

# Problem Solving And Python Programming Laboratory

## List of Programs

1. Developing flow charts
  - a. Electricity Billing,
  - b. Retail shop billing,
  - c. Sin series,
  - d. weight of a motorbike,
  - e. Weight of a steel bar,
  - f. compute Electrical Current in Three Phase AC Circuit,
2. Programs Using Simple Statements
  - a. Exchange the values of two variables,
  - b. Circulate the values of n variables,
  - c. Distance between two points.
3. Programs Using Conditionals and Iterative Statements
  - a. Number series
  - b. Number Patterns
  - c. Pyramid pattern

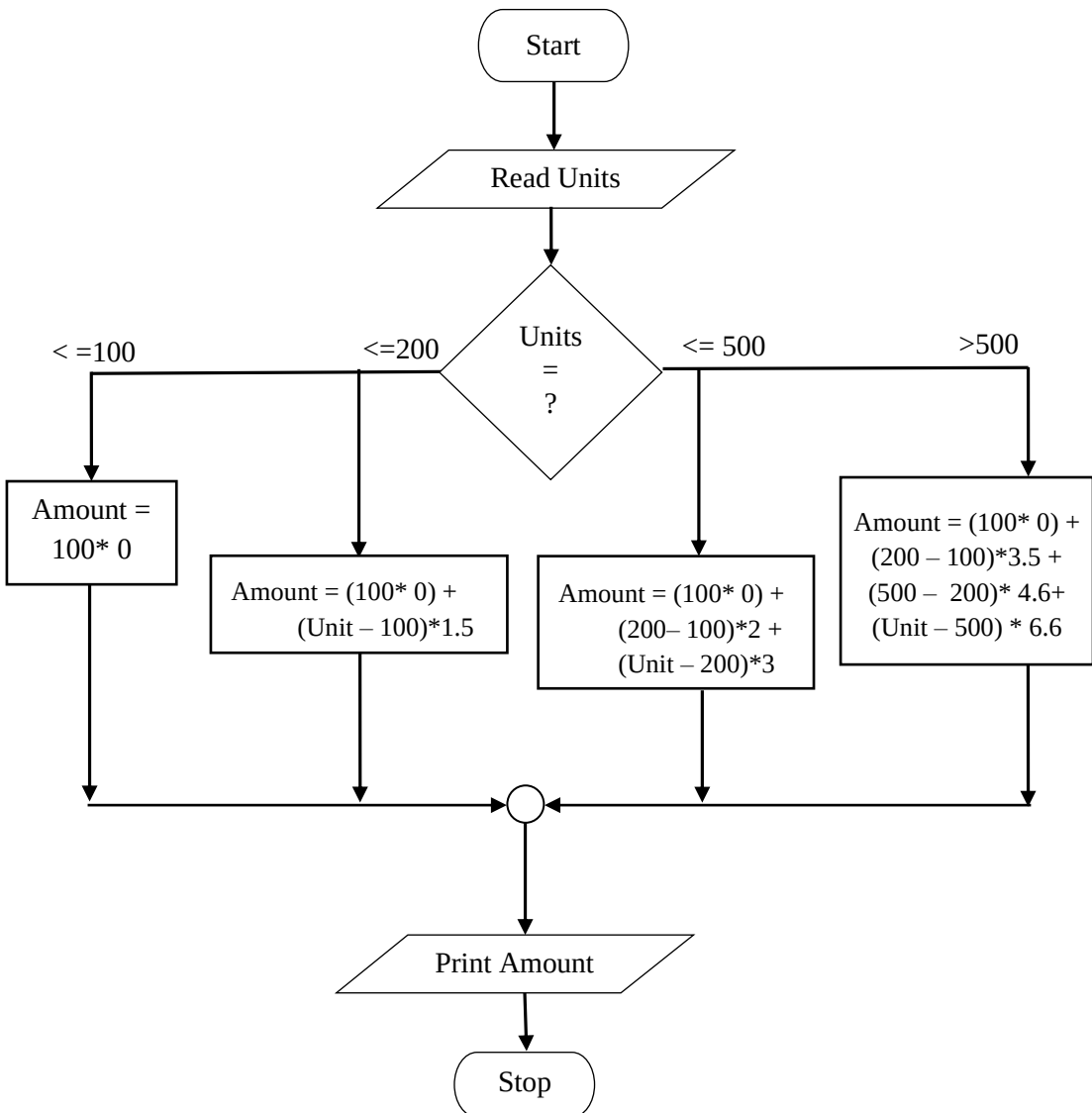
4. Operations of Lists and Tuples (Items present in a library/Components of a car/ Materials required for construction of a building)
5. Operations of Sets & Dictionaries (Language, components of an automobile, Elements of a civil structure, etc)
6. Programs using Functions
  - a. Factorial of a Number
  - b. Largest number in a list
  - c. Area of shape
7. Programs using Strings.
  - a. Reversing a String,
  - b. Checking Palindrome in a String,
  - c. Counting characters in a String
  - d. Replacing characters in a String
8. Programs Using modules and Python Standard Libraries (pandas, numpy. Matplotlib, scipy)
9. Programs using File handling.
  - a. Copy from one file to another,
  - b. Word count,
  - c. Longest word
10. Programs Using Exception handling.
  - a. Divide by zero error,
  - b. Voter's age validity,
  - c. Student mark range validation
11. Exploring Pygame tool.
12. Developing a game activity using Pygame like bouncing ball, car race etc.

**Ex. No :1A      DEVELOPING FLOWCHARTS ELECTRICITY BILLING****Problem Statement:**

- For 0 to 100 units the per unit is ₹ 0/-
- For 0 to 200 units, for the first 100 unit the per unit cost is zero and the next 100 units, the consumer shall pay ₹ 1.5 per unit.
- For 0 to 500 units, the consumer shall pay ₹ 0 for the first 100 units, for the next 100 units the consumer shall pay ₹ 2 per unit, for the next 300 units the unit cost is ₹3.00/-
- For above 500 units, the consumer shall pay ₹ 0 for the first 100 units, for the next 100 units the consumer shall pay ₹ 3.50 per unit, for the next 300 units the unit cost is ₹4.60/- and for the remaining units the unit cost is ₹6.60/-

**Algorithm:**

1. Start
2. Read number of units
3. If number of unit is  $\leq 100$  then calculate the amount as  $100 * 0$
4. If number of unit is  $\leq 200$  then calculate the amount as  $\text{Amount} = (100 * 0) + (\text{Unit} - 100) * 1.5$
5. If number of unit is  $\leq 500$  then calculate the amount as  $\text{Amount} = (100 * 0) + (200 - 100) * 2 + (\text{Unit} - 200) * 3$
6. If number of unit is  $> 500$  then calculate the amount as  $\text{Amount} = (100 * 0) + (200 - 100) * 3.5 + (500 - 200) * 4.6 + (\text{Unit} - 500) * 6.6$
7. Print Amount
8. Stop

**Flowchart:**

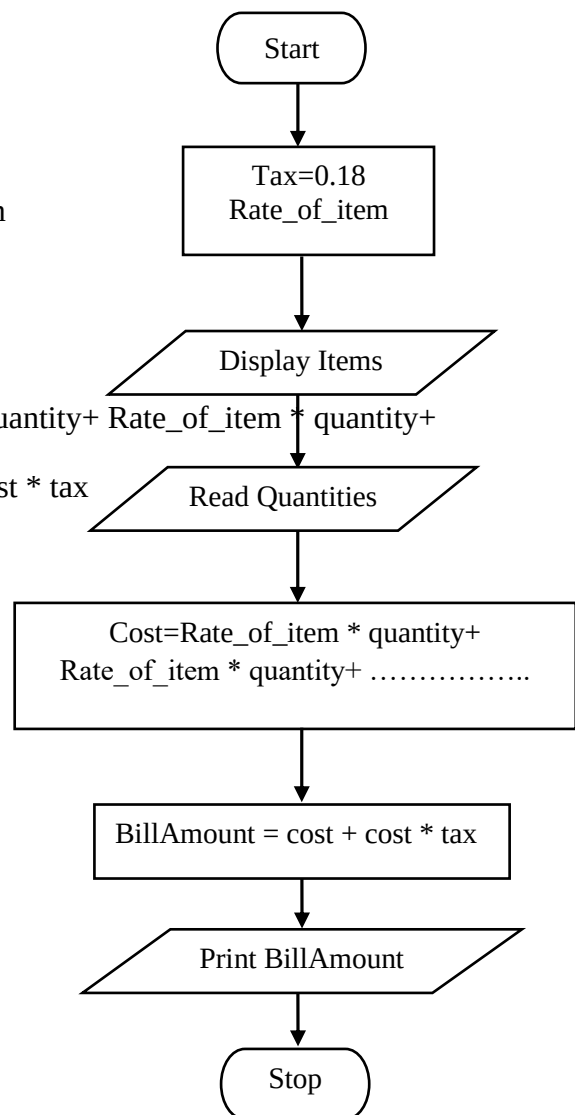
Result: Thus, the flow chart is drawn for the problem statement and it is verified.

**Ex: No: 1B****RETAIL SHOP BILLING****Problem Statement:**

To prepare Retail shop billing flowchart.

**Algorithm:**

1. Start
2. Assume Tax = 0.18, Rate\_of\_item
3. Display items
4. Read Quantities
5. Calculate Cost=Rate\_of\_item \* quantity+ Rate\_of\_item \* quantity+
6. Calculate BillAmount = cost + cost \* tax
7. Print BillAmount
8. Stop



Result: Thus, the flow chart is drawn for the problem statement and it is verified.

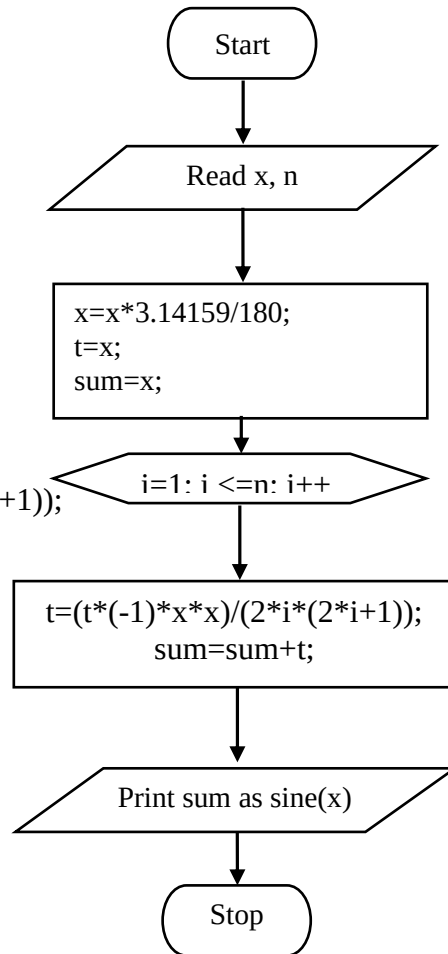
**Ex: No: 1 C****SINE SERIES****Problem Statement:**

To evaluate the sine series. The formula used to express the Sin(x) as

$$\sin x = \sum_{n=0}^{\infty} (-1)^n \frac{x^{2n+1}}{(2n+1)!}$$

**Algorithm:**

1. Start
2. Read x,n
3. Calculate  $x=x*3.14159/180$ ;  
     $t=x$ ;  
     $sum=x$ ;
4. Make a loop from 1 to n,
5. Calculate  $t=(t*(-1)*x*x)/(2*i*(2*i+1))$ ;
6.  $sum=sum+t$ ;
7. Print Sum
8. Stop



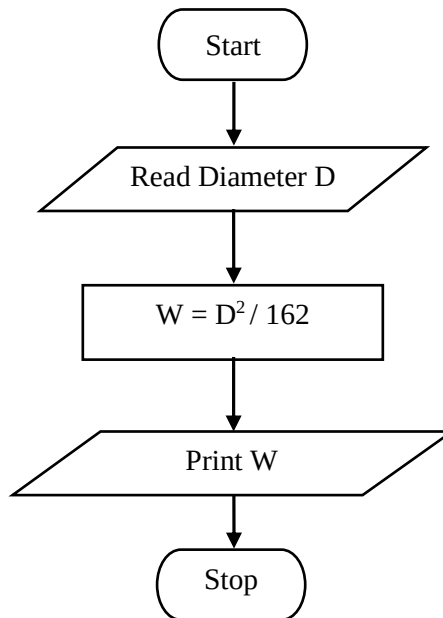
Result: Thus, the flow chart is drawn for the problem statement and it is verified.

**Ex: No: 1D****WEIGHT OF A STEEL BAR****Problem Statement:**

To find the weight of a steel bar.

**Algorithm:**

1. Start
2. Read Diameter D
3. Calculate  $W = D^2 / 162$
4. Print W
5. Stop



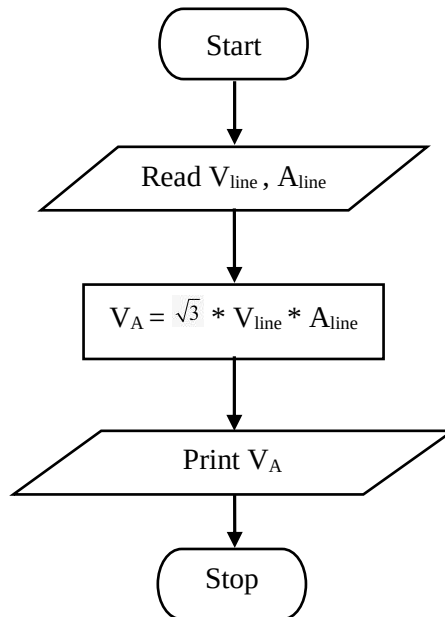
Result: Thus, the flow chart is drawn for the problem statement and it is verified.

**Ex No :1 E COMPUTE ELECTRICAL CURRENT IN THREE PHASE AC CIRCUIT****Problem Statement:**

To compute electrical current in three phase AC circuit.

**Algorithm:**

1. Start
2. Read  $V_{\text{line}}, A_{\text{line}}$
3. Calculate  $V_A = \sqrt{3} * V_{\text{line}} * A_{\text{line}}$
4. Print  $V_A$
5. Stop



Result: Thus, the flow chart is drawn for the problem statement and it is verified.

**Ex: No: 2 A****PROGRAMS USING SIMPLE STATEMENTS****EXCHANGE THE VALUES OF TWO VARIABLES****Aim:**

To write a python program to exchange the values of two variables.

**Algorithm:**

1. Start
2. Read x,y.
3. Print the numbers before exchange the numbers.
4. Swap the numbers by using assignment operator.  $x,y=y,x$ .
5. Print the numbers after exchanging the numbers.
6. Stop

**Program:**

```
def exchange(x,y):                                #Function Definition
    x,y=y,x                                       # swapping using tuple assignment
    print("After exchange of x,y")
    print("x =",x)
    print("Y= ",y)
x=input("Enter value of X ")                      #Main Function
y=input("Enter value of Y ")
print("Before exchange of x,y")
print("x =",x)
print("Y= ",y)
exchange(x,y)                                     # Function call
```

**Output:**

```
Enter value of X: 67
Enter value of Y: 56
Before exchange of x,y
x = 67
Y= 56
After exchange of x,y
x = 56
Y= 67
```

**Result:**

Thus, the python program is executed to exchange the values of two variables and the output was verified.

---

**Ex: No: 2 B      CIRCULATE THE VALUES OF N VARIABLES****Aim:**

To write a python program to circulate the values of N variables.

**Algorithm:**

1. Start
2. Read number of number of terms
3. Read the numbers and append to the list.
4. Pop first element from the list and append it to the end of the list.
5. Print the list.
6. Repeat steps 4 and 5 till number of terms.
7. Stop.

**Program:**

```
no_of_terms = int(input("Enter number of values : "))
#Read values
list1 = []
for val in range(0,no_of_terms,1):
    x = int(input("Enter integer : "))
    list1.append(x)
#Circulate and display values
print("Circulating the elements of list ", list1)

for val in range(0,no_of_terms,1):
    x = list1.pop(0)
    list1.append(x)
    print(list1)
```

**Output:**

```
Enter number of values : 5
Enter integer : 91
Enter integer : 92
Enter integer : 93
Enter integer : 94
```

Enter integer : 95

Circulating the elements of list [91, 92, 93, 94, 95]

[92, 93, 94, 95, 91]

[93, 94, 95, 91, 92]

[94, 95, 91, 92, 93]

[95, 91, 92, 93, 94]

[91, 92, 93, 94, 95]

Result:

Thus a python program to circulate the values of N variables is executed and the output was verified.

---

**Ex: No: 2 C****DISTANCE BETWEEN TWO VARIABLES****Aim:**

To write a python program to find distance between two variables.

**Algorithm:**

1. Start
2. Read the two coordinates (x1, y1) and (x2,y2)
3. Calculate distance =  $\text{math.sqrt}(((x2-x1)**2)+((y2-y1)**2))$
4. Print distance
5. Stop

**Program:**

```
import math
x1 = int(input("Enter a x1: "))
y1 = int(input("Enter a y1: "))
x2 = int(input("Enter a x2: "))
y2 = int(input("Enter a y2: "))
distance = math.sqrt(((x2-x1)**2)+((y2-y1)**2))
print("Distance = ",distance)
```

**Output:**

```
Enter a x1: 3
Enter a y1: 2
Enter a x2: 7
Enter a y2: 8
Distance = 7.211102550927978
```

**Result:**

Thus, a python program to find distance between two variables is executed and the output was verified.

**Ex: No: 3A PROGRAMS USING CONDITIONALS AND ITERATIVE LOOPS NUMBER SERIES****Aim:**

To write a python program to evaluate  $1^2+2^2+3^2+....+N^2$

**Algorithm:**

1. Start
2. Read n
3. Assign sum=0, i=1
4. Generate numbers from 1 to n and calculate sum=sum+i\*i and Increment i
5. Print sum.

**Program:**

```
n = int(input('Enter a number: '))
sum=0
i=1
while i<=n:
    sum=sum+i*i
    i+=1
print('Sum = ',sum)
```

**Output:**

Enter a number: 10

Sum = 385

**Result:**

Thus, a python program to evaluate  $1^2+2^2+3^2+....+N^2$  is executed and the output was verified.

**Ex: No: 3B****NUMBER PATTERN****Aim:**

To write a python program to print number pattern.

**Algorithm:**

1. Start
2. Assign number of rows N=5
3. Write an outer loop to iterate the number of rows using a for loop and range() function.
4. Write the inner loop or nested loop to handle the number of columns
5. Use the print() function in each iteration of nested for loop to display the symbol or number of a pattern (like a star (asterisk \*) or number).
6. Add a new line using the print() function after each iteration of the outer loop so that the pattern display appropriately.
7. Stop

**Program:**

```
N = 5
```

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

```
    for k in range(N,i, -1):
```

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

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

```
        print(j, end = ' ')
```

```
    for l in range(i-1,0,-1):
```

```
        print(l, end = ' ')
```

```
    print()
```

**Output:**

```
    1
  1 2 1
1 2 3 2 1
1 2 3 4 3 2 1
1 2 3 4 5 4 3 2 1
```

**Result:**

Thus a python program to print number pattern is executed and the output was verified.

---

**Ex: No: 3B****PYRAMID PATTERN****Aim:**

To write a python program to print Pyramid pattern.

**Algorithm:**

1. Start
2. Enter number of rows.
3. Write an outer loop to iterate the number of rows using a for loop and range() function.
4. Write the inner loop or nested loop to handle the number of columns
5. Use the print() function in each iteration of nested for loop to display the symbol or number of a pattern (like a star (asterisk \*) or number).
6. Add a new line using the print() function after each iteration of the outer loop so that the pattern display appropriately.
7. Stop

**Program:**

```
n = int(input("Enter the number of rows: "))
m = (2 * n) - 2
for i in range(0, n):
    for j in range(0, m):
        print(end=" ")
    m = m - 1 # decrementing m after each loop
    for j in range(0, i + 1):
        # printing full Triangle pyramid using stars
        print("* ", end=' ')
```

```
print(" ")
```

**Output:**

Enter the number of rows: 9

```

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

**Result:**

Thus a python program to print pyramid pattern is executed and the output was verified.

**Ex: No: 4 A****OPERATIONS OF LISTS****Aim:**

To Implement List operations in a library list.

**Algorithm:**

1. Start
2. Create a library list and assign values.
3. Print the list.
4. Perform different list operations and print the result.
5. Stop.

**Program:**

```
# declaring a list of items in a Library
```

```
library =
```

```
['Books','Periodicals','Newspaper','Manuscripts','Maps','Prints','Documents','Ebooks']
```

```
# printing the complete list
```

```
print('Library: ',library)
```

```
# printing first element
```

```
print('first element: ',library[0])
```

```
# printing fourth element
```

```
print('fourth element: ',library[3])
```

```
# printing list elements from 0th index to 4th index
```

```
print('Items in Library from 0 to 4 index: ',library[0: 5])
```

```
# printing list -7th or 3rd element from the list
```

```
print('3rd or -7th element: ',library[-7])
```

```
# appending an element to the list
```

```
library.append('Audiobooks')
print('Library list after append(): ',library)
# finding index of a specified element
print('index of \'Newspaper\': ',library.index('Newspaper'))
# sorting the elements of iList
library.sort()
print('after sorting: ', library);
# popping an element
print('Popped elements is: ',library.pop())
print('after pop(): ', library);
# removing specified element
library.remove('Maps')
print('after removing \'Maps\': ',library)
# inserting an element at specified index
# inserting 100 at 2nd index
library.insert(2, 'CDs')
print('after insert: ', library)
# Number of Elements in Library list
print(' Number of Elements in Library list : ',library.count('Ebooks'))
```

**Output:**

Library: ['Books', 'Periodicals', 'Newspaper', 'Manuscripts', 'Maps', 'Prints', 'Documents', 'Ebooks']

first element: Books

fourth element: Manuscripts

Items in Library from 0 to 4 index: ['Books', 'Periodicals', 'Newspaper', 'Manuscripts', 'Maps']

3rd or -7th element: Periodicals

Library list after append(): ['Books', 'Periodicals', 'Newspaper', 'Manuscripts', 'Maps', 'Prints', 'Documents', 'Ebooks', 'Audiobooks']

index of 'Newspaper': 2

after sorting: ['Audiobooks', 'Books', 'Documents', 'Ebooks', 'Manuscripts', 'Maps', 'Newspaper', 'Periodicals', 'Prints']

Popped elements is: Prints

after pop(): ['Audiobooks', 'Books', 'Documents', 'Ebooks', 'Manuscripts', 'Maps', 'Newspaper', 'Periodicals']

after removing 'Maps': ['Audiobooks', 'Books', 'Documents', 'Ebooks', 'Manuscripts', 'Newspaper', 'Periodicals']

after insert: ['Audiobooks', 'Books', 'CDs', 'Documents', 'Ebooks', 'Manuscripts', 'Newspaper', 'Periodicals']

Number of Elements in Library list : 1

**Result:**

Thus, the python program to implement list operations in a library list is executed and the output was verified.

**Ex: No: 4 B****OPERATIONS OF TUPLE****Aim:**

To Implement Tuple operations in a Car Tuple.

**Algorithm:**

1. Start
2. Create a Tuple and assign values.
3. Print the Tuple.
4. Perform different tuple operations and print the result.
5. Stop.

**Program:**

```
# Python code for various Tuple operation
# declaring a tuple of Components of a car
car = ('Engine','Battery','Alternator','Radiator','Steering','Break','Seat Belt')
# printing the complete tuple
print('Components of a car: ',car)
# printing first element
print('first element: ',car[0])
# printing fourth element
print('fourth element: ',car[3])
# printing tuple elements from 0th index to 4th index
print('Components of a car from 0 to 4 index: ',car[0: 5])
# printing tuple -7th or 3rd element from the list
print('3rd or -7th element: ',car[-7])
# finding index of a specified element
print('index of \'Alternator\': ',car.index('Alternator'))
# Number of Elements in car tuple
print(' Number of Elements in Car Tuple : ',car.count('Seat Belt'))
#Length of car tuple
print(' Length of Elements in Car Tuple : ',len(car))
```

**Output:**

```
Components of a car: ('Engine', 'Battery', 'Alternator', 'Radiator', 'Steering', 'Break', 'Seat Belt')
first element      : Engine
fourth element     : Radiator
Components of a car from 0 to 4 index: ('Engine', 'Battery', 'Alternator', 'Radiator', 'Steering')
3rd or -7th element: Engine
index of 'Alternator': 2
Number of Elements in Car Tuple : 7
Length of Elements in Car Tuple : 7
```

**Result:**

Thus, the python program to implement tuple operations in a car tuple is executed and the output was verified.

**Ex: No: 5 A****OPERATIONS OF SETS****Aim:**

To Implement operations in a set using Components of a Language as example.

**Algorithm:**

1. Start
2. Create a two sets and assign values.
3. Print the sets.
4. Perform different set operations and print the result.
5. Stop.

**Program:**

```
L1 = {'Pitch', 'Syllabus', 'Script', 'Grammar', 'Sentences'};
L2 = {'Grammar', 'Syllabus', 'Context', 'Words', 'Phonetics'};

# set union

print("Union of L1 and L2 is  ",L1 | L2)

# set intersection

print("Intersection of L1 and L2 is  ",L1 & L2)

# set difference

print("Difference of L1 and L2 is  ",L1 - L2)

# set symmetric difference

print("Symmetric difference of L1 and L2 is  ",L1 ^ L2)
```

**Output:**

```
Union of L1 and L2 is  {'Words', 'Pitch', 'Sentences', 'Phonetics', 'Script', 'Grammar',
'Syllabus', 'Context'}
Intersection of L1 and L2 is  {'Grammar', 'Syllabus'}
Difference of L1 and L2 is  {'Script', 'Pitch', 'Sentences'}
```

Symmetric difference of L1 and L2 is {'Words', 'Context', 'Script', 'Pitch', 'Sentences', 'Phonetics'}

**Result:**

Thus, the python program to implement set operations is executed and the output was verified.

**Ex: No: 6 A      FACTORIAL OF A NUMBER USING FUNCTION****Aim:**

To write a python program to find the factorial of a number using functions.

**Algorithm:**

1. Start
2. Read a number n
3. If  $n = 1$  print 1 as factorial.
4. Otherwise calculate  $n * \text{fact}(n-1)$  by calling recursive function and print fact.
5. Stop.

**Program:**

```
def fact(n):  
    if n == 1:  
        return n  
    else:  
        return n*fact(n-1)  
num = int(input("Enter a number: "))  
print("The factorial of",num,"is",fact(num))
```

**Output:**

Enter a number: 5  
The factorial of 5 is 120

**Result:**

Thus, the python program to calculate factorial is executed and the output was verified.

**Ex: No: 6 B FINDING LARGEST NUMBER IN A LIST USING FUNCTION****Aim:**

To write a python program to find the largest number in a list using functions.

**Algorithm:**

1. Start
2. Create a list and read the numbers
3. Append them in a list.
4. Use max built in function to find the largest element in a list.
5. Print the largest number
6. Stop.

**Program:**

```
def myMax(list1):  
    print("Largest element is:", max(list1))  
  
list1 = []  
num = int(input("Enter number of elements in list: "))  
for i in range(1, num + 1):  
    ele = int(input("Enter elements: "))  
    list1.append(ele)  
print("Largest element is:", myMax(list1))
```

**Output:**

```
Enter number of elements in list: 6  
Enter elements: 58  
Enter elements: 69  
Enter elements: 25  
Enter elements: 37  
Enter elements: 28  
Enter elements: 49  
Largest element is: 69
```

**Result:**

Thus, a python program to find the largest number in a list is executed and the output was verified.

---

**Ex: No: 6 B      FINDING AREA OF A CIRCLE USING FUNCTION****Aim:**

To write a python program to find the area of a circle using functions.

**Algorithm:**

1. Start
2. Read radius r value.
3. Assign  $PI=3.14$
4. Calculate  $PI*(r*r)$
5. Print result
6. Stop

**Program:**

```
def findArea(r):  
    PI = 3.142  
    return PI * (r*r);  
  
num=float(input("Enter r value:"))  
print("Area is %.6f" % findArea(num));
```

**Output:**

```
Enter r value:8  
Area is 201.088000
```

**Result:**

Thus, a python program to find the area of a circle is executed and the output was verified.

**Ex: No: 7A****REVERSING A STRING****Aim:**

To write a python program to reverse a string.

**Algorithm:**

1. Start
2. Read a string
3. Print the original string
4. Use reversed() built in function to reverse a string. The reversed() returns the reversed iterator of the given string and then its elements are joined empty string separated using join(). And reversed order string is formed.
5. Print the reversed string
6. Stop

**Program:**

```
def reverse(string):  
    string = "".join(reversed(string))  
    return string  
  
s = input("Enter any string: ")  
print ("The original string is : ",end="")  
print (s)  
  
print ("The reversed string(using reversed) is : ",end="")  
print (reverse(s))
```

**Output:**

```
Enter any string: Python  
The original string is : Python  
The reversed string(using reversed) is : nohtyP  
Result:
```

Thus, a python program to reverse a string is executed and the output was verified.

**Ex: No: 7B****CHECKING PALINDROME IN A STRING****Aim:**

To write a python program to check palindrome in a string.

**Algorithm:**

1. Start
2. Read a string
3. Reverse a string
4. Check whether original and reversed strings are same or not. If both are same then print it is palindrome string. Otherwise print it is not a palindrome string.
5. Stop.

**Program:**

```
string = input("Enter string: ")
string = string.casefold()
rev_string = reversed(string)
if list(string) == list(rev_string):
    print("It is palindrome")
else:
    print("It is not palindrome")
```

**Output:**

Enter string: Python

It is not palindrome

**Result:**

Thus the program to check palindrome in a string is executed and the output was verified.

**Ex: No: 7C                      COUNTING CHARACTERS IN A STRING****Aim:**

To write a python program to count number of characters in a string.

**Algorithm:**

1. Start
2. Read a string
3. Read the character to be count
4. Use count built in function to count the characters in a string.
5. Print number of characters
6. Stop.

**Program:**

```
string = input("Enter any string: ")  
char = input("Enter a character to count: ")  
val = string.count(char)  
print(val, "\n")
```

**Output:**

Enter any string: python programming

Enter a character to count: n

2

**Result:**

Thus the program to count number of characters in a string is executed and the output was verified.

**Ex: No: 7D****REPLACE CHARACTERS IN A STRING****Aim:**

To write a python program to replace characters in a string.

**Algorithm:**

1. Start
2. Read a string
3. Read a string to be replace and a new string
4. Using replace built in function to replace two strings.
5. Print the new string
6. Stop.

**Program:**

```
string = input("Enter any string: ")  
str1 = input("Enter old string: ")  
str2 = input("Enter new string: ")  
print(string.replace(str1, str2))
```

**Output:**

```
Enter any string: problem solving python programming  
Enter old string: python  
Enter new string: C  
problem solving C programming
```

**Result:**

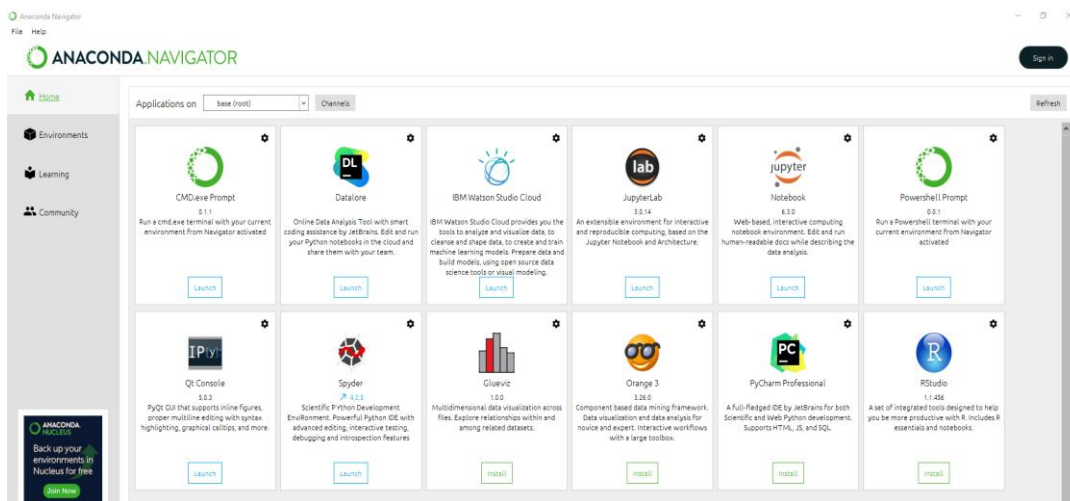
Thus, the program to replace characters in a string is executed and the output was verified.

## INSTALLING AND EXECUTING PYTHON PROGRAM IN ANACONDA NAVIGATOR

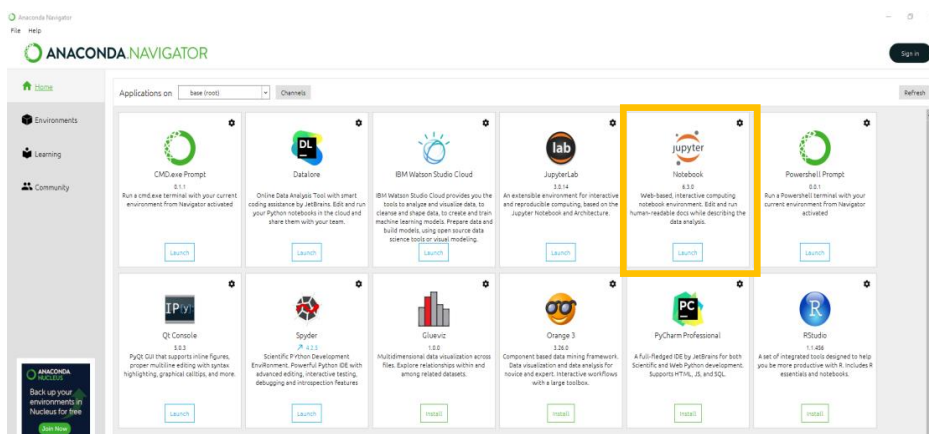
(To run pandas, numpy, matplotlib, scipy)

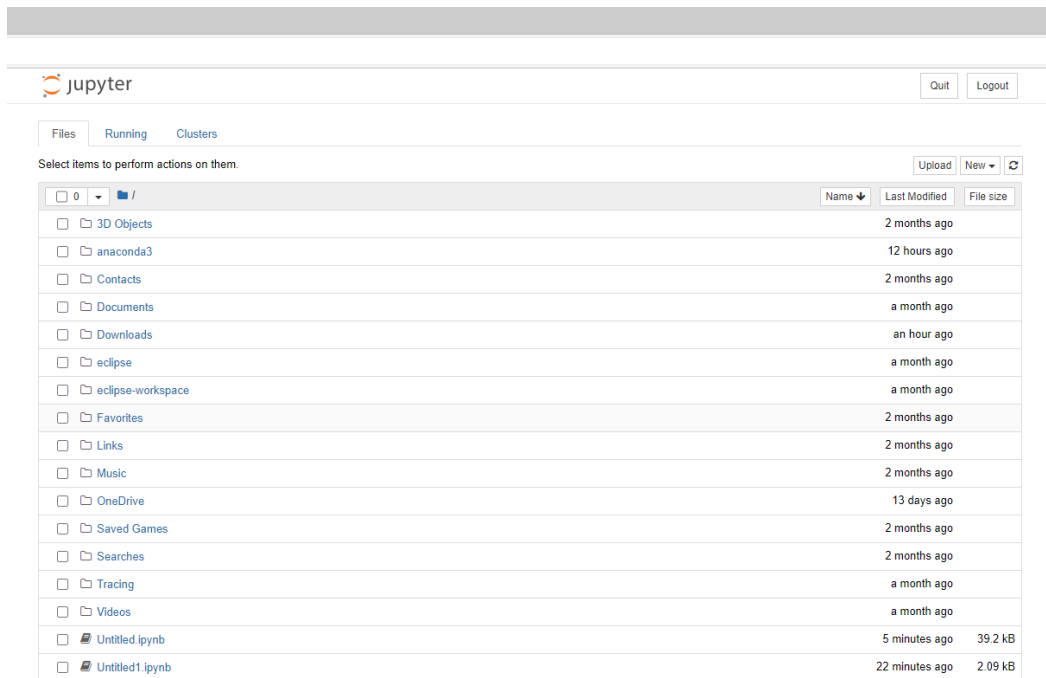
**Step 1:** Install anaconda individual edition for windows

**Step 2:** Open Anaconda navigator



**Step 3:** launch jupyter note book



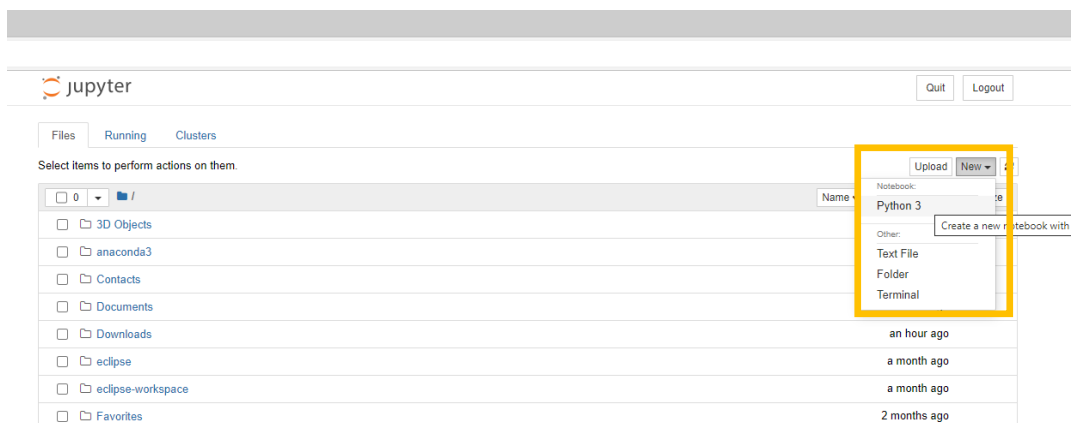


The JupyterLab interface shows the 'Files' tab. At the top, there are 'Quit' and 'Logout' buttons. Below the tabs, there are 'Upload' and 'New' buttons. The main area displays a file browser with a list of files and folders. The list includes folders like '3D Objects', 'anaconda3', 'Contacts', 'Documents', 'Downloads', 'eclipse', 'eclipse-workspace', 'Favorites', 'Links', 'Music', 'OneDrive', 'Saved Games', 'Searches', 'Tracing', 'Videos', and files like 'Untitled.ipynb' and 'Untitled1.ipynb'. The 'New' button is highlighted with a yellow box.

| Name              | Last Modified  | File size |
|-------------------|----------------|-----------|
| 3D Objects        | 2 months ago   |           |
| anaconda3         | 12 hours ago   |           |
| Contacts          | 2 months ago   |           |
| Documents         | a month ago    |           |
| Downloads         | an hour ago    |           |
| eclipse           | a month ago    |           |
| eclipse-workspace | a month ago    |           |
| Favorites         | 2 months ago   |           |
| Links             | 2 months ago   |           |
| Music             | 2 months ago   |           |
| OneDrive          | 13 days ago    |           |
| Saved Games       | 2 months ago   |           |
| Searches          | 2 months ago   |           |
| Tracing           | a month ago    |           |
| Videos            | a month ago    |           |
| Untitled.ipynb    | 5 minutes ago  | 39.2 kB   |
| Untitled1.ipynb   | 22 minutes ago | 2.09 kB   |

## Step 4:

Click new → Python3



The JupyterLab interface shows the 'Files' tab. The 'New' button is highlighted with a yellow box, and a dropdown menu is open, showing options: 'Notebook: Python 3', 'Other: Text File', 'Folder', and 'Terminal'. The 'Python 3' option is selected.

| Name              | Last Modified | File size |
|-------------------|---------------|-----------|
| 3D Objects        | 2 months ago  |           |
| anaconda3         | 12 hours ago  |           |
| Contacts          | 2 months ago  |           |
| Documents         | a month ago   |           |
| Downloads         | an hour ago   |           |
| eclipse           | a month ago   |           |
| eclipse-workspace | a month ago   |           |
| Favorites         | 2 months ago  |           |

## Step 5: Write or paste the python code

led2 - Jupyter Notebook x Home Page - Select or create a x Untitled3 - Jupyter Notebook x +

/Untitled3.ipynb?kernel\_name=python3

jupyter Untitled3 Last Checkpoint: a few seconds ago (unsaved changes) Logout

File Edit View Insert Cell Help Widgets Trusted Python 3

Run

```
In [ ]: import numpy as np
import matplotlib.pyplot as plt

x = np.array([80, 85, 90, 95, 100, 105, 110, 115, 120, 125])
y = np.array([240, 250, 260, 270, 280, 290, 300, 310, 320, 330])

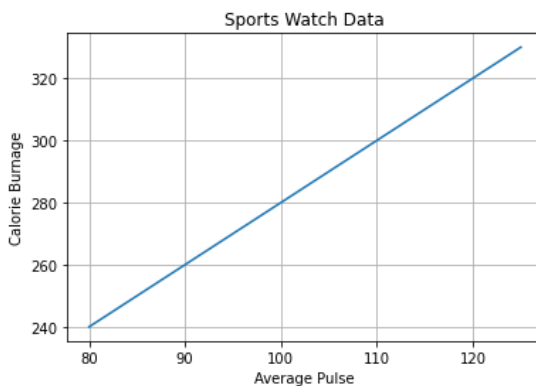
plt.title("Sports Watch Data")
plt.xlabel("Average Pulse")
plt.ylabel("Calorie Burnage")

plt.plot(x, y)

plt.grid()

plt.show()
```

## Step 6 : click run



In [ ]:

**Ex: No: 8A      COMPARE THE ELEMENTS OF THE TWO PANDAS SERIES USING PANDAS LIBRARY.****Aim:**

To write a python program to compare the elements of the two Pandas Series using Pandas library.

*Sample Series:* [2, 4, 6, 8, 10], [1, 3, 5, 7, 10]

**Algorithm:**

1. Start
2. Pandas is a Python library used for working with data sets. It has functions for analyzing, cleaning, exploring, and manipulating data.
3. Create two pandas series.
4. Compare both series using ==, > or < operators.
5. Print the result
6. Stop.

**Program:**

```
import pandas as pd
ds1 = pd.Series([2, 4, 6, 8, 10])
ds2 = pd.Series([1, 3, 5, 7, 10])
print("Series1:")
print(ds1)
print("Series2:")
print(ds2)
print("Compare the elements of the said Series:")
print("Equals:")
print(ds1 == ds2)
print("Greater than:")
print(ds1 > ds2)
print("Less than:")
print(ds1 < ds2)
```

**Output:**

```
Series1:
0    2
1    4
2    6
```

```
3    8
4   10
dtype: int64
Series2:
0     1
1     3
2     5
3     7
4    10
dtype: int64
Compare the elements of the said Series:
Equals:
0    False
1    False
2    False
3    False
4     True
dtype: bool
Greater than:
0     True
1     True
2     True
3     True
4    False
dtype: bool
Less than:
0    False
1    False
2    False
3    False
4    False
dtype: bool
```

**Result:**

Thus the python program is executed to compare two series in pandas and the output was verified.

**Ex: No: 8B      PROGRAM TO TEST WHETHER NONE OF  
THE ELEMENTS OF A GIVEN ARRAY IS ZERO  
USING NUMPY LIBRARY**

**Aim:**

To write a program to test whether none of the elements of a given array is zero using NumPy library.

**Algorithm:**

1. Start
2. Create an array using NUMPY.
3. Print the original array
4. Check all the elements whether the array contain zero or not
5. Print the result
6. stop

**Program:**

```
import numpy as np
x = np.array([1, 2, 3, 4])
print("Original array:")
print(x)
print("Test if none of the elements of the said array is zero:")
print(np.all(x))
x = np.array([0, 1, 2, 3])
print("Original array:")
print(x)
print("Test if none of the elements of the said array is zero:")
print(np.all(x))
```

**Output:**

```
Original array:
[1 2 3 4]
Test if none of the elements of the said array is zero:
True
Original array:
```

```
[0 1 2 3]
```

Test if none of the elements of the said array is zero:

False

**Result:**

Thus the python program is executed to test whether none of the elements of a given array is zero using NumPy library and the output was verified.

---

**EX: NO: 8C****PROGRAM TO PLOT A GRAPH USING  
MATPLOTLIB LIBRARY****Aim:**

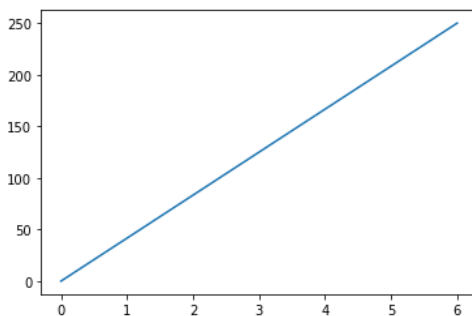
To write a python program to plot a graph using matplotlib library.

**Algorithm:**

1. Start
2. Read x and y coordinates
3. Draw a plot using the above values
4. Stop

**Program:**

```
import matplotlib.pyplot as plt
import numpy as np
xpoints = np.array([0, 6])
ypoints = np.array([0, 250])
plt.plot(xpoints, ypoints)
plt.show()
```

**Output:****Result:**

Thus, the python program is executed to plot a graph using matplotlib library and the output was verified.

**EX: NO: 8D****PYTHON PROGRAM TO RETURN****THE SPECIFIED UNIT IN SECONDS USING SCIPY LIBRARY****Aim:**

To write a python program to return the specified unit in seconds (e.g. hour returns 3600.0) using scipy library.

**Algorithm:**

1. Start
2. SciPy is a more focused on scientific implementations, it provides many built-in scientific constants. These constants can be helpful when working with Data Science.
3. Print the constants related to time from the scipy constants.
4. Stop

**Program:**

```
from scipy import constants
print(constants.minute)

print(constants.hour)
print(constants.day)
print(constants.week)
print(constants.year)
print(constants.Julian_year)
```

**Output:**

```
60.0
3600.0
86400.0
604800.0
31536000.0
31557600.0
```

**Result:**

Thus, a python program to return the specified unit in seconds using scipy library is executed and the output was verified.

**EX: NO: 9A****COPY FROM ONE FILE TO ANOTHER****Aim:**

To write a python program to copy from one file to another.

**Algorithm:**

The `shutil.copyfile()` method in Python is used to copy the content of the source file to the destination file. The metadata of the file is not copied. Source and destination must represent a file and destination must be writable. If the destination already exists then it will be replaced with the source file otherwise a new file will be created. If source and destination represent the same file then the `SameFileError` exception will be raised.

1. Start
2. Read the source file name
3. Read the destination file name
4. Copy the contents of source file to destination file using `copyfile` method
5. Print the destination file
6. Stop.

**Program:**

```
from shutil import copyfile
sourcefile = input("Enter source file name: ")
destinationfile = input("Enter destination file name: ")
copyfile(sourcefile, destinationfile)
print("File copied successfully!")
c = open(destinationfile, "r")
print(c.read())
c.close()
```

```
print()
```

```
print()
```

**Output:**

Enter source file name: file1.txt

Enter destination file name: file2.txt

File copied successfully!

Sunflower

Jasmine

Roses

Result:

Thus, a python program to copy from one file to another is executed and the output was verified.

---

**Ex: No : 9B****WORD COUNT FROM A FILE****Aim:**

To write a python program to count number of words in a file.

**Data.txt**

A file is a collection of data stored on a secondary storage device like hard disk. They can be easily retrieved when required. Python supports two types of files. They are Text files & Binary files.

**Algorithm:**

1. Start
2. Open the file and find number of words using data.split()
3. Print number of words
4. Stop

**Program:**

```
file = open("F:\Data.txt", "rt")
data = file.read()
words = data.split()

print('Number of words in text file :', len(words))
```

**Output:**

Number of words in text file : 36

**Result:**

Thus, a python program to count number of words in a file is executed and the output was verified.

**Ex: No : 9C****FINDING LONGEST WORD IN A FILE****Aim:**

To write a python program to find longest word in a file.

**Data.txt**

A file is a collection of data stored on a secondary storage device like hard disk. They can be easily retrieved when required. Python supports two types of files. They are Text files & Binary files.

**Algorithm:**

5. Start
6. Open the file and find the word length of each word by the `infile.read().split()` function.
7. In Python this function will read the entire file as a single string of text and then split it into a list of substrings based on a delimiter. The delimiter can be a character, a string, or a list of characters or strings. If no delimiter is specified, the string is split at white-space characters.
8. Print the word which has maximum length
9. Stop

**Program:**

```
def longest_word(filename):  
    with open(filename, 'r') as infile:  
        words = infile.read().split()  
        max_len = len(max(words, key=len))  
        return [word for word in words if len(word) == max_len]  
print(longest_word('F:\Data.txt'))
```

**Output:**

['collection']

**Result:**

Thus, a python program to find longest word in a file is executed and the output was verified.

**Ex: No : 10 A DIVIDE BY ZERO ERROR USING EXCEPTION HANDLING****Aim:**

To write a python program to handle divide by zero error using exception handling.

**Algorithm:**

1. Start
2. Read three numbers n,d,c
3. Calculate  $q=n/(d-c)$
4. Print quotient q
5. If any division by zero error occurs then catch the exception and print
6. Stop

**Program:**

```
n=int(input("Enter the value of n:"))
d=int(input("Enter the value of d:"))
c=int(input("Enter the value of c:"))
try:
    q=n/(d-c)
    print("Quotient:",q)
except ZeroDivisionError:
    print("Division by Zero!")
```

**Output:**

```
Enter the value of n:10
Enter the value of d:5
Enter the value of c:5
Division by Zero!
```

**Result:**

Thus, a python program to handle divide by zero error using exception handling is executed and the output was verified.

**Ex: No : 10 B****VOTERS AGE VALIDITY****Aim:**

To write a python program to check voters age validity.

**Algorithm:**

1. Start
2. Read the year of birth
3. Calculate the age and print
4. Check if age is less than 18 then print you are not eligible to vote. Otherwise print you are eligible to vote.
5. Stop

**Program:**

```
import datetime
Year_of_birth = int(input("In which year you took birth:- "))
current_year = datetime.datetime.now().year
Current_age = current_year - Year_of_birth
print("Your current age is ",Current_age)

if(Current_age<=18):
    print("You are not eligible to vote")
else:
    print("You are eligible to vote")
```

**Output:**

```
In which year you took birth:- 1981
Your current age is 40
You are eligible to vote
In which year you took birth:- 2011
Your current age is 10
You are not eligible to vote
```

**Result:**

Thus, a python program to check voters age validity is executed and the output was verified

**Ex: No : 10 C****STUDENT MARK RANGE VALIDATION****Aim:**

To write a python program to perform student mark range validation.

**Algorithm:**

1. Start
2. Read the mark
3. Check if  $\text{Mark} < 0$  or  $\text{Mark} > 100$  then print mark value is out of range.
4. Otherwise, that is if mark is in between 0 to 100 then print mark is in range
5. Stop

**Program:**

```
Mark = int(input("Enter the Mark: "))
if Mark < 0 or Mark > 100:
    print("The value is out of range, try again.")
else:
    print("The Mark is in the range")
```

**Output:**

Enter the Mark: 150  
The value is out of range, try again.

Enter the Mark: 98  
The Mark is in the range

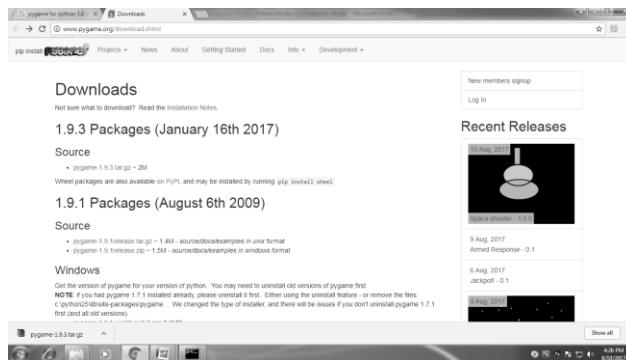
**Result:**

Thus, a python program to perform student mark range validation is executed and the output was verified

# EXPLORING PYGAME

## Steps

1. Install python 3.6.2 into C:\
2. Go to this link to install pygame [www.pygame.org/download.shtml](http://www.pygame.org/download.shtml)



3. Click  
pygame-1.9.3.tar.gz ~ 2M and download zip file
4. Extract the zip file into C:\Python36-32\Scripts folder
5. Open command prompt
6. Type the following command

```
C:\>py -m pip install pygame --user
```

## Collecting pygame

Downloading pygame-1.9.3-cp36-cp36m-win32.whl (4.0MB)

[illegible]

171kB/s

## Installing collected packages: pygame

Successfully installed pygame-1.9.3

7. Now, pygame installed successfully

8. To see if it works, run one of the included examples in pygame-1.9.3

- Open command prompt
- Type the following

```
C:\>cd Python36-32\Scripts\pygame-1.9.3
```

```
C:\Python36-32\Scripts\pygame-1.9.3>cd examples
```

```
C:\Python36-32\Scripts\pygame-1.9.3\examples>aliens.py
```

```
C:\Python36-32\Scripts\pygame-1.9.3\examples>
```

**Ex: No: 12                    SIMULATE BOUNCING BALL USING PYGAME****Aim:**

To write a Python program to bouncing ball in Pygame.

**Algorithm:**

1. Start the program
2. Set screen size and background color.
3. Set speed of moving ball.
4. Create a graphical window using set\_mode()
5. Set caption
6. Load the ball image and create a rectangle area covering the image
7. Use blit() method to copy the pixel color of the ball to the screen
8. Set background color of screen and use flip() method to make all images visible.
9. Move the ball in specified speed.
10. If ball hits the edges of the screen reverse the direction.
11. Create an infinite loop and Repeat steps 9 and 10 until user quits the program
12. Stop the program

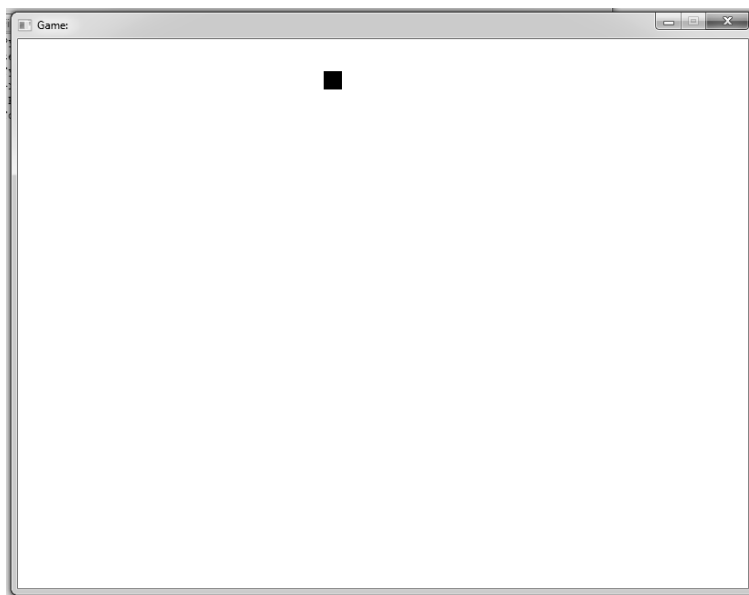
**Program:**

```
import pygame
pygame.init()
window_w = 800
window_h = 600
white = (255, 255, 255)
black = (0, 0, 0)
FPS = 120
window = pygame.display.set_mode((window_w, window_h))
pygame.display.set_caption("Game: ")
clock = pygame.time.Clock()
def game_loop():
    block_size = 20
    velocity = [1, 1]
    pos_x = window_w/2
```

```
pos_y = window_h/2
running = True
while running:
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            pygame.quit()
            quit()
    pos_x += velocity[0]
    pos_y += velocity[1]
    if pos_x + block_size > window_w or pos_x < 0:
        velocity[0] = -velocity[0]
    if pos_y + block_size > window_h or pos_y < 0:
        velocity[1] = -velocity[1]
    window.fill(white)

    pygame.draw.rect(window, black, [pos_x, pos_y, block_size, lock_size])
    pygame.display.update()
    clock.tick(FPS)
game_loop()
```

Output:



**Result:**

Thus, a python program to bouncing ball in Pygame is executed and the output was verified

---