

# **20MCA131 - PROGRAMMING LAB**

*Lab Report Submitted By*

**JENNY JOHNSON**

**Reg. No.: AJC22MCA-2053**

*In Partial Fulfilment for the Award of the Degree of*

**MASTER OF COMPUTER APPLICATIONS (2 Year) (MCA)**

**APJ ABDUL KALAM TECHNOLOGICAL UNIVERSITY**



**AMAL JYOTHI COLLEGE OF ENGINEERING**

**KANJIRAPPALLY**

[Affiliated to APJ Abdul Kalam Technological University, Kerala. Approved by AICTE,  
Accredited by NAAC with 'A' grade. Koovapally, Kanjirappally, Kottayam, Kerala 686518]

**2022-2023**

**DEPARTMENT OF COMPUTER APPLICATIONS**  
**AMAL JYOTHI COLLEGE OF ENGINEERING**  
**KANJIRAPPALLY**



**CERTIFICATE**

This is to certify that the lab report, “**20MCA131 PROGRAMMING LAB**” is the bona fide work of **JENNY JOHNSON (AJC22MCA-2053)** in partial fulfilment of the requirements for the award of the Degree of Master of Computer Applications under APJ Abdul Kalam Technological University during the year **2022-23**.

Sr. Elsin Chakkalackal S.H  
**Lab In- Charge**

Rev. Fr. Dr. Rubin Thottupurathu Jose  
**Head of the Department**

**Internal Examiner**

**External Examiner**

Course Code	Course Name	Syllabus Year	L-T-P-C
20MCA131	Programming Lab	2020	0-1-3-2

## VISION

To promote an academic and research environment conducive for innovation centric technical education.

## MISSION

- MS1 - Provide foundations and advanced technical education in both theoretical and applied Computer Applications in-line with Industry demands.
- MS2 - Create highly skilled computer professionals capable of designing and innovating real life solutions.
- MS3 - Sustain an academic environment conducive to research and teaching focused to generate up-skilled professionals with ethical values.
- MS4 - Promote entrepreneurial initiatives and innovations capable of bridging and contributing with sustainable, socially relevant technology solutions.

## COURSE OUTCOME

CO	Outcome	Target
CO1	Understands basics of Python Programming language including input/output functions, operators, basic and collection data types	60.5
CO2	Implement decision making, looping constructs and functions	60.5
CO3	Design modules and packages - built in and user defined packages	60.5
CO4	Implement object-oriented programming and exception handling.	60.5
CO5	Create files and form regular expressions for effective search operations on strings and files.	60.5

## COURSE END SURVEY

CO	Survey Question	Answer Format
CO1	To what extent you understands basics of Python Programming language including input/output functions, operators, basic and collection data types	Excellent/Very Good/Good Satisfactory/Needs improvement
CO2	To what extent you implement decision making, looping constructs and functions	Excellent/Very Good/Good Satisfactory/Needs improvement
CO3	To what extent you design modules and packages - built in and user defined packages	Excellent/Very Good/Good Satisfactory/Needs improvement
CO4	To what extent you implement object-oriented programming and exception handling.	Excellent/Very Good/Good Satisfactory/Needs improvement
CO5	To what extent you are able to create files and form regular expressions for effective search operations on strings and files.	Excellent/Very Good/Good Satisfactory/Needs improve ment

# CONTENT

<b>Sl. No.</b>	<b>Experiment</b>	<b>Date</b>	<b>CO</b>	<b>Page No.</b>
<b>1</b>	Display future leap years from current year to a final year entered by user.	31-10-2022	CO1	<b>1</b>
<b>2</b>	List comprehensions: (a) Generate positive list of numbers from a given list of integers (b) Square of N numbers (c) Form a list of vowels selected from a given word.	31-10-2022	CO1	<b>2</b>
<b>3</b>	Count the occurrences of each word in a line of text.	31-10-2022	CO1	<b>4</b>
<b>4</b>	Prompt the user for a list of integers. For all values greater than 100, store 'over' instead.	31-10-2022	CO1	<b>5</b>
<b>5</b>	Store a list of first names. Count the occurrences of 'a' within the list	31-10-2022	CO1	<b>7</b>
<b>6</b>	Enter 2 lists of integers. Check (a) Whether list are of same length (b) whether list sums to same value (c) whether any value occur in both	31-10-2022	CO1	<b>9</b>
<b>7</b>	Get a string from an input string where all occurrences of first character replaced with '\$', except first character.	31-10-2022	CO1	<b>12</b>
<b>8</b>	Create a string from given string where first and last characters exchanged.	31-10-2022	CO1	<b>13</b>
<b>9</b>	Accept the radius from user and find area of circle.	04-11-2022	CO1	<b>14</b>
<b>10</b>	Find biggest of 3 numbers entered.	04-11-2022	CO1	<b>15</b>
<b>11</b>	Accept a file name from user and print extension of that.	05-11-2022	CO1	<b>17</b>
<b>12</b>	Create a list of colors from comma-separated color names entered by user. Display first and last colors.	05-11-2022	CO1	<b>18</b>
<b>13</b>	Accept an integer n and compute n+nn+nnn	27-10-2022	CO1	<b>19</b>
<b>14</b>	Print out all colors from color-list1 not contained in color-list2.	05-11-2022	CO1	<b>20</b>
<b>15</b>	Create a single string separated with space from two strings by swapping the character at position 1.	05-11-2022	CO1	<b>21</b>

<b>Sl. No.</b>	<b>Experiment</b>	<b>Date</b>	<b>CO</b>	<b>Page No.</b>
<b>16</b>	Sort dictionary in ascending and descending order.	05-11-2022	CO1	<b>22</b>
<b>17</b>	Merge two dictionaries.	05-11-2022	CO1	<b>23</b>
<b>18</b>	Find gcd of 2 numbers.	05-11-2022	CO1	<b>25</b>
<b>19</b>	From a list of integers, create a list removing even numbers.	05-11-2022	CO1	<b>26</b>
<b>20</b>	Program to find the factorial of a number	07-11-2022	CO2	<b>27</b>
<b>21</b>	Generate Fibonacci series of N terms	07-11-2022	CO2	<b>28</b>
<b>22</b>	Find the sum of all items in a list	07-11-2022	CO2	<b>30</b>
<b>23</b>	Generate a list of four digit numbers in a given range with all their digits even and the number is a perfect square.	07-11-2022	CO2	<b>32</b>
<b>24</b>	Display the given pyramid with step number accepted from user. Eg: N=4 1 2 4 3 6 9 4 8 12 16	07-11-2022	CO2	<b>33</b>
<b>25</b>	Count the number of characters (character frequency) in a string.	07-11-2022	CO2	<b>34</b>
<b>26</b>	Add 'ing' at the end of a given string. If it already ends with 'ing', then add 'ly'.	11-11-2022	CO2	<b>35</b>
<b>27</b>	Accept a list of words and return length of longest word.	11-11-2022	CO2	<b>36</b>
<b>28</b>	Construct following pattern using nested loop *	14-11-2022	CO2	<b>38</b>

<b>Sl. No.</b>	<b>Experiment</b>	<b>Date</b>	<b>CO</b>	<b>Page No.</b>
<b>29</b>	Generate all factors of a number.	18-11-2022	CO2	<b>40</b>
<b>30</b>	Write lambda functions to find area of square, rectangle and triangle.	18-11-2022	CO2	<b>41</b>
<b>31</b>	Write a Python Program to subtract five days from the current date.	21-11-2022	CO3	<b>43</b>
<b>32</b>	Create a package graphics with modules rectangle, circle and sub-package 3D-graphics with modules cuboid and sphere. Include methods to find area and perimeter of respective figures in each module. Write programs that finds area and perimeter of figures by different importing statements. (Include selective import of modules and import * statements)	02-12-2022	CO3	<b>44</b>
<b>33</b>	Create Rectangle class with attributes length and breadth and methods to find area and perimeter. Compare two Rectangle objects by their area.	09-12-2022	CO4	<b>47</b>
<b>34</b>	Create a Bank account with members account number, name, type of account and balance. Write constructor and methods to deposit at the bank and withdraw an amount from the bank.	09-12-2022	CO4	<b>49</b>
<b>35</b>	Create a class Rectangle with private attributes length and width. Overload '<' operator to compare the area of 2 rectangles.	12-12-2022	CO4	<b>52</b>
<b>36</b>	Create a class Time with private attributes hour, minute and second. Overload '+' operator to find sum of 2 time.	16-12-2022	CO4	<b>54</b>
<b>37</b>	Create a class Publisher (name). Derive class Book from Publisher with attributes title and author. Derive class Python from Book with attributes price and no.of pages. Write a program that displays information about a Python book. Use base class constructor invocation and method overriding.	17-12-2022	CO4	<b>57</b>
<b>38</b>	Write a Python program to read a file line by line and store it into a list.	19-12-2022	CO5	<b>59</b>
<b>39</b>	Python program to copy odd lines of one file to other.	06-01-2023	CO5	<b>61</b>

<b>Sl. No.</b>	<b>Experiment</b>	<b>Date</b>	<b>CO</b>	<b>Page No.</b>
<b>40</b>	Write a Python program to read each row from a given csv file and print a list of strings.	06-01-2023	CO5	<b>63</b>
<b>41</b>	Write a Python program to read specific columns of a given CSV file and print the content of the columns.	09-01-2023	CO5	<b>65</b>
<b>42</b>	Write a Python program to write a Python dictionary to a csv file. After writing the CSV file read the CSV file and display the content.	09-01-2023	CO5	<b>67</b>
<b>43</b>	Micro Project	20-1-2023	CO1, CO2,C O3,CO 4,CO5	<b>69</b>





## **Experiment No.: 1**

### **Aim**

To display future leap years from current year to a final year entered by user.

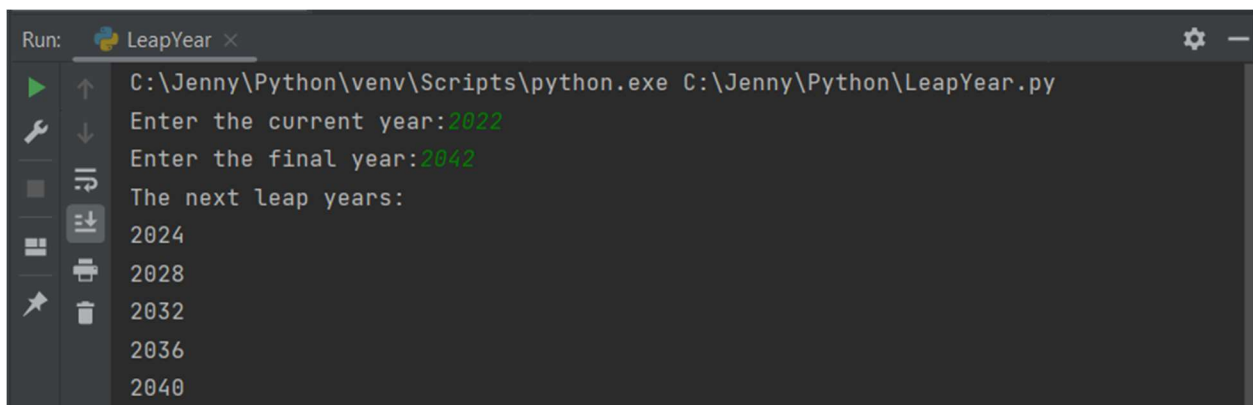
### **CO1**

Understands basics of Python Programming language including input/output functions, operators, basic and collection data types.

### **Procedure**

```
# Leap year #  
  
cr = int(input("Enter the current year:"))  
fi = int(input("Enter the final year:"))  
print("The next leap years:")  
while cr <= fi:  
    if cr % 4 == 0:  
        print(cr)  
        cr = cr+1  
    else:  
        cr = cr+1
```

### **Output Screenshot**



The screenshot shows a terminal window titled "Run: LeapYear x". The command executed is "C:\Jenny\Python\venv\Scripts\python.exe C:\Jenny\Python\LeapYear.py". The program prompts for "Enter the current year:" with input "2022" and "Enter the final year:" with input "2042". The output is "The next leap years:" followed by a list of years: 2024, 2028, 2032, 2036, and 2040.

### **Result**

The program was executed and the result was successfully obtained. Thus, CO1 was obtained.

## **Experiment No.: 2**

### **Aim**

List comprehensions: (a) Generate positive list of numbers from a given list of integers (b)

Square of N numbers (c) Form a list of vowels selected from a given word.

### **CO1**

Understands basics of Python Programming language including input/output functions, operators, basic and collection data types.

### **Procedure**

# list comprehensions #

(a)

```
li = int(input("Enter the size of the list:"))
list1 = []
list2 = []
for i in range(li):
    v = int(input("Enter the integer: "))
    list1.append(v)
print("First list: ", list1)
list2 = [list1[i] for i in range(li) if list1[i] >= 0]
print("List of positive numbers:", list2)
```

(b)

```
x = int(input("Enter the starting number : "))
y = int(input("Enter the ending number : "))
square = [i*i for i in range(x, y+1)]
print(square)
```

(c)

```
w = str(input("Enter the word : "))
v_list = [i for i in w if i == 'a' or i == 'e' or i == 'i' or i == "o" or
           i == 'u' or i == "A" or i == "E" or i == "I" or i == "O" or i == "U"]
print(v_list)
```

## Output Screenshot

(a)

```
Run: ListComprehensions(a) ×
C:\Jenny\Python\venv\Scripts\python.exe "C:\Jenny\Python\ListComprehensions(a).py"
Enter the size of the list: 5
Enter the integer: 0
Enter the integer: -1
Enter the integer: 9
Enter the integer: -4
Enter the integer: 34
First list: [0, -1, 9, -4, 34]
List of positive numbers: [0, 9, 34]
```

(b)

```
Run: ListComprehensions(b) ×
C:\Jenny\Python\venv\Scripts\python.exe "C:\Jenny\Python\ListComprehensions(b).py"
Enter the starting number : 2
Enter the ending number : 10
[4, 9, 16, 25, 36, 49, 64, 81, 100]
Process finished with exit code 0
```

(c)

```
Run: ListComprehensions(c) ×
C:\Jenny\Python\venv\Scripts\python.exe "C:\Jenny\Python\ListComprehensions(c).py"
Enter the word : justin
['u', 'i']
Process finished with exit code 0
```

## Result

The program was executed and the result was successfully obtained. Thus, CO1 was obtained.

## **Experiment No.: 3**

### **Aim**

To count the occurrences of each word in a line of text.

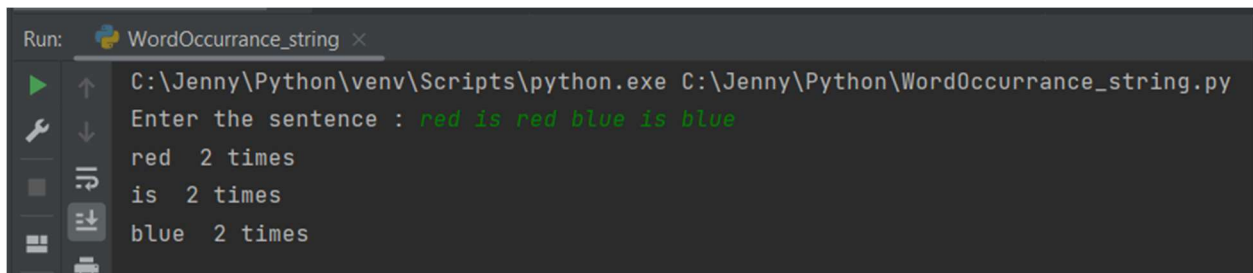
### **CO1**

Understands basics of Python Programming language including input/output functions, operators, basic and collection data types.

### **Procedure**

```
# Count of each word in a sentence #  
s = input("Enter the sentence : ").split()  
a = {}  
for i in s:  
    if i in a:  
        a[i] = a[i]+1  
    else:  
        a[i] = 1  
for m, n in a.items():  
    print(m, "", n, "times")
```

### **Output Screenshot**



The screenshot shows a terminal window titled 'Run: WordOccurance\_string'. The command executed is 'C:\Jenny\Python\venv\Scripts\python.exe C:\Jenny\Python\Word0ccurance\_string.py'. The input is 'Enter the sentence : red is red blue is blue'. The output is:

```
red 2 times  
is 2 times  
blue 2 times
```

### **Result**

The program was executed and the result was successfully obtained. Thus, CO1 was obtained.

**Experiment No.: 4****Aim**

Prompt the user for a list of integers. For all values greater than 100, store 'over' instead.

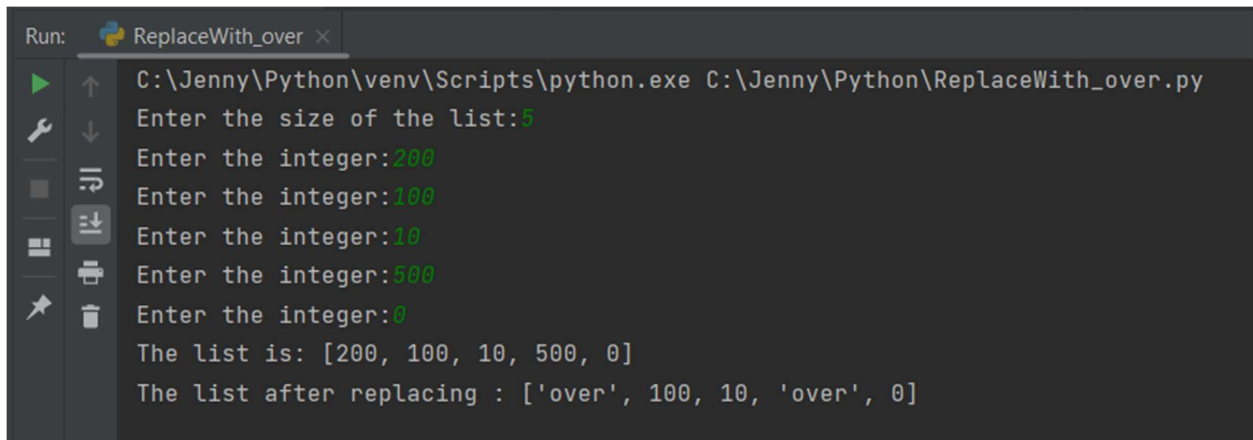
**CO1**

Understands basics of Python Programming language including input/output functions, operators, basic and collection data types.

**Procedure**

```
# for values greater than 100,store 'over' #  
li = []  
a = int(input("Enter the size of the list:"))  
for i in range(a):  
    v = int(input("Enter the integer:"))  
    li.append(v)  
print("The list is:", li)  
for i in range(a):  
    if li[i] > 100:  
        li[i] = "over"  
print("The list after replacing :", li)
```

## Output Screenshot



The screenshot shows a terminal window titled 'Run: ReplaceWith\_over x'. The command executed is 'C:\Jenny\Python\venv\Scripts\python.exe C:\Jenny\Python\ReplaceWith\_over.py'. The program prompts the user to enter the size of the list (5), followed by five integers (200, 100, 10, 500, 0). It then displays the list [200, 100, 10, 500, 0] and the list after replacing the first and fourth elements with the string 'over', resulting in ['over', 100, 10, 'over', 0].

```
Run: ReplaceWith_over x
C:\Jenny\Python\venv\Scripts\python.exe C:\Jenny\Python\ReplaceWith_over.py
Enter the size of the list:5
Enter the integer:200
Enter the integer:100
Enter the integer:10
Enter the integer:500
Enter the integer:0
The list is: [200, 100, 10, 500, 0]
The list after replacing : ['over', 100, 10, 'over', 0]
```

## Result

The program was executed and the result was successfully obtained. Thus, CO1 was obtained.

**Experiment No.: 5****Aim**

Store a list of first names. Count the occurrences of 'a' within the list

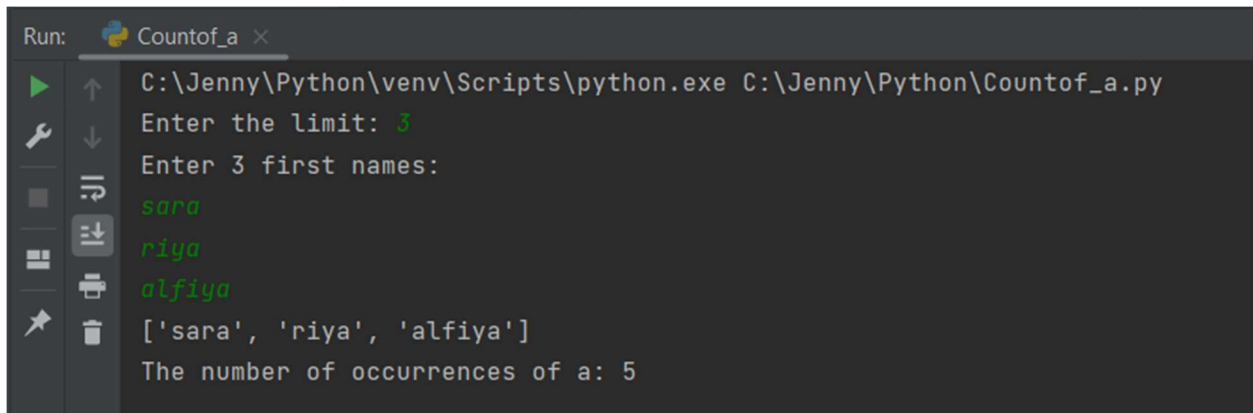
**CO1**

Understands basics of Python Programming language including input/output functions, operators, basic and collection data types.

**Procedure**

```
x = int(input("Enter the limit: "))
names = []
count = 0
print("Enter", x, "first names:")
for i in range(0, x):
    s = input()
    names.append(s)
print(names)
for i in names:
    for j in i:
        if j == "a":
            count = count+1
print("The number of occurrences of a:", count)
```

## Output Screenshot



The screenshot shows a terminal window titled 'Run: Countof\_a'. The command executed is 'C:\Jenny\Python\venv\Scripts\python.exe C:\Jenny\Python\Countof\_a.py'. The program prompts the user to 'Enter the limit:' and the input '3' is shown. It then prompts 'Enter 3 first names:' and the inputs 'sara', 'riya', and 'alfiya' are shown on separate lines. The program outputs the list ['sara', 'riya', 'alfiya'] and then states 'The number of occurrences of a: 5'.

```
Run: Countof_a ×
C:\Jenny\Python\venv\Scripts\python.exe C:\Jenny\Python\Countof_a.py
Enter the limit: 3
Enter 3 first names:
sara
riya
alfiya
['sara', 'riya', 'alfiya']
The number of occurrences of a: 5
```

## Result

The program was executed and the result was successfully obtained. Thus, CO1 was obtained.



**Experiment No.: 6****Aim**

Enter 2 lists of integers. Check (a) Whether list are of same length (b) whether list sums to same value (c) whether any value occur in both.

**CO1**

Understands basics of Python Programming language including input/output functions, operators, basic and collection data types.

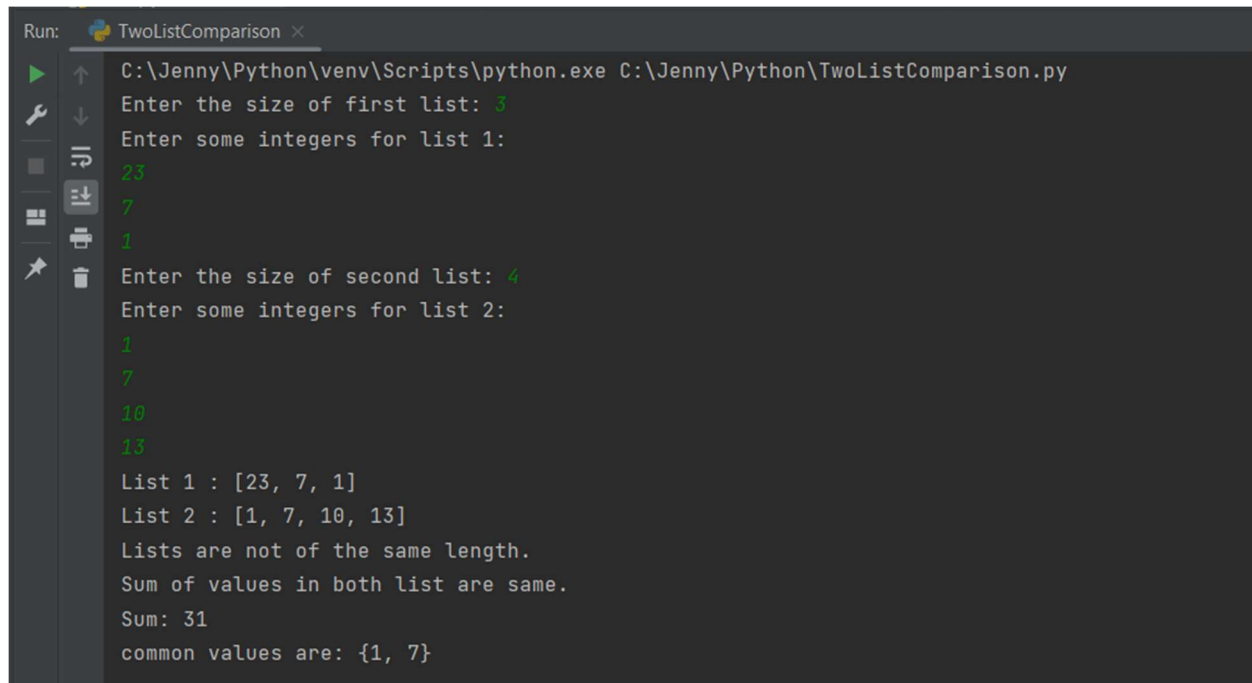
**Procedure**

# Two list comparison #

```
l1 = []
l2 = []
s1 = 0
s2 = 0
n1 = int(input("Enter the size of first list: "))
print("Enter some integers for list 1: ")
for i in range(n1):
    n = int(input())
    l1.append(n)
n2 = int(input("Enter the size of second list: "))
print("Enter some integers for list 2: ")
for i in range(n2):
    m = int(input())
    l2.append(m)
print("List 1 :", l1)
print("List 2 :", l2)
if len(l1) == len(l2):
```

```
print("List are of same length.\nLength:", len(l1))
else:
    print("Lists are not of the same length.")
for j in range(n1):
    s1 = s1+l1[j]
for k in range(n2):
    s2 = s2+l2[k]
if s1 == s2:
    print("Sum of values in both list are same.\nSum:", s1)
else:
    print("Sum of values in both list are not same.\nList1 sum: {}\nList2 sum: {}".format(s1, s2))
x = set(l1)
y = set(l2)
z = x.intersection(y)
if z == 0:
    print("No common values")
else:
    print("common values are:", z)
```

## Output Screenshot



The screenshot shows a terminal window titled 'TwoListComparison' with the following output:

```
Run: C:\Jenny\Python\venv\Scripts\python.exe C:\Jenny\Python\TwoListComparison.py
Enter the size of first list: 3
Enter some integers for list 1:
23
7
1
Enter the size of second list: 4
Enter some integers for list 2:
1
7
10
13
List 1 : [23, 7, 1]
List 2 : [1, 7, 10, 13]
Lists are not of the same length.
Sum of values in both list are same.
Sum: 31
common values are: {1, 7}
```

## Result

The program was executed and the result was successfully obtained. Thus, CO1 was obtained.

## **Experiment No.: 7**

### **Aim**

Get a string from an input string where all occurrences of first character replaced with '\$', except first character.

### **CO1**

Understands basics of Python Programming language including input/output functions, operators, basic and collection data types.

### **Procedure**

# Replacing first character occurrence with '\$'##

```
s = str(input("Enter a string:"))
```

```
f = s[0]
```

```
for i in s:
```

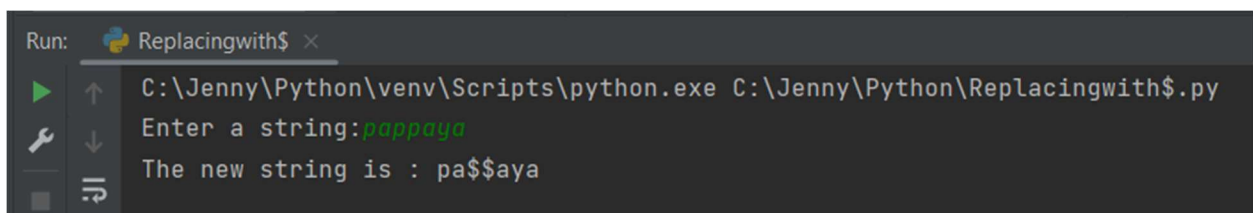
```
    if i == f:
```

```
        s = s.replace(i, "$")
```

```
    s = f+s[1:]
```

```
print("The new string is :", s)
```

### **Output Screenshot**



```
Run: Replacingwith$ x
C:\Jenny\Python\venv\Scripts\python.exe C:\Jenny\Python\Replacingwith$.py
Enter a string:pappaya
The new string is : pa$$aya
```

### **Result**

The program was executed and the result was successfully obtained. Thus, CO1 was obtained.

## **Experiment No.: 8**

### **Aim**

Create a string from given string where first and last characters exchanged.

### **CO1**

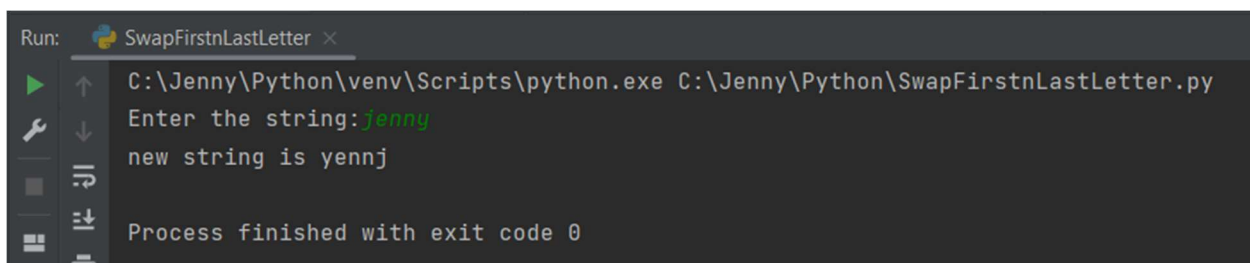
Understands basics of Python Programming language including input/output functions, operators, basic and collection data types.

### **Procedure**

```
# first and last char interchange #
```

```
string = input("Enter the string:")  
first = string[0]  
last = string[-1]  
n = last + string[1:-1] + first  
print("new string is", n)
```

### **Output Screenshot**



The screenshot shows a terminal window titled "Run: SwapFirstnLastLetter x". The command prompt shows the execution of the Python script: `C:\Jenny\Python\venv\Scripts\python.exe C:\Jenny\Python\SwapFirstnLastLetter.py`. The user input is "jenny", and the output is "new string is yennj". The process finished with exit code 0.

### **Result**

The program was executed and the result was successfully obtained. Thus, CO1 was obtained.

## **Experiment No.: 9**

### **Aim**

Accept the radius from user and find area of circle.

### **CO1**

Understands basics of Python Programming language including input/output functions, operators, basic and collection data types.

### **Procedure**

```
# To calculate Area of a Circle #
```

```
# value of pi=3.14 #
```

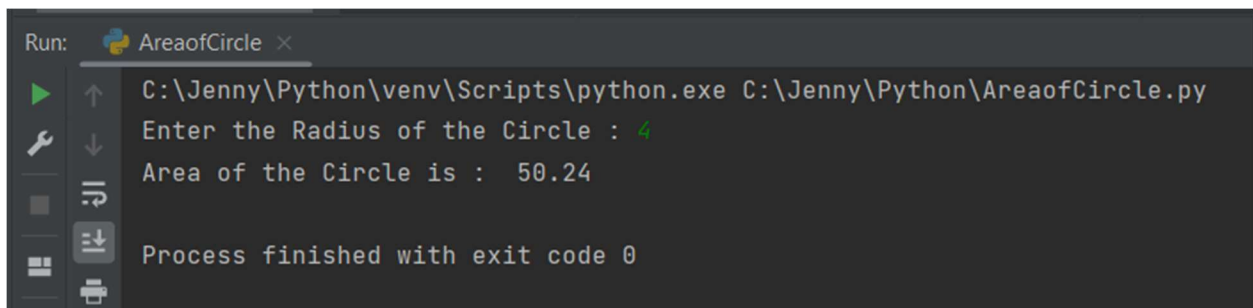
```
# Area=pi*(radius*radius) #
```

```
radius = int(input("Enter the Radius of the Circle : "))
```

```
area = 3.14*(radius*radius)
```

```
print("Area of the Circle is : ", area)
```

### **Output Screenshot**



```
Run: AreaofCircle x
C:\Jenny\Python\venv\Scripts\python.exe C:\Jenny\Python\AreaofCircle.py
Enter the Radius of the Circle : 4
Area of the Circle is : 50.24
Process finished with exit code 0
```

### **Result**

The program was executed and the result was successfully obtained. Thus, CO1 was obtained.

**Experiment No.: 10****Aim**

To find biggest of 3 numbers entered.

**CO1**

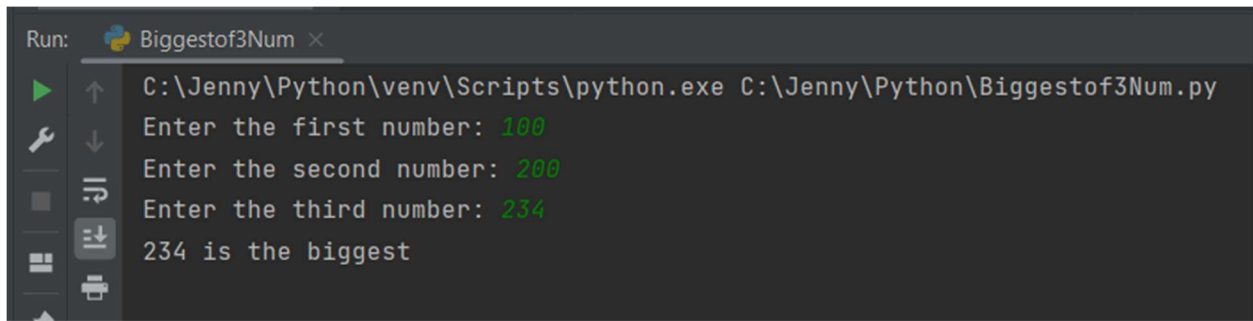
Understands basics of Python Programming language including input/output functions, operators, basic and collection data types.

**Procedure**

```
# biggest of 3 number
x = int(input("Enter the first number: "))
y = int(input("Enter the second number: "))
z = int(input("Enter the third number: "))

if x > y:
    if x > z:
        print("{} is the biggest".format(x))
    else:
        print("{} is the biggest".format(z))
elif y > z:
    print("{} is the biggest".format(y))
else:
    print("{} is the biggest".format(z))
```

## **Output Screenshot**



```
Run: Biggestof3Num x
C:\Jenny\Python\venv\Scripts\python.exe C:\Jenny\Python\Biggestof3Num.py
Enter the first number: 100
Enter the second number: 200
Enter the third number: 234
234 is the biggest
```

## **Result**

The program was executed and the result was successfully obtained. Thus, CO1 was obtained.



## **Experiment No.: 11**

### **Aim**

Accept a file name from user and print extension of that.

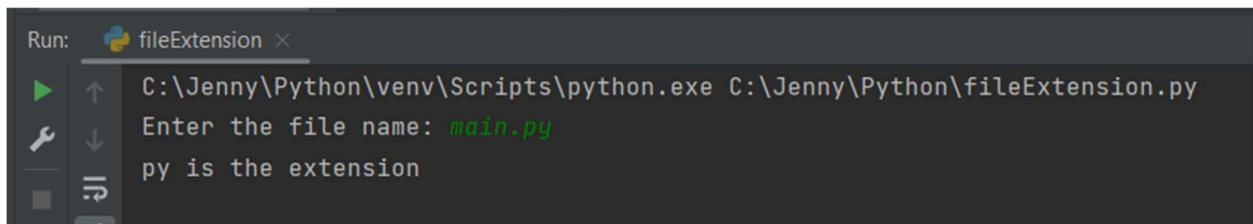
### **CO1**

Understands basics of Python Programming language including input/output functions, operators, basic and collection data types.

### **Procedure**

```
# To print extension of a file #  
x = input("Enter the file name: ")  
ex = x.split(".")  
print(ex[-1], "is the extension ")
```

### **Output Screenshot**

A screenshot of a Python IDE's 'Run' console. The title bar shows 'Run: fileExtension x'. The command prompt shows the execution of 'C:\Jenny\Python\venv\Scripts\python.exe C:\Jenny\Python\fileExtension.py'. The user input 'main.py' is shown in green, and the output 'py is the extension' is shown in white.

```
Run: fileExtension x  
C:\Jenny\Python\venv\Scripts\python.exe C:\Jenny\Python\fileExtension.py  
Enter the file name: main.py  
py is the extension
```

### **Result**

The program was executed and the result was successfully obtained. Thus, CO1 was obtained.

## **Experiment No.: 12**

### **Aim**

Create a list of colours from comma-separated colour names entered by user. Display first and last colours.

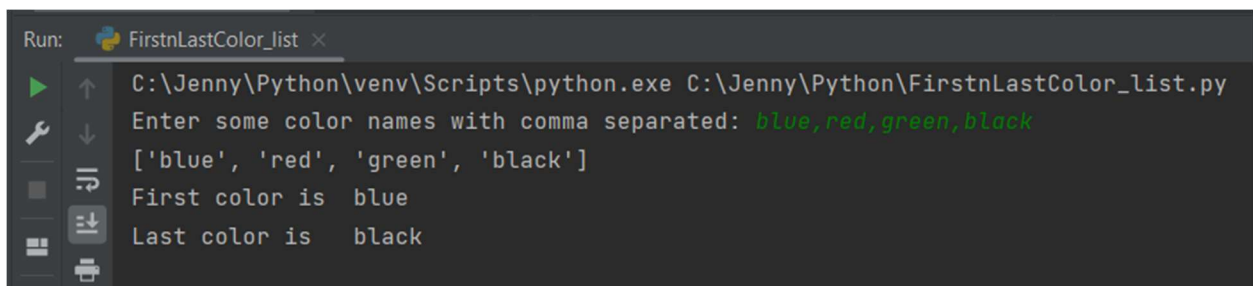
### **CO1**

Understands basics of Python Programming language including input/output functions, operators, basic and collection data types.

### **Procedure**

```
# last and first color name  
mylist = input("Enter some color names with comma separated: ").split(",")  
print(mylist)  
print("First color is ", mylist[0])  
print("Last color is ", mylist[-1])
```

### **Output Screenshot**



```
Run: FirstLastColor_list x  
C:\Jenny\Python\venv\Scripts\python.exe C:\Jenny\Python\FirstnLastColor_list.py  
Enter some color names with comma separated: blue,red,green,black  
['blue', 'red', 'green', 'black']  
First color is blue  
Last color is black
```

### **Result**

The program was executed and the result was successfully obtained. Thus, CO1 was obtained.

## **Experiment No.: 13**

### **Aim**

Accept an integer n and compute  $n+nn+nnn$

### **CO1**

Understands basics of Python Programming language including input/output functions, operators, basic and collection data types.

### **Procedure**

```
# n+n*n+n*n*n #
```

```
n = int(input("Enter a number: "))
```

```
p = n+n*n+n*n*n
```

```
print("Result:", p)
```

### **Output Screenshot**



```
Run: n+nn+nnn x
C:\Jenny\Python\venv\Scripts\python.exe C:\Jenny\Python\n+nn+nnn.py
Enter a number: 5
Result: 155
Process finished with exit code 0
```

### **Result**

The program was executed and the result was successfully obtained. Thus, CO1 was obtained.

## **Experiment No.: 14**

### **Aim**

Print out all colours from color-list1 not contained in color-list2.

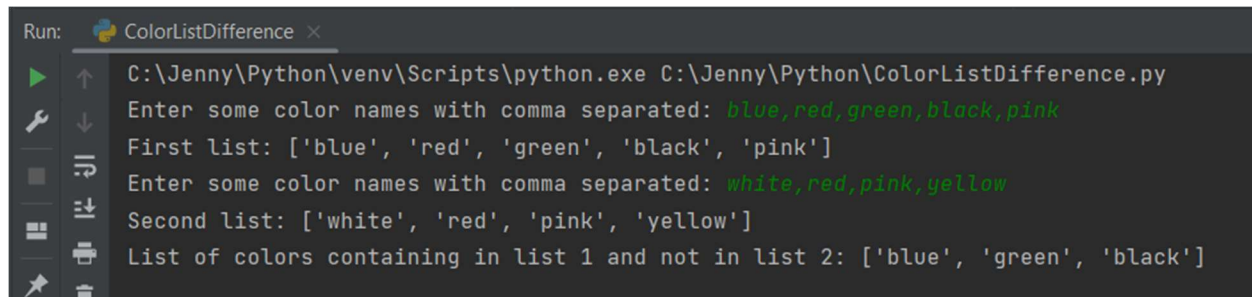
### **CO1**

Understands basics of Python Programming language including input/output functions, operators, basic and collection data types.

### **Procedure**

```
mylist1 = input("Enter some color names with comma separated: ").split(",")
print("First list:", mylist1)
mylist2 = input("Enter some color names with comma separated: ").split(",")
print("Second list:", mylist2)
color = set(mylist1).difference(set(mylist2))
print("List of colors containing in list 1 and not in list 2:", list(color))
```

### **Output Screenshot**

A screenshot of a Python IDE window titled 'Run: ColorListDifference'. The console output shows the following: 'Enter some color names with comma separated: blue,red,green,black,pink', 'First list: ['blue', 'red', 'green', 'black', 'pink']', 'Enter some color names with comma separated: white,red,pink,yellow', 'Second list: ['white', 'red', 'pink', 'yellow']', and 'List of colors containing in list 1 and not in list 2: ['blue', 'green', 'black']'.

```
Run: ColorListDifference ×
C:\Jenny\Python\venv\Scripts\python.exe C:\Jenny\Python\ColorListDifference.py
Enter some color names with comma separated: blue,red,green,black,pink
First list: ['blue', 'red', 'green', 'black', 'pink']
Enter some color names with comma separated: white,red,pink,yellow
Second list: ['white', 'red', 'pink', 'yellow']
List of colors containing in list 1 and not in list 2: ['blue', 'green', 'black']
```

### **Result**

The program was executed and the result was successfully obtained. Thus, CO1 was obtained.

## **Experiment No.: 15**

### **Aim**

Create a single string separated with space from two strings by swapping the character at position 1.

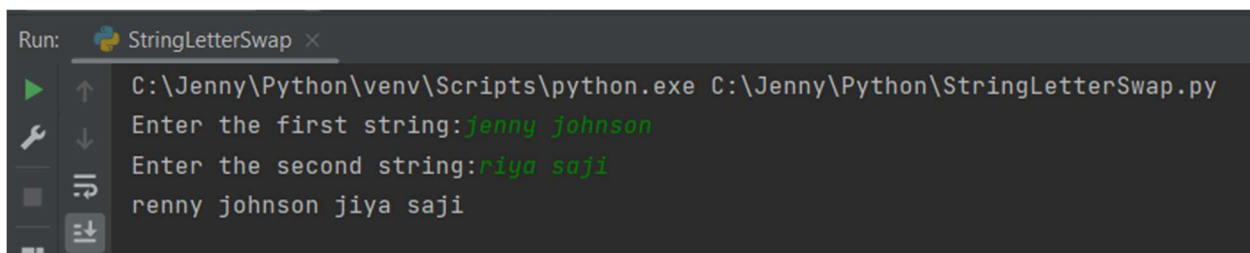
### **CO1**

Understands basics of Python Programming language including input/output functions, operators, basic and collection data types.

### **Procedure**

```
# exchanging first char of two string #  
a = input("Enter the first string:")  
a1 = a[0]  
b = input("Enter the second string:")  
b1 = b[0]  
ab = b1[0]+a[1:]  
ba = a1[0]+b[1:]  
print(ab + " " + ba)
```

### **Output Screenshot**



```
Run: StringLetterSwap ×  
C:\Jenny\Python\venv\Scripts\python.exe C:\Jenny\Python\StringLetterSwap.py  
Enter the first string:jenny johnson  
Enter the second string:riya saji  
renny johnson jiya saji
```

### **Result**

The program was executed and the result was successfully obtained. Thus, CO1 was obtained.

## **Experiment No.: 16**

### **Aim**

Sort dictionary in ascending and descending order.

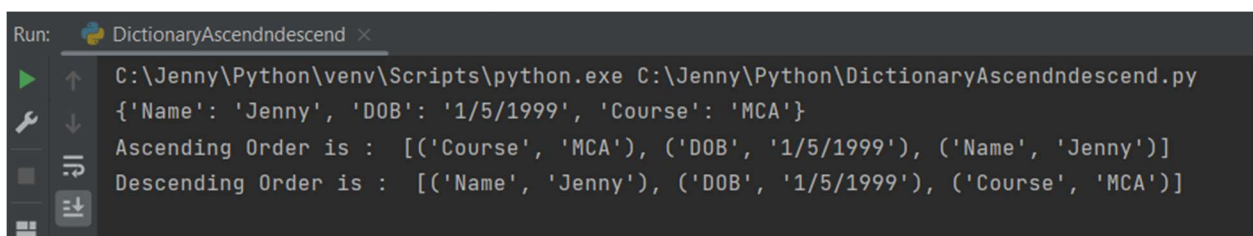
### **CO1**

Understands basics of Python Programming language including input/output functions, operators, basic and collection data types.

### **Procedure**

```
dict1 = {  
    "Name": "Jenny",  
    "DOB": "1/5/1999",  
    "Course": "MCA"  
}  
print(dict1)  
x = list(dict1.items())  
x.sort()  
print("Ascending Order is : ", x)  
y = list(dict1.items())  
y.sort(reverse=True)  
print("Descending Order is : ", y)
```

### **Output Screenshot**



```
Run: DictionaryAscendndescend ×  
C:\Jenny\Python\venv\Scripts\python.exe C:\Jenny\Python\DictionaryAscendndescend.py  
{'Name': 'Jenny', 'DOB': '1/5/1999', 'Course': 'MCA'}  
Ascending Order is : [('Course', 'MCA'), ('DOB', '1/5/1999'), ('Name', 'Jenny')]  
Descending Order is : [('Name', 'Jenny'), ('DOB', '1/5/1999'), ('Course', 'MCA')]
```

### **Result**

The program was executed and the result was successfully obtained. Thus, CO1 was obtained.

**Experiment No.: 17****Aim**

Merge two dictionaries.

**CO1**

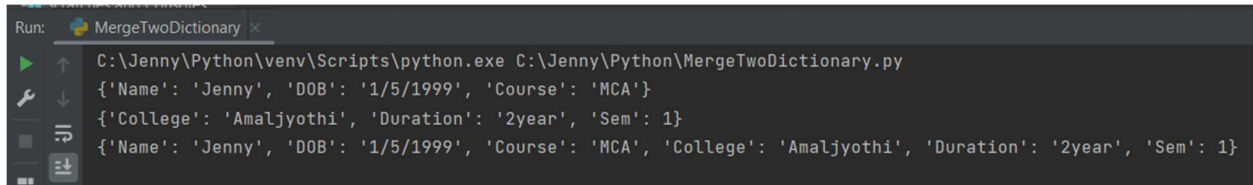
Understands basics of Python Programming language including input/output functions, operators, basic and collection data types.

**Procedure**

```
# merge two dictionary #
```

```
dict1 = {  
    "Name": "Jenny",  
    "DOB": "1/5/1999",  
    "Course": "MCA"  
}  
  
dict2 = {  
    "College": "Amaljyothi",  
    "Duration": "2year",  
    "Sem": 1  
}  
  
print(dict1)  
print(dict2)  
dict3 = dict1.copy()  
dict3.update(dict2)  
print(dict3)
```

## **Output Screenshot**



```
Run: MergeTwoDictionary x
C:\Jenny\Python\venv\Scripts\python.exe C:\Jenny\Python\MergeTwoDictionary.py
{'Name': 'Jenny', 'DOB': '1/5/1999', 'Course': 'MCA'}
{'College': 'Amaljyothi', 'Duration': '2year', 'Sem': 1}
{'Name': 'Jenny', 'DOB': '1/5/1999', 'Course': 'MCA', 'College': 'Amaljyothi', 'Duration': '2year', 'Sem': 1}
```

## **Result**

The program was executed and the result was successfully obtained. Thus, CO1 was obtained.



## **Experiment No.: 18**

### **Aim**

Find gcd of 2 numbers.

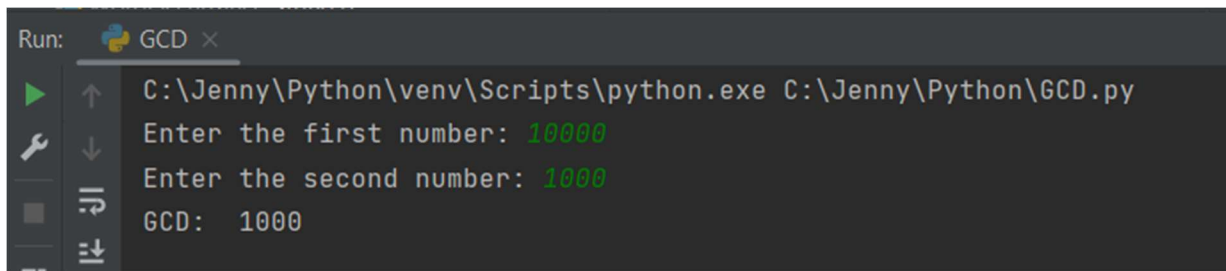
### **CO1**

Understands basics of Python Programming language including input/output functions, operators, basic and collection data types.

### **Procedure**

```
# GCD #  
  
a = int(input("Enter the first number: "))  
b = int(input("Enter the second number: "))  
  
i = 1  
while i <= a and i <= b:  
    if a % i == 0 and b % i == 0:  
        gcd = i  
        i = i+1  
print("GCD: ", gcd)
```

### **Output Screenshot**



```
Run: GCD ×  
C:\Jenny\Python\venv\Scripts\python.exe C:\Jenny\Python\GCD.py  
Enter the first number: 10000  
Enter the second number: 1000  
GCD: 1000
```

### **Result**

The program was executed and the result was successfully obtained. Thus, CO1 was obtained.

## **Experiment No.: 19**

### **Aim**

From a list of integers, create a list removing even numbers.

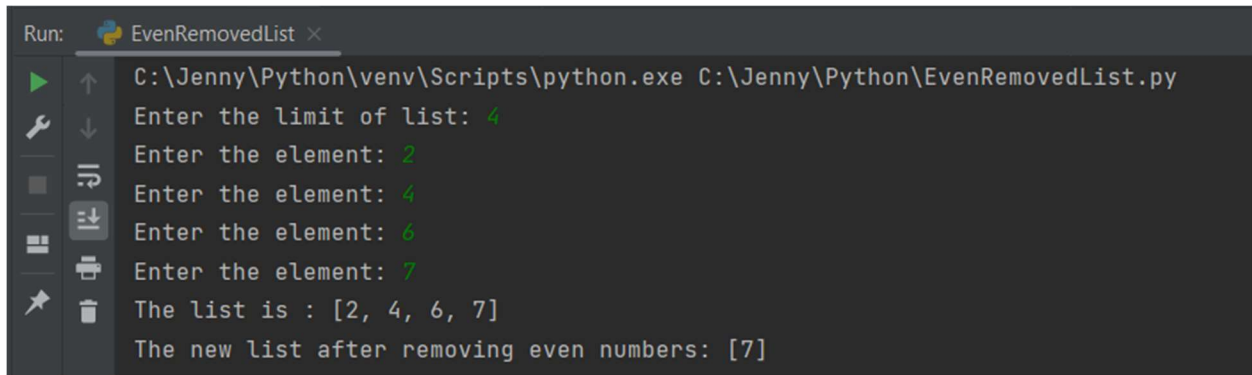
### **CO1**

Understands basics of Python Programming language including input/output functions, operators, basic and collection data types.

### **Procedure**

```
# even numbers removed list
list1 = []
newlist = []
li = int(input("Enter the limit of list: "))
for i in range(0, li, 1):
    element = int(input("Enter the element: "))
    list1.append(element)
print("The list is :", list1)
for i in range(len(list1)):
    if list1[i] % 2 == 0:
        continue
    else:
        newlist.append(list1[i])
print("The new list after removing even numbers:", newlist)
```

### **Output Screenshot**

The screenshot shows a Python IDE window titled 'EvenRemovedList'. The command prompt displays the following sequence of inputs and outputs: 'Enter the limit of list: 4', 'Enter the element: 2', 'Enter the element: 4', 'Enter the element: 6', 'Enter the element: 7', 'The list is : [2, 4, 6, 7]', and 'The new list after removing even numbers: [7]'. The program successfully filters out even numbers from the input list.

```
Run: EvenRemovedList x
C:\Jenny\Python\venv\Scripts\python.exe C:\Jenny\Python\EvenRemovedList.py
Enter the limit of list: 4
Enter the element: 2
Enter the element: 4
Enter the element: 6
Enter the element: 7
The list is : [2, 4, 6, 7]
The new list after removing even numbers: [7]
```

### **Result**

The program was executed and the result was successfully obtained. Thus, CO1 was obtained.

## **Experiment No.: 20**

### **Aim**

To find the factorial of a number.

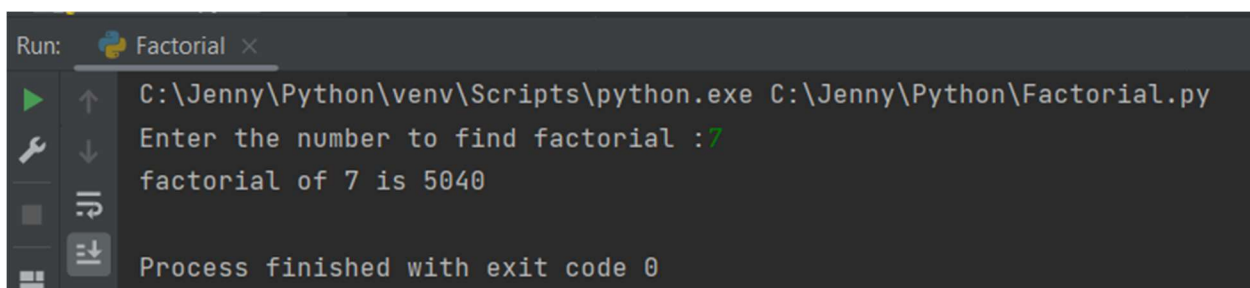
### **CO2**

Implement decision making, looping constructs and functions.

### **Procedure**

```
# factorial of a number  
a = int(input("Enter the number to find factorial :"))  
i = 1  
f = 1  
while i <= a:  
    f = f*i  
    i = i+1  
print("factorial of {} is".format(a), f)
```

### **Output Screenshot**



```
Run: Factorial x  
C:\Jenny\Python\venv\Scripts\python.exe C:\Jenny\Python\Factorial.py  
Enter the number to find factorial :7  
factorial of 7 is 5040  
Process finished with exit code 0
```

### **Result**

The program was executed and the result was successfully obtained. Thus, CO2 was obtained.

**Experiment No.: 21****Aim**

To generate Fibonacci series of N terms.

**CO2**

Implement decision making, looping constructs and functions.

**Procedure**

```
# fibonacci series #
```

```
li = int(input("Enter the limit: "))
```

```
a = 0
```

```
b = 1
```

```
c = 0
```

```
i = 0
```

```
while i <= li:
```

```
    print(a)
```

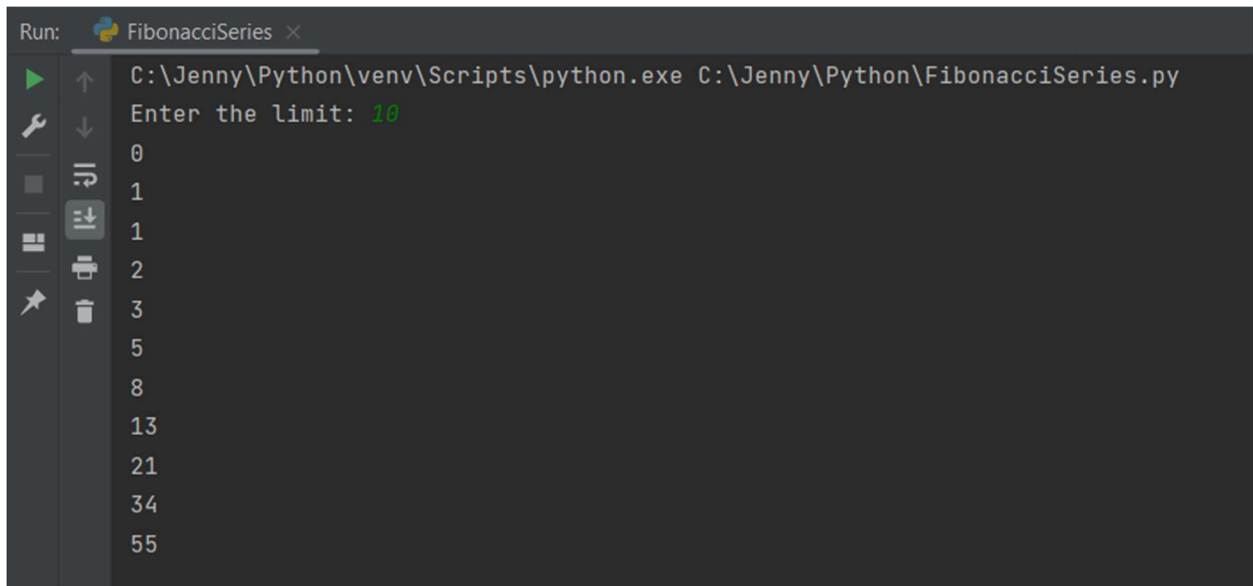
```
    c = a+b
```

```
    a = b
```

```
    b = c
```

```
    i = i+1
```

## **Output Screenshot**



The screenshot shows a terminal window titled "Run: FibonacciSeries". The command executed is `C:\Jenny\Python\venv\Scripts\python.exe C:\Jenny\Python\FibonacciSeries.py`. The program prompts "Enter the limit: 10". The output displays the Fibonacci series: 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55.

```
Run: FibonacciSeries x
C:\Jenny\Python\venv\Scripts\python.exe C:\Jenny\Python\FibonacciSeries.py
Enter the limit: 10
0
1
1
2
3
5
8
13
21
34
55
```

## **Result**

The program was executed and the result was successfully obtained. Thus, CO2 was obtained.

**Experiment No.: 22****Aim**

To find the sum of all items in a list.

**CO2**

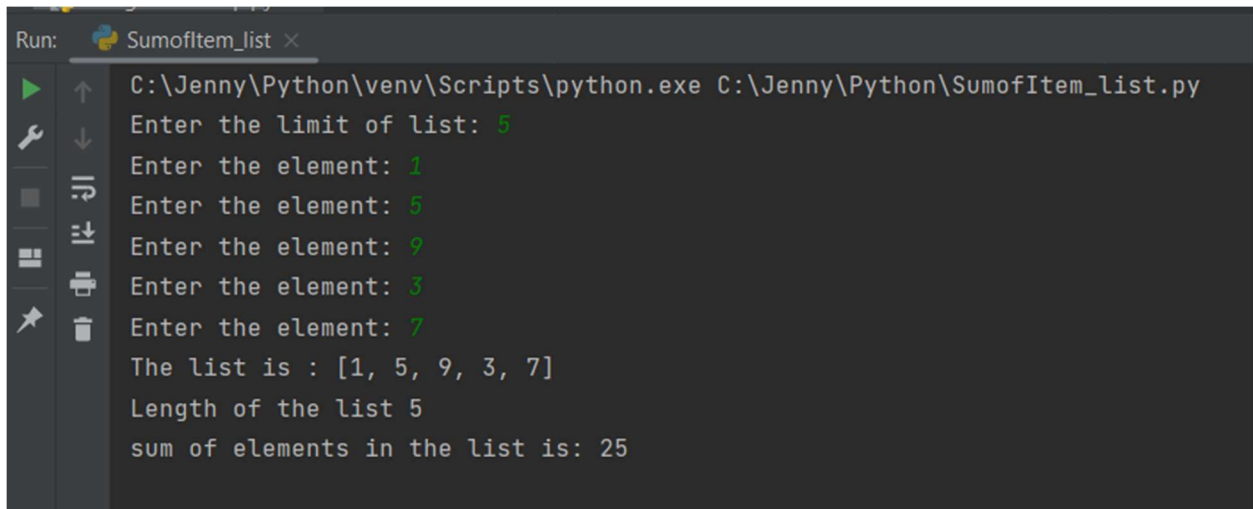
Implement decision making, looping constructs and functions

**Procedure**

# sum of elements in a list #

```
list1 = []
li = int(input("Enter the limit of list: "))
for i in range(0, li, 1):
    element = int(input("Enter the element: "))
    list1.append(element)
print("The list is :", list1)
x = len(list1)
print("Length of the list", x)
s = 0
for i in range(x):
    s = s+list1[i]
print("sum of elements in the list is:", s)
```

## Output Screenshot

A screenshot of a terminal window titled 'SumofItem\_list'. The window shows the execution of a Python script. The prompt 'C:\Jenny\Python\venv\Scripts\python.exe C:\Jenny\Python\SumofItem\_list.py' is at the top. Below it, the program prompts the user for the limit of the list (5), then for five elements (1, 5, 9, 3, 7). It then displays the resulting list [1, 5, 9, 3, 7], its length (5), and the sum of its elements (25).

```
Run: SumofItem_list x
C:\Jenny\Python\venv\Scripts\python.exe C:\Jenny\Python\SumofItem_list.py
Enter the limit of list: 5
Enter the element: 1
Enter the element: 5
Enter the element: 9
Enter the element: 3
Enter the element: 7
The list is : [1, 5, 9, 3, 7]
Length of the list 5
sum of elements in the list is: 25
```

## Result

The program was executed and the result was successfully obtained. Thus, CO2 was obtained.

## **Experiment No.: 23**

### **Aim**

To generate a list of four-digit numbers in a given range with all their digits even and the number is a perfect square.

### **CO2**

Implement decision making, looping constructs and functions

### **Procedure**

```
print("Four digit number with all their digits even and the number is a perfect square")
```

```
for i in range(1000, 10000):
```

```
    for j in range(32, 100):
```

```
        if i == j*j:
```

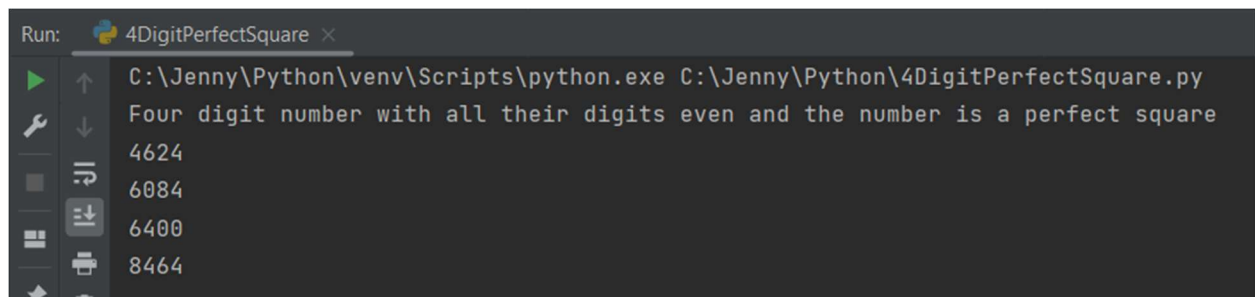
```
            string = str(i)
```

```
            if (int(string[0]) % 2 == 0) and (int(string[1]) % 2 == 0) and \
```

```
                (int(string[2]) % 2 == 0) and (int(string[3]) % 2 == 0):
```

```
                print(i)
```

### **Output Screenshot**



```
Run: 4DigitPerfectSquare x
C:\Jenny\Python\venv\Scripts\python.exe C:\Jenny\Python\4DigitPerfectSquare.py
Four digit number with all their digits even and the number is a perfect square
4624
6084
6400
8464
```

### **Result**

The program was executed and the result was successfully obtained. Thus, CO2 was obtained.



## **Experiment No.: 24**

### **Aim**

Display the given pyramid with step number accepted from user. Eg: N=4

```
1
2 4
3 6 9
4 8 12 16
```

### **CO2**

Implement decision making, looping constructs and functions.

### **Procedure**

```
n = int(input("Enter the limit: "))
```

```
for i in range(1, n+1):
```

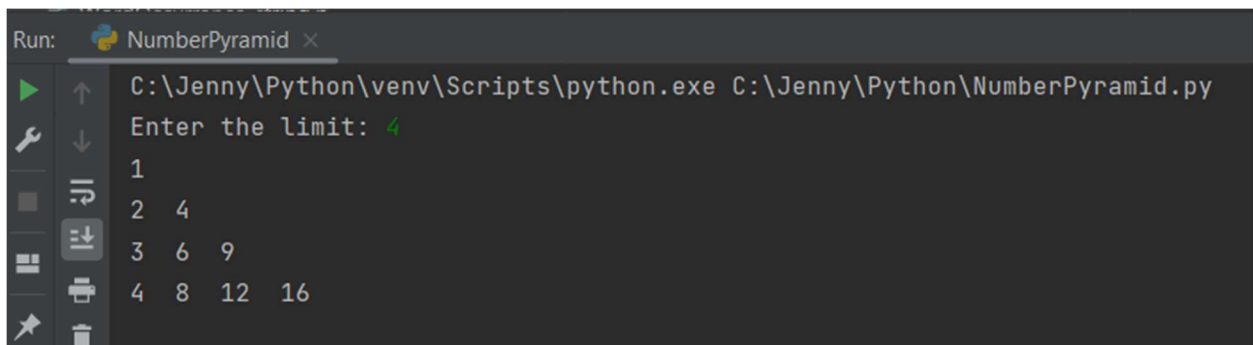
```
    for j in range(1, i+1):
```

```
        s = j*i
```

```
        print(s, " ", end="")
```

```
    print("")
```

### **Output Screenshot**



```
Run: NumberPyramid x
C:\Jenny\Python\venv\Scripts\python.exe C:\Jenny\Python\NumberPyramid.py
Enter the limit: 4
1
2 4
3 6 9
4 8 12 16
```

### **Result**

The program was executed and the result was successfully obtained. Thus, CO2 was obtained.

## **Experiment No.: 25**

### **Aim**

Count the number of characters (character frequency) in a string.

### **CO2**

Implement decision making, looping constructs and functions

### **Procedure**

```
n = input("Enter the String: ")
```

```
s = {}
```

```
for i in n:
```

```
    if i in s:
```

```
        s[i] = s[i]+1
```

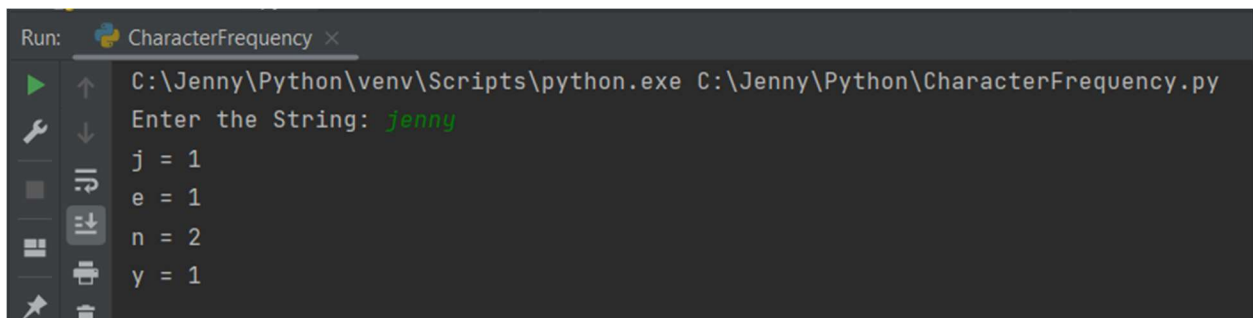
```
    else:
```

```
        s[i] = 1
```

```
for m, s in s.items():
```

```
    print(m, "=", s)
```

### **Output Screenshot**



```
Run: CharacterFrequency x
C:\Jenny\Python\venv\Scripts\python.exe C:\Jenny\Python\CharacterFrequency.py
Enter the String: jenny
j = 1
e = 1
n = 2
y = 1
```

### **Result**

The program was executed and the result was successfully obtained. Thus, CO2 was obtained.

## **Experiment No.: 26**

### **Aim**

Add 'ing' at the end of a given string. If it already ends with 'ing', then add 'ly'.

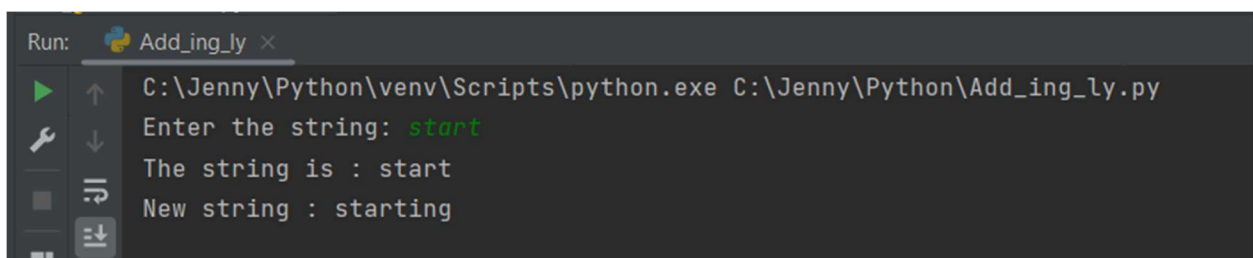
### **CO2**

Implement decision making, looping constructs and functions

### **Procedure**

```
# adding 'ing' or 'ly' #  
s = str(input("Enter the string: "))  
print("The string is :", s)  
if s[-3:] == 'ing':  
    print("New string :", s+"ly")  
else:  
    print("New string :", s+"ing")
```

### **Output Screenshot**



```
Run: Add_ing_ly x  
C:\Jenny\Python\venv\Scripts\python.exe C:\Jenny\Python\Add_ing_ly.py  
Enter the string: start  
The string is : start  
New string : starting
```

### **Result**

The program was executed and the result was successfully obtained. Thus, CO2 was obtained.

**Experiment No.: 27****Aim**

Accept a list of words and return length of longest word.

**CO2**

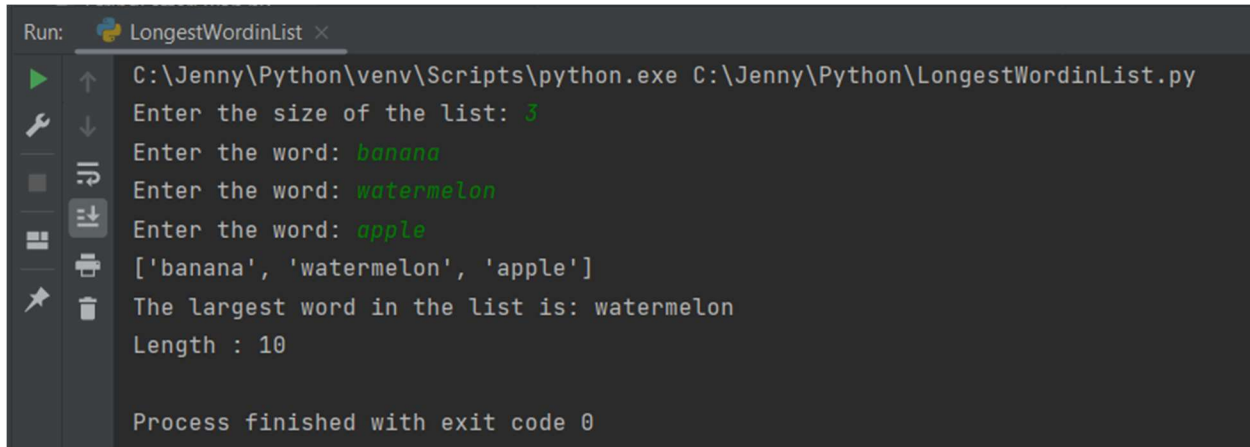
Implement decision making, looping constructs and functions

**Procedure**

# length of the largest word in the list #

```
lis = []
li = int(input("Enter the size of the list: "))
for i in range(li):
    w = str(input("Enter the word: "))
    lis.append(w)
print(lis)
t = lis[1]
m = len(lis[1])
for i in range(li):
    if m < len(lis[i]):
        m = len(lis[i])
        t = lis[i]
    else:
        i = i+1
print("The largest word in the list is:", t)
print("Length :", m)
```

## Output Screenshot

A screenshot of a terminal window titled 'LongestWordinList'. The window shows the execution of a Python script. The command prompt is 'C:\Jenny\Python\venv\Scripts\python.exe C:\Jenny\Python\LongestWordinList.py'. The script prompts the user to 'Enter the size of the list: 3', then 'Enter the word: banana', 'Enter the word: watermelon', and 'Enter the word: apple'. It then displays the list ['banana', 'watermelon', 'apple'] and the output 'The largest word in the list is: watermelon' with a 'Length : 10'. The process finished with exit code 0.

```
Run: LongestWordinList x
C:\Jenny\Python\venv\Scripts\python.exe C:\Jenny\Python\LongestWordinList.py
Enter the size of the list: 3
Enter the word: banana
Enter the word: watermelon
Enter the word: apple
['banana', 'watermelon', 'apple']
The largest word in the list is: watermelon
Length : 10

Process finished with exit code 0
```

## Result

The program was executed and the result was successfully obtained. Thus, CO2 was obtained.

**Experiment No.: 28****Aim**

Construct following pattern using nested loop

```
*  
  
* *  
  
* * *  
  
* * * *  
  
* * * * *  
  
* * * *  
  
* * *  
  
* *  
  
*
```

**CO2**

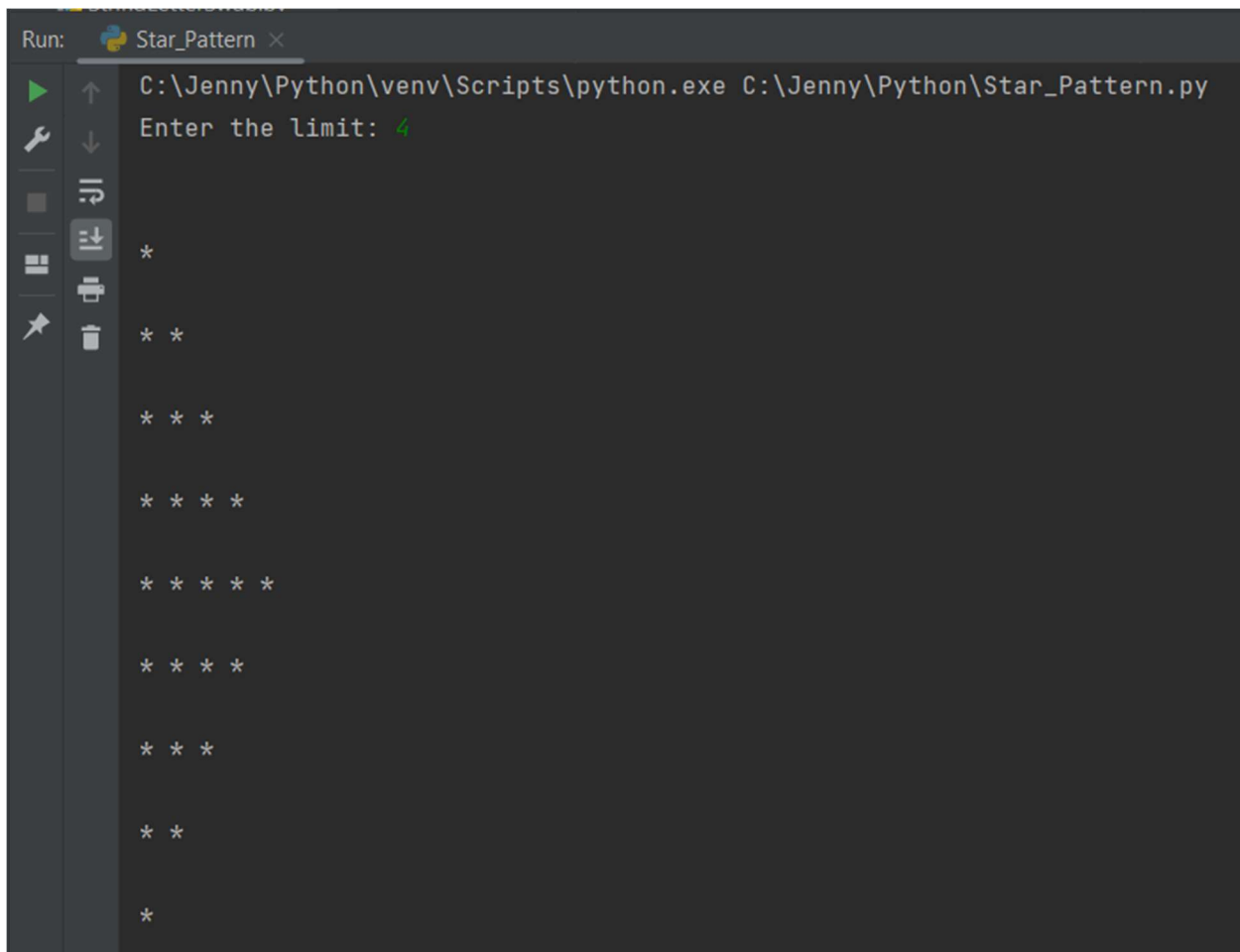
Implement decision making, looping constructs and functions.

**Procedure**

```
# * pattern #  
  
limit = int(input("Enter the limit: "))  
  
i = 0  
while i <= limit:  
    j = 0  
    while j < i:  
        print("*", end=" ")  
        j = j+1  
    i = i+1  
    print("\n")
```

```
i = 0
while i <= limit:
    j = limit
    while j >= i:
        print("*", end=" ")
        j = j-1
    print("\n")
    i = i+1
```

### Output Screenshot



The screenshot shows a Python IDE window titled "Star\_Pattern". The terminal output displays the execution of the program. The user enters the limit "4". The program prints a diamond-shaped star pattern:

```
Enter the limit: 4
*
* *
* * *
* * * *
* * * * *
* * * *
* * *
* *
*
```

### Result

The program was executed and the result was successfully obtained. Thus, CO2 was obtained.

## **Experiment No.: 29**

### **Aim**

Generate all factors of a number.

### **CO2**

Implement decision making, looping constructs and functions

### **Procedure**

# Factors of a number #

```
n = int(input("Enter the number: "))
```

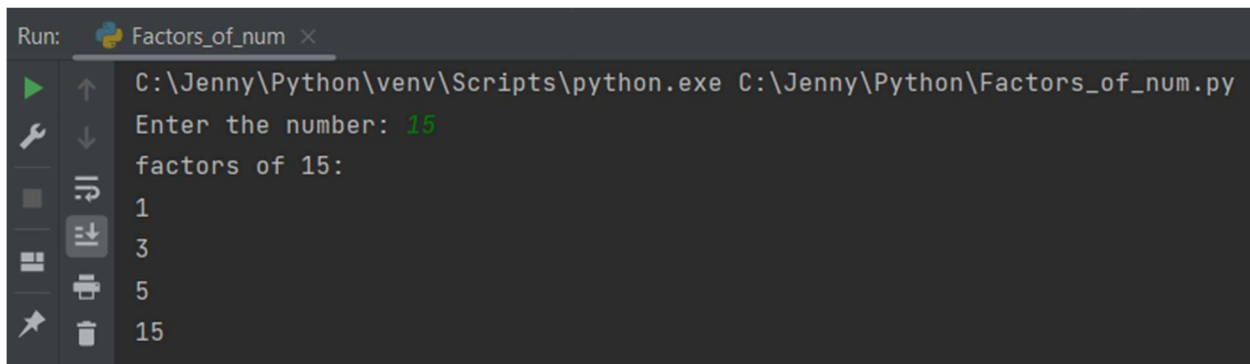
```
print("factors of {}".format(n))
```

```
for i in range(1, n+1, 1):
```

```
    if n % i == 0:
```

```
        print(i)
```

### **Output Screenshot**



```
Run: Factors_of_num x
C:\Jenny\Python\venv\Scripts\python.exe C:\Jenny\Python\Factors_of_num.py
Enter the number: 15
factors of 15:
1
3
5
15
```

### **Result**

The program was executed and the result was successfully obtained. Thus, CO2 was obtained.



**Experiment No.: 30****Aim**

Write lambda functions to find area of square, rectangle and triangle.

**CO2**

Implement decision making, looping constructs and functions

**Procedure**

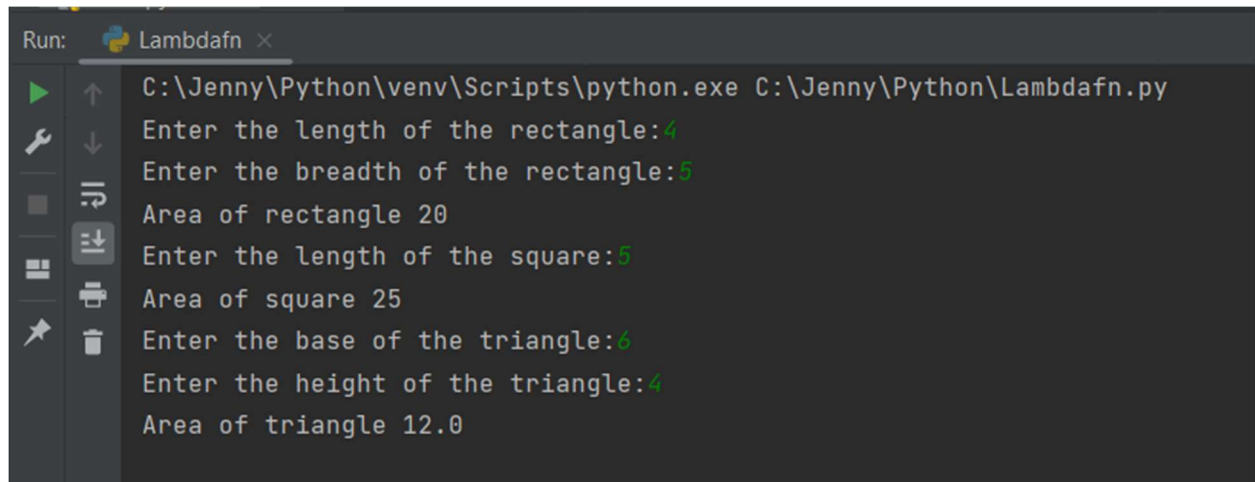
# using lambda function

```
l = int(input("Enter the length of the rectangle:"))
b = int(input("Enter the breadth of the rectangle:"))
area = lambda l, b: l * b
print("Area of rectangle", area(l, b))

s = int(input("Enter the length of the square:"))
areas = lambda s: s * s
print("Area of square", areas(s))

ba = int(input("Enter the base of the triangle:"))
h = int(input("Enter the height of the triangle:"))
ar = lambda ba, h: 0.5 * ba * h
print("Area of triangle", ar(ba, h))
```

## Output Screenshot



The screenshot shows a terminal window titled 'Run: Lambdafn'. The command executed is 'C:\Jenny\Python\venv\Scripts\python.exe C:\Jenny\Python\Lambdafn.py'. The output shows the program prompting for inputs and calculating areas for a rectangle, a square, and a triangle.

```
Run: Lambdafn ×
C:\Jenny\Python\venv\Scripts\python.exe C:\Jenny\Python\Lambdafn.py
Enter the length of the rectangle:4
Enter the breadth of the rectangle:5
Area of rectangle 20
Enter the length of the square:5
Area of square 25
Enter the base of the triangle:6
Enter the height of the triangle:4
Area of triangle 12.0
```

## Result

The program was executed and the result was successfully obtained. Thus, CO2 was obtained.

## **Experiment No.: 31**

### **Aim**

To subtract five days from the current date.

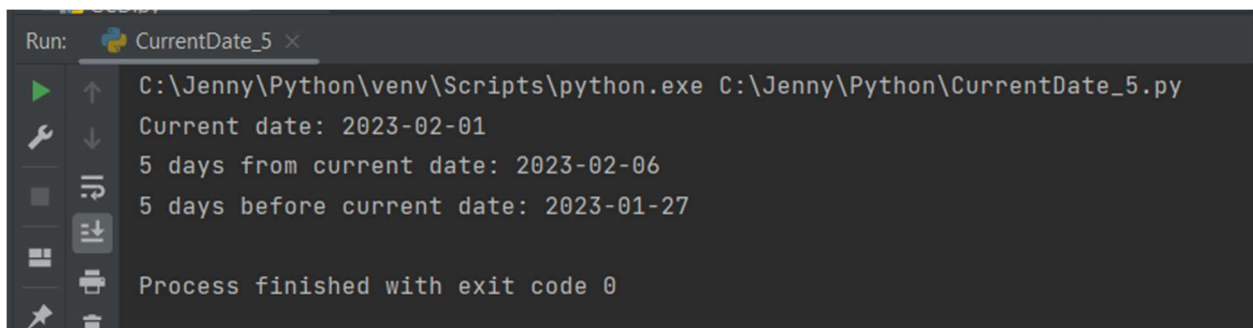
### **CO3**

Design modules and packages - built in and user defined packages.

### **Procedure**

```
from datetime import date, timedelta  
dt = date.today() + timedelta(5)  
dt1 = date.today() - timedelta(5)  
print('Current date:', date.today())  
print('5 days from current date:', dt)  
print('5 days before current date:', dt1)
```

### **Output Screenshot**



```
Run: CurrentDate_5 ×  
C:\Jenny\Python\venv\Scripts\python.exe C:\Jenny\Python\CurrentDate_5.py  
Current date: 2023-02-01  
5 days from current date: 2023-02-06  
5 days before current date: 2023-01-27  
Process finished with exit code 0
```

### **Result**

The program was executed and the result was successfully obtained. Thus, CO3 was obtained.

**Experiment No.: 32****Aim**

Create a package graphics with modules rectangle, circle and sub-package 3D-graphics with modules cuboid and sphere. Include methods to find area and perimeter of respective figures in each module. Write programs that finds area and perimeter of figures by different importing statements.

**CO3**

Design modules and packages - built in and user defined packages.

**Procedure**rectangle.py

```
def area(length, breadth):  
    return length * breadth  
  
def perimeter(length, breadth):  
    return 2 * (length + breadth)
```

circle.py

```
def area(radius):  
    return radius * radius * 3.14
```

```
def perimeter(radius):  
    return 2 * 3.14 * radius
```

cuboid.py

```
def area(length, width, height):  
    return 2*(length*width + width*height + height*length)  
  
def perimeter(length, width, height):  
    return 4 * (length + width + height)
```

sphere.py

```
def area(radius):
```

```
    return radius * radius * 3.14 * 4
```

```
def perimeter(radius):
```

```
    return 6.2832 * radius
```

main.py

```
from graphics.rectangle import area, perimeter
```

```
length = int(input("Enter the Length : "))
```

```
breadth = int(input("Enter the Breadth : "))
```

```
radius = int(input("Enter the Radius :"))
```

```
width = int(input("Enter the Width : "))
```

```
height = int(input("Enter the Height :"))
```

```
area_rect = area(length, breadth)
```

```
print("\nThe Area of Rectangle is : ", area_rect)
```

```
peri_rect = perimeter(length, breadth)
```

```
print("The Perimeter of Rectangle is : ", peri_rect)
```

```
from graphics.circle import area, perimeter
```

```
area_cir = area(radius)
```

```
print("\nThe Area of Circle is : ", area_cir)
```

```
peri_cir = perimeter(radius)
```

```
print("The Perimeter of Circle is : ", peri_cir)
```

```
from graphics.graphics_3d.sphere import area, perimeter
```

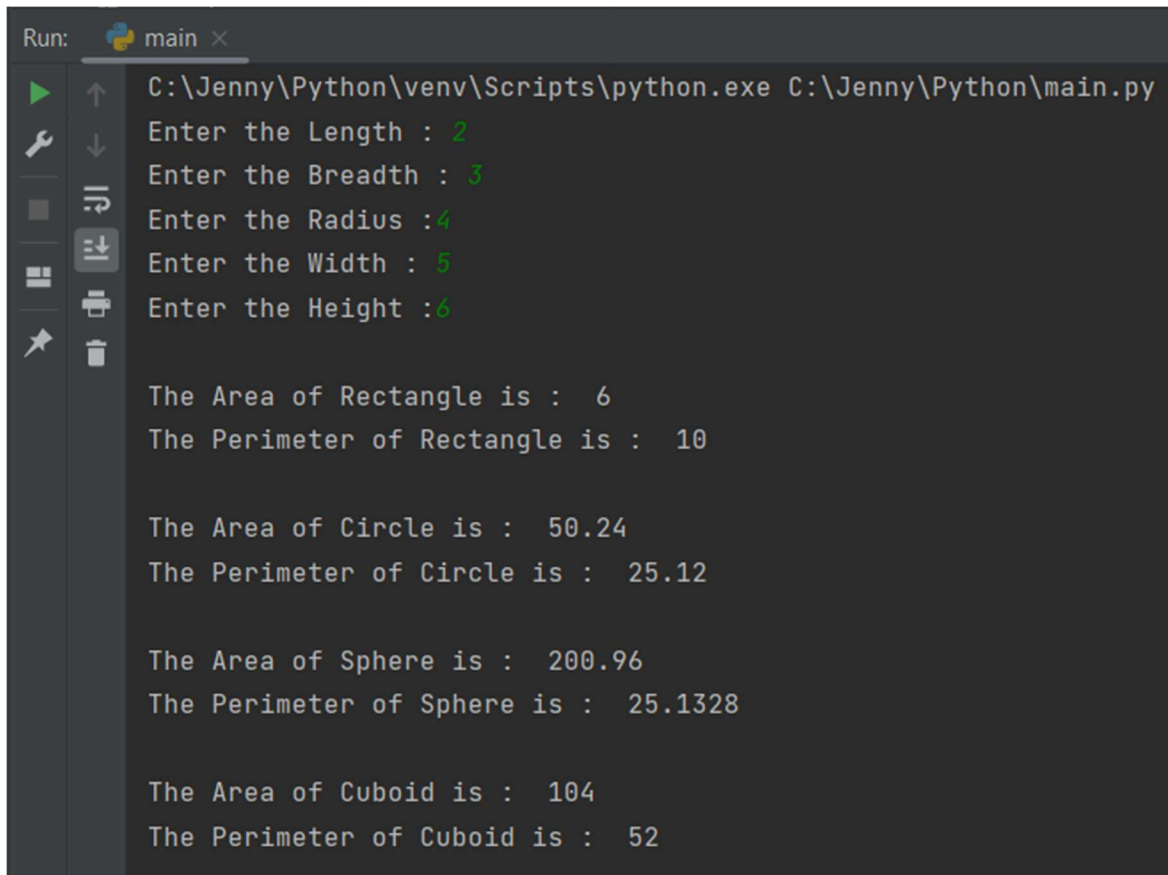
```
area_sph = area(radius)
```

```
print("\nThe Area of Sphere is : ", area_sph)
```

```
peri_sph = perimeter(radius)
print("The Perimeter of Sphere is : ", peri_sph)
from graphics.graphics_3d.cuboid import area, perimeter

area_cub = area(length, width, height)
print("\nThe Area of Cuboid is : ", area_cub)
peri_cub = perimeter(length, width, height)
print("The Perimeter of Cuboid is : ", peri_cub)
```

## Output Screenshot



```
Run: main x
C:\Jenny\Python\venv\Scripts\python.exe C:\Jenny\Python\main.py
Enter the Length : 2
Enter the Breadth : 3
Enter the Radius : 4
Enter the Width : 5
Enter the Height : 6

The Area of Rectangle is : 6
The Perimeter of Rectangle is : 10

The Area of Circle is : 50.24
The Perimeter of Circle is : 25.12

The Area of Sphere is : 200.96
The Perimeter of Sphere is : 25.1328

The Area of Cuboid is : 104
The Perimeter of Cuboid is : 52
```

## Result

The program was executed and the result was successfully obtained. Thus, CO3 was obtained.

**Experiment No.: 33****Aim**

Create Rectangle class with attributes length and breadth and methods to find area and perimeter. Compare two Rectangle objects by their area.

**CO4**

Implement object-oriented programming and exception handling.

**Procedure**

```
class Rectangle:
```

```
    def __init__(self, length, breadth):
```

```
        self.length = length
```

```
        self.breadth = breadth
```

```
    def area(self):
```

```
        return self.length * self.breadth
```

```
    def perimeter(self):
```

```
        return 2 * (self.length + self.breadth)
```

```
l1 = float(input("Enter length of rectangle 1 : "))
```

```
b1 = float(input("Enter breadth of rectangle 1 : "))
```

```
l2 = float(input("Enter length of rectangle 2 : "))
```

```
b2 = float(input("Enter breadth of rectangle 2 : "))
```

```
rect1 = Rectangle(l1, b1)
```

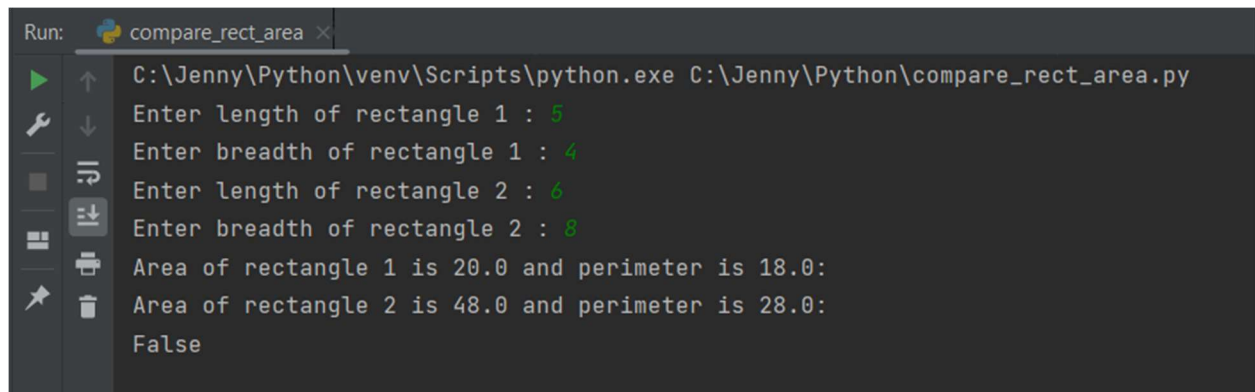
```
rect2 = Rectangle(l2, b2)
```

```
print("Area of rectangle 1 is {} and perimeter is {}".format(rect1.area(), rect1.perimeter()))
```

```
print("Area of rectangle 2 is {} and perimeter is {}".format(rect2.area(), rect2.perimeter()))
```

```
print(rect1.area() > rect2.area())
```

## **Output Screenshot**



```
Run: compare_rect_area x
C:\Jenny\Python\venv\Scripts\python.exe C:\Jenny\Python\compare_rect_area.py
Enter length of rectangle 1 : 5
Enter breadth of rectangle 1 : 4
Enter length of rectangle 2 : 6
Enter breadth of rectangle 2 : 8
Area of rectangle 1 is 20.0 and perimeter is 18.0:
Area of rectangle 2 is 48.0 and perimeter is 28.0:
False
```

## **Result**

The program was executed and the result was successfully obtained. Thus, CO4 was obtained.



**Experiment No.: 34****Aim**

Create a Bank account with members account number, name, type of account and balance. Write constructor and methods to deposit at the bank and withdraw an amount from the bank.

**CO4**

Implement object-oriented programming and exception handling.

**Procedure**

```
def details():
```

```
    name = input("Enter your name: ")
```

```
    number = int(input("Enter your account number: "))
```

```
    ac_type = int(input("1.Fixed\n2.Savings\n3.Current\nEnter your account type: "))
```

```
    print("Name:", name)
```

```
    print("Account number:", number)
```

```
    print("Account Type:", ac_type)
```

```
class BankAccount:
```

```
    def __init__(self):
```

```
        self.balance = 0
```

```
        print("Hello!! welcome to the Deposit and Withdrawal machine")
```

```
    def deposit(self):
```

```
        amount = float(input("Enter the amount to be deposited: "))
```

```
        self.balance += amount
```

```
        print("\n Amount Deposited:", amount)
```

```
def withdraw(self):

amount = float(input("Enter amount to withdraw: "))

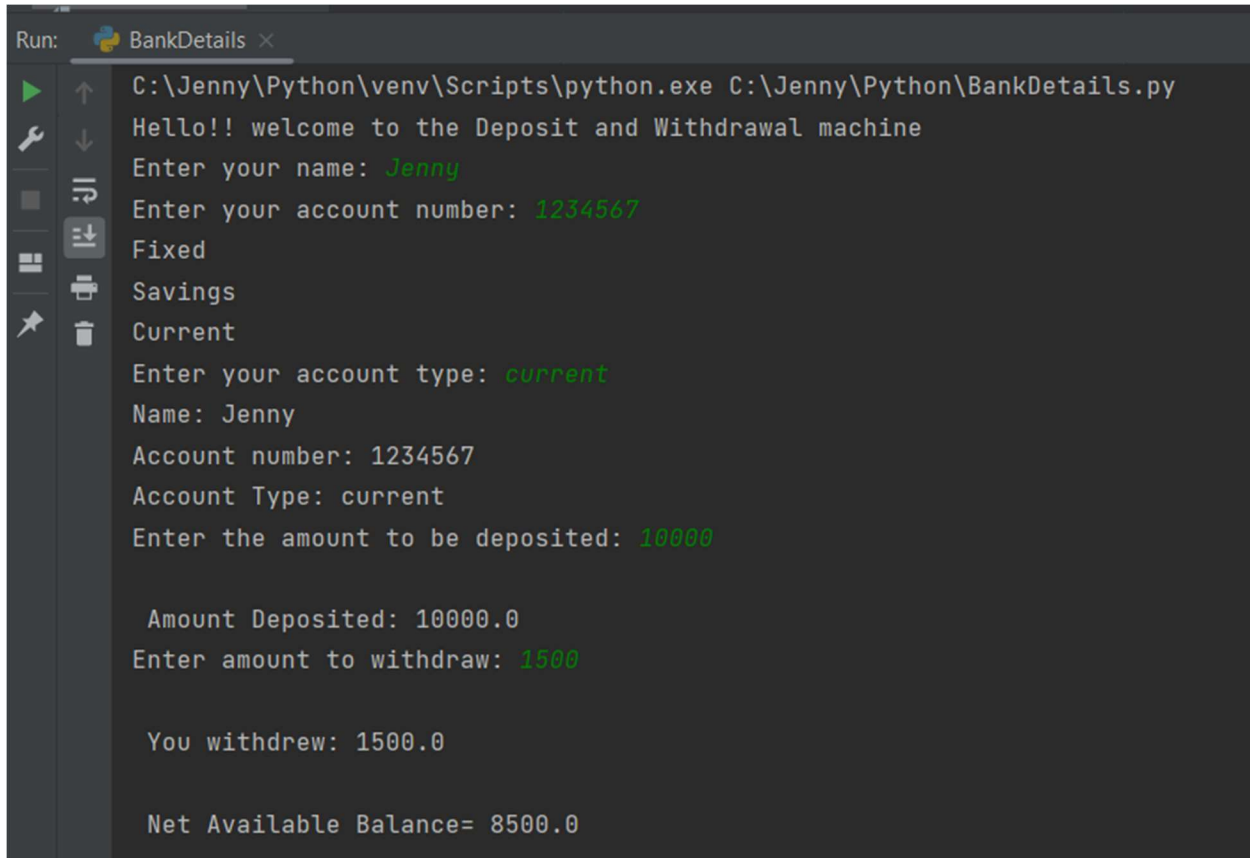
    if self.balance >= amount:
        self.balance -= amount
        print("\n You withdrew:", amount)
    else:
        print("\n Insufficient Balance")


def display(self):
    print("\n Net Available Balance=", self.balance)


s = BankAccount()
```

```
details()
s.deposit()
s.withdraw()
s.display()
```

## Output Screenshot



```
Run: BankDetails x
C:\Jenny\Python\venv\Scripts\python.exe C:\Jenny\Python\BankDetails.py
Hello!! welcome to the Deposit and Withdrawal machine
Enter your name: Jenny
Enter your account number: 1234567
Fixed
Savings
Current
Enter your account type: current
Name: Jenny
Account number: 1234567
Account Type: current
Enter the amount to be deposited: 10000

Amount Deposited: 10000.0
Enter amount to withdraw: 1500

You withdrew: 1500.0

Net Available Balance= 8500.0
```

## Result

The program was executed and the result was successfully obtained. Thus, CO4 was obtained.

**Experiment No.: 35****Aim**

Create a class Rectangle with private attributes length and width. Overload '<' operator to compare the area of 2 rectangles.

**CO4**

Implement object-oriented programming and exception handling.

**Procedure**

class Rectangle:

```
    def __init__(self, length, width):
```

```
        self.__length = length
```

```
        self.__width = width
```

```
        self.area = length*width
```

```
    def __lt__(self, other):
```

```
        if self.area < other.area:
```

```
            return "Rectangle 1 is smaller in Area"
```

```
        else:
```

```
            return "Rectangle 2 is smaller in Area"
```

```
l1 = int(input("Enter the length of rectangle 1 : "))
```

```
b1 = int(input("Enter the breadth of rectangle 1 : "))
```

```
l2 = int(input("Enter the length of rectangle 2 : "))
```

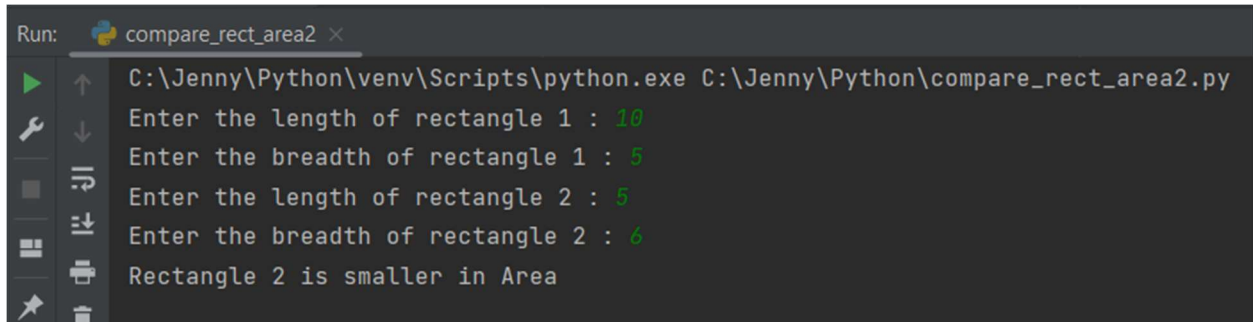
```
b2 = int(input("Enter the breadth of rectangle 2 : "))
```

```
r1 = Rectangle(l1, b1)
```

```
r2 = Rectangle(l2, b2)
```

```
print(r1 < r2)
```

## **Output Screenshot**



```
Run: compare_rect_area2 ×
C:\Jenny\Python\venv\Scripts\python.exe C:\Jenny\Python\compare_rect_area2.py
Enter the length of rectangle 1 : 10
Enter the breadth of rectangle 1 : 5
Enter the length of rectangle 2 : 5
Enter the breadth of rectangle 2 : 6
Rectangle 2 is smaller in Area
```

## **Result**

The program was executed and the result was successfully obtained. Thus, CO4 was obtained.

**Experiment No.: 36****Aim**

Create a class Time with private attributes hour, minute and second. Overload '+' operator to find sum of 2 time.

**CO4**

Implement object-oriented programming and exception handling.

**Procedure**

class Time:

```
def __init__(self, hour, minute, second):
```

```
    self.__hour = hour
```

```
    self.__minute = minute
```

```
    self.__second = second
```

```
def __add__(self, other):
```

```
    t3.__hour = t1.__hour + t2.__hour
```

```
    t3.__minute = t1.__minute + t2.__minute
```

```
    t3.__second = t1.__second + t2.__second
```

```
    if t3.__second > 59:
```

```
        t3.__second -= 60
```

```
        t3.__minute = t3.__minute + 1
```

```
    if t3.__minute > 59:
```

```
        t3.__minute -= 60
```

```
        t3.__hour = t3.__hour + 1
```

```
    return str(t3.__hour) + ':' + str(t3.__minute) + ':' + str(t3.__second)
```

```
def __lt__(self, other):
```

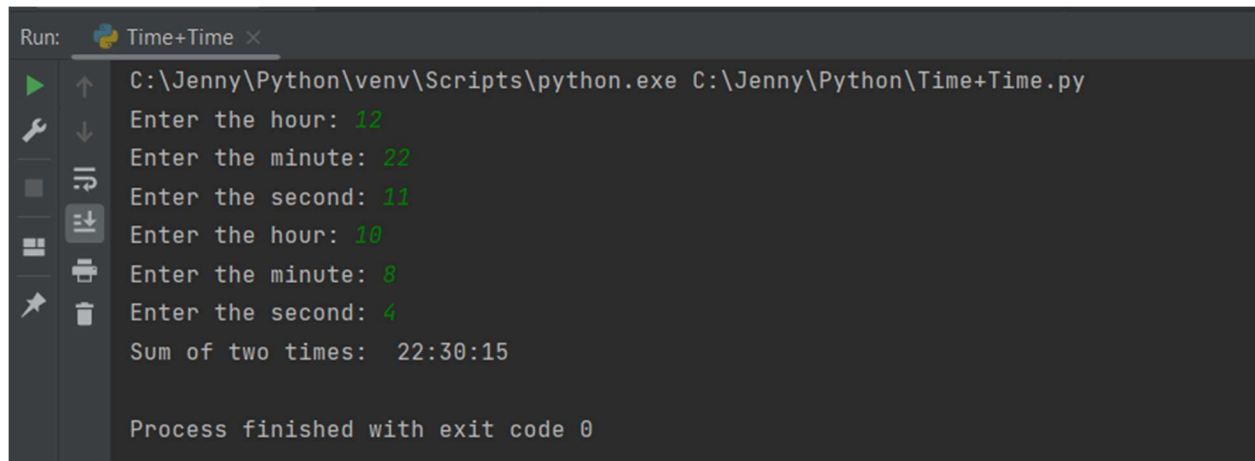
```
    if self.__hour < other.__hour:
```

```
return "True"

elif self.__hour == other.__hour:
    if self.__minute < other.__minute:
        return "True"
    elif self.__minute == other.__minute:
        if self.__second < other.__second:
            return "True"
        else:
            return "False"
    else:
        return "False"
```

```
h1 = int(input("Enter the hour: "))
m1 = int(input("Enter the minute: "))
s1 = int(input("Enter the second: "))
h2 = int(input("Enter the hour: "))
m2 = int(input("Enter the minute: "))
s2 = int(input("Enter the second: "))
t1 = Time(h1, m1, s1)
t2 = Time(h2, m2, s2)
t3 = Time(0, 0, 0)
print("Sum of two times: ", t1 + t2)
```

## **Output Screenshot**



The screenshot shows a terminal window titled 'Run: Time+Time x'. The command executed is 'C:\Jenny\Python\venv\Scripts\python.exe C:\Jenny\Python\Time+Time.py'. The output shows two sets of user input for hours, minutes, and seconds, followed by the sum of the two times. The process finished with exit code 0.

```
Run: Time+Time x
C:\Jenny\Python\venv\Scripts\python.exe C:\Jenny\Python\Time+Time.py
Enter the hour: 12
Enter the minute: 22
Enter the second: 11
Enter the hour: 10
Enter the minute: 8
Enter the second: 4
Sum of two times: 22:30:15

Process finished with exit code 0
```

## **Result**

The program was executed and the result was successfully obtained. Thus, CO4 was obtained.



**Experiment No.: 37****Aim**

Create a class Publisher (name). Derive class Book from Publisher with attributes title and author. Derive class Python from Book with attributes price and no.of pages. Write a program that displays information about a Python book. Use base class constructor invocation and method overriding.

**CO4**

Implement object-oriented programming and exception handling.

**Procedure**

```
class Publisher:
```

```
    def get_name(self):
```

```
        self.__name = input("Enter the publisher name: ")
```

```
    def print_name(self):
```

```
        print("Publisher's name:", self.__name)
```

```
class Book(Publisher):
```

```
    def get_book(self):
```

```
        self.get_name()
```

```
        self.__name = input("Enter the book name: ")
```

```
        self.__author = input("Enter author name: ")
```

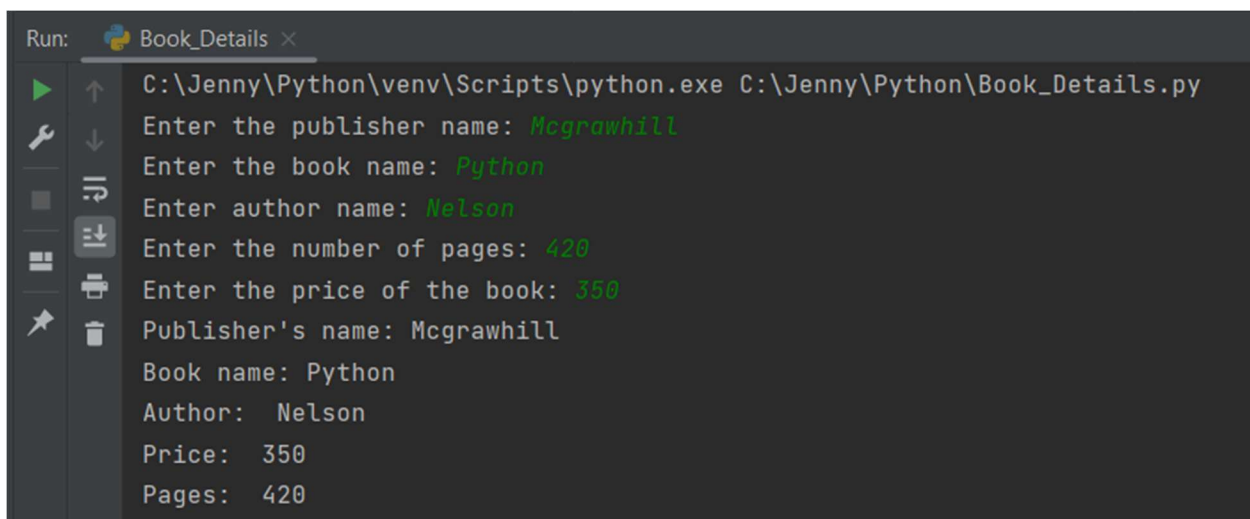
```
    def print_book(self):
```

```
        self.print_name()
```

```
        print("Book name:", self.__name, "\nAuthor: ", self.__author)
```

```
class Python(Book):  
    def get_py(self):  
        self.get_book()  
        self.__pages = input("Enter the number of pages: ")  
        self.__price = input("Enter the price of the book: ")  
  
    def print_py(self):  
        self.print_book()  
        print("Price: ", self.__price, "\nPages: ", self.__pages)  
  
p = Python()  
p.get_py()  
p.print_py()
```

### **Output Screenshot**



```
Run: C:\Jenny\Python\venv\Scripts\python.exe C:\Jenny\Python\Book_Details.py  
Enter the publisher name: Mcgrawhill  
Enter the book name: Python  
Enter author name: Nelson  
Enter the number of pages: 420  
Enter the price of the book: 350  
Publisher's name: Mcgrawhill  
Book name: Python  
Author: Nelson  
Price: 350  
Pages: 420
```

### **Result**

The program was executed and the result was successfully obtained. Thus, CO4 was obtained.

**Experiment No.: 38****Aim**

To read a file line by line and store it into a list.

**CO5**

Create files and form regular expressions for effective search operations on strings and files.

**Procedure**

```
# Program to read file content and store it into a list.
```

```
# using readlines()
```

```
open_file = open('readfile.txt')
```

```
File_Lines = open_file.readlines()
```

```
# Without using strip
```

```
print("\nFile content stored in list:")
```

```
print(File_Lines)
```

```
# By using strip
```

```
print("\nFile content after removing newline character:")
```

```
File_Lines = [X.strip() for X in File_Lines]
```

```
print(File_Lines)
```

```
# print([X.strip() for X in File_Lines])
```

```
open_file.close()
```

**readline.txt**

Jenny Johnson

Amal Jyothi College of Engineering

Koovappally P.O


Kanjirappally

Kottayam

Kerala

686518

## **Output Screenshot**



```
Run: Read_file_store_list x
C:\Jenny\Python\venv\Scripts\python.exe C:\Jenny\Python\Read_file_store_list.py

File content stored in list:
['Jenny Johnson\n', 'Amal Jyothi College of Engineering\n', 'Koovappally P.O\n', 'Kanjirappally\n', 'Kottayam\n', 'Kerala\n', '686518']

File content after removing newline character:
['Jenny Johnson', 'Amal Jyothi College of Engineering', 'Koovappally P.O', 'Kanjirappally', 'Kottayam', 'Kerala', '686518']

Process finished with exit code 0
```

## **Result**

The program was executed and the result was successfully obtained. Thus, CO5 was obtained.

**Experiment No.: 39****Aim**

To copy odd lines of one file to other.

**CO5**

Create files and form regular expressions for effective search operations on strings and files.

**Procedure**

```
# Program to copy odd lines of one file to another.
# Opening files for reading and writing data
input_file = open('readfile.txt')
output_file = open('WriteData.txt', 'w')

# Coping/reading contents from read_file to copy_data
copy_data = input_file.readlines()
print("\nActual File Content is:")
print(copy_data, "\n")

for i in range(0, len(copy_data)):
    if i % 2 == 0:
        output_file.write(copy_data[i])
    else:
        pass

# Closing file after writing
output_file.close()

# Opening write file in read mode and printing values
output_file = open('WriteData.txt', 'r')
```

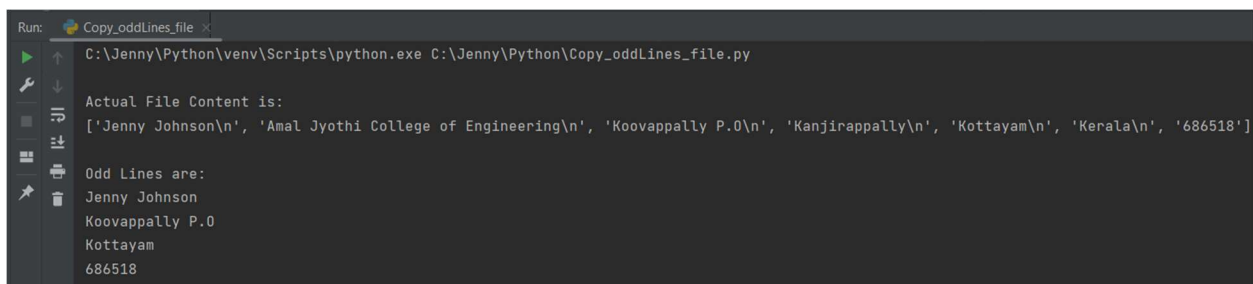
```
print("Odd Lines are:")  
print(output_file.read())
```

```
# Closing files  
input_file.close()  
output_file.close()
```

readfile.txt

Jenny Johnson  
Amal Jyothi College of Engineering  
Koovappally P.O  
Kanjirappally  
Kottayam  
Kerala  
686518

## **Output Screenshot**



```
Run: Copy_oddLines_file  
C:\Jenny\Python\venv\Scripts\python.exe C:\Jenny\Python\Copy_oddLines_file.py  
  
Actual File Content is:  
['Jenny Johnson\n', 'Amal Jyothi College of Engineering\n', 'Koovappally P.O\n', 'Kanjirappally\n', 'Kottayam\n', 'Kerala\n', '686518']  
  
Odd Lines are:  
Jenny Johnson  
Koovappally P.O  
Kottayam  
686518
```

## **Result**

The program was executed and the result was successfully obtained. Thus, CO5 was obtained.

**Experiment No.: 40****Aim**

To read each row from a given csv file and print a list of strings.

**CO5**

Create files and form regular expressions for effective search operations on strings and files.

**Procedure**

```
import csv
# open csv file
with open("Book1.csv", 'r') as file:
    # create a csv reader
    reader = csv.reader(file)
    # fetching each line from csv file
    for row in reader:
        print(row)
```

Book1.csv

Name,Branch,Year,Batch

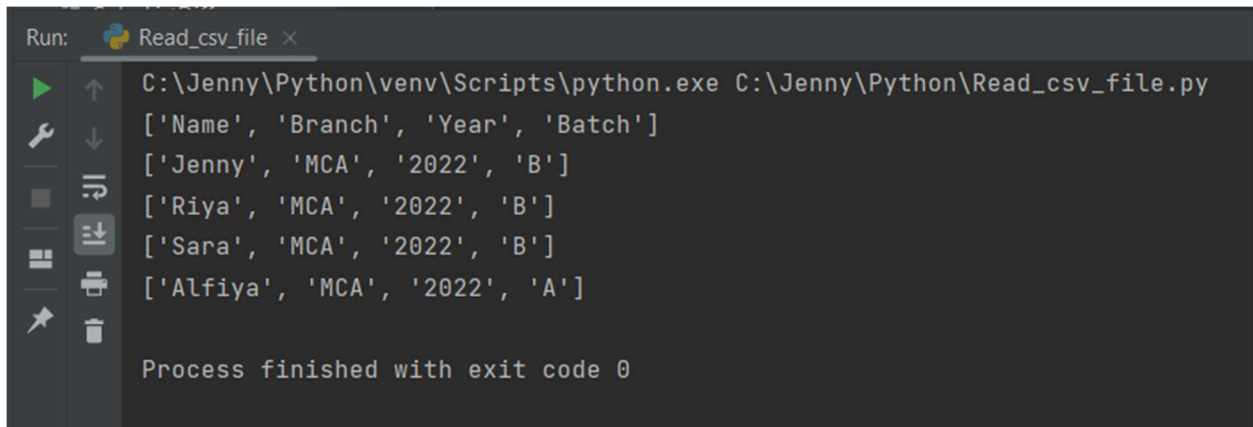
Jenny,MCA,2022,B

Riya,MCA,2022,B

Sara,MCA,2022,B

Alfiya,MCA,2022,A

## **Output Screenshot**



```
Run: Read_csv_file x
C:\Jenny\Python\venv\Scripts\python.exe C:\Jenny\Python\Read_csv_file.py
['Name', 'Branch', 'Year', 'Batch']
['Jenny', 'MCA', '2022', 'B']
['Riya', 'MCA', '2022', 'B']
['Sara', 'MCA', '2022', 'B']
['Alfiya', 'MCA', '2022', 'A']

Process finished with exit code 0
```

## **Result**

The program was executed and the result was successfully obtained. Thus, CO5 was obtained.



**Experiment No.: 41****Aim**

To read specific columns of a given CSV file and print the content of the columns.

**CO5**

Create files and form regular expressions for effective search operations on strings and files.

**Procedure**

```
import csv

# specify the no.of columns to be displayed
columns_read = [0, 2]

# open csv file
with open("Book1.csv", 'r') as file:
    # create a csv reader
    reader = csv.reader(file)
    # fetching each line from csv file
    for row in reader:
        print([row[i] for i in columns_read])
```

Book1.csv

Name,Branch,Year,Batch

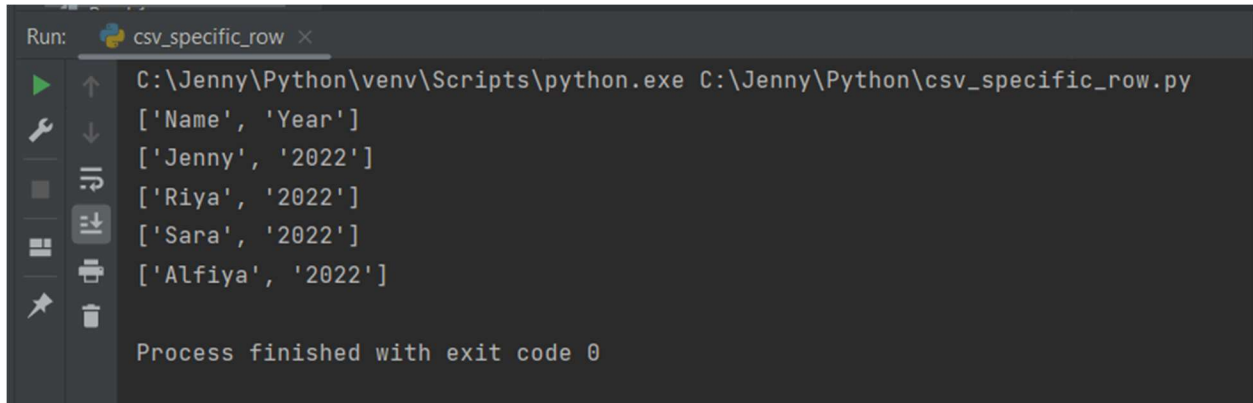
Jenny,MCA,2022,B

Riya,MCA,2022,B

Sara,MCA,2022,B

Alfiya,MCA,2022,A

## **Output Screenshot**



```
Run: csv_specific_row x
C:\Jenny\Python\venv\Scripts\python.exe C:\Jenny\Python\csv_specific_row.py
['Name', 'Year']
['Jenny', '2022']
['Riya', '2022']
['Sara', '2022']
['Alfiya', '2022']

Process finished with exit code 0
```

## **Result**

The program was executed and the result was successfully obtained. Thus, CO5 was obtained.

**Experiment No.: 42****Aim**

To write a Python dictionary to a csv file. After writing the CSV file read the CSV file and display the content.

**CO5**

Create files and form regular expressions for effective search operations on strings and files.

**Procedure**

# Write a Python program to write a Python dictionary to a csv file.

# After writing the CSV file read the CSV file and display the content.

```
import csv

# Data to be inserted
data = [{ 'Name': 'John', 'Age': 25, 'Country': 'United States'},
        { 'Name': 'Mike', 'Age': 32, 'Country': 'Canada'},
        { 'Name': 'Sarah', 'Age': 35, 'Country': 'United Kingdom'}]

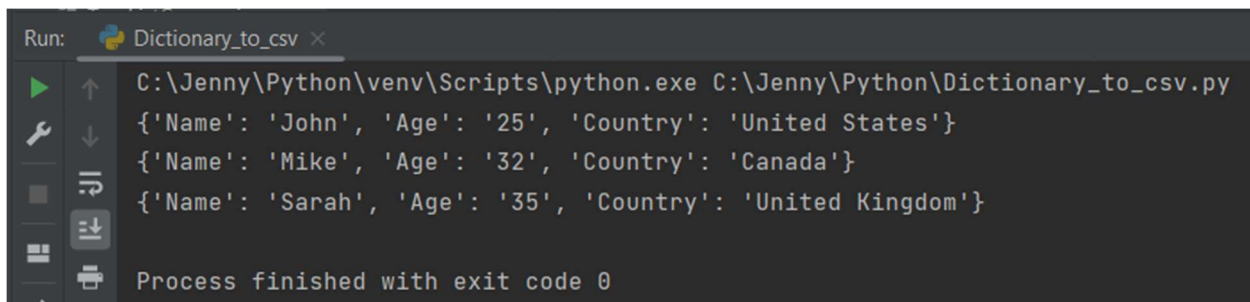
# Write to CSV file
with open('people.csv', 'w') as csvfile:
    headernames = ['Name', 'Age', 'Country']
    csvwriter = csv.DictWriter(csvfile, fieldnames=headernames)
    csvwriter.writeheader()
    for row in data:
        csvwriter.writerow(row)

# Read from CSV file and print contents
with open('people.csv', 'r') as csvfile:
```

```
reader = csv.DictReader(csvfile)

for row in reader:
    print(row)
```

## **Output Screenshot**



```
Run: Dictionary_to_csv x
C:\Jenny\Python\venv\Scripts\python.exe C:\Jenny\Python\Dictionary_to_csv.py
{'Name': 'John', 'Age': '25', 'Country': 'United States'}
{'Name': 'Mike', 'Age': '32', 'Country': 'Canada'}
{'Name': 'Sarah', 'Age': '35', 'Country': 'United Kingdom'}
Process finished with exit code 0
```

## **Result**

The program was executed and the result was successfully obtained. Thus, CO5 was obtained.

**Experiment No.: 43****Aim**

Micro Project: Crud operations using Django (student details).

**COs**

CO1, CO2, CO3, CO4 and CO5.

**Procedure**

views.py

```
from django.shortcuts import render,redirect
```

```
from .models import Student
```

```
# Create your views here.
```

```
def index(request):
```

```
    data=Student.objects.all()
```

```
    print(data)
```

```
    context={"data":data}
```

```
    return render(request,"index.html",context)
```

```
def insertData(request):
```

```
    if request.method=="POST":
```

```
        name=request.POST.get('name')
```

```
        course=request.POST.get('course')
```

```
        email=request.POST.get('email')
```

```
        phone=request.POST.get('phone')
```

```
        gender=request.POST.get('gender')
```

```
        print(name,course,email,phone,gender)
```

```
        query=Student(name=name,course=course,email=email,phone=phone,gender=gender)
```

```
        query.save()
```

```
return render(request,"index.html")
```

```
def updateData(request,id):
```

```
    if request.method=="POST":
```

```
        name=request.POST['name']
```

```
        course=request.POST['course']
```

```
        email=request.POST['email']
```

```
        phone=request.POST['phone']
```

```
        gender=request.POST['gender']
```

```
        print(name,course,email,phone,gender)
```

```
        query=Student(name=name,course=course,email=email,phone=phone,gender=gender)
```

```
        query.save()
```

```
        return redirect("/")
```

```
d=Student.objects.get(id=id)
```

```
context={"d":d}
```

```
return render(request,"edit.html",context)
```

```
def deleteData(request,id):
```

```
    if request.method=="POST":
```

```
        de.delete()
```

```
        return redirect("/")
```

```
de=Student.objects.get(id=id)
```

```
context={"de":de}
```

```
return render(request,"delete.html",context)
```

```
def about(request):
```

```
return render(request,"about.html")
```

### urls.py

```
from django.urls import path
```

```
from app import views
```

```
urlpatterns = [  
    path("",views.index,name="index"),  
    path('about',views.about,name="about"),  
    path('insert',views.insertData,name="insertData"),  
    path('update/<id>',views.updateData,name="updateData"),  
    path('insert/<id>',views.deleteData,name="deleteData"),  
]
```

### Manage.py

```
#!/usr/bin/env python
```

```
"""Django's command-line utility for administrative tasks."""
```

```
import os
```

```
import sys
```

```
def main():
```

```
    """Run administrative tasks."""
```

```
    os.environ.setdefault('DJANGO_SETTINGS_MODULE', 'project.settings')
```

```
    try:
```

```
        from django.core.management import execute_from_command_line
```

```
    except ImportError as exc:
```

```
        raise ImportError(
```

"Couldn't import Django. Are you sure it's installed and "

"available on your PYTHONPATH environment variable? Did you "

"forget to activate a virtual environment?"

) from exc

execute\_from\_command\_line(sys.argv)

if \_\_name\_\_ == '\_\_main\_\_':

main()

## Output Screenshot

crud operation x +

127.0.0.1:8000

CRUD

**Insert Student Details**

Enter student name

Enter course

Enter email id

Enter contact number

Select gender

Submit

**Student Details**

Id	Name	Course	Email	Phone	Edit	Remove
12	rugma	mca	rugma@gmail.com	12345	female	<a href="#">Edit</a> <a href="#">Delete</a>
15	riya	mca	riya@gmail.com	34567	female	<a href="#">Edit</a> <a href="#">Delete</a>
16	sara	mca	sara@gmail.com	34568	female	<a href="#">Edit</a> <a href="#">Delete</a>
17	jenny	mca	jenny@gmail.com	34568	female	<a href="#">Edit</a> <a href="#">Delete</a>
18	rugma	mca	rug@gmail.com	12345	female	<a href="#">Edit</a> <a href="#">Delete</a>

## Result

The program was executed and the result was successfully obtained. Thus, CO1, CO2, CO3, CO4 and CO5 was obtained.