

# LOTUS VALLEY INTERNATIONAL SCHOOL: Computer Science Practical File

Pranav Verma (Roll No. 7, XII Aryabhata)

October 15, 2025



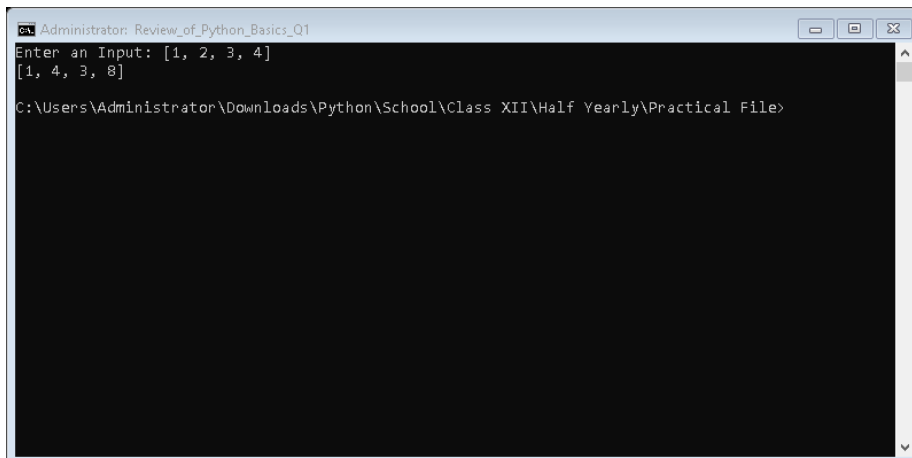
## Contents

1	Review of Python Basics	2
2	Functions	16
3	Data File Handling	24
4	Stack Operations	33
5	Relational Databases and SQL	41

## 1 Review of Python Basics

Q1) Write a program to multiply an element by two, if it is an odd index for a given list containing both numbers and strings.

```
1 l=eval(input("Enter an Input: "))
2 n=len(l)
3 for i in range(n):
4     if i%2!=0:
5         l[i]=l[i]*2
6 print(l)
```



Q2) Write a program to count the frequency of an element in a list.

```
1 l=eval(input("Enter an Input: "))
2 e=eval(input('Enter the element to be counted: '))
3 print(l.count(e))
```

```
Administrator: Review_of_Python_Basics_Q2
Enter an Input: [1, 2, 3, 4]
Enter the element to be counted: 1
1
C:\Users\Administrator\Downloads\Python\School\Class XII\Half Yearly\Practical File>
```

**Q3) Write a program to shift elements of a list so that the first element moves to the second index and second index move to the third index and so on, and the last element shifts to the first position.**

```
1 l=eval(input('Enter an Input: '))
2 x=l[-1]
3 l.pop(-1)
4 l.insert(0,x)
5 print(l)
```

```
Administrator: Review_of_Python_Basics_Q3
Enter an Input: [1, 2, 3, 4]
[4, 1, 2, 3]
C:\Users\Administrator\Downloads\Python\School\Class XII\Half Yearly\Practical File>
```

**Q4) A list NUM contains the elements: 3,25,13,6,35,8,14,45. Write a program to swap the content with the next value divisible by 5 so that the resultant list will look like: 25,3,13,35,6,8,45,14.**

```

1 l=eval(input('Enter an Input: '))
2 n=len(l)
3 for i in range(n):
4     if l[i]%5==0:
5         l[i-1],l[i]=l[i],l[i-1]
6 print(l)

```

**Q5)** Write a program to accept values from a user in a tuple. Add a tuple to it and display its elements one by one. Also display its maximum and minimum values.

```

1 n=int(input('Enter No. of Elements: '))
2 t=()
3 for i in range(n):
4     x=eval(input('Enter a Value: '))
5     t+=x,
6 print(t)
7 for i in t:
8     print(i)
9 print(max(t),min(t),end='\n')

```

```
Administrator: Review_of_Python_Basics_Q5
Enter No. of Elements: 1
Enter a Value: 2
(2,)
2
2 2
C:\Users\Administrator\Downloads\Python\School\Class XII\Half Yearly\Practical File>
```

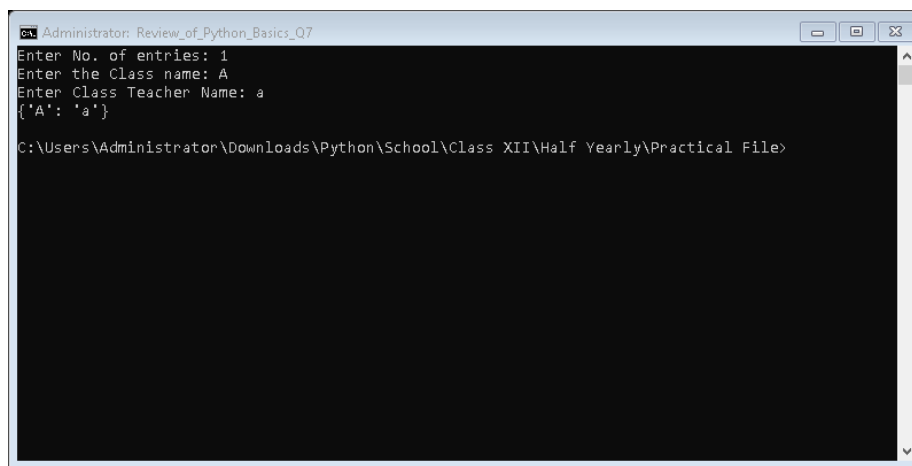
**Q6)** Write a program to input any values for two tuples. Print it, interchange it and then compare them.

```
1 t1=eval(input("Enter the Input: "))
2 t2=eval(input("Enter the Input: "))
3 print(t1, t2)
4 x=t2
5 t2=t1
6 t1=x
7 print(t1, t2)
8 if t1<t2:
9     print('t2 is greater')
10 else:
11     print('t1 is greater')
```

```
Administrator: Review_of_Python_Basics_Q6
Enter the Input: (1, 2)
Enter the Input: (3, 4)
(1, 2) (3, 4)
(3, 4) (1, 2)
t1 is greater
C:\Users\Administrator\Downloads\Python\School\Class XII\Half Yearly\Practical File>
```

Q7) Write a python program to input 'n' classes and names of class teachers to store them in a dictionary and display the same. Also accept a particular class from the user and display the name of the class teacher of that class.

```
1 n=int(input('Enter No. of entries: '))
2 d={}
3 for i in range(n):
4     x=input('Enter the Class name: ')
5     y=input('Enter Class Teacher Name: ')
6     d[x]=y
7 print(d)
```



```
Administrator: Review_of_Python_Basics_Q7
Enter No. of entries: 1
Enter the Class name: A
Enter Class Teacher Name: a
{'A': 'a'}

C:\Users\Administrator\Downloads\Python\School\Class XII\Half Yearly\Practical File>
```

Q8) Write a program to store students names and their percentage in a dictionary and delete a particular student name from the dictionary. Also display the dictionary after deletion.

```
1 d=eval(input('Enter a Dictionary: '))
2 x=input('Enter the name of the student to be deleted: ')
3 if x in d:
4     del d[x]
5 print(d)
```

```
Administrator: Review_of_Python_Basics_Q8
Enter a Dictionary: {"a":20}
Enter the name of the student to be deleted: a
{}
C:\Users\Administrator\Downloads\Python\School\Class XII\Half Yearly\Practical File>
```

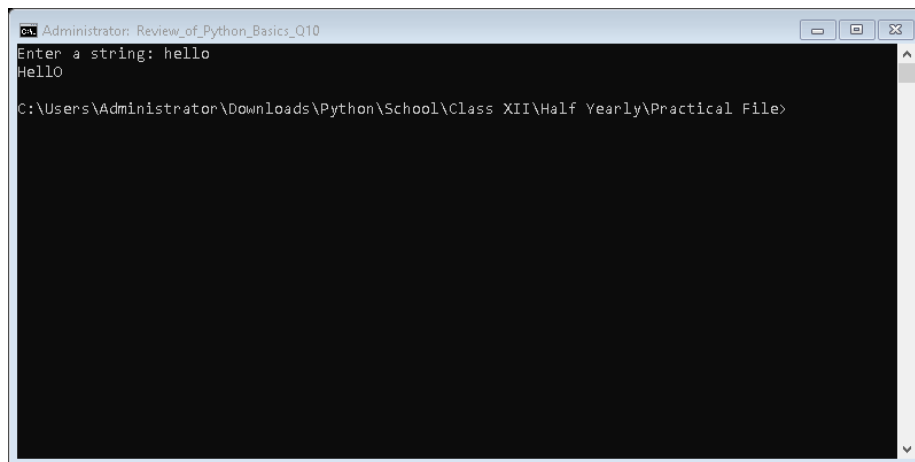
**Q9)** Write a Python program to input names of 'n' customers and their details like items bought, cost and phone number, etc., store them in a dictionary and display all the details in a tabular form.

```
1 d={}
2 n=int(input('Enter No. of Customers: '))
3 for i in range(n):
4     name=input('Enter name of Customer: ')
5     item=input('Enter item bought: ')
6     cost=eval(input('Enter the cost of item: '))
7     ph_no=int(input('Enter contact no.: '))
8     d[name]=[item, cost, ph_no]
9 print('Name', '\t Item', '\t Cost', '\t Contact Number')
10 for i in d:
11     print(i, '\t ', d[i][0], '\t ', d[i][1], '\t ', d[i][2])
```

```
Administrator: Review_of_Python_Basics_Q9
Enter No. of Customers: 1
Enter name of Customer: a
Enter item bought: Test
Enter the cost of item: 1000
Enter contact no.: 98234
Name      Item      Cost      Contact Number
a         Test      1000      98234
C:\Users\Administrator\Downloads\Python\School\Class XII\Half Yearly\Practical File>
```

**Q10)** Write a Python program to capitalize the first and last letters of each word of a given string.

```
1 s=input('Enter a string: ')
2 j=''
3 x=''
4 l=s.split()
5 for i in l:
6     j=i[0].upper()+i[1:-1]+i[-1].upper()
7     x+=j+' '
8 print(x)
```

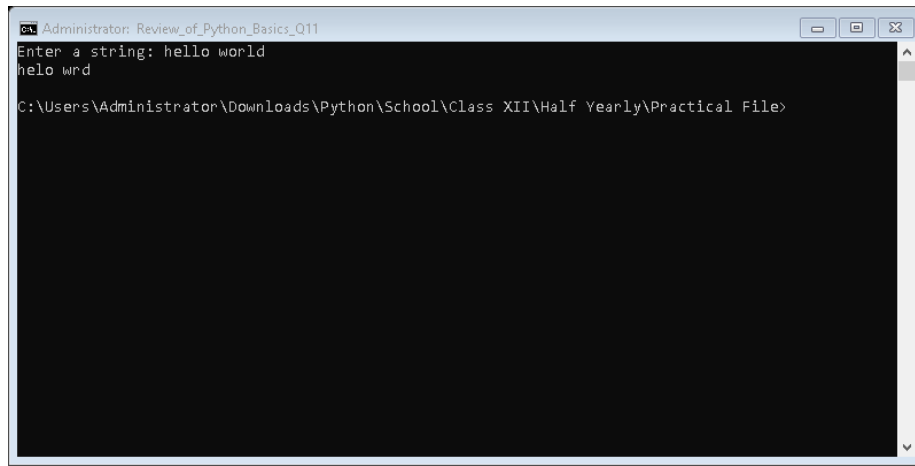


```
Administrator: Review_of_Python_Basics_Q10
Enter a string: hello
Hello
C:\Users\Administrator\Downloads\Python\School\Class XII\Half Yearly\Practical File>
```

**Q11)** Write a Python program to remove duplicate characters of a given string.

```
1 s=input('Enter a string: ')
2 x=''
3 for i in s:
4     if i not in x:
5         x+=i
6 print(x)
```



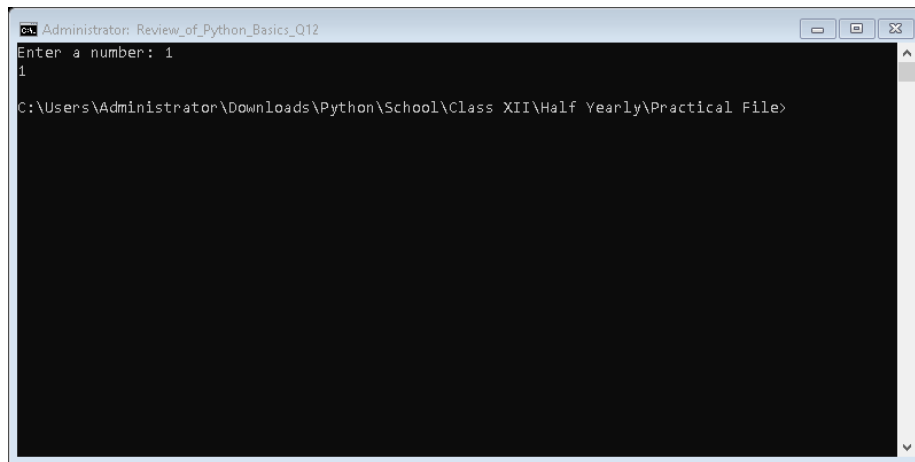


```
Administrator: Review_of_Python_Basics_Q11
Enter a string: hello world
helo wrd

C:\Users\Administrator\Downloads\Python\School\Class XII\Half Yearly\Practical File>
```

**Q12)** Write a Python program compute the sum of digits of a given number

```
1 x=int(input('Enter a number: '))
2 s=0
3 while x>0:
4     b=x%10
5     s+=b
6     x=x//10
7 print(s)
```



```
Administrator: Review_of_Python_Basics_Q12
Enter a number: 1
1

C:\Users\Administrator\Downloads\Python\School\Class XII\Half Yearly\Practical File>
```

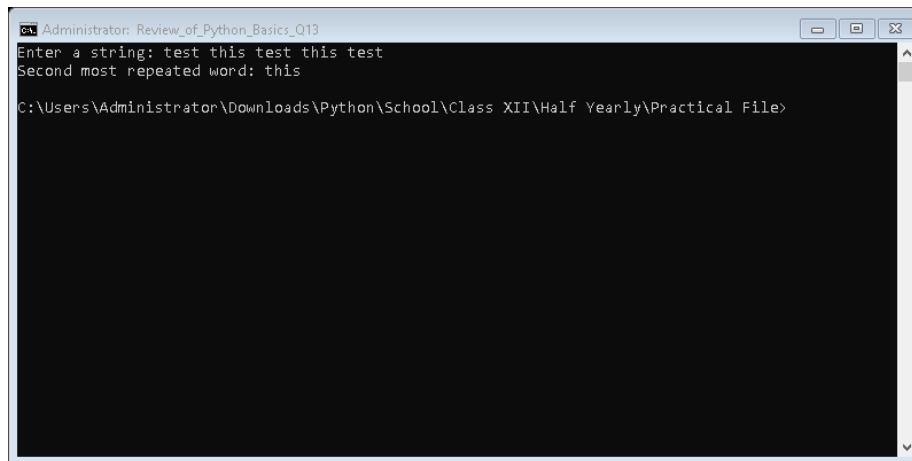
**Q13)** Write a Python program to find the second most repeated word in a given string.

```
1 s = input("Enter a string: ")
2 words = s.split()
```

```

3 freq = {}
4 for word in words:
5     freq[word] = freq.get(word, 0) + 1
6 sorted_freq = sorted(freq.items(), key=lambda x: x[1], reverse=True
7 )
8 if len(sorted_freq) < 2:
9     print("No second most repeated word found.")
10 else:
11     print("Second most repeated word:", sorted_freq[1][0])

```



```

Administrator: Review_of_Python_Basics_Q13
Enter a string: test this test this test
Second most repeated word: this

C:\Users\Administrator\Downloads\Python\School\Class XII\Half Yearly\Practical File>

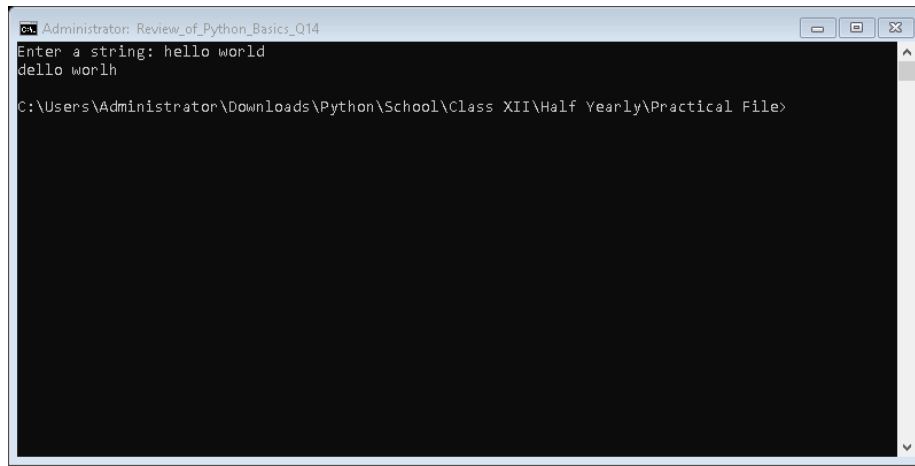
```

**Q14) Write a Python program to change a given string to a new string where the first and last string have been exchanged.**

```

1 s=input('Enter a string: ')
2 x=len(s)
3 a=s[-1]+s[1:x-1]+s[0]
4 print(a)

```

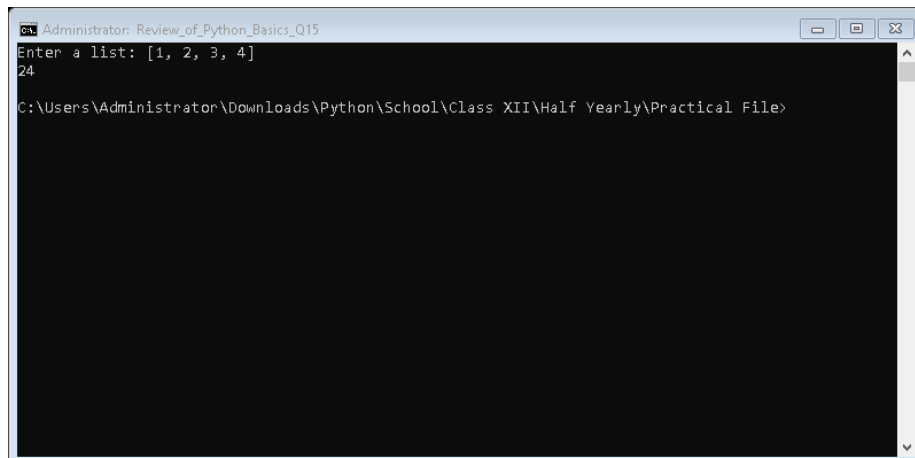


```
Administrator: Review_of_Python_Basics_Q14
Enter a string: hello world
dello worlh

C:\Users\Administrator\Downloads\Python\School\Class XII\Half Yearly\Practical File>
```

**Q15) Write a Python program to multiply all the elements in a list.**

```
1 l=eval(input('Enter a list: '))
2 p=1
3 for i in l:
4     p*=i
5 print(p)
```

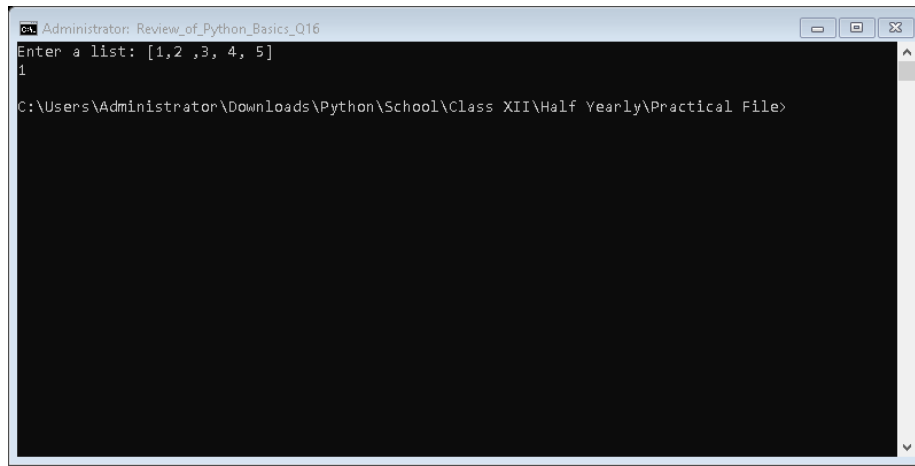


```
Administrator: Review_of_Python_Basics_Q15
Enter a list: [1, 2, 3, 4]
24

C:\Users\Administrator\Downloads\Python\School\Class XII\Half Yearly\Practical File>
```

**Q16) Write a Python program to get the smallest number from a list.**

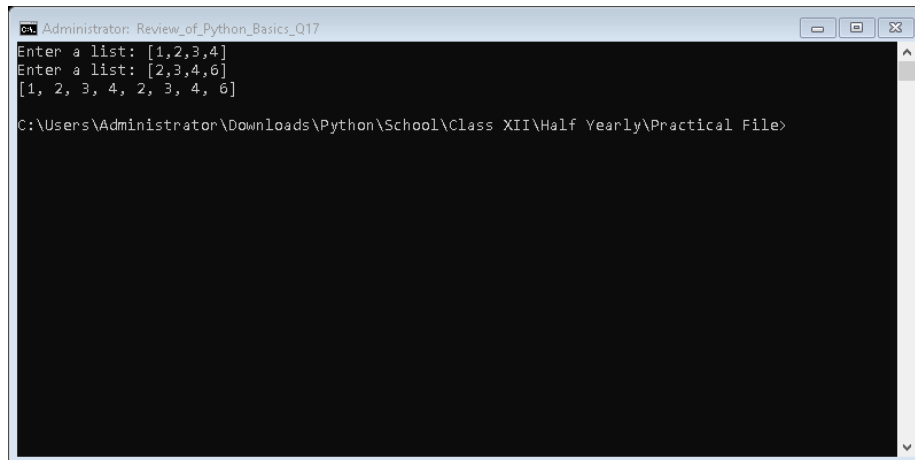
```
1 l=eval(input('Enter a list: '))
2 print(min(l))
```



```
Administrator: Review_of_Python_Basics_Q16
Enter a list: [1,2 ,3, 4, 5]
1
C:\Users\Administrator\Downloads\Python\School\Class XII\Half Yearly\Practical File>
```

**Q17) Write a Python program to append a list to the second list.**

```
1 l1=eval(input('Enter a list: '))
2 l2=eval(input('Enter a list: '))
3 l1.extend(l2)
4 print(l1)
```

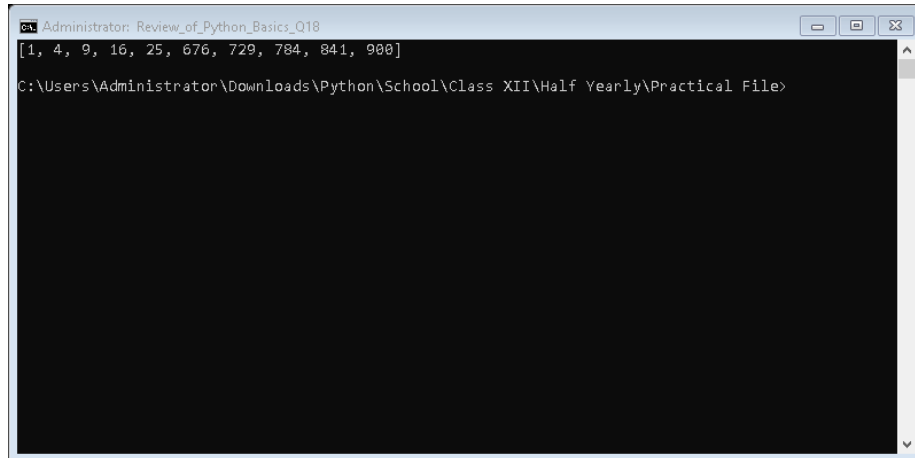


```
Administrator: Review_of_Python_Basics_Q17
Enter a list: [1,2,3,4]
Enter a list: [2,3,4,6]
[1, 2, 3, 4, 2, 3, 4, 6]
C:\Users\Administrator\Downloads\Python\School\Class XII\Half Yearly\Practical File>
```

**Q18) Write a Python program to generate and print a list of first five and last five elements where the values are square of numbers between one and 30 (both included).**

```
1 l=[]
2 for i in range(1,31):
3     l.append(i**2)
```

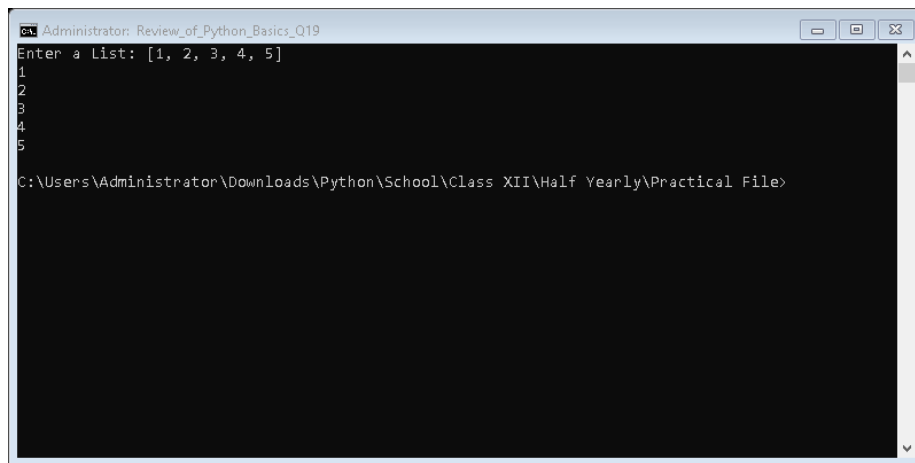
```
4 x=l[:5]+l[-5:]
5 print(x)
```



```
Administrator: Review_of_Python_Basics_Q18
[1, 4, 9, 16, 25, 676, 729, 784, 841, 900]
C:\Users\Administrator\Downloads\Python\School\Class XII\Half Yearly\Practical File>
```

**Q19) Write a Python program to get unique values from a list.**

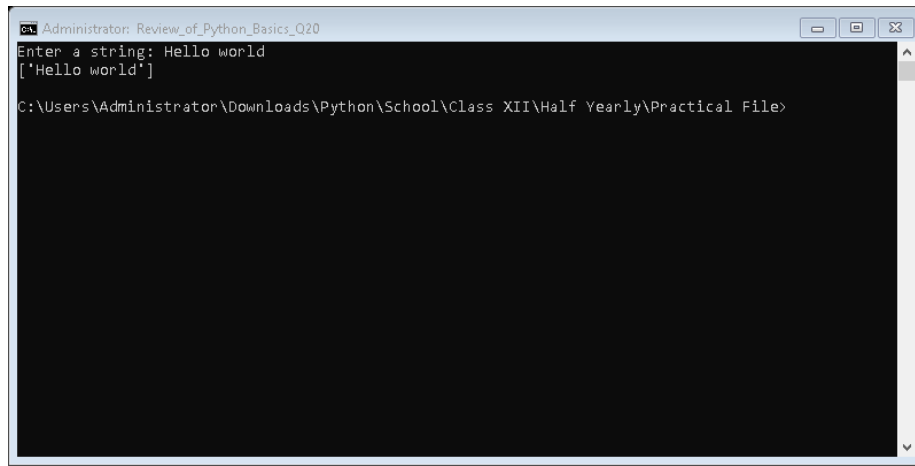
```
1 l=eval(input('Enter a List: '))
2 for i in l:
3     if l.count(i)==1:
4         print(i)
```



```
Administrator: Review_of_Python_Basics_Q19
Enter a List: [1, 2, 3, 4, 5]
1
2
3
4
5
C:\Users\Administrator\Downloads\Python\School\Class XII\Half Yearly\Practical File>
```

**Q20) Write a python program to convert a string to a list.**

```
1 s=input('Enter a string: ')
2 l=[]
3 l.append(s)
4 print(l)
```

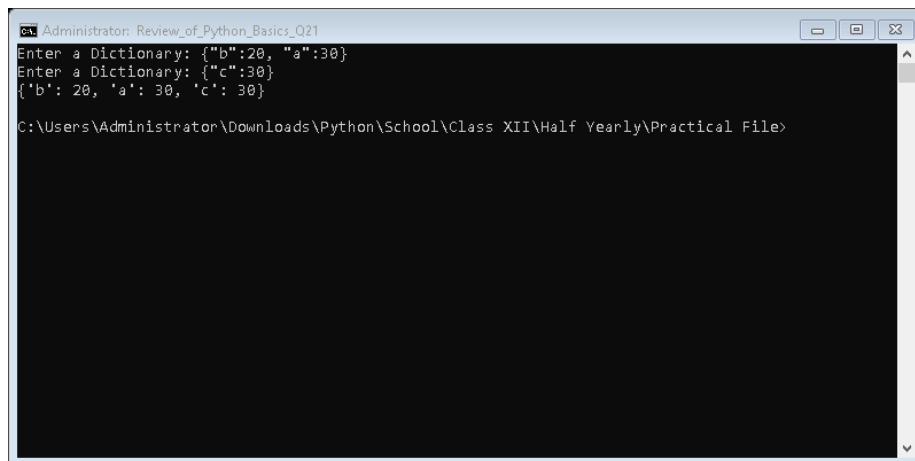


```
Administrator: Review_of_Python_Basics_Q20
Enter a string: Hello world
['Hello world']

C:\Users\Administrator\Downloads\Python\School\Class XII\Half Yearly\Practical File>
```

**Q21)** Write a Python script to concatenate the following dictionaries to create a new one: d1: {'A':1, 'B':2, 'C': 3}, d2: {'D':4}, Output should be: {'A':1, 'B':2, 'C':3, 'D':4}

```
1 d1=eval(input("Enter a Dictionary: "))
2 d2=eval(input("Enter a Dictionary: "))
3 d1.update(d2)
4 print(d1)
```



```
Administrator: Review_of_Python_Basics_Q21
Enter a Dictionary: {"b":20, "a":30}
Enter a Dictionary: {"c":30}
{'b': 20, 'a': 30, 'c': 30}

C:\Users\Administrator\Downloads\Python\School\Class XII\Half Yearly\Practical File>
```

**Q22)** Write a Python script to check if a given key already exists in a dictionary.

```
1 d=eval(input('Enter a dictionary: '))
2 x=input('Enter a key: ')
3 if x in d:
```

```

4     print('Exists')
5 else:
6     print("Key Does Not Exist")

```

```

Administrator: Review_of_Python_Basics_Q22
Enter a dictionary: {"a": 20, "b": 30}
Enter a key: a
Exists
C:\Users\Administrator\Downloads\Python\School\Class XII\Half Yearly\Practical File>

```

**Q23) Write a Python script to print a dictionary where the keys are numbers between 1 and 15 (both included) and the values are square of keys.**

```

1 d={}
2 for i in range(1,16):
3     d[i]=i**2
4 print(d)

```

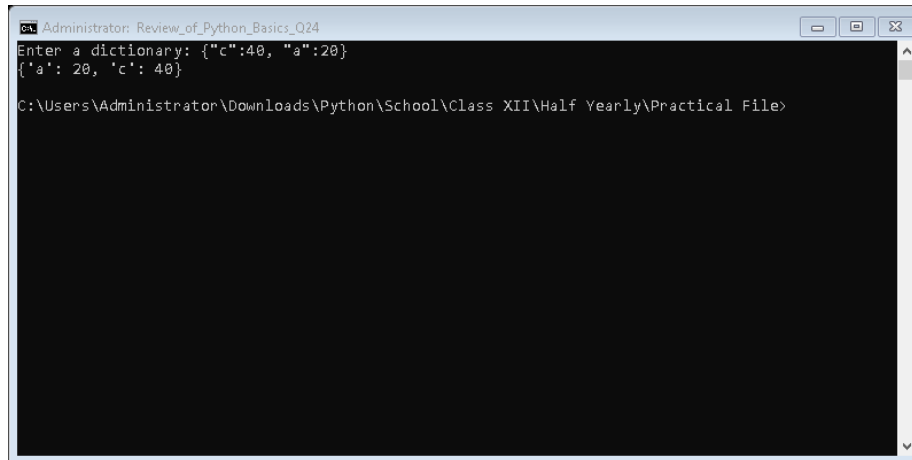
```

Administrator: Review_of_Python_Basics_Q23
C:\Users\Administrator\Downloads\Python\School\Class XII\Half Yearly\Practical File>

```

**Q24) Write a Python program to sort a dictionary by key.**

```
1 d=eval(input('Enter a dictionary: '))
2 print(dict(sorted(d.items())))
```

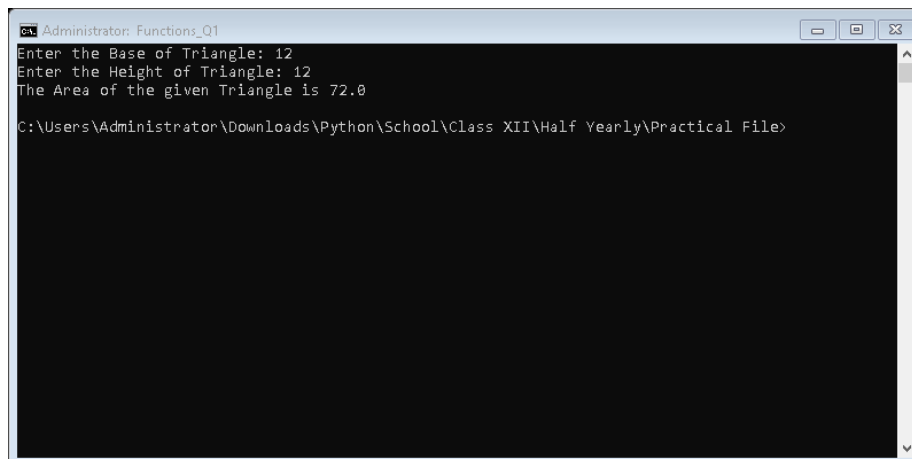


```
Administrator: Review_of_Python_Basics_Q24
Enter a dictionary: {"c":40, "a":20}
{'a': 20, 'c': 40}
C:\Users\Administrator\Downloads\Python\School\Class XII\Half Yearly\Practical File>
```

## 2 Functions

Q1) Write a function, `calculate_area()`, that takes base and height as input arguments and returns the area of a triangle as an output. The formula used is:  $\text{Triangle Area} = \frac{1}{2} * \text{base} * \text{height}$

```
1 def calculate_area(B,H):
2     A=(1/2)*B*H
3     return A
4 b=float(input("Enter the Base of Triangle: "))
5 h=float(input("Enter the Height of Triangle: "))
6 print("The Area of the given Triangle is",calculate_area(b,h))
```



```
Administrator: Functions_Q1
Enter the Base of Triangle: 12
Enter the Height of Triangle: 12
The Area of the given Triangle is 72.0
C:\Users\Administrator\Downloads\Python\School\Class XII\Half Yearly\Practical File>
```



Q2) Modify the function given in the previous question to take a third parameter called shape type. Shaped type should be either triangle or rectangle. Based on the shape, it should calculate the area.

```

1 def calculate_area(Shape_Type,B,H):
2     if Shape_Type=="Rectangle":
3         A=B*H
4     elif Shape_Type=="Triangle":
5         A=(1/2)*B*H
6     return A
7 shape=input("Enter the Type of Shape: ")
8 if shape.lower() in ("rect","rectangle","r","sqare","s"):
9     shape="Rectangle"
10 elif shape.lower() in ("triangle","t","tri"):
11     shape="Triangle"
12 else:
13     print("Shape NOT DEFINED")
14     quit()
15 b=float(input("Enter the Dimensions: "))
16 h=float(input("Enter the Dimensions: "))
17 print("The Area of the given Figure is",calculate_area(shape,b,h))

```

```

Administrator: Functions_Q2
Enter the Type of Shape: rect
Enter the Dimensions: 12
Enter the Dimensions: 12
The Area of the given Figure is 144.0

C:\Users\Administrator\Downloads\Python\School\Class XII\Half Yearly\Practical File>

```

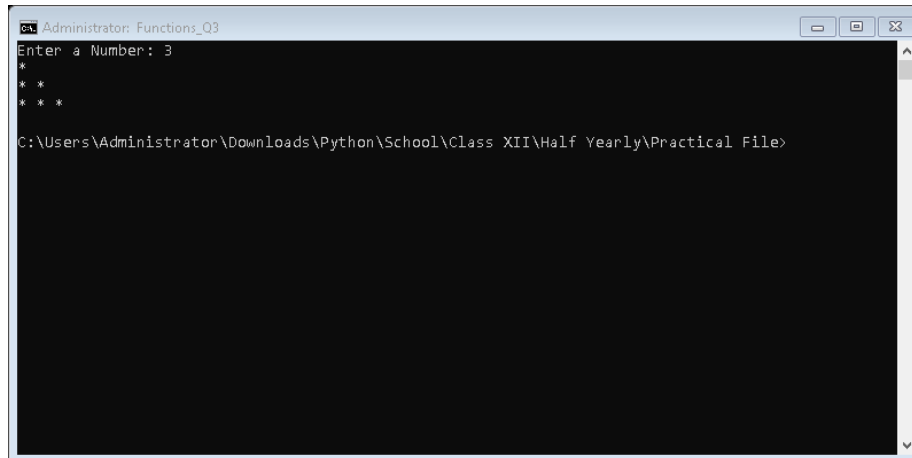
Q3) Write a function, print\_pattern(), that takes integer number as argument and print the following pattern if the input is 3: \*, \*\*, \*\*\* If the input is 4, then it should print: \*, \*\*, \*\*\*, \*\*\*\*.

```

1 def pattern(N):
2     for i in range(1,N+1):
3         for j in range(i):
4             print("*",end=" ")
5         print()
6 n=int(input("Enter a Number: "))

```

```
7 pattern(n)
```

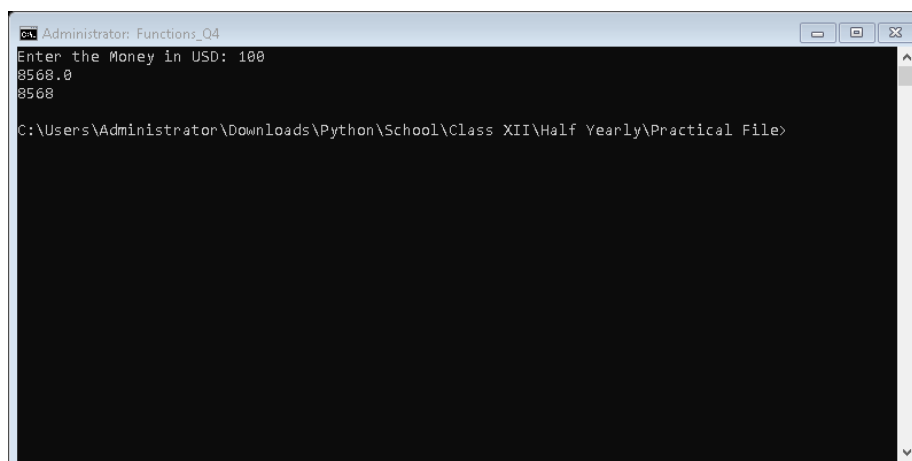


```
Administrator: Functions_Q3
Enter a Number: 3
*
* *
* * *

C:\Users\Administrator\Downloads\Python\School\Class XII\Half Yearly\Practical File>
```

Q4) Write a function that takes amount dollars and dollar-to-rupee conversion price and then returns the amount converted to rupees. Create the function in both void and non-void forms.

```
1 def Void(M):
2     print(M*85.68)
3 def Non_Void(M):
4     x=M*85.68
5     return int(x)
6 m=float(input("Enter the Money in USD: "))
7 Void(m)
8 print(Non_Void(m))
```

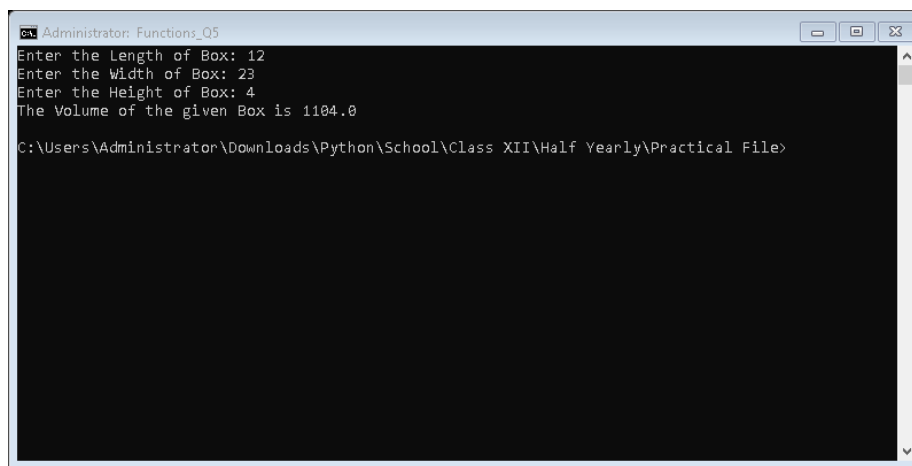


```
Administrator: Functions_Q4
Enter the Money in USD: 100
8568.0
8568

C:\Users\Administrator\Downloads\Python\School\Class XII\Half Yearly\Practical File>
```

Q5) Write a function to calculate the value of a box with appropriate default values for its parameters. Your function should have the following input parameters: Length of box, Width of box, Height of box.

```
1 def Volume(L,W,H):
2     V=L*W*H
3     return V
4 l=float(input("Enter the Length of Box: "))
5 w=float(input("Enter the Width of Box: "))
6 h=float(input("Enter the Height of Box: "))
7 print("The Volume of the given Box is",Volume(l,w,h))
```

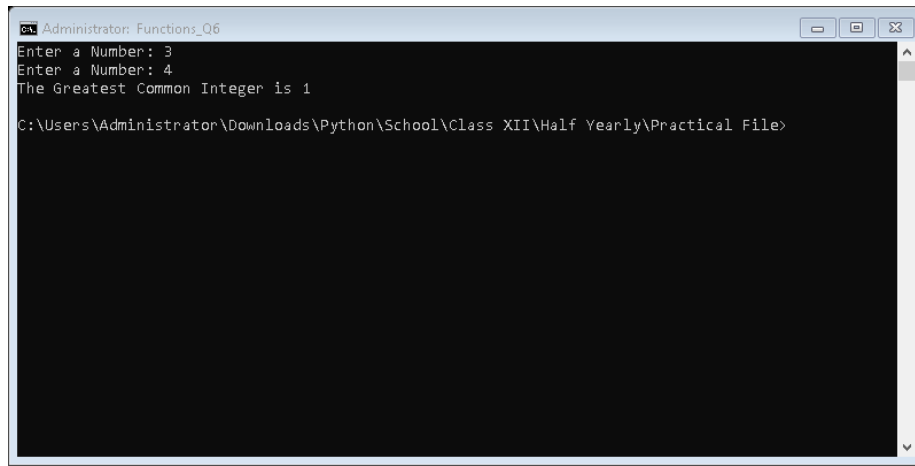


```
Administrator: Functions_Q5
Enter the Length of Box: 12
Enter the Width of Box: 23
Enter the Height of Box: 4
The Volume of the given Box is 1104.0

C:\Users\Administrator\Downloads\Python\School\Class XII\Half Yearly\Practical File>
```

Q6) Write a program to find the greatest common divisor between two numbers.

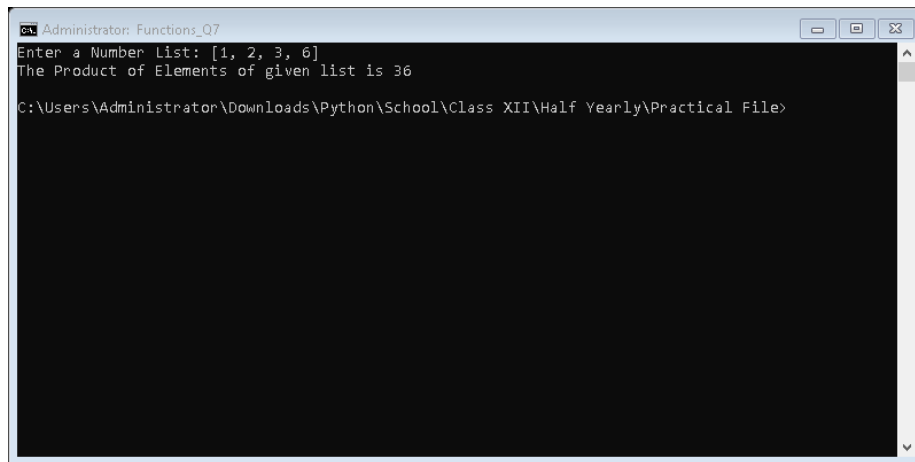
```
1 def GCF(N1,N2):
2     l1=[]
3     l2=[]
4     for i in range(1,N1+1):
5         if N1%i==0:
6             l1.append(i)
7     for i in l1:
8         if N2%i==0:
9             l2.append(i)
10    print("The Greatest Common Integer is",max(l2))
11 n1=int(input("Enter a Number: "))
12 n2=int(input("Enter a Number: "))
13 GCF(n1,n2)
```



```
Administrator: Functions_Q6
Enter a Number: 3
Enter a Number: 4
The Greatest Common Integer is 1
C:\Users\Administrator\Downloads\Python\School\Class XII\Half Yearly\Practical File>
```

**Q7) Write a Python function to multiply all the numbers in a list.**

```
1 def Multiply(l1):
2     p=1
3     for i in l1:
4         p*=i
5     return p
6 l=eval(input("Enter a Number List: "))
7 print("The Product of Elements of given list is",Multiply(l))
```



```
Administrator: Functions_Q7
Enter a Number List: [1, 2, 3, 6]
The Product of Elements of given list is 36
C:\Users\Administrator\Downloads\Python\School\Class XII\Half Yearly\Practical File>
```

**Q8) Write a Python function to calculate the factorial of a number (a non-negative integer). The function accepts the number whose factorial is to be calculated as the argument.**

```

1 def Multiply(l1):
2     p=1
3     for i in range(1,l1+1):
4         p*=i
5     return p
6 l=eval(input("Enter a Number List: "))
7 print("The Factorial of",l,"is",Multiply(l))

```

```

Administrator: Functions_Q8
Enter a Number List: 10
The Factorial of 10 is 3628800

C:\Users\Administrator\Downloads\Python\School\Class XII\Half Yearly\Practical File>

```

**Q9)** Write a Python function that takes a number as a parameter and checks whether the number is prime or not.

```

1 def Prime(l1):
2     a=0
3     for i in range(2,l1):
4         if l1%i==0:
5             a=0
6             break
7         else:
8             a=1
9     if a==1:
10        print("Number is Prime")
11    else:
12        print("Number is NOT Prime")
13 l=eval(input("Enter a Number List: "))
14 Prime(l)

```

```
Administrator: Functions_Q9
Enter a Number List: 20
Number is NOT Prime

C:\Users\Administrator\Downloads\Python\School\Class XII\Half Yearly\Practical File>
```

**Q10) Write a Python function that checks whether a passed string is a palindrome or not.**

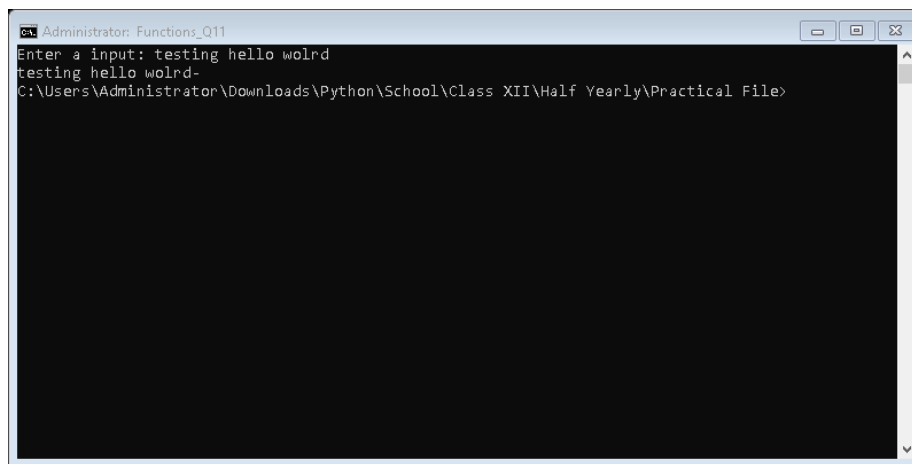
```
1 def palindrome(x):
2     i=''
3     for j in x:
4         if j.isalpha()==True:
5             i+=j
6     if i==i[::-1]:
7         print("It is a Palindrome")
8     else:
9         print("It is NOT a Palindrome")
10 s=input("Enter a Input: ")
11 palindrome(s)
```

```
Administrator: Functions_Q10
Enter a Input: test this test
It is NOT a Palindrome

C:\Users\Administrator\Downloads\Python\School\Class XII\Half Yearly\Practical File>
```

Q11) Write a Python program that accepts a hyphen-separated sequence of words as input and prints the words in a hyphen-separated after sorting them alphabetically.

```
1 def Sorter(s):
2     l1=s.split("-")
3     l1.sort()
4     for i in l1:
5         print(i,end="-")
6 S=input("Enter a input: ")
7 Sorter(S)
```



```
Administrator: Functions_Q11
Enter a input: testing hello wolrd
testing hello wolrd-
C:\Users\Administrator\Downloads\Python\School\Class XII\Half Yearly\Practical File>
```

Q12) Write a method in Python to find and display prime numbers from 2 to N. The value of N should be passed as an argument to the method.

```
1 def P_S(N):
2     for i in range(2,N+1):
3         a=0
4         for j in range(2,i):
5             if j%i==0:
6                 a=0
7                 break
8             else:
9                 a=1
10                break
11        if a==1:
12            print(i,": Number is Prime")
13        else:
14            print(i,": Number is NOT Prime")
15 l=eval(input("Enter a Number List: "))
16 P_S(l)
```

```
Administrator: Functions_Q12
204 : Number is Prime
205 : Number is Prime
206 : Number is Prime
207 : Number is Prime
208 : Number is Prime
209 : Number is Prime
210 : Number is Prime
211 : Number is Prime
212 : Number is Prime
213 : Number is Prime
214 : Number is Prime
215 : Number is Prime
216 : Number is Prime
217 : Number is Prime
218 : Number is Prime
219 : Number is Prime
220 : Number is Prime
221 : Number is Prime
222 : Number is Prime
223 : Number is Prime

C:\Users\Administrator\Downloads\Python\School\Class XII\Half Yearly\Practical File>
```

### 3 Data File Handling

Q1) File 'sports.dat' contains information in the following format: EventName, Participant. Write a function that read contents from file 'sports.dat' and create a file named 'Athletic.dat', copying only those records from 'sports.dat' in which the event name is 'Athletics'.

```
1 import pickle
2 ch = input("Make a New Sports file? (yes/no): ")
3 y = ("yes", "y")
4 n = ("no", "n")
5 if ch.lower() in y:
6     with open("sports.dat", "wb") as f:
7         while True:
8             l = eval(input("Enter Data (e.g., ('Type','Sport','Participant')): "))
9             pickle.dump(l, f)
10            c = input("Continue? (yes/no): ")
11            if c.lower() in n:
12                break
13 def Athletics():
14     with open("sports.dat", "rb") as f:
15         with open("Athletics.dat", "wb") as f2:
16             try:
17                 while True:
18                     x = pickle.load(f)
19                     if isinstance(x, (list, tuple)) and len(x)>0
20                     and x[0].lower() == "athletics":
21                         pickle.dump(x, f2)
22             except EOFError:
23                 print("File Loading Completed!")
24     with open("Athletics.dat", "rb") as f:
25         try:
```



```

25         while True:
26             print(pickle.load(f))
27         except EOFError:
28             print("End of File!!")
29 # Athletics()

```

```

Administrator: Data_File_Handling_Q1
Make a New Sports file? (yes/no): yes
Enter Data (e.g., ('Type','Sport','Participant')): ("Test", "Basketball", "Pranav")
Continue? (yes/no): no

C:\Users\Administrator\Downloads\Python\School\Class XII\Half Yearly\Practical File>

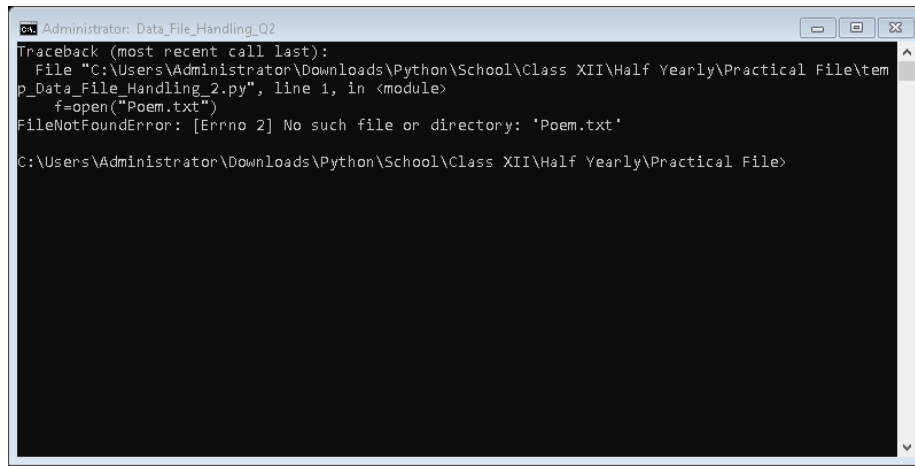
```

**Q2) Write a program to count the words 'to' and 'the' present in text file 'Poem.txt'.**

```

1 f=open("Poem.txt")
2 lines=f.read()
3 f.close()
4 count=0
5 line=lines.split()
6 x=line.index("to")
7 y=line.index("the")
8 for i in range(x,y+1):
9     count+=1
10 print(count)

```

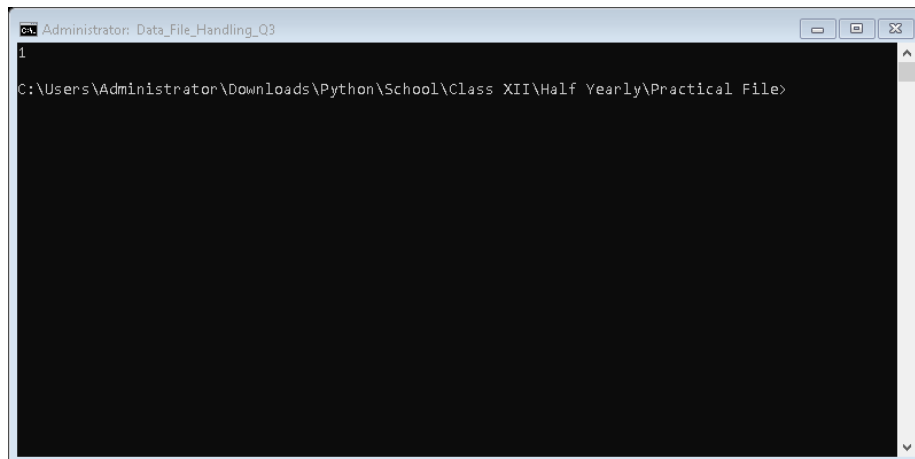


```
Administrator: Data_File_Handling_Q2
Traceback (most recent call last):
  File "C:\Users\Administrator\Downloads\Python\School\Class XII\Half Yearly\Practical File\tem
p_Data_File_Handling_2.py", line 1, in <module>
    f=open("Poem.txt")
FileNotFoundError: [Errno 2] No such file or directory: 'Poem.txt'

C:\Users\Administrator\Downloads\Python\School\Class XII\Half Yearly\Practical File>
```

Q3) Write a program to count the number of uppercase alphabets present in text file 'Poem.txt'.

```
1 f=open("Poem.txt")
2 x=f.read()
3 f.close()
4 count=0
5 y="QWERTYUIOPASDFGHJKLZXCVBNM"
6 for i in x:
7     if i in y:
8         count+=1
9 print(count)
```



```
Administrator: Data_File_Handling_Q3
1
C:\Users\Administrator\Downloads\Python\School\Class XII\Half Yearly\Practical File>
```

Q4) Write a program that copies one file to another and reads the filenames from the user.

```

1 b=input("Enter a File Name/Path to read from: ")
2 with open(b,"r") as f:
3     MyS=f.read()
4     f.close()
5 a=input("Enter a New Name for Location on Desktop: ")
6 with open(a,"w") as F:
7     F.writelines(MyS)
8     F.close()

```

```

Administrator: Data_File_Handling_Q4
Enter a File Name/Path to read from: Poem.txt
Enter a New Name for Location on Desktop: New_Poem.txt
C:\Users\Administrator\Downloads\Python\School\Class XII\Half Yearly\Practical File>

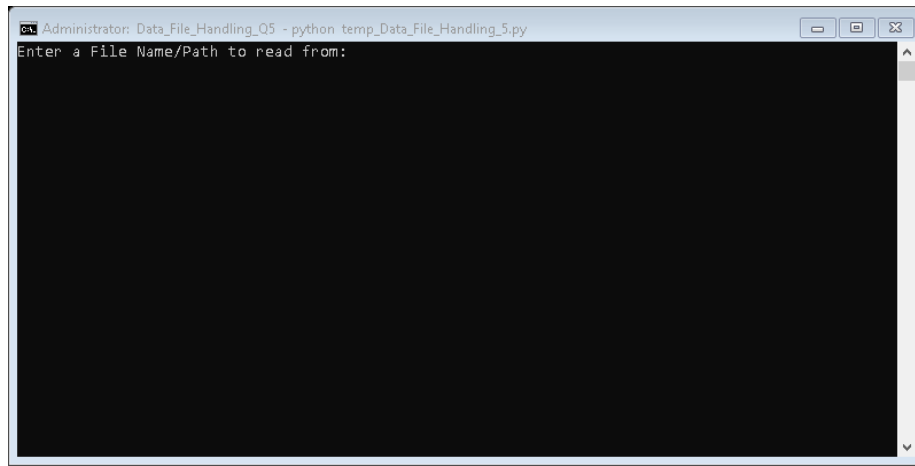
```

**Q5) Write a program that appends the contents of one file to another and takes the filenames from the user.**

```

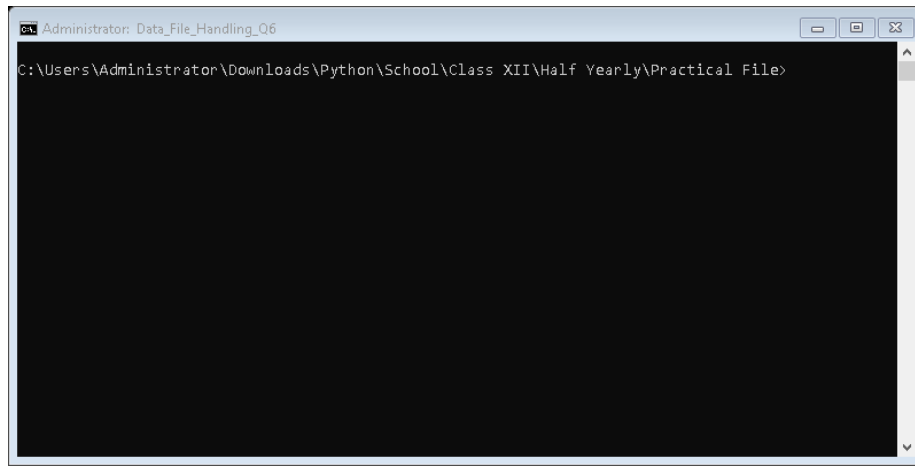
1 b=input("Enter a File Name/Path to read from: ")
2 with open(b,"r") as f:
3     MyS=f.read()
4     f.close()
5 a=input("Enter a New Name for Location on Desktop: ")
6 with open(a,"a") as F:
7     F.writelines(MyS)

```



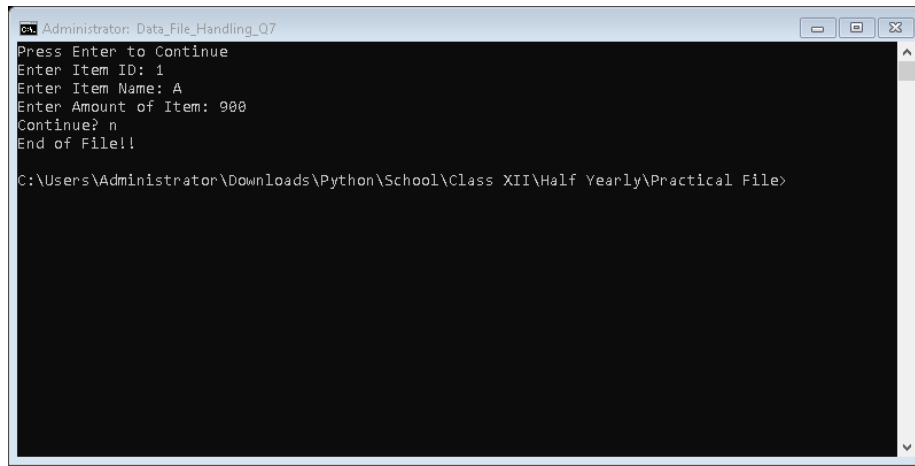
**Q6)** Write a program that reads characters from the keyboard one by one. All lowercase characters get stored in the file 'LOWER', all uppercase characters get stored in the file 'UPPER' and all the other characters get stored in the file 'OTHERS'.

```
1 f=open("Poem.txt")
2 a=f.read()
3 f.close()
4 l="asdfghjklqwertyuiopzxcvbnm"
5 u="QWERTYUIOPASDFGHJKLZXCVBNM"
6 for i in a:
7     if i in l:
8         with open("LOWER.txt", "a") as f:
9             f.write(i)
10            f.close()
11    elif i in u:
12        with open("UPPER.txt", "a") as f:
13            f.write(i)
14            f.close()
15    elif i!=" ":
16        with open("OTHER.txt", "a") as f:
17            f.write(i)
18            f.close()
```



Q7) Consider binary file 'items.dat' containing records stored in the given format: {item\_id: [item\_name, amount]}. Write a function, copy\_new(), that copies all records whose amount is greater than 1000 from 'items.dat' to 'new\_items.dat'.

```
1 import pickle
2 with open("items.dat","wb") as f:
3     d={}
4     c=input("Press Enter to Continue")
5     while c.lower()!='n':
6         x=input("Enter Item ID: ")
7         y=input("Enter Item Name: ")
8         z=int(input("Enter Amount of Item: "))
9         l=[]
10        l.append(y)
11        l.append(z)
12        d[x]=l
13        c=input("Continue? ")
14    pickle.dump(d,f)
15 def copy_new():
16     with open("items.dat","rb") as f:
17         try:
18             d=pickle.load(f)
19         except:
20             print("End of File!!")
21     with open("new_items.dat","wb") as f:
22         for i in d:
23             if d[i][1]>=1000:
24                 pickle.dump(d[i],f)
25     with open("new_items.dat","rb") as f:
26         try:
27             while True:
28                 print(pickle.load(f))
29         except:
30             print("End of File!!")
31 copy_new()
```

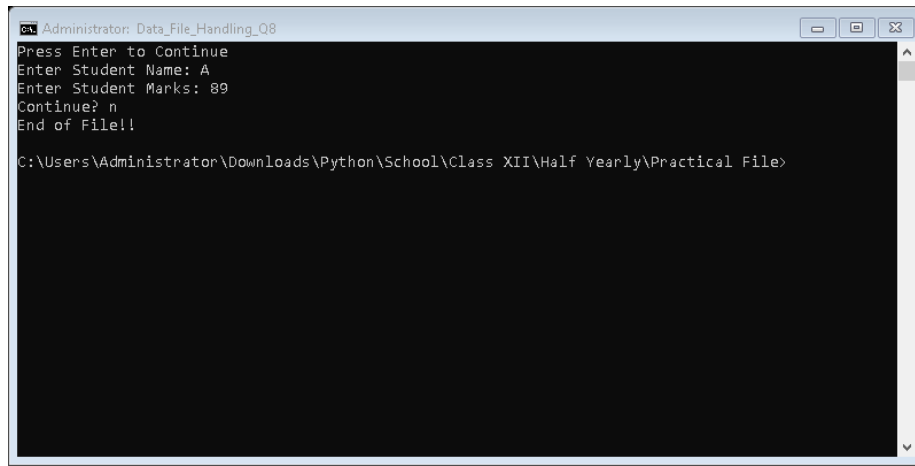


```
Administrator: Data_File_Handling_Q7
Press Enter to Continue
Enter Item ID: 1
Enter Item Name: A
Enter Amount of Item: 900
Continue? n
End of File!!

C:\Users\Administrator\Downloads\Python\School\Class XII\Half Yearly\Practical File>
```

Q8) Anant has been asked to display the names of all students who have scored less than 40 for Remedial Classes. Write a user-defined function to display the names of the students from the binary file 'Student.dat' who have less than 40.

```
1 import pickle
2 with open("Students.dat","wb") as f:
3     d={}
4     c=input("Press Enter to Continue")
5     while c.lower()!='n':
6         x=input("Enter Student Name: ")
7         l=int(input("Enter Student Marks: "))
8         d[x]=l
9         c=input("Continue? ")
10    pickle.dump(d,f)
11 def copy_new():
12     with open("Students.dat","rb") as f:
13         try:
14             d=pickle.load(f)
15         except:
16             print("End of File!!")
17     with open("Remedial.dat","wb") as f:
18         for i in d:
19             if d[i]<=40:
20                 pickle.dump(i,f)
21     with open("Remedial.dat","rb") as f:
22         try:
23             while True:
24                 print(pickle.load(f))
25         except:
26             print("End of File!!")
27 copy_new()
```



```
Administrator: Data_File_Handling_Q8
Press Enter to Continue
Enter Student Name: A
Enter Student Marks: 89
Continue? n
End of File!!

C:\Users\Administrator\Downloads\Python\School\Class XII\Half Yearly\Practical File>
```

Q9) Given a binary file, 'STUDENT.dat', containing records of the following type: [S\_Admno, S\_Name, Percentage] Where these three values are: S\_Admno - Admission Number of student (string) S\_Name - Name of student (string) Percentage - percentage obtained by student (float) Write a function in Python that would read the contents of the file 'STUDENT.dat' and that would display the details of those students whose percentage is below 65.

```
1 import pickle
2 with open("items.dat","wb") as f:
3     d={}
4     c=input("Press Enter to Continue")
5     while c.lower()!='n':
6         x=input("Enter Student Admission No.: ")
7         y=input("Enter Student Name: ")
8         z=int(input("Enter Percentage: "))
9         l=[]
10        l.append(y)
11        l.append(z)
12        d[x]=l
13        c=input("Continue? ")
14    pickle.dump(d,f)
15 def copy_new():
16     with open("items.dat","rb") as f:
17         try:
18             d=pickle.load(f)
19         except:
20             print("End of File!!")
21     with open("new_items.dat","wb") as f:
22         for i in d:
23             if d[i][1]<=65:
24                 D={}
25                 D[i]=d[i]
```

```

26         pickle.dump(D,f)
27     with open("new_items.dat","rb") as f:
28         try:
29             while True:
30                 print(pickle.load(f))
31             except:
32                 print("End of File!!")
33 copy_new()

```

```

Administrator: Data_File_Handling_Q9
Press Enter to Continue
Enter Student Admission No.: 1
Enter Student Name: A
Enter Percentage: 59
Continue? n
{'1': ['A', 59]}
End of File!!

C:\Users\Administrator\Downloads\Python\School\Class XII\Half Yearly\Practical File>

```

**Q10)** Create CSV file 'Groceries' to store information of different items existing in a shop. The information is to be stored with respect to each item code, name, price, qty. Write a program to accept the data from the user and store it permanently in the CSV file.

```

1 import csv
2 try:
3     with open("Groceries.csv", "r") as f_check:
4         is_empty = f_check.readline() == ''
5 except FileNotFoundError:
6     is_empty = True
7 with open("Groceries.csv", "a", newline='') as f:
8     writer = csv.writer(f)
9     if is_empty:
10        writer.writerow(["Item Code", "Item Name", "Price", "
11        Quantity"])
12    num_rows = int(input("Enter number of items to add: "))
13    rows = []
14    for _ in range(num_rows):
15        code = input("Enter Item Code: ")
16        name = input("Enter Item Name: ")
17        price = input("Enter Price: ")
18        qty = input("Enter Quantity: ")
19        rows.append([code, name, price, qty])
20    writer.writerows(rows)

```



```

20 with open("Groceries.csv", "r", newline='') as f:
21     reader = csv.reader(f)
22     next(reader)
23     for row in reader:
24         print(row)

```

```

Administrator: Data_File_Handling_Q10
Enter number of items to add: 1
Enter Item Code: test
Enter Item Name: test12
Enter Price: 341
Enter Quantity: 1
['test', 'test12', '341', '1']
C:\Users\Administrator\Downloads\Python\School\Class XII\Half Yearly\Practical File>

```

## 4 Stack Operations

Q1) Write a program to implement pop and push functions on a stack containing package name as records.

```

1 stack = []
2
3 def push(item):
4     stack.append(item)
5
6 def pop():
7     if not stack:
8         return "Underflow"
9     return stack.pop()
10
11
12 push("Package1")
13 push("Package2")
14 print(stack)
15 popped_item = pop()
16 print(popped_item)
17 print(stack)

```

Q2) Write a program to sort a stack into ascending order without using another stack.

```

1 def sort_stack(stack):

```

```

2     if stack:
3         # Remove the top element
4         temp = stack.pop()
5         # Sort the remaining stack
6         sort_stack(stack)
7         # Insert the temp back in sorted position
8         sorted_insert(stack, temp)
9
10    def sorted_insert(stack, element):
11        if not stack or stack[-1] <= element:
12            stack.append(element)
13        else:
14            temp = stack.pop()
15            sorted_insert(stack, element)
16            stack.append(temp)
17
18
19    stack = [3, 1, 2]
20    sort_stack(stack)
21    print(stack)

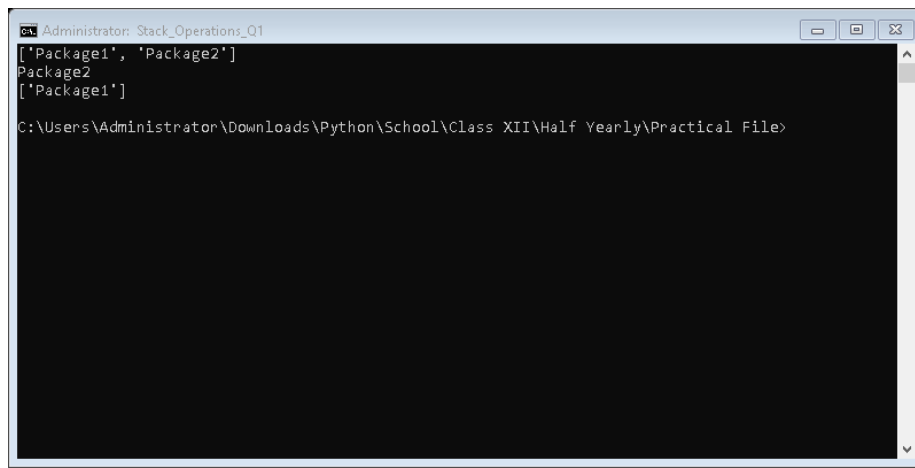
```

**Q1) Write a program to implement pop and push functions on a stack containing package name as records.**

```

1    stack = []
2
3    def push(item):
4        stack.append(item)
5
6    def pop():
7        if not stack:
8            return "Underflow"
9        return stack.pop()
10
11
12    push("Package1")
13    push("Package2")
14    print(stack)
15    popped_item = pop()
16    print(popped_item)
17    print(stack)

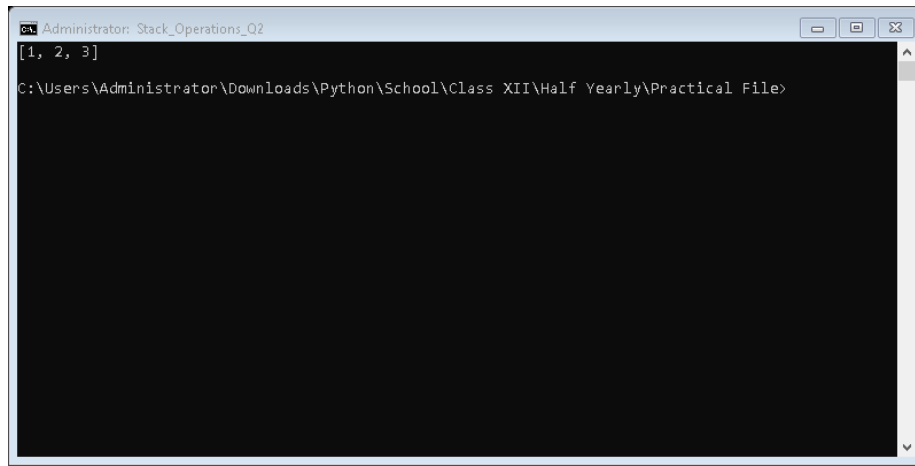
```



```
Administrator: Stack_Operations_Q1
['Package1', 'Package2']
Package2
['Package1']
C:\Users\Administrator\Downloads\Python\School\Class XII\Half Yearly\Practical File>
```

Q2) Write a program to sort a stack into ascending order without using another stack.

```
1 def sort_stack(stack):
2     if stack:
3         # Remove the top element
4         temp = stack.pop()
5         # Sort the remaining stack
6         sort_stack(stack)
7         # Insert the temp back in sorted position
8         sorted_insert(stack, temp)
9
10 def sorted_insert(stack, element):
11     if not stack or stack[-1] <= element:
12         stack.append(element)
13     else:
14         temp = stack.pop()
15         sorted_insert(stack, element)
16         stack.append(temp)
17
18
19 stack = [3, 1, 2]
20 sort_stack(stack)
21 print(stack)
```



Q3) Write a program to implement pop and push operations on a stack. The push operation should add numbers from a list which have 5 digits or more. The pop operation should print underflow if stack is empty.

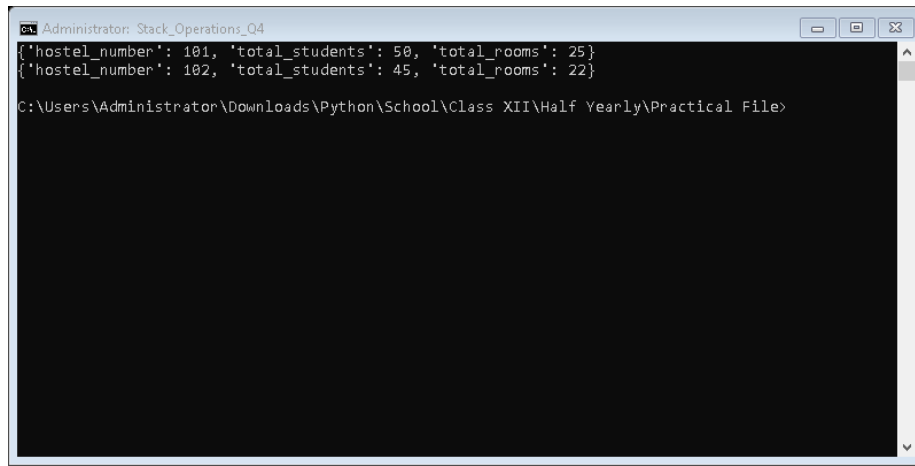
```
1 stack = []
2
3 def push_from_list(lst):
4     for num in lst:
5         if len(str(abs(num))) >= 5: # Check if the number has 5 or
6             more digits
7                 stack.append(num)
8
9 def pop():
10     if not stack:
11         return "Underflow"
12     return stack.pop()
13
14 numbers_list = [12345, 1234, 123456, 123, 1234567]
15 push_from_list(numbers_list)
16 print(stack)
17 print(pop())
18 print(stack)
19 print(pop())
20 print(stack)
21 print(pop())
22 print(stack)
23 print(pop())
24 print(stack)
25 print(pop())
26 print(stack)
27 print(pop()) # Should print "Underflow"
```

```
Administrator: Stack_Operations_Q3
[12345, 123456, 1234567]
1234567
[12345, 123456]
123456
[12345]
12345
[]
Underflow
[]
Underflow
[]
Underflow

C:\Users\Administrator\Downloads\Python\School\Class XII\Half Yearly\Practical File>
```

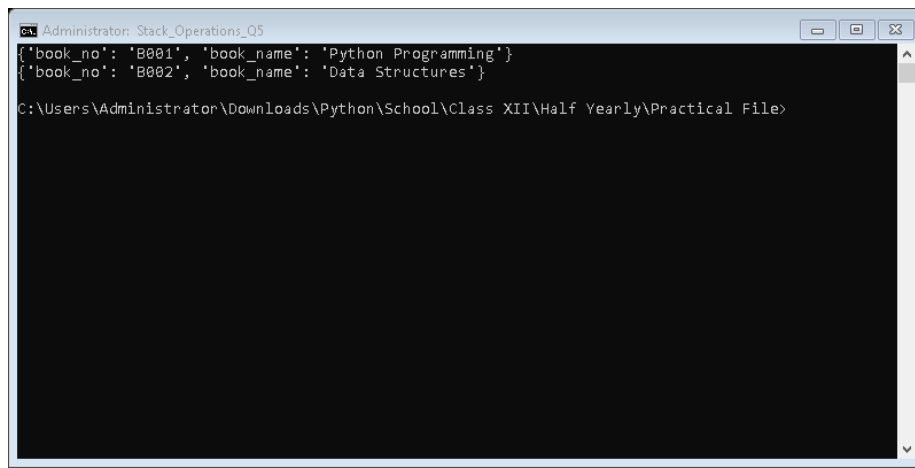
**Q4) Write a program to implement push and display operations on a stack with hostel number, total students and total rooms as record.**

```
1 stack = []
2
3 def push_hostel_record(hostel_number, total_students, total_rooms):
4     record = {
5         'hostel_number': hostel_number,
6         'total_students': total_students,
7         'total_rooms': total_rooms
8     }
9     stack.append(record)
10
11 def display_stack():
12     for record in stack:
13         print(record)
14
15
16 push_hostel_record(101, 50, 25)
17 push_hostel_record(102, 45, 22)
18 display_stack()
```



**Q5)** Write a program to implement push and display operations on a stack which has book number and name as records.

```
1 stack = []
2
3 def push_book_record(book_no, book_name):
4     record = {
5         'book_no': book_no,
6         'book_name': book_name
7     }
8     stack.append(record)
9
10 def display_stack():
11     for record in stack:
12         print(record)
13
14
15 push_book_record("B001", "Python Programming")
16 push_book_record("B002", "Data Structures")
17 display_stack()
```

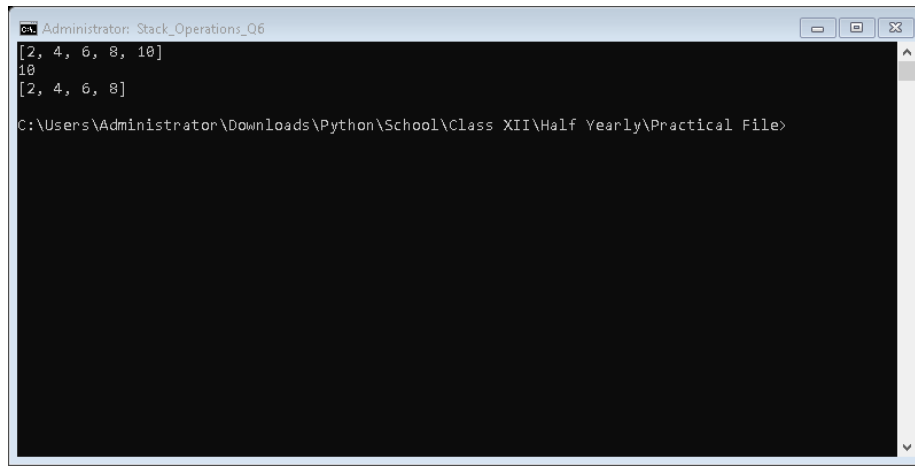


```
Administrator: Stack_Operations_Q5
{'book_no': 'B001', 'book_name': 'Python Programming'}
{'book_no': 'B002', 'book_name': 'Data Structures'}

C:\Users\Administrator\Downloads\Python\School\Class XII\Half Yearly\Practical File>
```

**Q6)** Write a program to push elements in a stack from a list that are even. Also implement pop function.

```
1 stack = []
2
3 def push_evens_from_list(lst):
4     for num in lst:
5         if num % 2 == 0:
6             stack.append(num)
7
8 def pop():
9     if not stack:
10        return "Underflow"
11    return stack.pop()
12
13
14 numbers_list = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
15 push_evens_from_list(numbers_list)
16 print(stack)
17 print(pop())
18 print(stack)
```



```
Administrator: Stack_Operations_Q6
[2, 4, 6, 8, 10]
10
[2, 4, 6, 8]
C:\Users\Administrator\Downloads\Python\School\Class XII\Half Yearly\Practical File>
```

**Q7)** Write a program to push student names from a dictionary into a stack who have won more than 3 medals. Also it prints the the no of names pushed and overflow if the no of items exceed 15.

```
1 stack = []
2 max_size = 15
3
4 def push_students(dictionary):
5     count = 0
6     for name, medals in dictionary.items():
7         if medals > 3:
8             if len(stack) >= max_size:
9                 print("Overflow")
10                return
11                stack.append(name)
12                count += 1
13                print(f"Number of names pushed: {count}")
14
15 def pop():
16     if not stack:
17         return "Underflow"
18     return stack.pop()
19
20
21 students = {
22     "Alice": 4,
23     "Bob": 2,
24     "Charlie": 5,
25     "David": 3,
26     "Eve": 6
27 }
28 push_students(students)
29 print(stack)
30 print(pop())
31 print(stack)
```



```
Administrator: Stack_Operations_Q7
Number of names pushed: 3
['Alice', 'Charlie', 'Eve']
Eve
['Alice', 'Charlie']
C:\Users\Administrator\Downloads\Python\School\Class XII\Half Yearly\Practical File>
```

## 5 Relational Databases and SQL

Q20) Ms Veda created a table named GAME, containing columns Game\_id, P\_Age, and G\_Name. After creating the table, Category has to be added. Help her to write the command to add the Category column. Thereafter, write the command to insert the following record in the table:

Table: GAME

Game_id	P_Age	G_Name	Category
G42	18	Chess	Senior

```
1 ALTER TABLE GAME ADD Category VARCHAR(20);
2
3 INSERT INTO GAME VALUES ('G42', 18, 'Chess', 'Senior');
```

Q21) Define the following terms:

- (a) **Database:** A database is an organized collection of structured data stored electronically in a computer system, typically controlled by a database management system (DBMS).
- (b) **Data Inconsistency:** Data inconsistency occurs when the same data exists in different formats or values in different places, leading to conflicting information.
- (c) **Primary Key:** A primary key is a column or set of columns that uniquely identifies each row in a table. It cannot contain NULL values and must be unique.

- **(d) Candidate Key:** A candidate key is a column or set of columns that can uniquely identify any record in a table. A table can have multiple candidate keys, and one of them is chosen as the primary key.
- **(e) Foreign Key:** A foreign key is a column or set of columns in one table that refers to the primary key in another table, establishing a relationship between the two tables.

**Q22) Differentiate between Primary key and Unique constraints.**

- **Primary Key:**
  - Cannot contain NULL values
  - Only one primary key per table
  - Automatically creates a clustered index
  - Used to establish relationships with other tables
- **Unique Constraint:**
  - Can contain one NULL value
  - Multiple unique constraints per table
  - Creates a non-clustered index
  - Ensures uniqueness but doesn't establish relationships

**Q26) Based on the given table, write SQL queries for the following:**

```

1 ALTER TABLE Projects ADD PRIMARY KEY (P_id);
2
3 UPDATE Projects SET language = 'Python' WHERE id = 'P002';
4
5 DROP TABLE Projects;
```

**Q27) Write SQL commands for (a) to (f) on the basis of relations given below:**

**Table: BOOKS**

Book_ID	Book_name	Author_name	Publishers	Price	Type	Qty
K0001	Let Us C	Y. Kanetkar	EPB	450	Prog	15

```

1 SELECT * FROM BOOKS
2 WHERE Publishers = 'FIRST PUBL' AND Author_name = 'P. Purohit';
3
4 SELECT Book_name, Price FROM BOOKS
5 WHERE Publishers = 'FIRST PUBL';
6
7 UPDATE BOOKS SET Price = Price * 0.95
8 WHERE Publishers = 'EPB';
9
10 SELECT Book_name, Price FROM BOOKS
11 WHERE Qty > 3;
12
13 SELECT Type, SUM(Price) AS Total_Cost FROM BOOKS
14 GROUP BY Type;
15
16 SELECT * FROM BOOKS
17 WHERE Price = (SELECT MAX(Price) FROM BOOKS);

```

## Q28) What are DDL and DML?

- **DDL (Data Definition Language):** DDL commands are used to define and modify database structures. Examples include CREATE, ALTER, DROP, and TRUNCATE.
- **DML (Data Manipulation Language):** DML commands are used to manipulate data stored in the database. Examples include SELECT, INSERT, UPDATE, and DELETE.

## Q29) Differentiate between primary key and candidate key in a relation.

- **Primary Key:**
  - Selected from candidate keys
  - Only one per table
  - Used to establish relationships
  - Cannot be NULL
- **Candidate Key:**
  - Set of all possible keys
  - Can be multiple in a table
  - Any candidate key can become primary key
  - Uniquely identifies records

**Q30) What do you understand by the terms Cardinality and Degree of a relation in relational database?**

- **Cardinality:** The number of rows (tuples) in a table.
- **Degree:** The number of columns (attributes) in a table.

**Q31) Differentiate between DDL and DML. Mention the two commands for each category.**

- **DDL (Data Definition Language):**
  - Defines database structure
  - Auto-commit (changes are permanent)
  - Commands: CREATE, ALTER, DROP, TRUNCATE
- **DML (Data Manipulation Language):**
  - Manipulates data in tables
  - Can be rolled back
  - Commands: SELECT, INSERT, UPDATE, DELETE

**Q32) Write SQL Commands for (a) to (h) on the basis of the following table: FURNITURE**

**Table: FURNITURE**

NO	ITEM	TYPE	DATEOFSTOCK	PRICE	DISCOUNT
1	WhiteLotus	DoubleBed	2002-02-23	3000	25

```
1 SELECT * FROM FURNITURE WHERE PRICE > 10000;  
2  
3 SELECT ITEM, PRICE FROM FURNITURE  
4 WHERE DISCOUNT BETWEEN 10 AND 20;  
5  
6 DELETE FROM FURNITURE WHERE DISCOUNT = 30;  
7  
8 SELECT ITEM, TYPE, PRICE FROM FURNITURE  
9 WHERE ITEM LIKE 'B%';  
10  
11 SELECT PRICE FROM FURNITURE WHERE ITEM = 'BabyCot';  
12  
13 SELECT DISTINCT Type FROM Furniture;  
14  
15 SELECT MAX(Price) FROM Furniture  
16 WHERE DateOfStock > '2002-02-15';
```

**Q32h) Write SQL Commands/output for the following on the basis of the given table GRADUATE:**

**Table: GRADUATE**

S.No	NAME	STIPEND	SUBJECT	AVERAGE	RANK
1	KARAN	400	PHYSICS	68	1

```

1 SELECT NAME FROM GRADUATE
2 WHERE RANK = 1
3 ORDER BY NAME;
4
5 SELECT NAME FROM GRADUATE
6 WHERE AVERAGE > 65;
7
8 SELECT NAME FROM GRADUATE
9 WHERE SUBJECT = 'COMPUTER' AND AVERAGE > 60;
10
11 SELECT NAME FROM GRADUATE
12 ORDER BY NAME;
13
14 SELECT * FROM GRADUATE
15 WHERE NAME LIKE '%i%';
16
17 SELECT DISTINCT RANK FROM GRADUATE;

```

**Q33a) What is the difference between Candidate key and Alternate key?**

- **Candidate Key:** All keys that can uniquely identify records in a table.
- **Alternate Key:** Candidate keys that are not chosen as the primary key.

**Q33b) What is the degree and cardinality of a table having 10 rows and 5 columns?**

- **Degree:** 5 (number of columns)
- **Cardinality:** 10 (number of rows)

**Q33c) For the given table, do as directed:**

**Table: STUDENT**

ColumnName	Data type	Size	Constraint
ROLLNO	Integer	4	Primary Key
SNAME	Varchar	25	NOT NULL

```

1 CREATE TABLE STUDENT (
2     ROLLNO INT(4) PRIMARY KEY,
3     SNAME VARCHAR(25) NOT NULL
4 );
5
6 ALTER TABLE STUDENT MODIFY SNAME VARCHAR(30);
7
8 ALTER TABLE STUDENT DROP COLUMN HOBBY;
9
10 INSERT INTO STUDENT VALUES (101, 'Rahul Kumar');

```

**Q34) Write the SQL commands to perform the following tasks:**

```

1 SELECT MIN(salary), MAX(salary), AVG(salary)
2 FROM employees
3 WHERE designation = 'Manager';
4
5 SELECT COUNT(*) FROM employees
6 WHERE designation = 'Clerk';
7
8 SELECT name, salary, date_of_joining, designation
9 FROM employees
10 ORDER BY designation;
11
12 SELECT COUNT(*) FROM employees
13 WHERE commission IS NULL;
14
15 SELECT AVG(salary) FROM employees
16 WHERE salary > 2000
17 GROUP BY DeptID;
18
19 SELECT DeptID, COUNT(*) FROM employees
20 GROUP BY DeptID;
21
22 SELECT DeptID, MAX(salary) FROM employees
23 GROUP BY DeptID;
24
25 SELECT e.name, e.designation, d.department_name
26 FROM employees e
27 JOIN departments d ON e.DeptID = d.DeptID;
28
29 SELECT COUNT(*) FROM employees
30 WHERE DeptID = (SELECT DeptID FROM departments
31 WHERE department_name = 'ACCOUNTS');

```

**Q34 Additional) Based on Table: PRODUCTS**

**Table: PRODUCTS**

PCODE	PNAME	COMPANY	PRICE	STOCK	MANUFACTURE	WARRANTY
P001	TV	BPL	10000	200	2018-01-12	3

```

1 SELECT * FROM PRODUCTS
2 WHERE PNAME = 'PC' AND STOCK > 110;
3
4 SELECT COMPANY FROM PRODUCTS
5 WHERE WARRANTY > 2;
6
7 SELECT SUM(PRICE * STOCK) AS Stock_Value
8 FROM PRODUCTS
9 WHERE COMPANY = 'BPL';
10
11 SELECT COMPANY, COUNT(*) AS Product_Count
12 FROM PRODUCTS
13 GROUP BY COMPANY;
14
15 SELECT COUNT(*) FROM PRODUCTS
16 WHERE DATE_ADD(MANUFACTURE, INTERVAL WARRANTY YEAR) < '2020-11-20';
17
18 SELECT PNAME FROM PRODUCTS
19 WHERE DATE_ADD(MANUFACTURE, INTERVAL WARRANTY YEAR) >= CURDATE();

```

**Q35) Write SQL queries based on the following tables:**

**Table: PRODUCT**

P_ID	ProductName	Manufacturer	Price	Discount
TP01	Talcum Powder	LAK	40	NULL

**Table: CLIENT**

C_ID	ClientName	City	P_ID
01	Cosmetic Shop	Delhi	TP01

```

1 CREATE TABLE STUDENT (
2     ROLLNO INT PRIMARY KEY,
3     SNAME VARCHAR(25) NOT NULL,
4     HOBBY VARCHAR(20)
5 );
6
7 ALTER TABLE STUDENT MODIFY SNAME VARCHAR(30);
8
9 ALTER TABLE STUDENT DROP COLUMN HOBBY;
10
11 INSERT INTO STUDENT VALUES (101, 'Amit Singh', 'Reading');

```

**Q36) Write SQL queries based on the following tables:**

**Table: PRODUCT and CLIENT (as shown above)**

```

1 SELECT ProductName, Price FROM PRODUCT
2 WHERE Price BETWEEN 50 AND 150;
3
4 SELECT * FROM PRODUCT
5 WHERE Manufacturer IN ('XYZ', 'ABC');
6

```

```

7 SELECT ProductName, Manufacturer, Price FROM PRODUCT
8 WHERE Discount IS NULL;
9
10 SELECT ProductName, Price FROM PRODUCT
11 WHERE ProductName LIKE '%h';
12
13 SELECT ClientName, City, C.P_ID, ProductName
14 FROM CLIENT C
15 JOIN PRODUCT P ON C.P_ID = P.P_ID
16 WHERE City = 'Delhi';

```

**Answer (f):** P\_ID is used as Foreign Key in the CLIENT table.

**Q37) Answer the questions based on the table given below:**

**Table: HOSPITAL**

S.No	Name	Age	Department	DateOfAdm	Charges	Sex
1	Arpit	62	Surgery	1998-01-15	800	M

```

1 SELECT Name FROM HOSPITAL
2 WHERE DateOfAdm > '1998-01-15';
3
4 SELECT Name FROM HOSPITAL
5 WHERE Sex = 'F' AND Department = 'ENT';
6
7 SELECT Name, DateOfAdm FROM HOSPITAL
8 WHERE AVERAGE > 65;
9
10 SELECT Name, Charges, Age FROM HOSPITAL
11 WHERE Sex = 'F'
12 ORDER BY Age;
13
14 SELECT COUNT(DISTINCT Charges) FROM HOSPITAL;
15
16 SELECT MIN(Age) FROM HOSPITAL
17 WHERE Sex = 'F';

```

**Q38) A department store MyStore is considering to maintain their inventory using SQL**

**Table: STORE**

ItemNo	ItemName	SCode	Quantity
2005	Sharpener Classic	23	60

**(a)** ItemNo is best suitable as primary key.

**(b)** Degree: 4 (number of columns), Cardinality: 1 (number of rows shown)

```

1 INSERT INTO STORE (ItemNo, ItemName, SCode)
2 VALUES (2010, 'Note Book', 25);

```

**(d)** Answer: (ii) DROP TABLE STORE;

```

1 DESCRIBE STORE;

```



**Q39)** Consider the tables Admin and Transport given below:

**Table: Admin**

S_id	S_name	Address	S_type
S001	Sandhya	Rohini	Day Boarder

**Table: Transport**

S_id	S_name	S_type
S002	TSS10	Sarai Kale Khan

```
1 SELECT A.S_name, T.S_name AS Stop_Name
2 FROM Admin A
3 JOIN Transport T ON A.S_id = T.S_id;
4
5 SELECT COUNT(*) FROM Admin
6 WHERE S_type IS NULL;
7
8 SELECT * FROM Admin
9 WHERE S_name LIKE 'V%';
10
11 SELECT S_id, Address FROM Admin
12 ORDER BY S_name;
```