users\ADMIN\Desktop\Algo_practicals\pract6_countingsort.py =
===
sorted character array is: ['a', 'a', 'a', 'a', 'b', 'b', 'b', 'b', 'c', 'c',
'c', 'd', 'd', 'f', 'f', 'g', 'r', 's', 't', 'x']
>>>
```

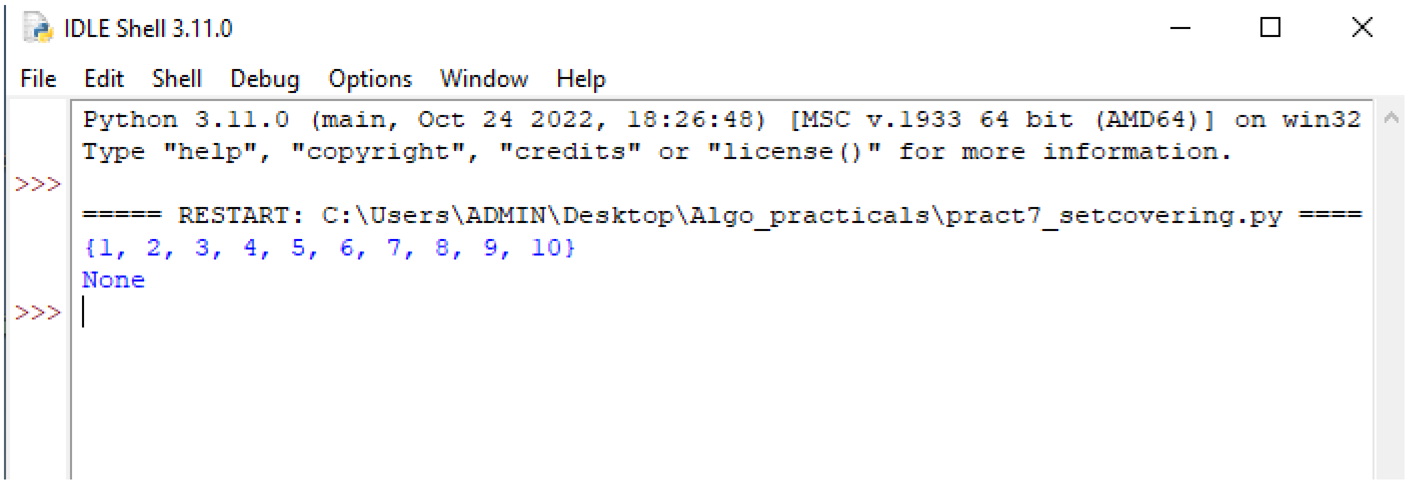
7) Write a program for Set Covering Problem

```
def set_cover(universe, subsets):
    elements = set(e for s in subsets for e in s)

    if elements != universe:
        return None
    covered = set(i)
    cover = []

    while covered != elements:
        subset = max(subsets, key=lambda s: len(s - covered))
        cover.append(subset)
        covered != subset

    return cover
```

Anjalee Pagare
Roll no:525

```
def main():

    universe = set(range(1, 11))

    print(universe)

    subsets = [set([1, 7, 3, 8, 5, 10]),

               set([]),

               set([]),

               set([]),

               set([])]

    cover = set_cover(universe, subsets)

    print(cover)


if __name__ == '__main__':

    main()
```

- **Output**



IDLE Shell 3.11.0

File Edit Shell Debug Options Window Help

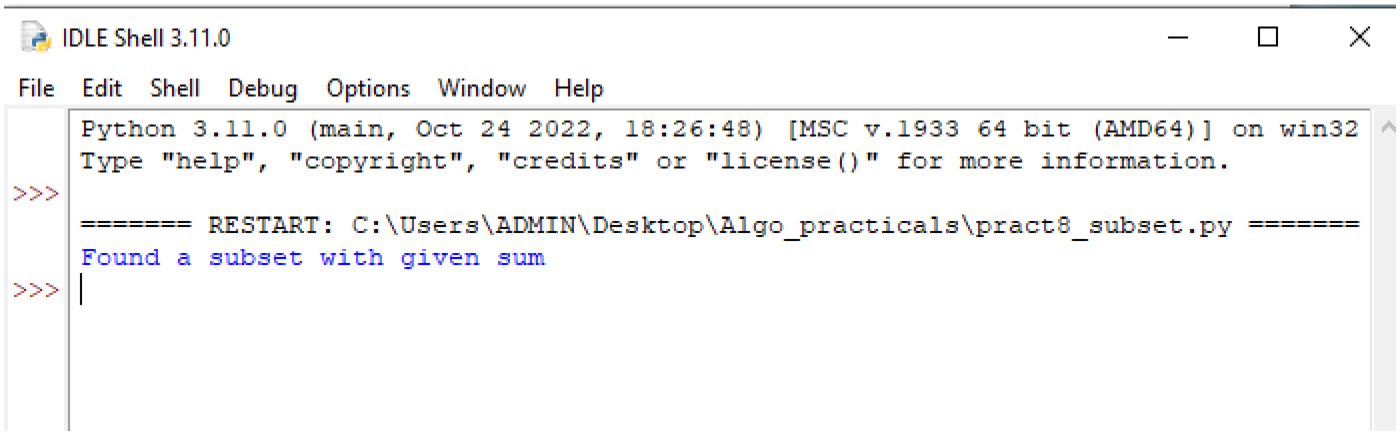
```
Python 3.11.0 (main, Oct 24 2022, 18:26:48) [MSC v.1933 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: C:\Users\ADMIN\Desktop\Algo_practicals\pract7_setcovering.py ====
{1, 2, 3, 4, 5, 6, 7, 8, 9, 10}
None
>>> |
```

Anjalee Pagare
Roll no:525

8)Write a Program for found a subset with given sum

```
 def isSubsetSum(set,n, sum):
  if (sum == 0) :
      return True
  if (n == 0 and sum != 0) :
      return False
  if (set[n - 1] > sum) :
      return isSubsetSum(set, n - 1,sum);
  return isSubsetSum(set, n-1, sum) or isSubsetSum(set, n-1, sum-set[n-1])
set = [3, 34, 4, 10, 8, 2]
sum = 9
n = len(set)
if (isSubsetSum(set, n, sum) == True) :
  print ("Found a subset with given sum")
else :
  print ("No subset with given sum")
```

- **Output**



```
Python 3.11.0 (main, Oct 24 2022, 18:26:48) [MSC v.1933 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
======= RESTART: C:\Users\ADMIN\Desktop\Algo_practicals\pract8_subset.py =======
Found a subset with given sum
>>> |
```

Anjalee Pagare
Roll no:525