LOTUS VALLEY INTERNATIONAL SCHOOL:
Computer Science Practical File

Pranav Verma (Roll No. 7, XII Aryabhatta)

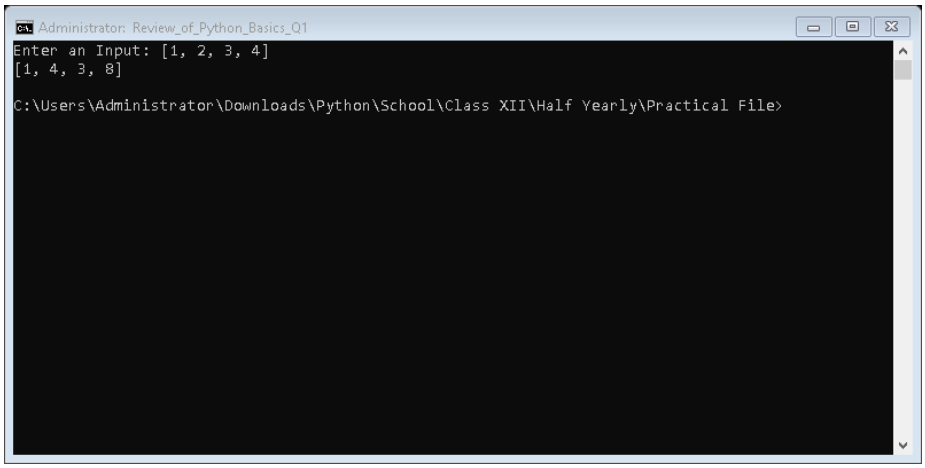
November 1, 2025

Contents

1 Review of Python Basics

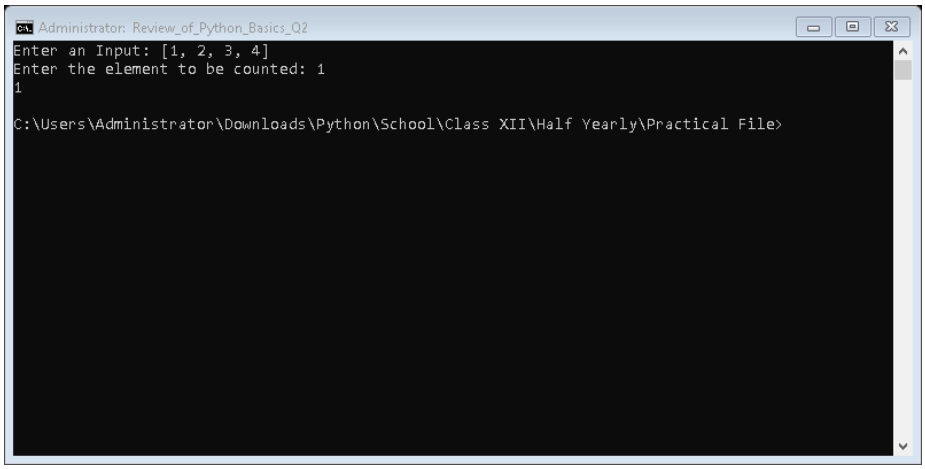
Q1) Write a program to multiply an element by two, if it is an odd index for a given list containing both numbers and strings.

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



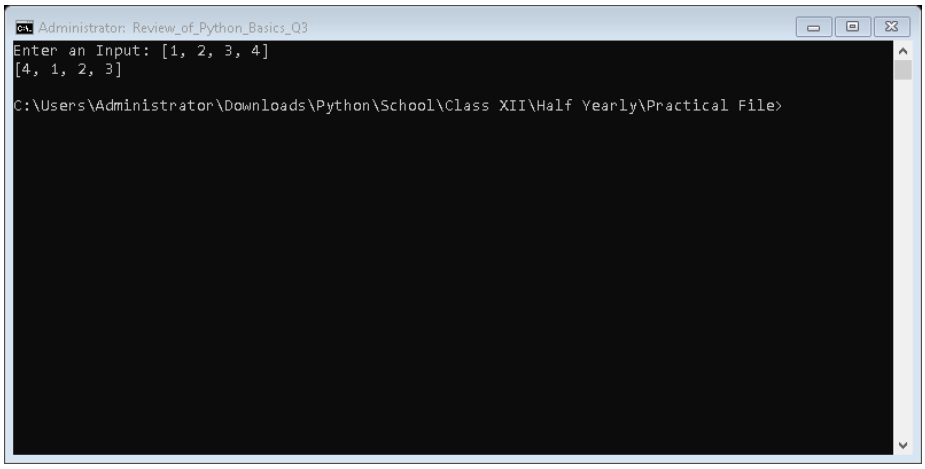
Q2) Write a program to count the frequency of an element in a list.

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



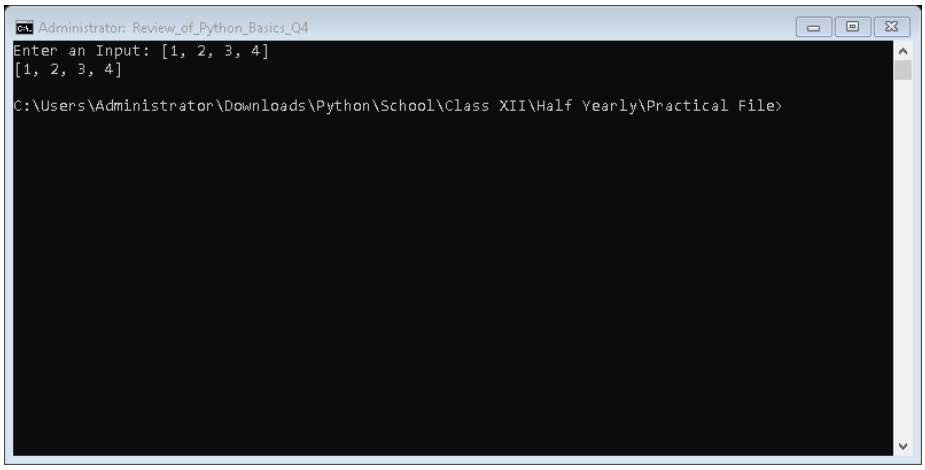
Q3) Write a program to shift elements of a list so that the first element moves to the second index and second index move to the third index and so on, and the last element shifts to the first position.

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



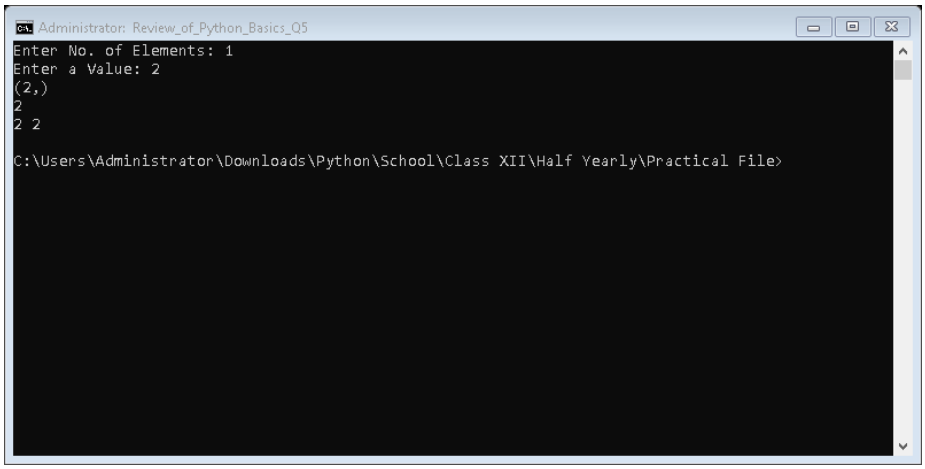
Q4) A list NUM contains the elements: 3,25,13,6,35,8,14,45. Write a program to swap the content with the next value divisible by 5 so that the resultant list will look like: 25,3,13,35,6,8,45,14.

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



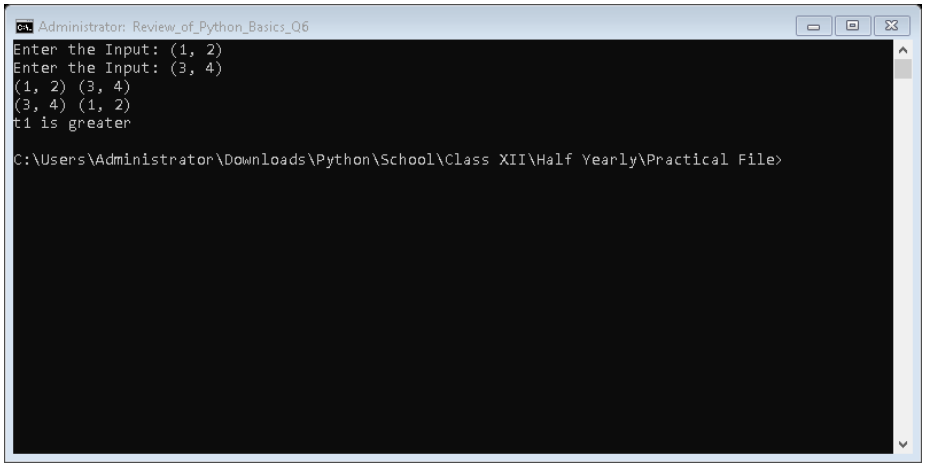
Q5) Write a program to accept values from a user in a tuple. Add a tuple to it and display its elements one by one. Also display its maximum and minimum values.

```python
n=int(input('Enter No. of Elements: '))
t=()
for i in range(n):
    x=eval(input('Enter a Value: '))
    t+=x,
print(t)
for i in t:
    print(i)
print(max(t),min(t),end='\n')
```



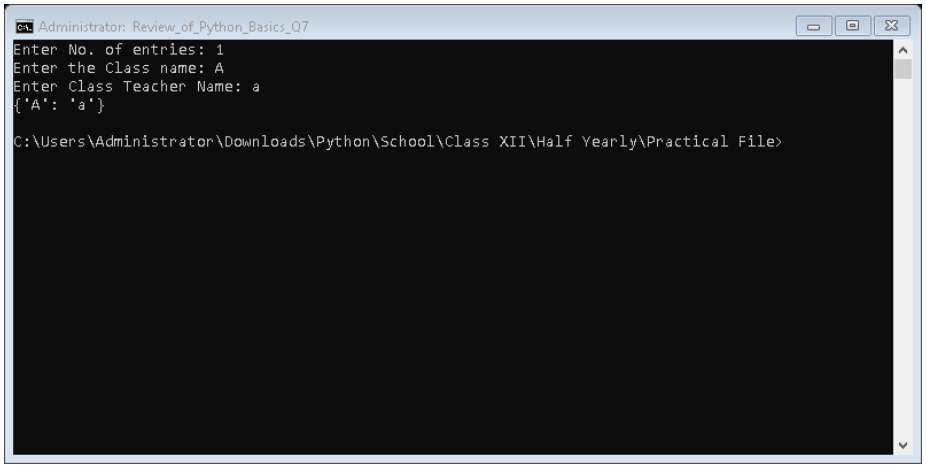
Q6) Write a program to input any values for two tuples. Print it, interchange it and then compare them.

```
t1=eval(input("Enter the Input: "))
t2=eval(input("Enter the Input: "))
print(t1, t2)
x=t2
t2=t1
t1=x
print(t1, t2)
if t1<t2:
    print('t2 is greater')
else:
    print('t1 is greater')
```



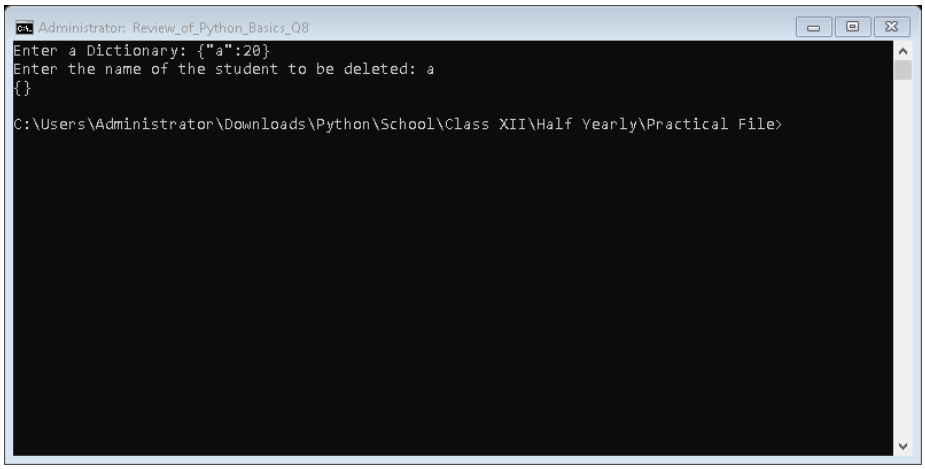
Q7) Write a python program to input 'n' classes and names of class teachers to store them in a dictionary and display the same. Also accept a particular class from the user and display the name of the class teacher of that class.

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



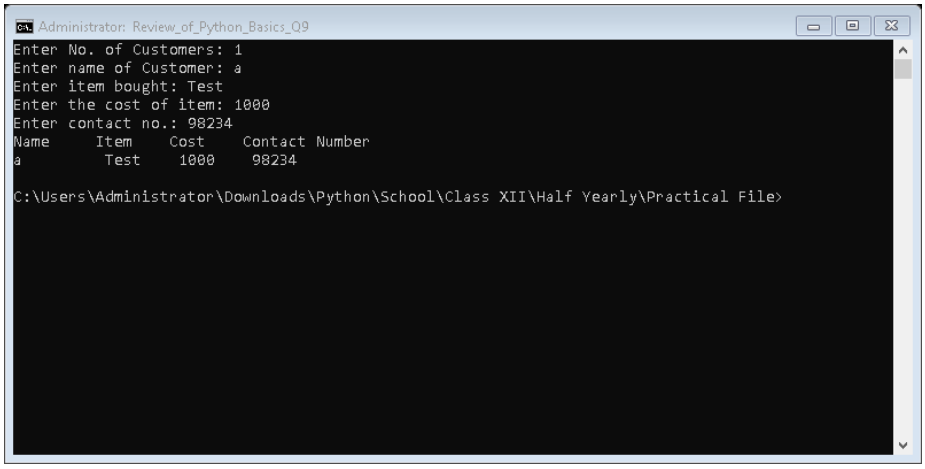
Q8) Write a program to store students names and their percentage in a dictionary and delete a particular student name from the dictionary. Also display the dictionary after deletion.

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



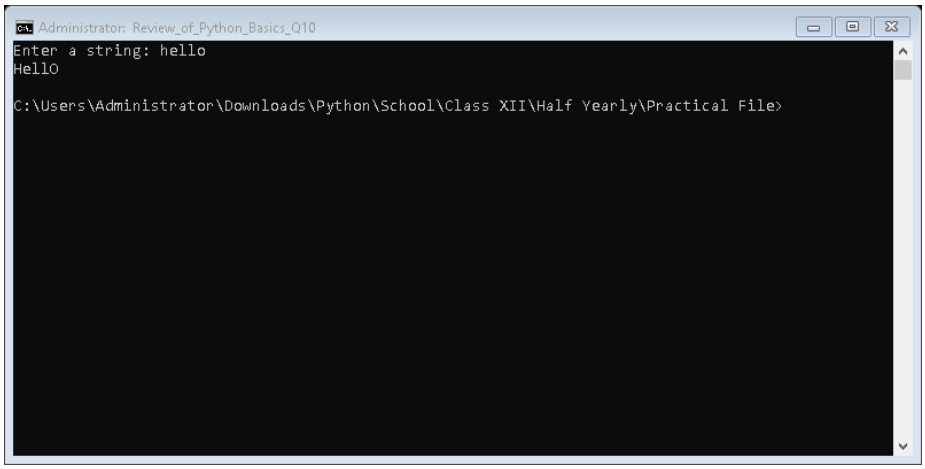
Q9) Write a Python program to input names of 'n' customers and their details like items bought, cost and phone number, etc., store them in a dictionary and display all the details in a tabular form.

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



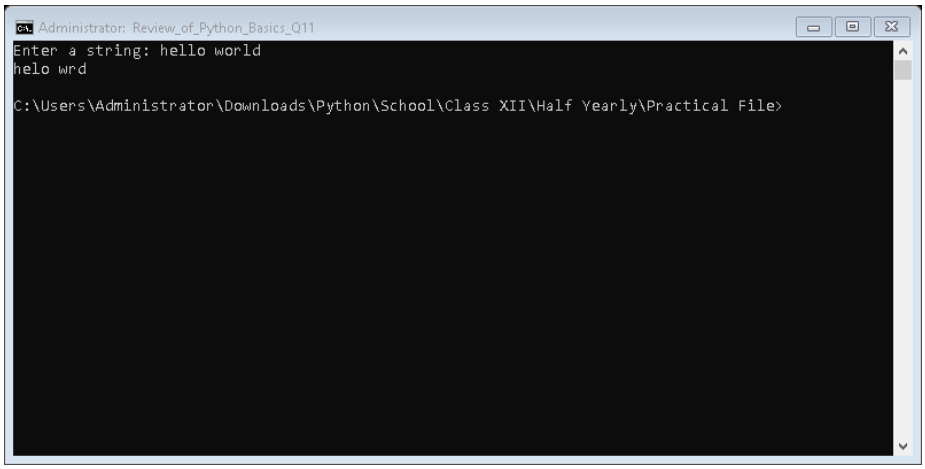
Q10) Write a Python program to capitalize the first and last letters of each word of a given string.

```
s=input('Enter a string: ')
j=''
x=''
l=s.split()
for i in l:
    j=i[0].upper()+i[1:-1]+i[-1].upper()
    x+=j+''
print(x)
```



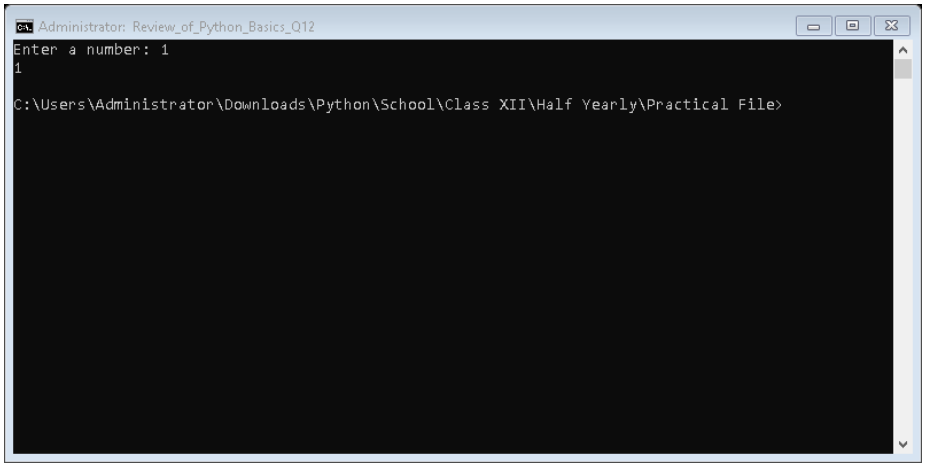
Q11) Write a Python program to remove duplicate characters of a given string.

```
s=input('Enter a string: ')
x=''
for i in s:
    if i not in x:
        x+=i
print(x)
```



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

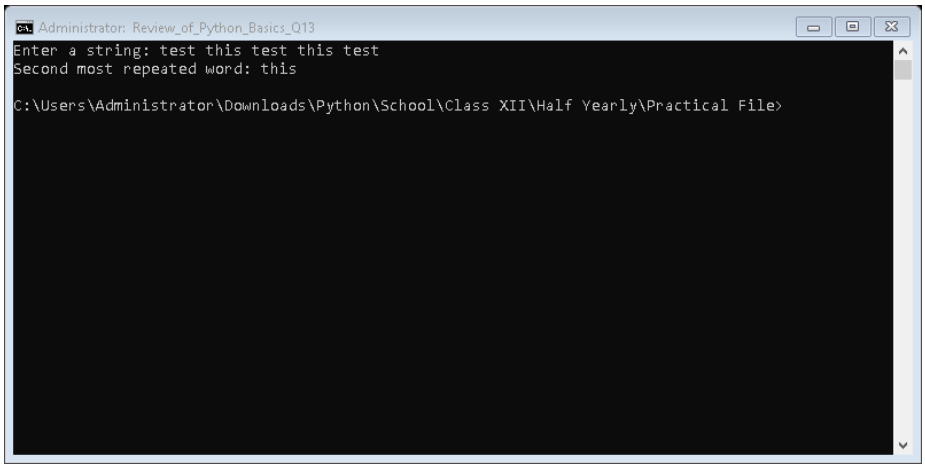
```
x=int(input('Enter a number: '))
s=0
while x>0:
    b=x%10
    s+=b
    x=x//10
print(s)
```



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

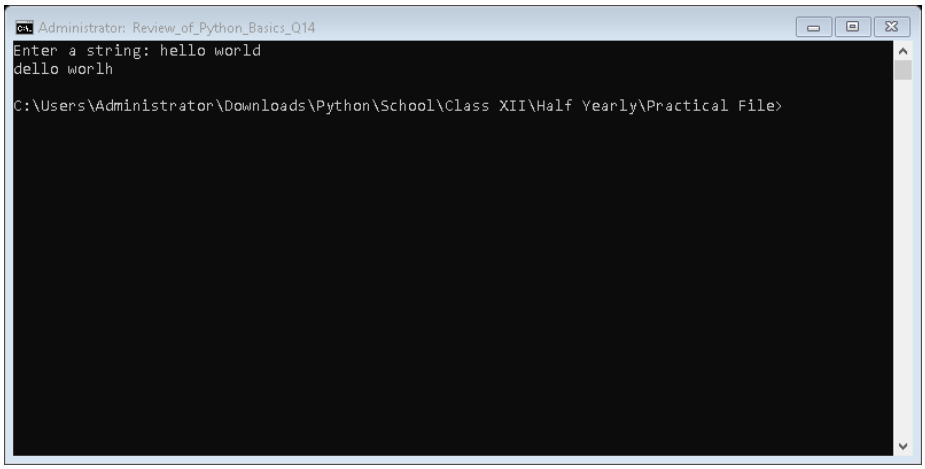
```
s = input("Enter a string: ")
words = s.split()
```

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



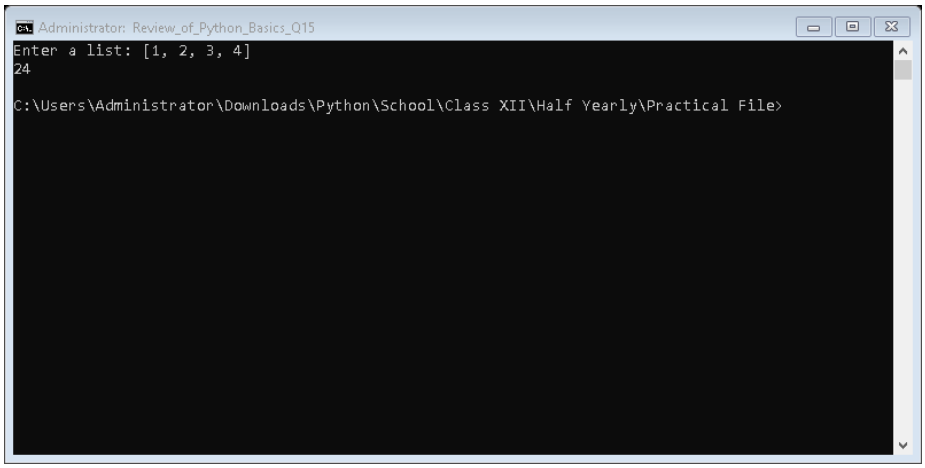
Q14) Write a Python program to change a given string to a new string where the first and last string have been exchanged.

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



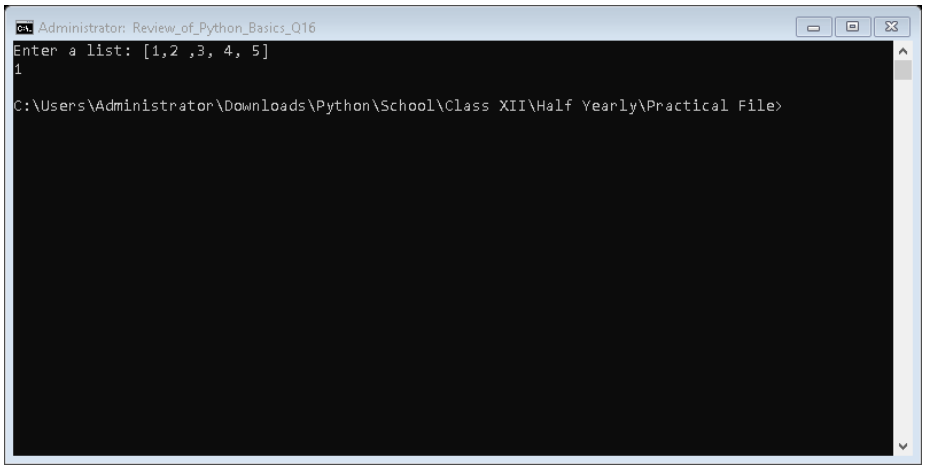
Q15) Write a Python program to multiply all the elements in a list.

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



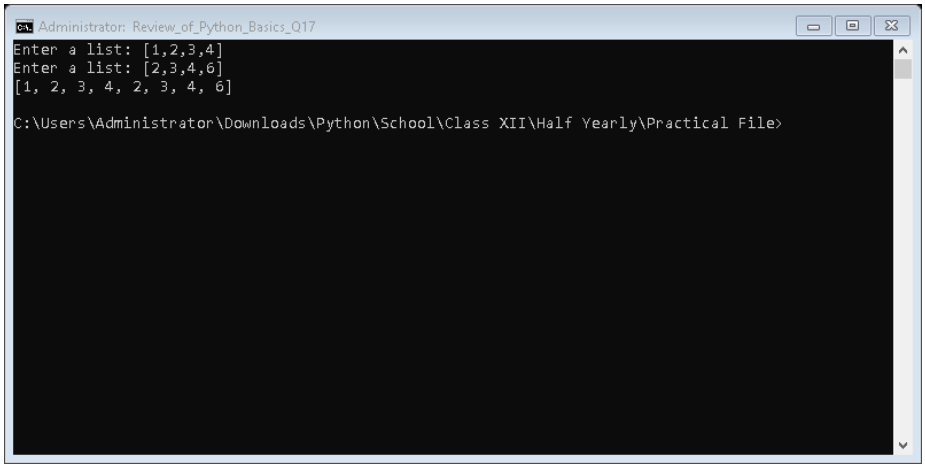
Q16) Write a Python program to get the smallest number from a list.

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



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

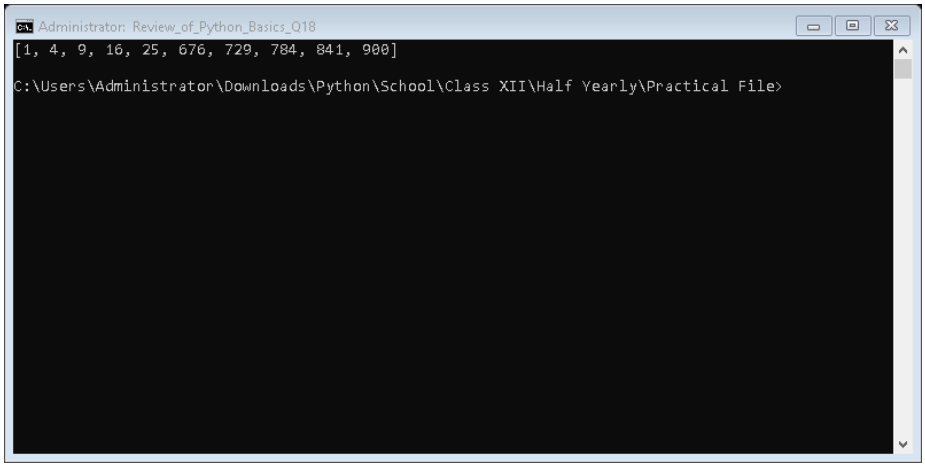
```
l1=eval(input('Enter a list: '))
l2=eval(input('Enter a list: '))
l1.extend(l2)
print(l1)
```



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).

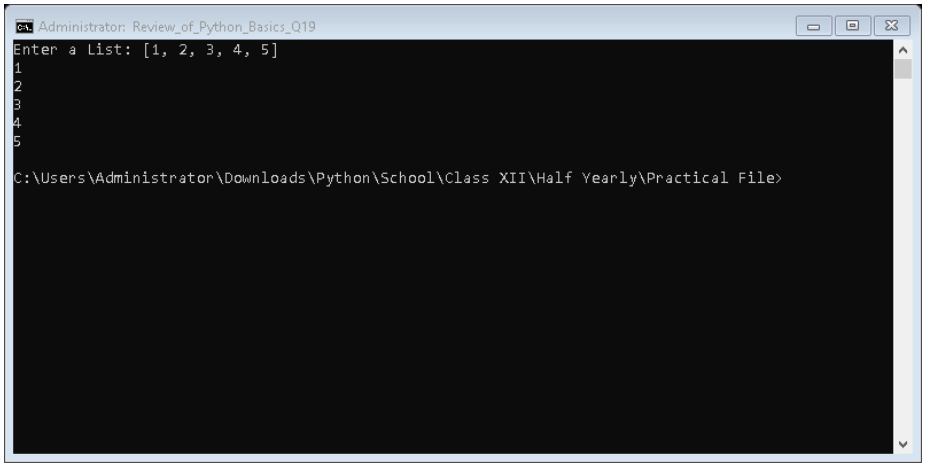
```
l=[]
for i in range(1,31):
    l.append(i**2)
```

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



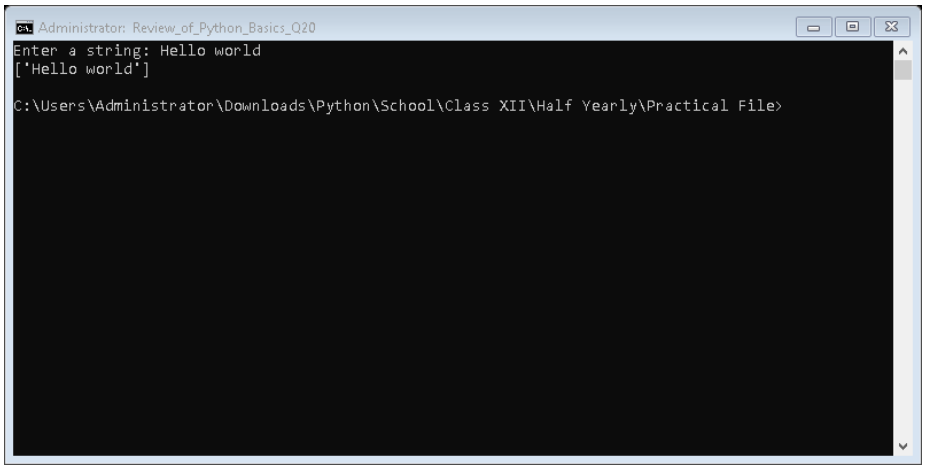
Q19) Write a Python program to get unique values from a list.

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



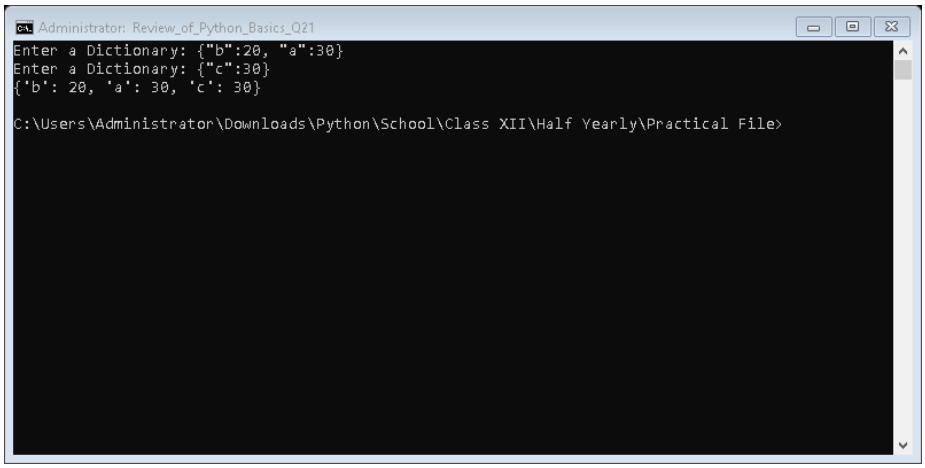
Q20) Write a python program to convert a string to a list.

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



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}

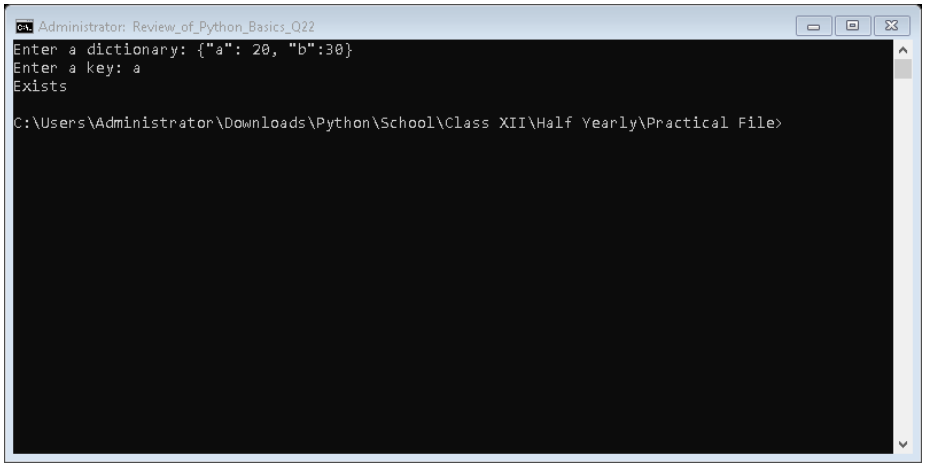
```
d1=eval(input("Enter a Dictionary: "))
d2=eval(input("Enter a Dictionary: "))
d1.update(d2)
print(d1)
```



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

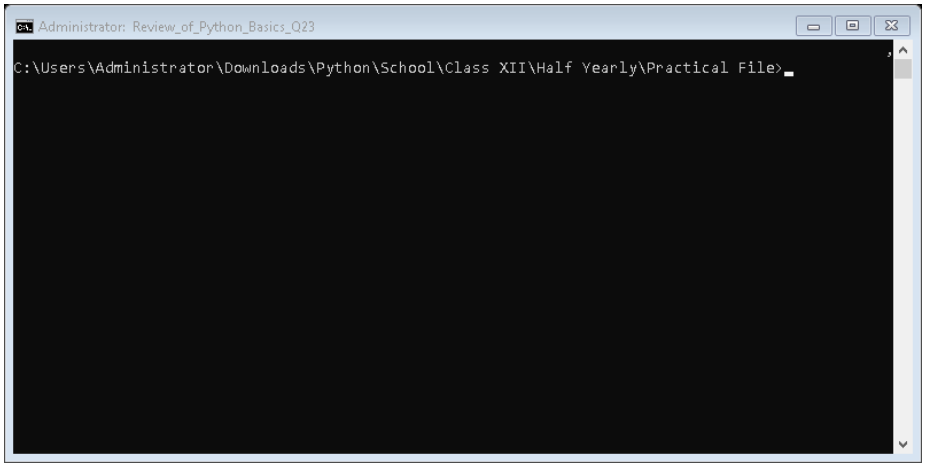
```
d=eval(input('Enter a dictionary: '))
x=input('Enter a key: ')
if x in d:
```

```
    print('Exists')
else:
    print("Key Does Not Exist")
```



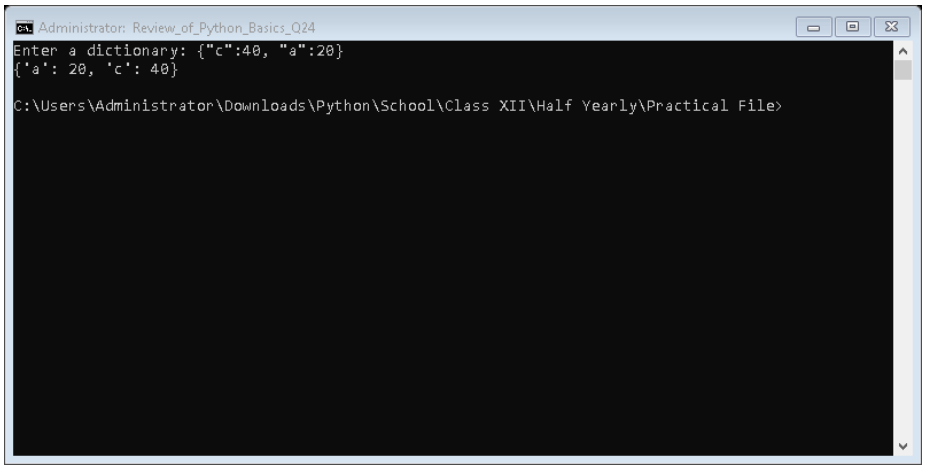
Q23) Write a Python script to print a dictionary where the keys are numbers between 1 and 15 (both included) and the values are square of keys.

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



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

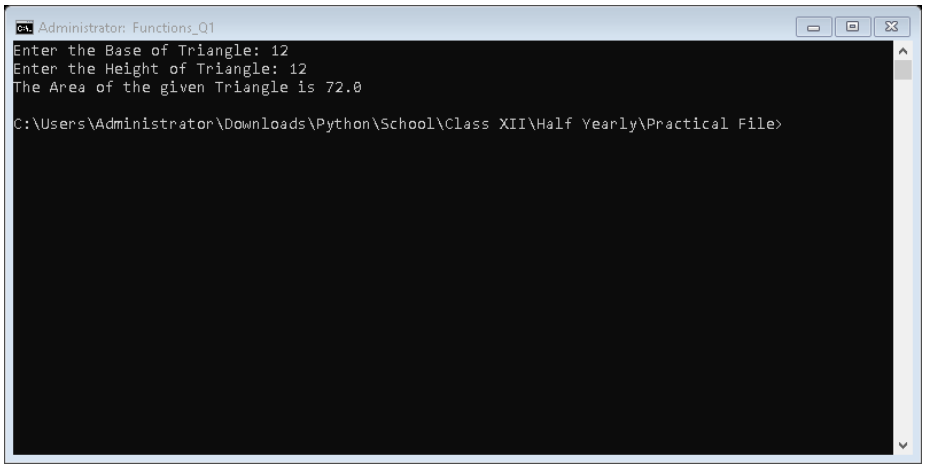
```python
d=eval(input('Enter a dictionary: '))
print(dict(sorted(d.items())))
```



2 Functions

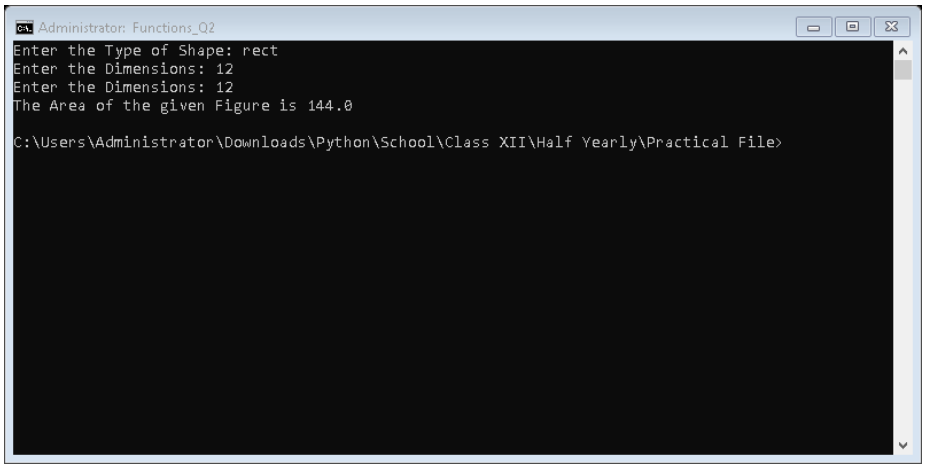
Q1) Write a function, calculate_area(), that takes base and height as input arguments and returns the area of a triangle as an output. The formula used is: Triangle Area = 1/2 * base * height

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



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.

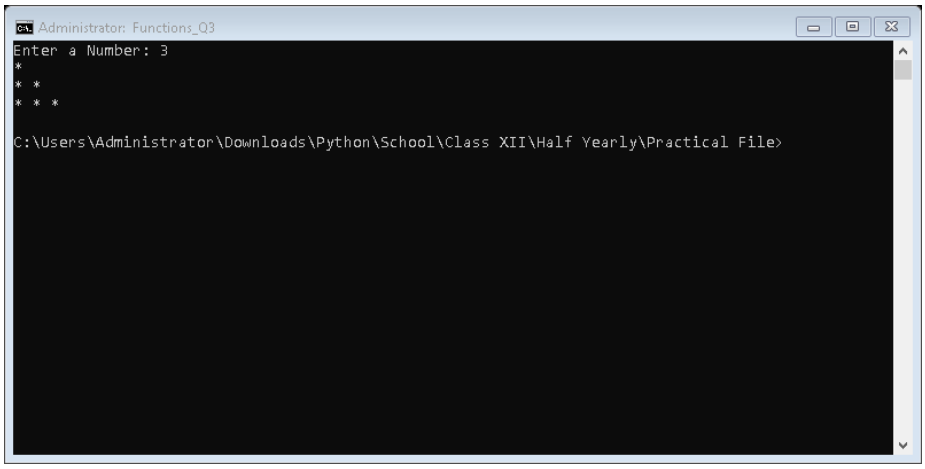
```
def calculate_area(Shape_Type,B,H):
    if Shape_Type=="Rectangle":
        A=B*H
    elif Shape_Type=="Triangle":
        A=(1/2)*B*H
    return A
shape=input("Enter the Type of Shape: ")
if shape.lower() in ("rect","rectangle","r","sqaure","s"):
    shape="Rectangle"
elif shape.lower() in ("triangle","t","tri"):
    shape="Triangle"
else:
    print("Shape NOT DEFINED")
    quit()
b=float(input("Enter the Dimensions: "))
h=float(input("Enter the Dimensions: "))
print("The Area of the given Figure is",calculate_area(shape,b,h))
```



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: *, **, ***, ****.**

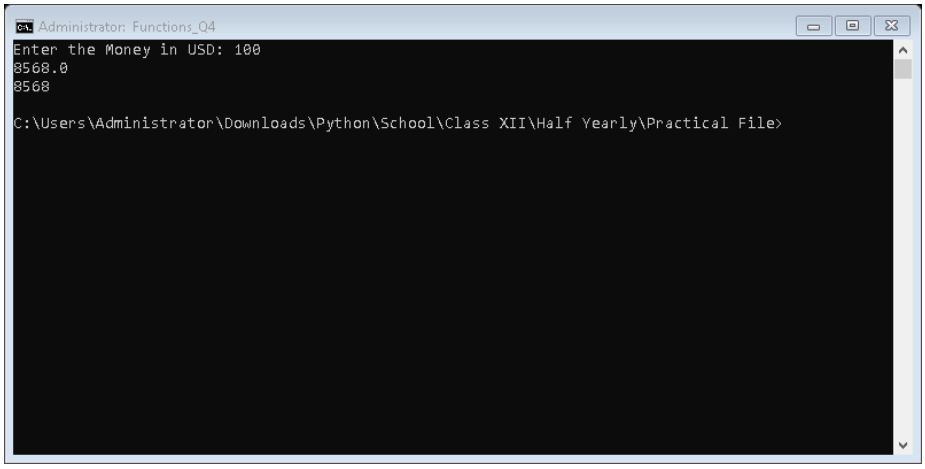
```
def pattern(N):
    for i in range(1,N+1):
        for j in range(i):
            print("*",end=" ")
        print()
n=int(input("Enter a Number: "))
```

```
pattern(n)
```



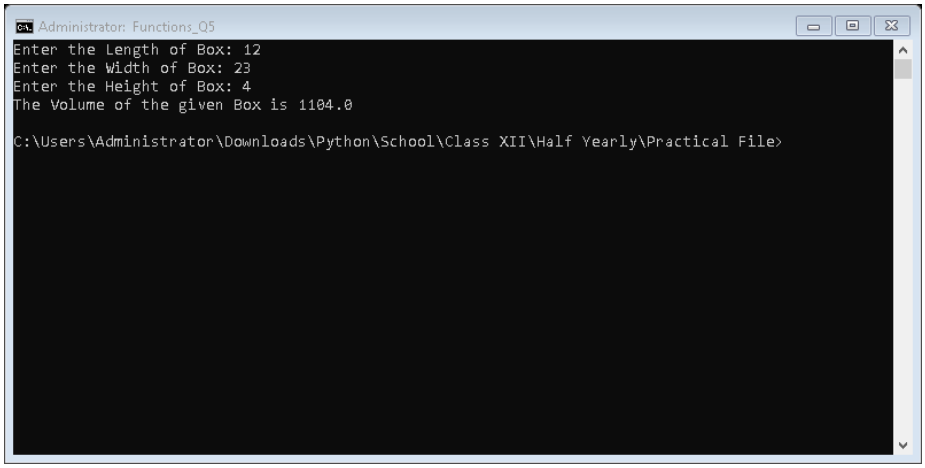
Q4) Write a function that takes amount dollars and dollar-to-rupee conversion price and then returns the amount converted to rupees. Create the function in both void and non-void forms.

```
def Void(M):
    print(M*85.68)
def Non_Void(M):
    x=M*85.68
    return int(x)
m=float(input("Enter the Money in USD: "))
Void(m)
print(Non_Void(m))
```



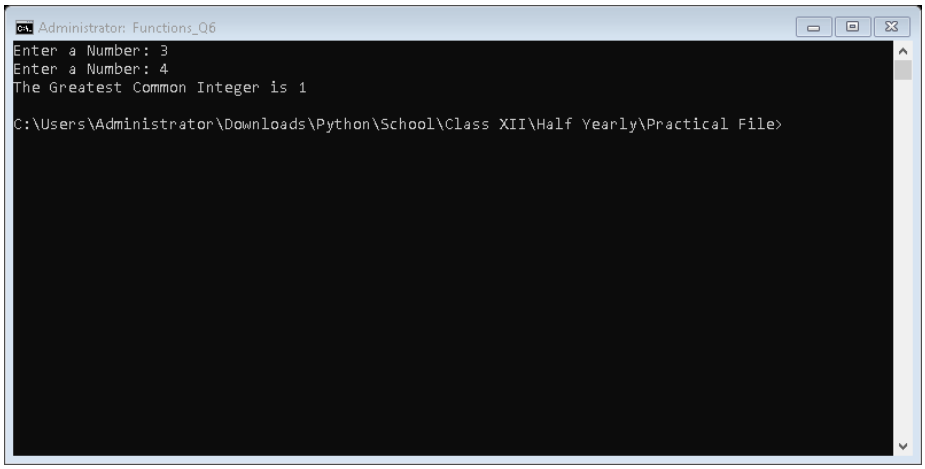
Q5) Write a function to calculate the value of a box with appropriate default values for its parameters. Your function should have the following input parameters: Length of box, Width of box, Height of box.

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



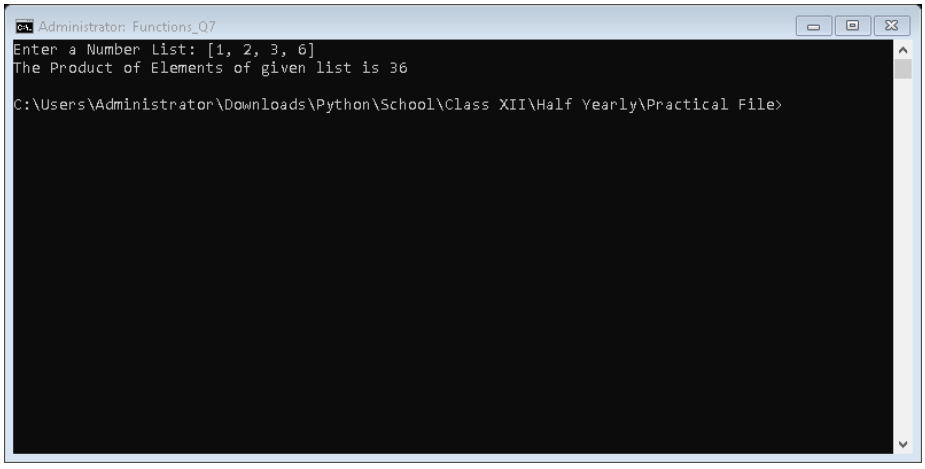
Q6) Write a program to find the greatest common divisor between two numbers.

```
def GCF(N1,N2):
    l1=[]
    l2=[]
    for i in range(1,N1+1):
        if N1%i==0:
            l1.append(i)
    for i in l1:
        if N2%i==0:
            l2.append(i)
    print("The Greatest Common Integer is",max(l2))
n1=int(input("Enter a Number: "))
n2=int(input("Enter a Number: "))
GCF(n1,n2)
```



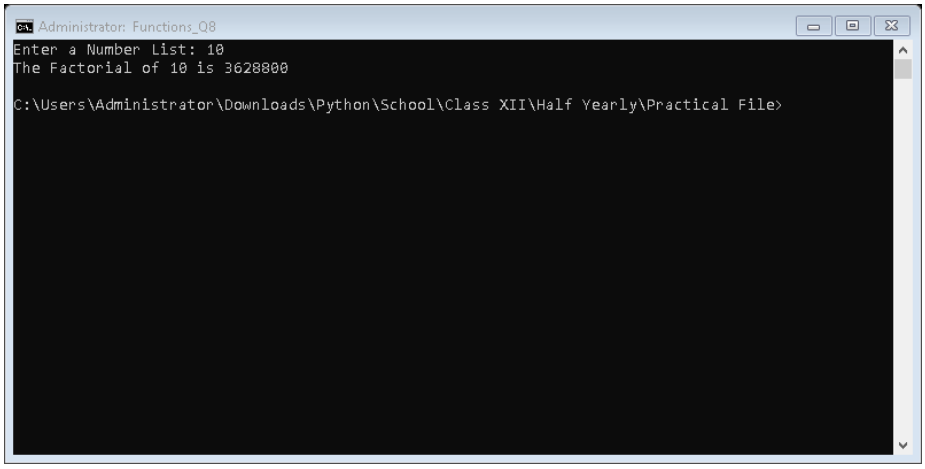
Q7) Write a Python function to multiply all the numbers in a list.

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



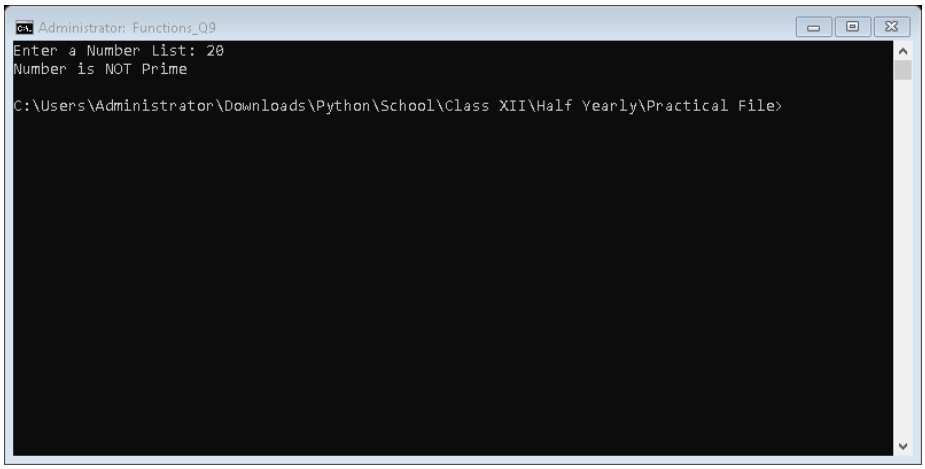
Q8) Write a Python function to calculate the factorial of a number (a non-negative integer). The function accepts the number whose factorial is to be calculated as the argument.

```python
def Multiply(l1):
    p=1
    for i in range(1,l1+1):
        p*=i
    return p
l=eval(input("Enter a Number List: "))
print("The Factorial of",l,"is",Multiply(l))
```



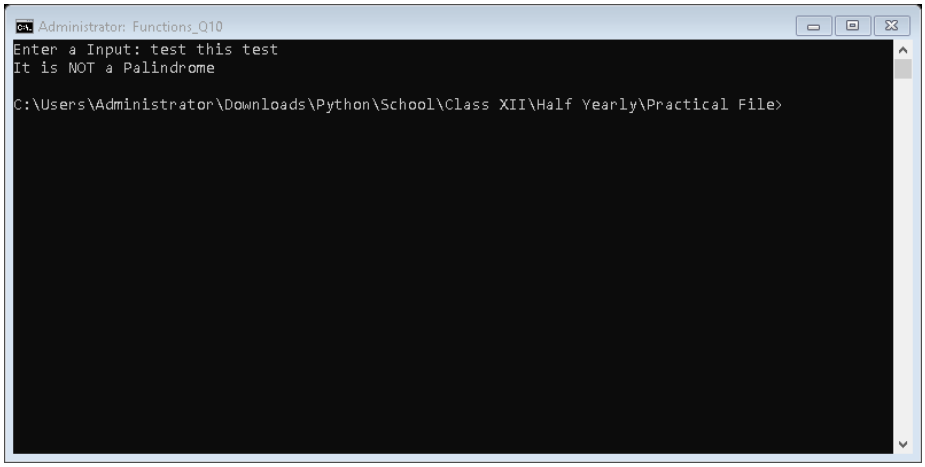
Q9) Write a Python function that takes a number as a parameter and checks whether the number is prime or not.

```python
def Prime(l1):
    a=0
    for i in range(2,l1):
        if l1%i==0:
            a=0
            break
        else:
            a=1
    if a==1:
        print("Number is Prime")
    else:
        print("Number is NOT Prime")
l=eval(input("Enter a Number List: "))
Prime(l)
```



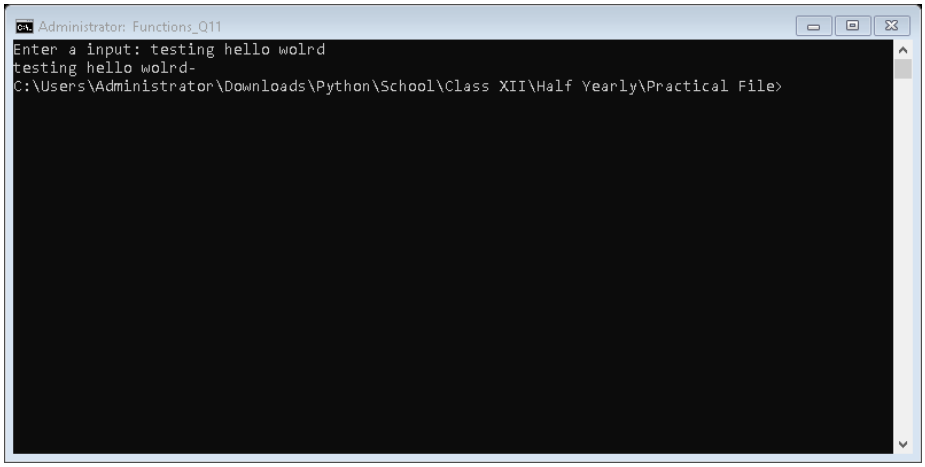
Q10) Write a Python function that checks whether a passed string is a palindrome or not.

```
def palindrome(x):
    i=''
    for j in x:
        if j.isalpha()==True:
            i+=j
    if i==i[:-1]:
        print("It is a Palindrome")
    else:
        print("It is NOT a Palindrome")
s=input("Enter a Input: ")
palindrome(s)
```



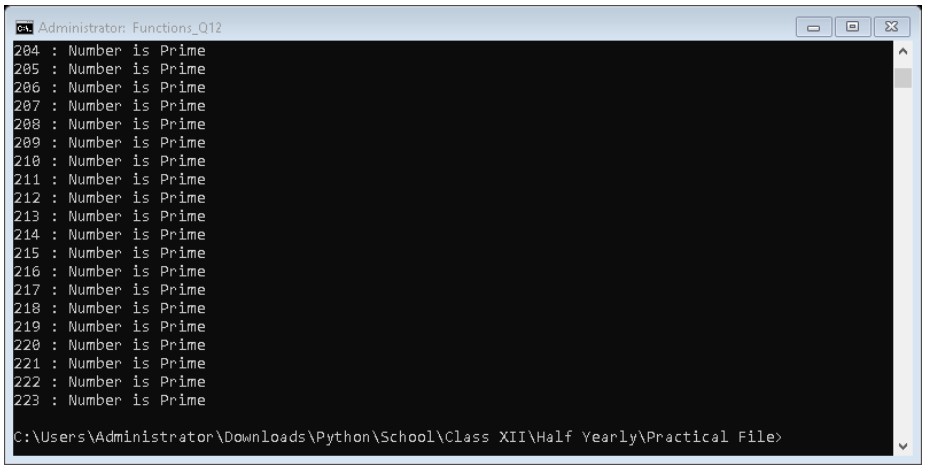
Q11) Write a Python program that accepts a hyphen-separated sequence of words as input and prints the words in a hyphen-separated after sorting them alphabetically.

```
def Sorter(s):
    l1=s.split("-")
    l1.sort()
    for i in l1:
        print(i,end="-")
S=input("Enter a input: ")
Sorter(S)
```



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.

```
def P_S(N):
    for i in range(2,N+1):
        a=0
        for j in range(2,i):
            if j%i==0:
                a=0
                break
            else:
                a=1
                break
        if a==1:
            print(i,": Number is Prime")
        else:
            print(i,": Number is NOT Prime")
l=eval(input("Enter a Number List: "))
P_S(l)
```

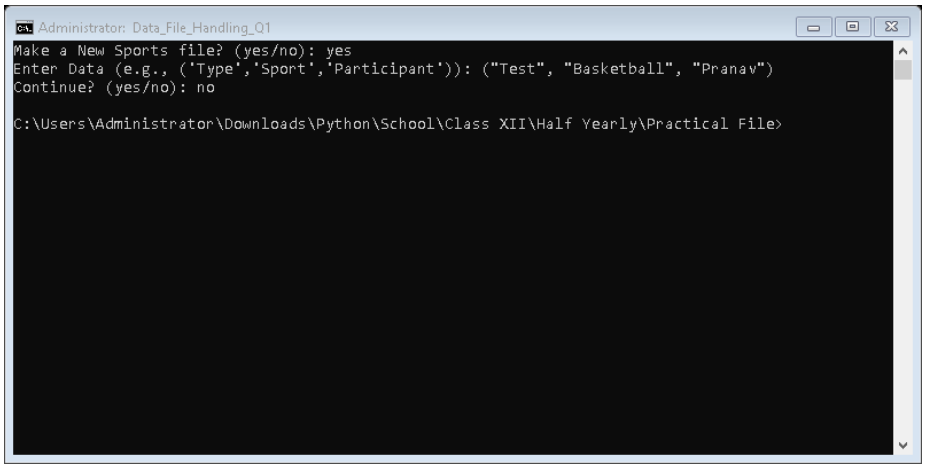


3 Data File Handling

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'.

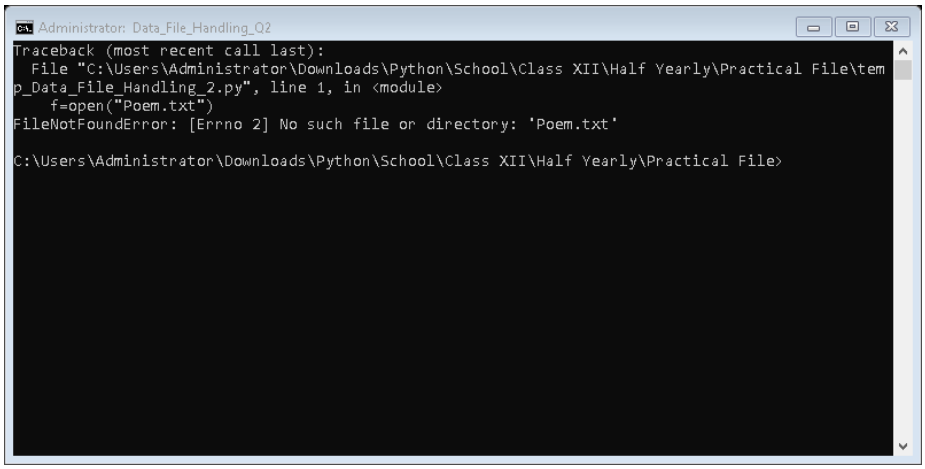
```
import pickle
ch = input("Make a New Sports file? (yes/no): ")
y = ("yes", "y")
n = ("no", "n")
if ch.lower() in y:
    with open("sports.dat", "wb") as f:
        while True:
            l = eval(input("Enter Data (e.g., ('Type','Sport','
    Participant')): "))
            pickle.dump(l, f)
            c = input("Continue? (yes/no): ")
            if c.lower() in n:
                break
def Athletics():
    with open("sports.dat", "rb") as f:
        with open("Athletics.dat", "wb") as f2:
            try:
                while True:
                    x = pickle.load(f)
                    if isinstance(x, (list, tuple)) and len(x)>0
    and x[0].lower() == "athletics":
                        pickle.dump(x, f2)
            except EOFError:
                print("File Loading Completed!")
    with open("Athletics.dat", "rb") as f:
        try:
```

```
            while True:
                print(pickle.load(f))
        except EOFError:
            print("End of File!!")
# Athletics()
```



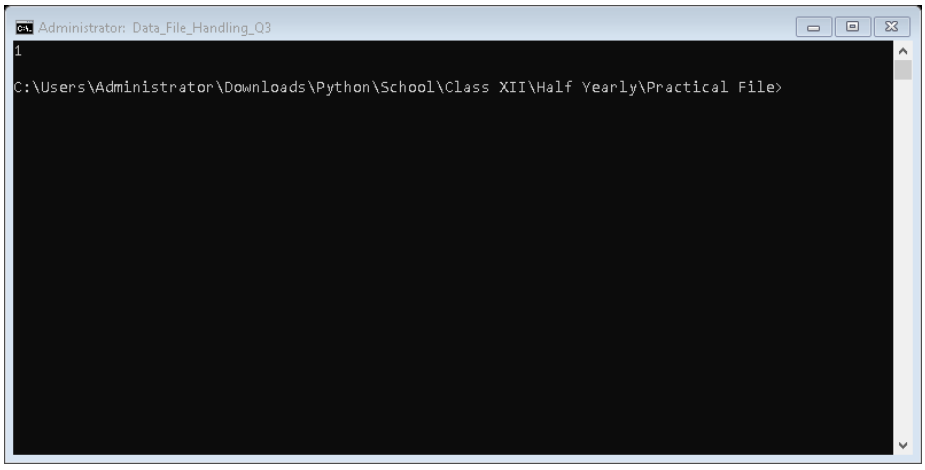
Q2) Write a program to count the words 'to' and 'the' present in text file 'Poem.txt'.

```
f=open("Poem.txt")
lines=f.read()
f.close()
count=0
line=lines.split()
x=line.index("to")
y=line.index("the")
for i in range(x,y+1):
    count+=1
print(count)
```



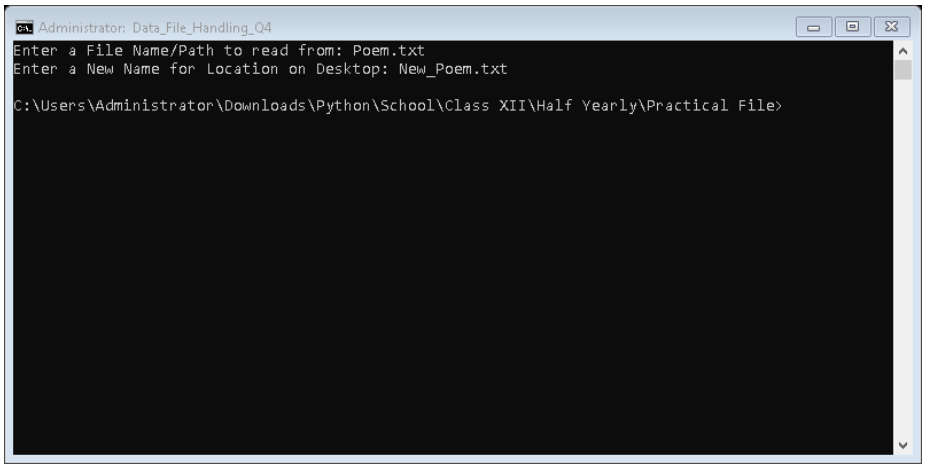
Q3) Write a program to count the number of uppercase alphabets present in text file 'Poem.txt'.

```
f=open("Poem.txt")
x=f.read()
f.close()
count=0
y="QWERTYUIOPASDFGHJKLZXCVBNM"
for i in x:
    if i in y:
        count+=1
print(count)
```



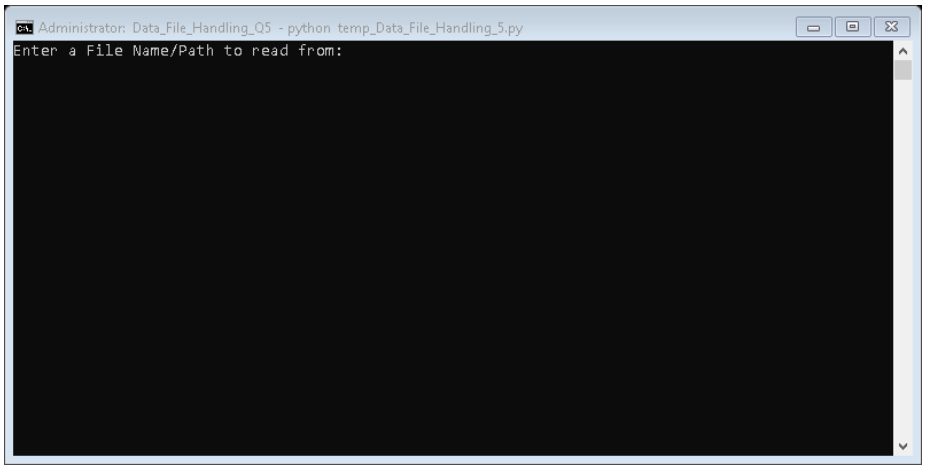
Q4) Write a program that copies one file to another and reads the filenames from the user.

```
b=input("Enter a File Name/Path to read from: ")
with open(b,"r") as f:
    MyS=f.read()
    f.close()
a=input("Enter a New Name for Location on Desktop: ")
with open(a,"w") as F:
    F.writelines(MyS)
    F.close()
```



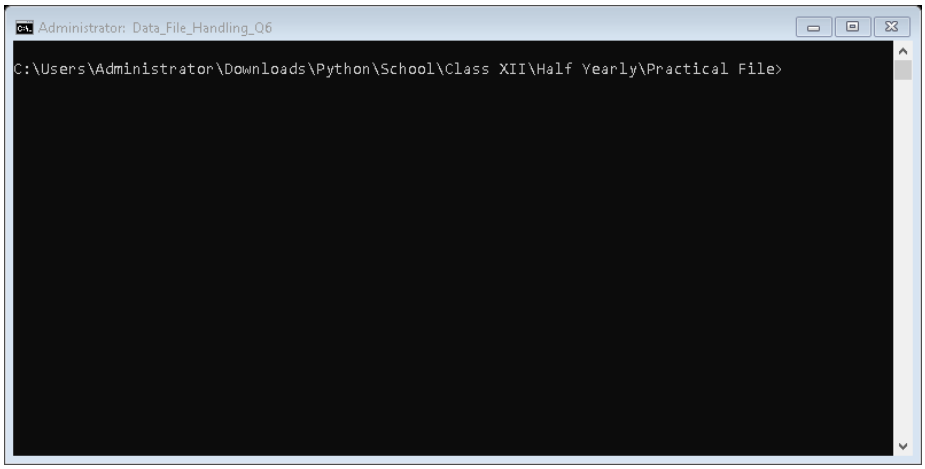
Q5) Write a program that appends the contents of one file to another and takes the filenames from the user.

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



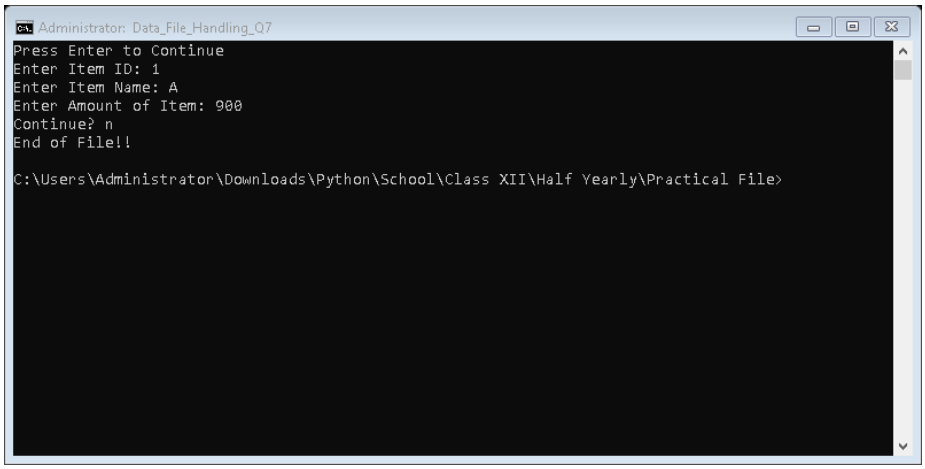
Q6) Write a program that reads characters from the keyboard one by one. All lowercase characters get stored in the file 'LOWER', all uppercase characters get stored in the file 'UPPER' and all the other characters get stored in the file 'OTHERS'.

```
f=open("Poem.txt")
a=f.read()
f.close()
l="asdfghjklqwertyuiopzxcvbnm"
u="QWERTYUIOPASDFGHJKLZXCVBNM"
for i in a:
    if i in l:
        with open("LOWER.txt", "a") as f:
            f.write(i)
            f.close()
    elif i in u:
        with open("UPPER.txt", "a") as f:
            f.write(i)
            f.close()
    elif i!=" ":
        with open("OTHER.txt", "a") as f:
            f.write(i)
            f.close()
```



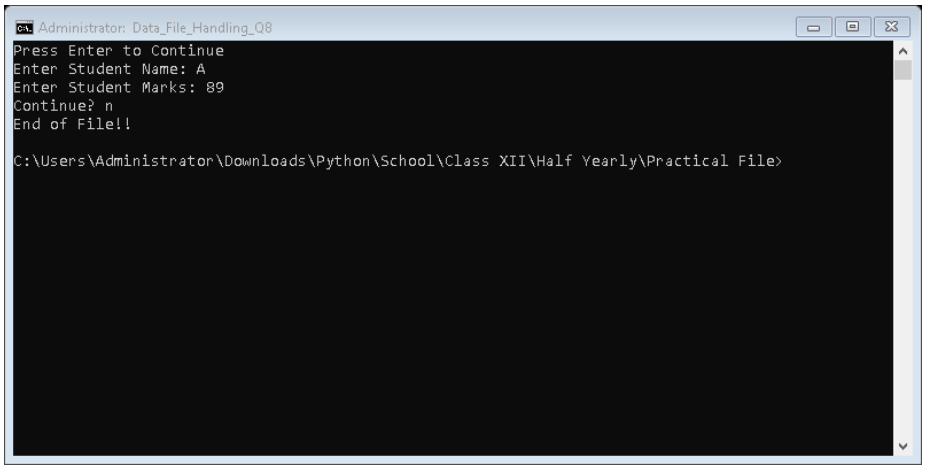
Q7) Consider binary file 'items.dat' containing records stored in the given format: {item_id: [item_name, amount]}. Write a function, copy_new(), that copies all records whose amount is greater than 1000 from 'items.dat' to 'new_items.dat'.

```
import pickle
with open("items.dat","wb") as f:
    d={}
    c=input("Press Enter to Continue")
    while c.lower()!='n':
        x=input("Enter Item ID: ")
        y=input("Enter Item Name: ")
        z=int(input("Enter Amount of Item: "))
        l=[]
        l.append(y)
        l.append(z)
        d[x]=l
        c=input("Continue? ")
    pickle.dump(d,f)
def copy_new():
    with open("items.dat","rb") as f:
        try:
            d=pickle.load(f)
        except:
            print("End of File!!")
    with open("new_items.dat","wb") as f:
        for i in d:
            if d[i][1]>=1000:
                pickle.dump(d[i],f)
    with open("new_items.dat","rb") as f:
        try:
            while True:
                print(pickle.load(f))
        except:
            print("End of File!!")
copy_new()
```



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.

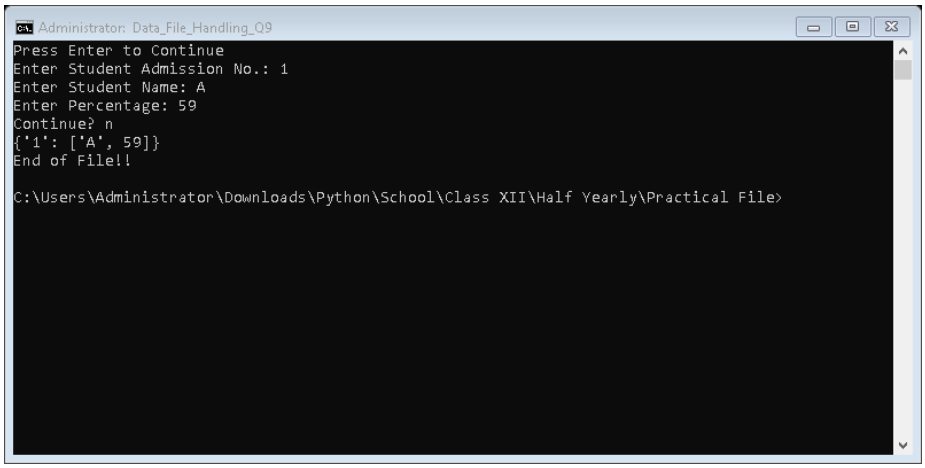
```
import pickle
with open("Students.dat","wb") as f:
    d={}
    c=input("Press Enter to Continue")
    while c.lower()!='n':
        x=input("Enter Student Name: ")
        l=int(input("Enter Student Marks: "))
        d[x]=l
        c=input("Continue? ")
    pickle.dump(d,f)
def copy_new():
    with open("Students.dat","rb") as f:
        try:
            d=pickle.load(f)
        except:
            print("End of File!!")
    with open("Remedial.dat","wb") as f:
        for i in d:
            if d[i]<=40:
                pickle.dump(i,f)
    with open("Remedial.dat","rb") as f:
        try:
            while True:
                print(pickle.load(f))
        except:
            print("End of File!!")
copy_new()
```



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.

```
import pickle
with open("items.dat","wb") as f:
    d={}
    c=input("Press Enter to Continue")
    while c.lower()!='n':
        x=input("Enter Student Admission No.: ")
        y=input("Enter Student Name: ")
        z=int(input("Enter Percentage: "))
        l=[]
        l.append(y)
        l.append(z)
        d[x]=l
        c=input("Continue? ")
    pickle.dump(d,f)
def copy_new():
    with open("items.dat","rb") as f:
        try:
            d=pickle.load(f)
        except:
            print("End of File!!")
    with open("new_items.dat","wb") as f:
        for i in d:
            if d[i][1]<=65:
                D={}
                D[i]=d[i]
```

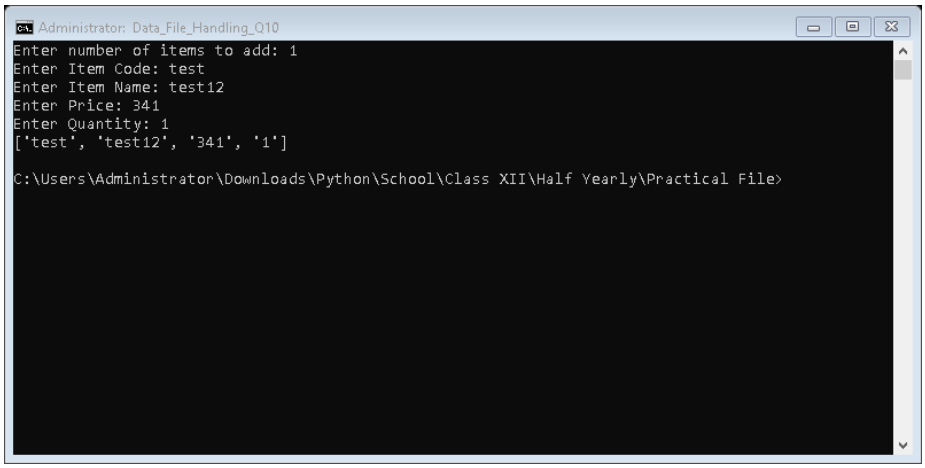
```
                pickle.dump(D,f)
    with open("new_items.dat","rb") as f:
        try:
            while True:
                print(pickle.load(f))
        except:
            print("End of File!!")
copy_new()
```



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.

```
import csv
try:
    with open("Groceries.csv", "r") as f_check:
        is_empty = f_check.readline() == ''
except FileNotFoundError:
    is_empty = True
with open("Groceries.csv", "a", newline='') as f:
    writer = csv.writer(f)
    if is_empty:
        writer.writerow(["Item Code", "Item Name", "Price", "
    Quantity"])
    num_rows = int(input("Enter number of items to add: "))
    rows = []
    for _ in range(num_rows):
        code = input("Enter Item Code: ")
        name = input("Enter Item Name: ")
        price = input("Enter Price: ")
        qty = input("Enter Quantity: ")
        rows.append([code, name, price, qty])
    writer.writerows(rows)
```

```
with open("Groceries.csv", "r", newline='') as f:
    reader = csv.reader(f)
    next(reader)
    for row in reader:
        print(row)
```



4 Stack Operations

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

```
stack = []

def push(item):
    stack.append(item)

def pop():
    if not stack:
        return "Underflow"
    return stack.pop()

# Example usage
push("Package1")
push("Package2")
print(stack)
popped_item = pop()
print(popped_item)
print(stack)
```

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

```
def sort_stack(stack):
```

```
    if stack:
        # Remove the top element
        temp = stack.pop()
        # Sort the remaining stack
        sort_stack(stack)
        # Insert the temp back in sorted position
        sorted_insert(stack, temp)

def sorted_insert(stack, element):
    if not stack or stack[-1] <= element:
        stack.append(element)
    else:
        temp = stack.pop()
        sorted_insert(stack, element)
        stack.append(temp)

# Example usage
stack = [3, 1, 2]
sort_stack(stack)
print(stack)
```

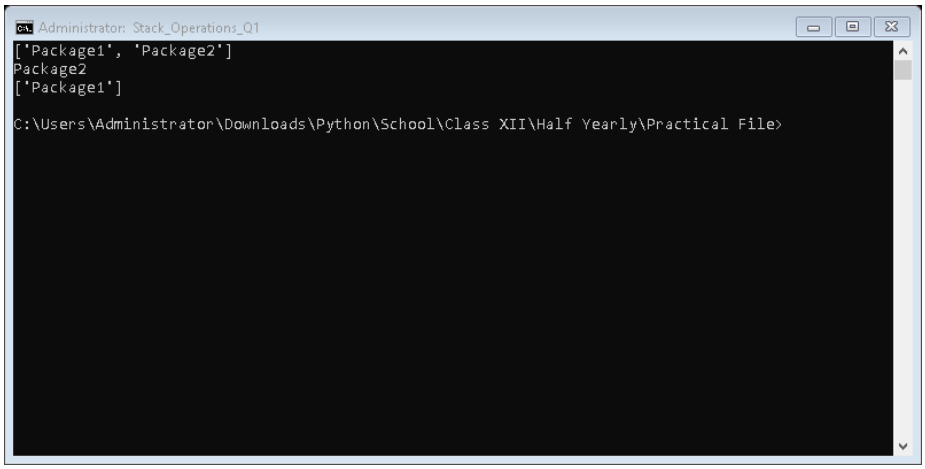
Q1) Write a program to implement pop and push functions on a stack containing package name as records.

```
stack = []

def push(item):
    stack.append(item)

def pop():
    if not stack:
        return "Underflow"
    return stack.pop()

# Example usage
push("Package1")
push("Package2")
print(stack)
popped_item = pop()
print(popped_item)
print(stack)
```

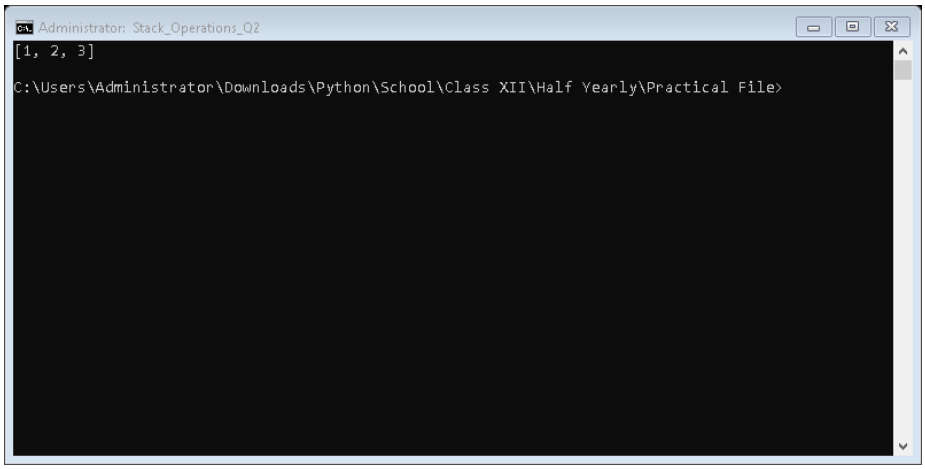


Q2) Write a program to sort a stack into ascending order without using another stack.

```
def sort_stack(stack):
    if stack:
        # Remove the top element
        temp = stack.pop()
        # Sort the remaining stack
        sort_stack(stack)
        # Insert the temp back in sorted position
        sorted_insert(stack, temp)

def sorted_insert(stack, element):
    if not stack or stack[-1] <= element:
        stack.append(element)
    else:
        temp = stack.pop()
        sorted_insert(stack, element)
        stack.append(temp)

# Example usage
stack = [3, 1, 2]
sort_stack(stack)
print(stack)
```



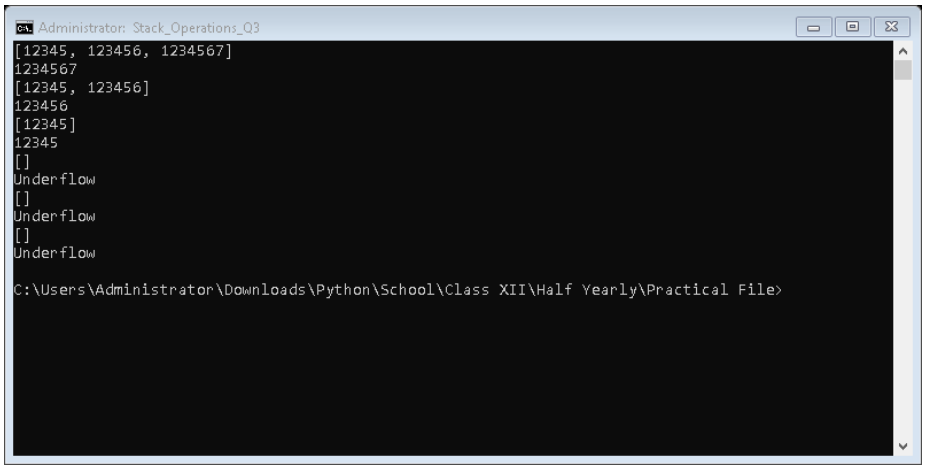
Q3) Write a program to implement pop and push operations on a stack. The push operation should add numbers from a list which have 5 digits or more. The pop operation should print underflow if stack is empty.

```
stack = []

def push_from_list(lst):
    for num in lst:
        if len(str(abs(num))) >= 5:  # Check if the number has 5 or more digits
            stack.append(num)

def pop():
    if not stack:
        return "Underflow"
    return stack.pop()

# Example usage
numbers_list = [12345, 1234, 123456, 123, 1234567]
push_from_list(numbers_list)
print(stack)
print(pop())
print(stack)
print(pop())
print(stack)
print(pop())
print(stack)
print(pop())
print(stack)
print(pop())
print(stack)
print(pop())  # Should print "Underflow"
```



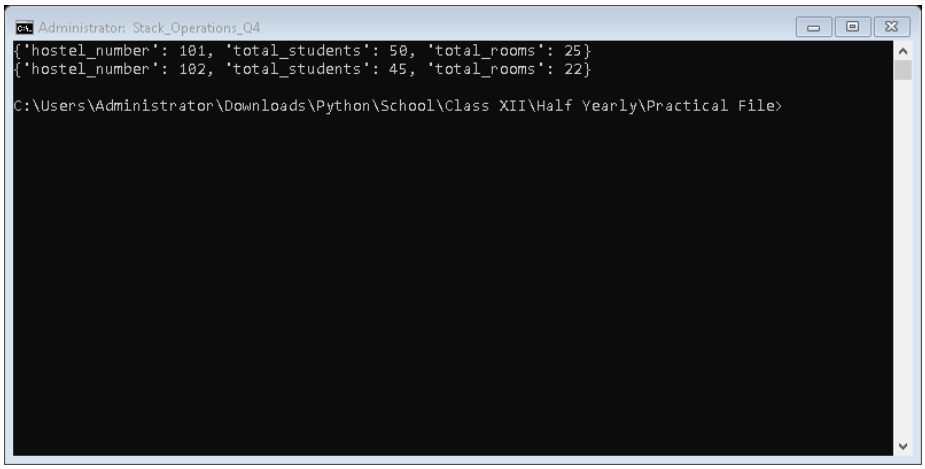
Q4) Write a program to implement push and display operations on a stack with hostel number, total students and total rooms as record.

```
stack = []

def push_hostel_record(hostel_number, total_students, total_rooms):
    record = {
        'hostel_number': hostel_number,
        'total_students': total_students,
        'total_rooms': total_rooms
    }
    stack.append(record)

def display_stack():
    for record in stack:
        print(record)

# Example usage
push_hostel_record(101, 50, 25)
push_hostel_record(102, 45, 22)
display_stack()
```



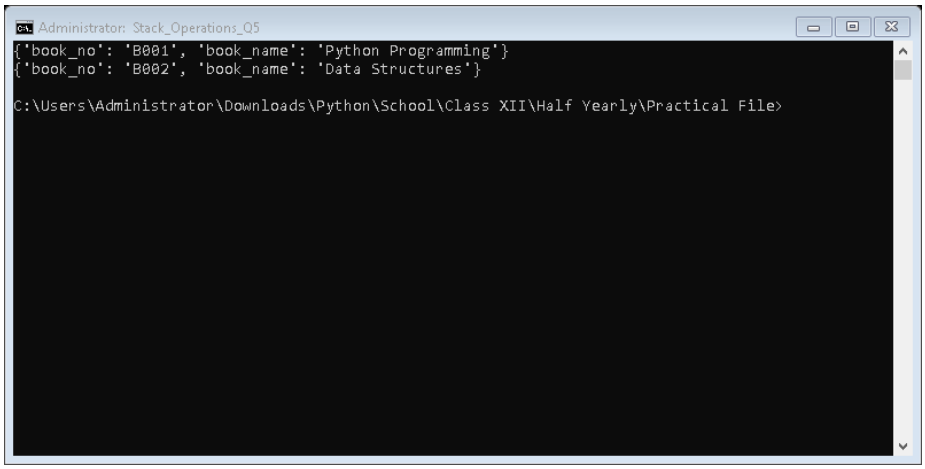
Q5) Write a program to implement push and display operations on a stack which has book number and name as records.

```
stack = []

def push_book_record(book_no, book_name):
    record = {
        'book_no': book_no,
        'book_name': book_name
    }
    stack.append(record)

def display_stack():
    for record in stack:
        print(record)

# Example usage
push_book_record("B001", "Python Programming")
push_book_record("B002", "Data Structures")
display_stack()
```



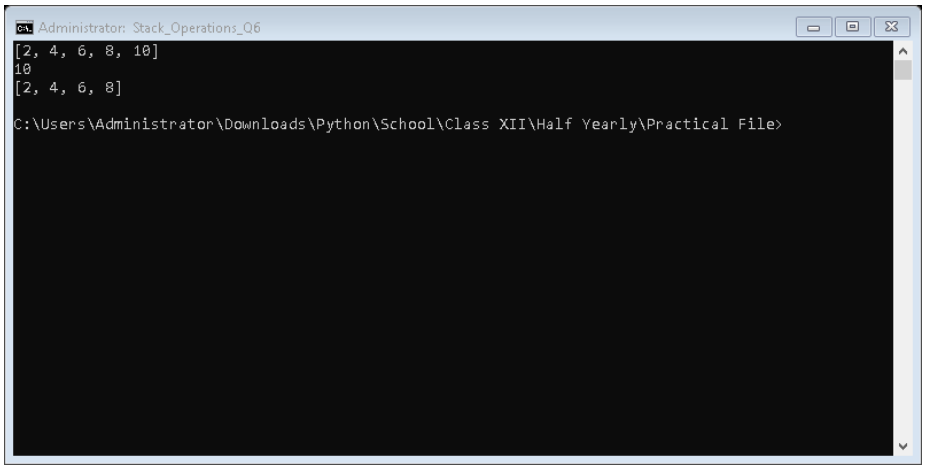
Q6) Write a program to push elements in a stack from a list that are even. Also implement pop function.

```
stack = []

def push_evens_from_list(lst):
    for num in lst:
        if num % 2 == 0:
            stack.append(num)

def pop():
    if not stack:
        return "Underflow"
    return stack.pop()

# Example usage
numbers_list = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
push_evens_from_list(numbers_list)
print(stack)
print(pop())
print(stack)
```



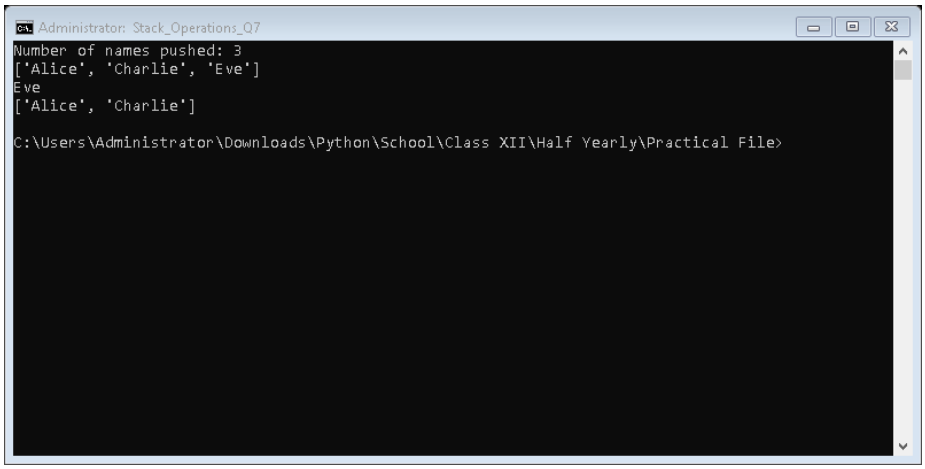
Q7) Write a program to push student names from a dictionary into a stack who have won more than 3 medals. Also it prints the the no of names pushed and overflow if the no of items exceed 15.

```
stack = []
max_size = 15

def push_students(dictionary):
    count = 0
    for name, medals in dictionary.items():
        if medals > 3:
            if len(stack) >= max_size:
                print("Overflow")
                return
            stack.append(name)
            count += 1
    print(f"Number of names pushed: {count}")

def pop():
    if not stack:
        return "Underflow"
    return stack.pop()

# Example usage
students = {
    "Alice": 4,
    "Bob": 2,
    "Charlie": 5,
    "David": 3,
    "Eve": 6
}
push_students(students)
print(stack)
print(pop())
print(stack)
```



5 Relational Databases

Q17) Write SQL queries to perform the following based on the table PRODUCT having fields as (prod_id, prod_name, quantity, unit_rate, price, city)

(i) Display those records from table PRODUCT where prod_id is more than 100.

```
SELECT * FROM PRODUCT WHERE prod_id > 100;
```

(ii) List records from table PRODUCT where prod_name is 'Almirah'.

```
SELECT * FROM PRODUCT WHERE prod_name = 'Almirah';
```

(iii) List all those records whose price is between 200 and 500.

```
SELECT * FROM PRODUCT WHERE price BETWEEN 200 AND 500;
```

(iv) Display the product names whose price is less than the average of price.

```
SELECT prod_name FROM PRODUCT WHERE price < (SELECT AVG(price) FROM
    PRODUCT);
```

(v) Show the total number of records in the table PRODUCT.

```
SELECT COUNT(*) AS total_records FROM PRODUCT;
```

Q20) Consider the following EMP and DEPT tables:

Table: EMP

EmpNo	EmpName	City	Designation	DOJ	Sal	Comm	DeptID
8369	SMITH	Mumbai	CLERK	1990-12-18	800.00	NULL	20
8499	ANYA	Varanasi	SALESMAN	1991-02-20	1600.00	300.00	30
8521	SETH	Jaipur	SALESMAN	1991-02-22	1250.00	500.00	30
8566	MAHADEVAN	Delhi	MANAGER	1991-04-02	2985.00	NULL	20

Table: DEPT

DeptID	DeptName	MgrID	Location
10	SALES	8566	Mumbai
20	PERSONNEL	9698	Delhi
30	ACCOUNTS	4578	Delhi
40	RESEARCH	8839	Bengaluru

Write the SQL command to get the following:

(a) Show the minimum, maximum and average salary of managers.

```
SELECT MIN(Sal) AS MinSalary, MAX(Sal) AS MaxSalary, AVG(Sal) AS
    AvgSalary
FROM EMP
WHERE Designation = 'MANAGER';
```

(b) Count the number of clerks in the organization.

```
SELECT COUNT(*) AS ClerkCount
FROM EMP
WHERE Designation = 'CLERK';
```

(c) Display the designation-wise list of employees with name, salary and date of joining.

```
SELECT Designation, EmpName, Sal, DOJ
FROM EMP
ORDER BY Designation;
```

(d) Count the number of employees who are not getting commission.

```
SELECT COUNT(*) AS No_comm
FROM EMP
WHERE comm is NULL;
```

(e) Show the average salary for all departments having salary ¿ 2000.

```
SELECT D.DeptName, AVG(E.Sal) AS AvgSalary
FROM EMP E, DEPT D
WHERE E.DeptID = D.DeptID AND E.Sal > 2000
GROUP BY D.DeptName;
```

(f) List the count of employees grouped by DeptID.

```
SELECT DeptID, COUNT(*) AS EmpCount
FROM EMP
GROUP BY DeptID;
```

(g) Display the maximum salary of employees in each department.

```
SELECT D.DeptName, MAX(E.Sal) AS MaxSalary
FROM EMP E, DEPT D
WHERE E.DeptID = D.DeptID
GROUP BY D.DeptName;
```

(h) Display the name of employees along with their designation and department name.

```
SELECT EMP.EMPNAME, EMP.DESIGNATION, DEPT.DEPTNAME
FROM EMP, DEPT
WHERE EMP.DEPTID = DEPT.DEPTID;
```

(i) Count the number of employees working in ACCOUNTS department.

```sql
SELECT COUNT(*) AS NUM_EMP
FROM EMP, DEPT
WHERE EMP.DEPTID = DEPT.DEPTID
AND DEPTNAME = 'ACCOUNTS';
```

Q21) Write SQL commands for (i) to (vi) and write output for (vii) on the basis of PRODUCTS relation given below:

Table: PRODUCTS

PCODE	PNAME	COMPANY	PRICE	STOCK	MANUFACTURE	WA
P001	TV	BPL	10000	200	2018-01-12	3
P002	TV	SONY	12000	150	2017-03-23	4
P003	PC	LENOVO	39000	100	2018-04-09	2
P004	PC	COMPAQ	38000	120	2019-06-20	2
P005	HANDYCAM	SONY	18000	250	2017-03-23	3

(i) To show details of all PCs with stock more than 110.

```sql
SELECT * FROM PRODUCTS WHERE PNAME = 'PC' AND STOCK > 110;
```

(ii) To list the company which gives warranty of more than 2 years.

```sql
SELECT DISTINCT COMPANY FROM PRODUCTS WHERE WARRANTY > 2;
```

(iii) To find stock value of the BPL company where stock value is the sum of the products of price and stock.

```sql
SELECT COMPANY, SUM(PRICE * STOCK) AS StockValue FROM PRODUCTS
    WHERE COMPANY = 'BPL';
```

(iv) To show number of products from each company.

```sql
SELECT COMPANY, COUNT(*) AS ProductCount FROM PRODUCTS GROUP BY
    COMPANY;
```

(v) To count the number of PRODUCTS which shall be out of warranty on 20-NOV-2020.

```sql
SELECT COUNT(*) FROM PRODUCTS WHERE DATE_ADD(MANUFACTURE, INTERVAL
    WARRANTY YEAR) <= '2020-11-20';
```

(vi) To show the PRODUCT name of the products which are within warranty as on date.

```sql
SELECT PNAME FROM PRODUCTS WHERE DATE_ADD(MANUFACTURE, INTERVAL
    WARRANTY YEAR) >= CURDATE();
```

There is no output produced because the warranty of all products has expired.

(vii) Give the output of the following statements: (a) SELECT COUNT(DISTINCT COMPANY) FROM PRODUCTS; (b) SELECT MAX(PRICE) FROM PRODUCTS WHERE WARRANTY ¡= 3;

Q22) Write SQL commands for (i) to (vi) on the basis of relations given below:

Table: BOOKS

Book_ID	Book_name	Author_name	Publishers	Price	Typ
K0001	Let us C	Y. Kanetkar	EPB	450	Prog
P0001	Computer Networks	B. Agarwal	FIRST PUBL	755	Com
M0001	Mastering C++	K.R. Venugopal	EPB	165	Prog
N0002	VC++ advance	P. Purohit	TDH	250	Prog
K0002	Programming with Python	Sanjeev	FIRST PUBL	350	Prog
L02	Computer Science with Python	Sumita Arora	Dhanpat rai	655	Prog
L04	Computer Science with Python	Preeti Arora	Sultan chand	550	Prog
L05	Concise Mathematics	R.K.Bansal	Selina	600	Mat

Table: ISSUED

Book_ID	Qty_Issued
L02	13
L04	5
L05	21

(i) To show the books of "FIRST PUBL" Publishers written by P.Purohit.

```
SELECT BOOK_NAME FROM BOOKS WHERE PUBLISHERS = "FIRST PUBL" AND
    AUTHOR_NAME = "P.PUROHIT";
```

There is no output produced because there are no books published by "FIRST PUBL" and written by "P.Purohit" in the table BOOKS.

(ii) To display cost of all the books published for FIRST PUBL.

```
SELECT SUM(Price * qty) AS Cost FROM BOOKS WHERE Publishers = '
    FIRST PUBL' GROUP BY publishers;
```

(iii) Depreciate the price of all books of EPB publishers by 5%.

```
UPDATE BOOKS SET Price = Price - (Price * 5/100) WHERE Publishers =
     'EPB';
```

(iv) To display the bookname and price of the books, more than 3 copies of which have been issued.

```
SELECT b.Book_name, b.Price FROM BOOKS b, ISSUED i WHERE b.Book_ID
    = i.Book_ID and i.Qty_Issued > 3;
```

(v) To show total cost of books of each type.

```
SELECT Type, SUM(Price * qty) AS Total_Cost FROM BOOKS GROUP BY
    Type;
```

(vi) To show the details of the costliest book.

```
SELECT * FROM BOOKS WHERE Price = (SELECT MAX(Price) FROM BOOKS);
```

Q27) Write SQL Commands for (i) to (v) and write the outputs for (vi) to (viii) on the basis of the following table:

Table: FURNITURE

NO	ITEM	TYPE	DATEOFSTOCK	PRICE	DISCOUNT
1	White Lotus	DoubleBed	2002-02-23	3000	25
2	Pinkfeathers	BabyCot	2002-01-29	7000	20
3	Dolphin	BabyCot	2002-02-19	9500	20
4	Decent	OfficeTable	2002-02-01	25000	30
5	Comfortzone	DoubleBed	2002-02-12	25000	30
6	Donald	BabyCot	2002-02-24	6500	15

(i) To list the details of furniture whose price is more than 10000.

```
SELECT * FROM FURNITURE WHERE PRICE > 10000;
```

(ii) To list the Item name and Price of furniture whose discount is between 10 and 20.

```
SELECT ITEM, PRICE FROM FURNITURE WHERE DISCOUNT BETWEEN 10 AND 20;
```

(iii) To delete the record of all items where discount is 30.

```
DELETE FROM FURNITURE WHERE DISCOUNT = 30;
```

(iv) To display the price of 'Babycot'.

```
SELECT PRICE FROM FURNITURE WHERE TYPE = 'BabyCot';
```

(v) To list item name, type and price of all items whose names start with 'D'.

```
SELECT ITEM, TYPE, PRICE FROM FURNITURE WHERE ITEM LIKE 'D%';
```

(vi) SELECT DISTINCT Type FROM Furniture;
(vii) SELECT MAX(Price) FROM Furniture WHERE DateofStock ¿ '2002-02-15' ;
(viii) SELECT COUNT(*) FROM Furniture WHERE Discount ¡ 25 ;

Q28) 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
2	RAJ	450	CHEMISTRY	68	1
3	DEEP	300	MATHS	62	2
4	DIVYA	350	CHEMISTRY	63	1
5	GAURAV	500	PHYSICS	70	1
6	MANAV	400	CHEMISTRY	55	2
7	VARUN	250	MATHS	64	1
8	LIZA	450	COMPUTER	68	1
9	PUJA	500	PHYSICS	62	1
10	NISHA	300	COMPUTER	57	2

(i) List the names of those students who have obtained rank 1 sorted by NAME.

```sql
SELECT NAME FROM GRADUATE WHERE `RANK` = 1 ORDER BY NAME;
```

(ii) Display a list of all those names whose AVERAGE is greater than 65.

```sql
SELECT NAME FROM GRADUATE WHERE AVERAGE > 65;
```

(iii) Display the names of those students who have opted COMPUTER as a SUBJECT with an AVERAGE of more than 60.

```sql
SELECT NAME FROM GRADUATE WHERE SUBJECT = 'COMPUTER' AND AVERAGE >
    60;
```

(iv) List the names of all the students in alphabetical order.

```sql
SELECT NAME FROM GRADUATE ORDER BY NAME;
```

(v) SELECT * FROM GRADUATE WHERE NAME LIKE "%I%";

(vi) SELECT DISTINCT RANK FROM GRADUATE; Since 'RANK' is a reserved keyword in SQL, we encounter an error while running this query. To avoid such errors, we can enclose the column name 'RANK' in backticks to treat it as a literal identifier. The corrected query is :

```sql
SELECT DISTINCT `RANK` FROM GRADUATE;
```

Q29(c)) For the given table, do as directed:

Table: STUDENT

ColumnName	Data type	size	Constraint
ROLLNO	Integer	4	Primary Key
SNAME	Varchar	25	Not Null
GENDER	Char	1	Not Null
DOB	Date		Not Null
FEES	Integer	4	Not Null
HOBBY	Varchar	15	Null

(i) Write SQL query to create the table.

```sql
CREATE TABLE STUDENT(
    ROLLNO INT(4) PRIMARY KEY,
    SNAME VARCHAR(25) NOT NULL,
    GENDER CHAR(1) NOT NULL,
    DOB DATE NOT NULL,
    FEES INT(4) NOT NULL,
    HOBBY VARCHAR(15)
);
```

(ii) Write SQL query to increase the size of SNAME to hold 30 characters.

```sql
ALTER TABLE STUDENT MODIFY SNAME VARCHAR(30);
```

(iii) Write SQL query to remove the column HOBBY.

```sql
ALTER TABLE STUDENT DROP HOBBY;
```

(iv) Write SQL query to insert a row in the table with any values of your choice that can be accommodated there.

```sql
INSERT INTO STUDENT(ROLLNO, SNAME, GENDER, DOB, FEES, HOBBY) VALUES
    (1, 'ANANYA', 'F', '2000-01-01', 5000, 'COOKING');
```

Q30) Write SQL queries based on the following tables:

Table: PRODUCT

P_ID	ProductName	Manufacturer	Price	Discount
TP01	Talcum Powder	LAK	40	NULL
FW05	Face Wash	ABC	45	5
BS01	Bath Soap	ABC	55	NULL
5H06	Shampoo	XYZ	120	10
FW12	Face Wash	XYZ	95	NULL

Table: CLIENT

C_ID	ClientName	City	P_ID
01	Cosmetic Shop	Delhi	TP01
02	Total Health	Mumbai	FW05
03	Live Life	Delhi	BS01
04	Pretty Woman	Delhi	5H06
05	Dreams	Delhi	FW12

(i) Write SQL Query to display ProductName and Price for all products whose Price is in the range 50 to 150.

```
SELECT ProductName, Price FROM PRODUCT WHERE Price BETWEEN 50 AND
    150;
```

(ii) Write SQL Query to display details of products whose manufacturer is either XYZ or ABC.

```
SELECT * FROM PRODUCT WHERE Manufacturer = 'XYZ' OR Manufacturer =
    'ABC';
```

(iii) Write SQL query to display ProductName, Manufacturer and Price for all products that are not giving any discount.

```
SELECT ProductName, Manufacturer, Price FROM PRODUCT WHERE Discount
     IS NULL;
```

(iv) Write SQL query to display ProductName and price for all products whose ProductName ends with 'h'.

```
SELECT ProductName, Price FROM PRODUCT WHERE ProductName LIKE '%h';
```

(v) Write SQL query to display ClientName, City, P_ID and ProductName for all clients whose city is Delhi.

```
SELECT C.ClientName, C.City, P.P_ID, P.ProductName FROM PRODUCT P,
    CLIENT C WHERE P.P_ID = C.P_ID AND C.CITY = 'DELHI';
```

(vi) Which column is used as Foreign Key and name the table where it has been used as Foreign key? The column used as a Foreign Key is P_ID in the CLIENT table, and it references the P_ID column in the PRODUCT table.

Q31) Answer the questions based on the table given below:

Table: HOSPITAL

S.No.	Name	Age	Department	Dateofadm	Charges	Sex
1	Arpit	62	Surgery	1998-01-21	300	M
2	Zareena	22	ENT	1997-12-12	250	F
3	Kareem	32	Orthopaedic	1998-02-19	200	M
4	Arun	12	Surgery	1998-01-11	300	M
5	Zubin	30	ENT	1998-01-12	250	M
6	Ketaki	16	ENT	1998-02-24	250	F
7	Ankit	29	Cardiology	1998-02-20	800	F
8	Zareen	45	Gynaecology	1998-02-22	300	F
9	Kush	19	Cardiology	1998-01-13	800	M
10	Shilpa	23	Nuclear Medicine	1998-02-21	400	F

(a) To list the names of all the patients admitted after 1998-01-15.

```
SELECT NAME FROM HOSPITAL WHERE DATEOFADM > '1998-01-15';
```

(b) To list the names of female patients who are in ENT department.

```
SELECT NAME FROM HOSPITAL WHERE SEX = 'F' AND DEPARTMENT = 'ENT';
```

(c) To list the names of all patients with their date of admission in ascending order.

```
SELECT NAME, DATEOFADM FROM HOSPITAL ORDER BY DATEOFADM;
```

(d) To display Patient's Name, Charges, Age for only female patients.

```
SELECT NAME, CHARGES, AGE FROM HOSPITAL WHERE SEX = 'F';
```

(e) Find out the output of the following SQL commands: SELECT COUNT(DISTINCT Charges) FROM HOSPITAL; SELECT MIN(Age) FROM HOSPITAL WHERE Sex = "F";

Q32) A department store MyStore is considering to maintain their inventory using SQL to store the data. As a database administrator, Abhay has decided that:

- Name of the database — mystore
- Name of the table — STORE

The attributes of STORE are as follows: ItemNo — numeric ItemName — character of size 20 Scode — numeric Quantity — numeric

Table: STORE

ItemNo	ItemName	Scode	Quantity
2005	Sharpener Classic	23	60
2003	Ball Pen 0.25	22	50
2002	Gel Pen Premium	21	150
2006	Gel Pen Classic	21	250
2001	Eraser Small	22	220
2004	Eraser Big	22	110
2009	Ball Pen 0.5	21	180

(a) Identify the attribute best suitable to be declared as a primary key. ItemNo attribute is best suitable to be declared as a primary key as it uniquely identifies each item in the inventory.

(b) Write the degree and cardinality of the table STORE. The degree of the table STORE is 4, and the cardinality of the table STORE is 7.

(c) Insert the following data into the attributes ItemNo, ItemName and SCode respectively in the given table STORE.ItemNo = 2010, ItemName = "Note Book" and Scode = 25.

```
INSERT INTO STORE(ItemNo, ItemName, Scode) VALUES(2010, 'Note Book'
    , 25);
```

(d) Abhay wants to remove the table STORE from the database MyStore. Which command will he use from the following?

```
DELETE FROM STORE;
DROP TABLE STORE;
DROP DATABASE MYSTORE;
DELETE STORE FROM MYSTORE;
```

DROP TABLE STORE; Reason — DROP TABLE command is used to remove/delete a table permanently. The syntax is : DROP TABLE ¡table_name¿;. Hence, according to this DROP TABLE STORE; is the correct command to remove the STORE table from the database MyStore.

(e) Now Abhay wants to display the structure of the table STORE, i.e., name of the attributes and their respective data types that he has used in the table. Write the query to display the same.

```
DESCRIBE STORE;
```