Python Laboratory

SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE AWARD OF THE DEGREE OF

BACHELOR OF TECHNOLOGY

(Information Technology)

SUBMITTED BY:
Jay Kumar (2203844)

SUBMITTED TO:
Prof. Reema Varma

DEPARTMENT OF INFORMATION TECHNOLOGY

GURU NANAK DEV ENGINEERING COLLEGE LUDHIANA

(An Autonomous College Under UGC ACT)

Python Practical File

Name: Raushan Kumar

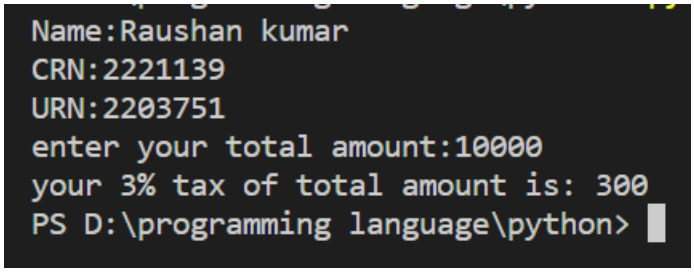
CRN:2221139

URN:2203751

Branch: IT(B2)

1. The tax calculator program of the case study outputs a floating-point number that might show more than two digits of precision. Use the round function to modify the program to display at most two digits of precision in the output number.

```
print("Name:Raushan kumar\nCRN:2221139\nURN:2203751")
total=int(input("enter your total amount:"))
tax=(3/100)*total
tax1=round(tax)
print("your 3% tax of total amount is:",tax1)
```



2. You can calculate the surface area of a cube if you know the length of an edge. Write a program that takes the length of an edge (an integer) as input and prints the cube's surface area as output.

```
print("Name:Raushan kumar\nCRN:2221139\nURN:2203751")
edge=int(input("enter the length of an edge:"))
print("the surface area of a cube is:",6*edge*edge)
```

```
Name:Raushan kumar
CRN:2221139
URN:2203751
enter the length of an edge:10
the surface area of a cube is: 600
PS D:\programming language\python>
```

3. Five Star Retro Video rents VHS tapes and DVDs to the same connoisseurs who like to buy LP record albums. The store rents new videos for $3.00 a night, and oldies for $2.00 a night. Write a program that the clerks at Five Star Retro Video can use to calculate the total charge for a customer's video rentals. The program should prompt the user for the number of each type of video and output the total.

```
print("Name:Raushan kumar\nCRN:2221139\nURN:2203751")

rents_new_video=int(input("enter the store rents new videos:"))

oldies=int(input("enter the oldies:"))

total_rents_new_video=3*rents_new_video

total_oldies=2*oldies

total=total_rents_new_video + total_oldies

print("The total cost for the rentals is:$",total)
```

```
Name:Raushan kumar
CRN:2221139
URN:2203751
enter the store rents new videos:12
enter the oldies:5
The total cost for the rentals is:$ 46
```

4.Write a program that takes the radius of a sphere (a floating-point number) as input and then outputs the sphere's diameter, circumference, surface area, and volume.

```
import math

print("Name:Raushan kumar\nCRN:2221139\nURN:2203751")

radius=float(input("Enter the radius of a sphere:"))

diameter=2*radius

circumference=2*math.pi*radius

sarea=4*math.pi*radius*radius

volume=(4/3)*math.pi*radius*radius*radius

print("The diameter of sphere is:",diameter)
```

print("The circumference of sphere is:",circumference)

print("The surface area of sphere is:",sarea)

print("The volume of sphere is:",volume)

```
Name:Raushan kumar
CRN:2221139
URN:2203751
Enter the radius of a sphere:5
The diameter of sphere is: 10.0
The circumference of sphere is: 31.41592653589793
The surface area of sphere is: 314.1592653589793
The volume of sphere is: 523.5987755982989
PS D:\programming language\python>
```

5.An object's momentum is its mass multiplied by its velocity. Write a program that accepts an object's mass (in kilograms) and velocity (in meters per second) as inputs and then outputs its momentum.

print("Name:Raushan kumar\nCRN:2221139\nURN:2203751")

mass=int(input("Enter object's mass in kg:"))

velocity=int(input("Enter object's velocity in meters per second:"))

momentum=mass*velocity

print("The monemtum of object is:",momentum)

```
Name:Raushan kumar
CRN:2221139
URN:2203751
Enter object's mass in kg:5
Enter object's velocity in meters per second:6
The monemtum of object is: 30
PS D:\programming language\python>
```

Methods

6.write a program to differentiate append vs extend method in list.

print("Name:Raushan kumar\nCRN:2221139\nURN:2203751")

list=[1,2,3]

print("before append:",list)

list.append(4)

print("after append:",list)

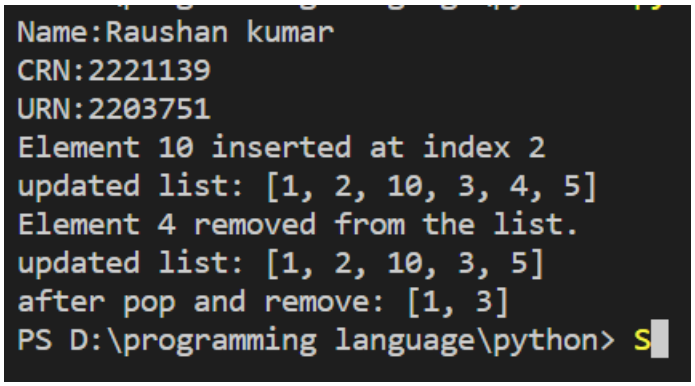
```
#n list.append(5,6) -->error not allowed
print("before extend:",list)
list.extend([4,5])
print("after extend:",list)
list2=[6,7]
print("before append:",list)
list.append(list2)
print("after append:",list)
list3=[8,9]
print("before extend:",list)
list.extend(list3)
print("after extend:",list)
```

```
Name:Raushan kumar
CRN:2221139
URN:2203751
before append: [1, 2, 3]
after append: [1, 2, 3, 4]
before extend: [1, 2, 3, 4]
after extend: [1, 2, 3, 4, 4, 5]
before append: [1, 2, 3, 4, 4, 5]
after append: [1, 2, 3, 4, 4, 5, [6, 7]]
before extend: [1, 2, 3, 4, 4, 5, [6, 7]]
after extend: [1, 2, 3, 4, 4, 5, [6, 7], 8, 9]
PS D:\programming language\python>
```

7. write a program to demonstrate how to insert element and how to remove element from a list.

```
print("Name:Raushan kumar\nCRN:2221139\nURN:2203751")
def insert_element(lst, index, element):
    # lst.insert(element) -->error
    lst.insert(index, element)
    print(f"Element {element} inserted at index {index}")
    print("updated list:",lst)


def remove_element(lst, element):
```

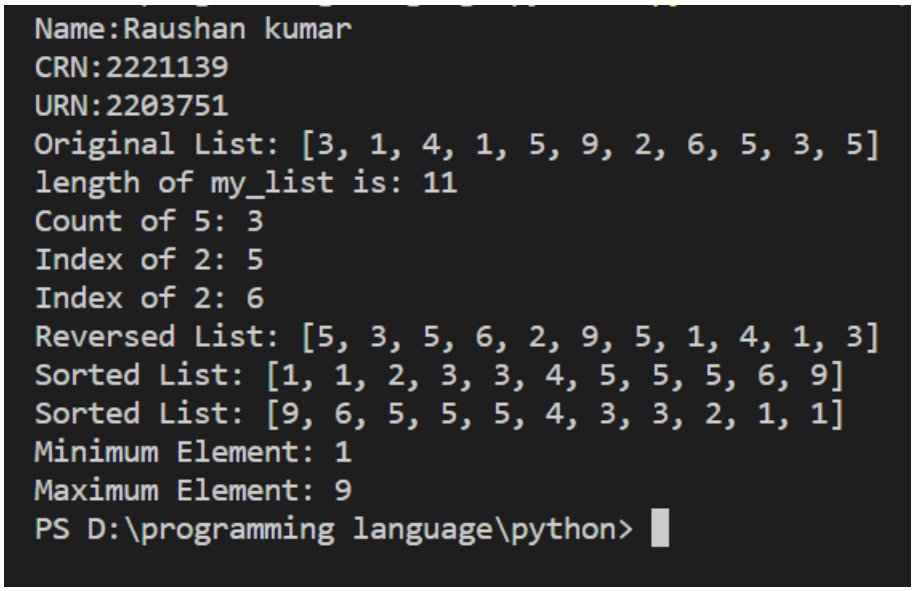
```
        lst.remove(element)
        print(f"Element {element} removed from the list.")
        print("updated list:",lst)
mylist = [1, 2, 3, 4, 5]
insert_element(mylist, 2, 10)
remove_element(mylist, 4)
mylist.pop()
mylist.pop(2)#by index
mylist.remove(2)# by element
print("after pop and remove:",mylist)
```



8.write a program to demonstrate count , index ,reverse ,sort ,min ,max function in list.

```
print("Name:Raushan kumar\nCRN:2221139\nURN:2203751")
my_list = [3, 1, 4, 1, 5, 9, 2, 6, 5, 3, 5]
print("Original List:", my_list)
# Count function: Count occurrences of a specific element
count_of_5 = my_list.count(5)
length=len(my_list)
print("length of my_list is:", length)
print("Count of 5:", count_of_5)
# Index function: Find the index of the first occurrence of a specific element
index_of_9 = my_list.index(9)
print("Index of 2:", index_of_9)
# Index function: Find the index of the first occurrence of a  element in between sub list
```

```
index_of_2 = my_list.index(2,4,8)
print("Index of 2:", index_of_2)
# Reverse function: Reverse the elements of the list
my_list.reverse()
print("Reversed List:", my_list)
# Sort function: Sort the elements of the list in ascending order
my_list.sort()
print("Sorted List:", my_list)
# Sort function: Sort the elements of the list in decending order
my_list.sort(reverse=True)
print("Sorted List:", my_list)
# Min function: Find the minimum element in the list
min_element = min(my_list)
print("Minimum Element:", min_element)
# Max function: Find the maximum element in the list
max_element = max(my_list)
print("Maximum Element:", max_element)
```



PRACTICAL NO.5

Design with Functions

1.A list is sorted in ascending order if it is empty or each item except the last one is less than or equal to its successor. Define a predicate isSorted that expects a list as an argument and returns True if the list is sorted, or returns False otherwise.

CODE:

```
def isSorted(lst):
    if len(lst) <= 1:
        return True
    for i in range(len(lst) - 1):
        if lst[i] > lst[i + 1]:
            return False
    return True


# Test cases
print(isSorted([]))         # Output: True
print(isSorted([1]))        # Output: True
print(isSorted([1, 2, 3]))  # Output: True
print(isSorted([3, 2, 1]))  # Output: False
print(isSorted([1, 3, 2]))  # Output: False
```

OUTPUT:

```
True
True
True
False
False
```

2.Write a recursive function that expects a pathname as an argument. The pathname can be either the name of a file or the name of a directory. If the pathname refers to a file, its name is displayed, followed by its contents. Otherwise, if the pathname refers to a directory, the function is applied to each name in the directory. Test this function in a new program.
CODE:

```
import os

def display_contents(path):
    if os.path.isfile(path):
        with open(path, 'r') as file:
            print(f"File: {os.path.basename(path)}")
            print("Contents:")
            print(file.read())
    elif os.path.isdir(path):
        print(f"Directory: {os.path.basename(path)}")
        print("Contents:")
        for item in os.listdir(path):
            display_contents(os.path.join(path, item))
```

Test

display_contents('example_directory')

OUTPUT:

```
Directory: example_directory
Contents:
File: file1.txt
Contents:
Hello, this is file 1.

File: file2.txt
Contents:
Hello, this is file 2.

Directory: subdirectory
Contents:
File: file3.txt
Contents:
Hello, this is file 3.
```

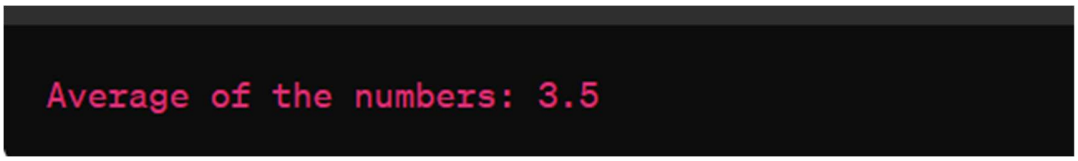
3.Write a program that computes and prints the average of the numbers in a text file. You should make use of two higher-order functions to simplify the design.

CODE:

```
def read_numbers(filename):
    with open(filename, 'r') as file:
        numbers = [float(line.strip()) for line in file]
    return numbers

def compute_average(numbers):
    if not numbers:
        return 0
    total = sum(numbers)
```

```
    return total / len(numbers)

# Main function
def main():
    filename = 'numbers.txt'
    numbers = read_numbers(filename)
    average = compute_average(numbers)
    print("Average of the numbers:", average)

# Test
main()
```

OUTPUT:



PRACTICAL NO.6

Design with Classes

1.Add three methods to the Student class that compare two Student objects. One method should test for equality. A second method should test for less than. The third method should test for greater than or equal to. In each case, the method returns the result of the comparison of the two students' names. Include a main function that tests all of the comparison operators.

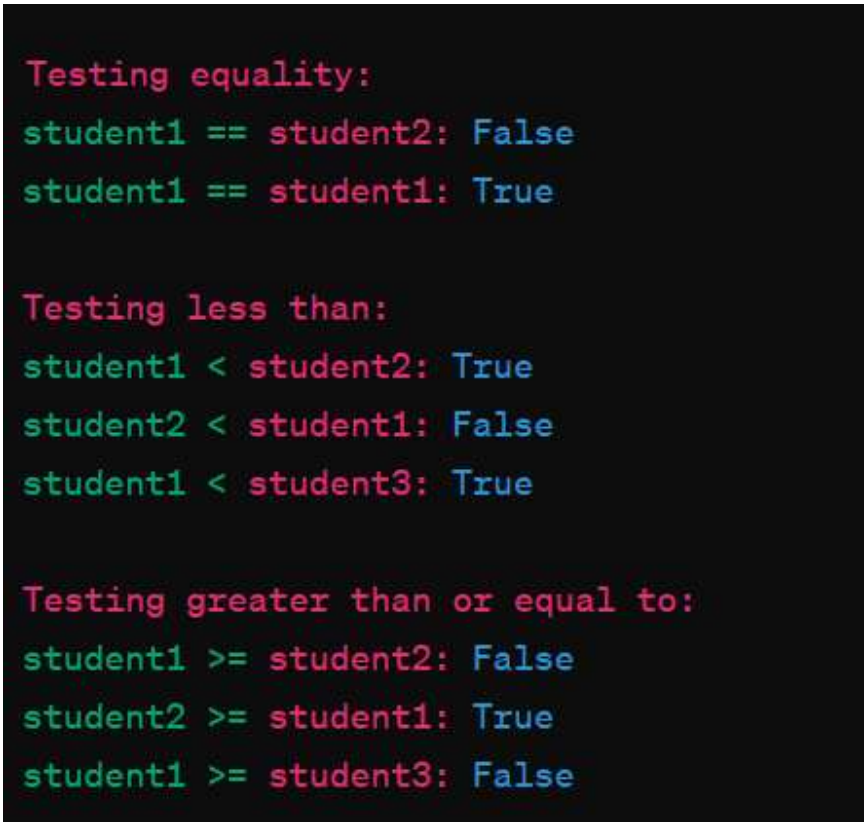
C0DE:

```
class Student:
    def __init__(self, name):
        self.name = name

    def __eq__(self, other):
        return self.name == other.name

    def __lt__(self, other):
        return self.name < other.name

    def __ge__(self, other):
        return self.name >= other.name

def main():
    # Creating student objects
    student1 = Student("Alice")
    student2 = Student("Bob")
    student3 = Student("Charlie")
```

```
# Testing equality
print("Testing equality:")
print("student1 == student2:", student1 == student2)
print("student1 == student1:", student1 == student1)


# Testing less than
print("\nTesting less than:")
print("student1 < student2:", student1 < student2)
print("student2 < student1:", student2 < student1)
print("student1 < student3:", student1 < student3)


# Testing greater than or equal to
print("\nTesting greater than or equal to:")
print("student1 >= student2:", student1 >= student2)
print("student2 >= student1:", student2 >= student1)
print("student1 >= student3:", student1 >= student3)
```

```
if __name__ == "__main__":
    main()
```

OUTPUT:



2. This project assumes that you have completed Project 1. Place several Student objects into a list and shuffle it. Then run the sort method with this list and display all of the students' information.

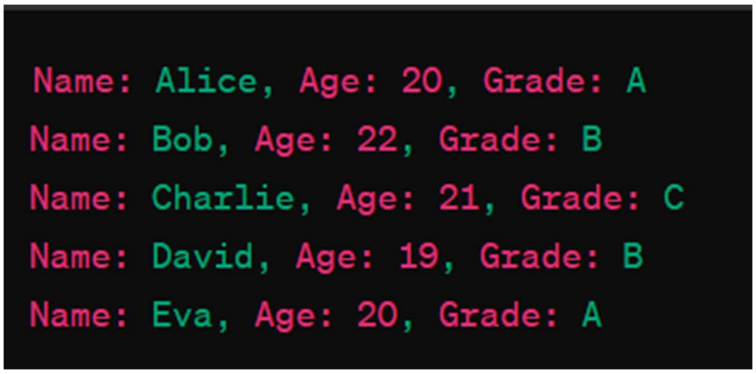
CODE:

```
import random
```

```
class Student:
    def __init__(self, name, age, grade):
        self.name = name
        self.age = age
        self.grade = grade

    def __str__(self):
        return f"Name: {self.name}, Age: {self.age}, Grade: {self.grade}"

def main():
    # Creating a list of Student objects
    students = [
        Student("Alice", 20, "A"),
        Student("Bob", 22, "B"),
        Student("Charlie", 21, "C"),
        Student("David", 19, "B"),
        Student("Eva", 20, "A")
    ]

    # Shuffling the list of students
    random.shuffle(students)

    # Sorting the list of students by their names
    students.sort(key=lambda student: student.name)
```

```
    # Displaying all students' information
    for student in students:
        print(student)


if __name__ == "__main__":
    main()
```

OUTPUT:



3.The str method of the Bank class returns a string containing the accounts in random order. Design and implement a change that causes the accounts to be placed in the string by order of name. (Hint: You will also have to define some methods in the SavingsAccount class.)

CODE:

```
class SavingsAccount:
    def __init__(self, account_number, holder_name, balance):
        self.account_number = account_number
        self.holder_name = holder_name
        self.balance = balance

    def __str__(self):
        return f"Account Number: {self.account_number}, Holder Name:
{self.holder_name}, Balance: {self.balance}"

    def __lt__(self, other):
        return self.holder_name < other.holder_name
```

```
    def __eq__(self, other):
        return self.holder_name == other.holder_name


class Bank:
    def __init__(self, accounts):
        self.accounts = accounts

    def __str__(self):
        sorted_accounts = sorted(self.accounts, key=lambda acc: acc.holder_name)
        return '\n'.join(str(acc) for acc in sorted_accounts)


# Testing
if __name__ == "__main__":
    account1 = SavingsAccount(12345, "Alice", 1000)
    account2 = SavingsAccount(54321, "Bob", 2000)
    account3 = SavingsAccount(98765, "Charlie", 1500)

    bank = Bank([account1, account2, account3])
    print(bank)
```

OUTPUT:

```
Account Number: 12345, Holder Name: Alice, Balance: 1000
Account Number: 54321, Holder Name: Bob, Balance: 2000
Account Number: 98765, Holder Name: Charlie, Balance: 1500
```