EE-527: Machine Learning Laboratory

Assignment 1: Python Basics

Group Members:

1. RAJENDRA KUJUR (214161008)
2. ROHIT RAJ SINGH CHAUHAN (214161009)

1. Print 'Hello World!'.

In [1]:

```python
print('Hello World')
```

```
Hello World
```

2. User input two numbers a and b. Perform the following algebraic operations c = a+b, d= a-b, e= a*b, f=a/b and g=a%b and print their results.

In [2]:

```python
a = int(input('Enter a :'))
b = int(input('Enter b :'))
print(f'{a} + {b} : {a+b}')
print(f'{a} - {b} : {a-b}')
print(f'{a} * {b} : {a*b}')
print(f'{a} / {b} : {a/b}')
print(f'{a} % {b} : {a%b}')
```

```
Enter a :4
Enter b :6
4 + 6 : 10
4 - 6 : -2
4 * 6 : 24
4 / 6 : 0.6666666666666666
4 % 6 : 4
```

3. Print the factorial of a positive number 'a' given as a user input.

In [3]:

```
# Returns the factorial of given number
def factorial(number):

    result = 1
    if number == 0:
        return result
    else:
        factor = 2
        while factor <= number:
            result *= factor
            factor += 1
    return result

a = int(input('Enter a positive number :'))
print(factorial(a))
```

```
Enter a positive number :2
2
```

4. Write a function to print all prime numbers in an interval [a,b]. Interval is to be obtained as a user input.

In [4]:

```
# checks whether a number is prime or not
def isPrime(number):
#   1 is neither prime nor composite
    if number == 1:
        return False
#     2 and 3 both are prime numbers
    elif number == 2 or number == 3:
        return True
#     if number is completely divisible by 2 or 3 then number is not prime
    elif number % 2 == 0 or number % 3 == 0:
        return False
    else:
        divisor = 5
        while divisor**2 <= number:
            if number % divisor == 0 or number % (divisor + 2) == 0:
                return False
            divisor += 5
    return True


# prints prime number in the given range
def primeInRange(start, end):
    for number in range(start,end+1):
        if isPrime(number):
            print(number)

# Execution begins here
a = int(input('Enter a : '))
b = int(input('Enter b :'))
primeInRange(a, b)
```

```
Enter a : 2
Enter b :15
2
3
5
7
11
13
```

5. User input two numbers a and b. Print their lowest common multiple (LCM).

In [5]:

```
# extended euclidean algorithm to find GCD of two numbers
def gcd(num_1, num_2):
    if num_2 == 0:
        return num_1
    else:
        return gcd(num_2, num_1 % num_2)


#     returns the LCM of two given numbers
def LCM(num_1, num_2):
#     since product of two numbers is same as product of LCM and GCD of those numbe
    return (num_1 * num_2)/gcd(num_1, num_2)

# Execution begins here
a = int(input('Enter a : '))
b = int(input('Enter b : '))
print(f'LCM({a},{b}) : {LCM(a,b)}')
```

```
Enter a : 4
Enter b : 15
LCM(4,15) : 60.0
```

6. Create a list of length n = 15. Sort in descending order and print the sorted list as well as the sorted indices. Use bubble sort algorithm.

In [6]:

```
import random

list_of_15 = [random.randint(1,100) for i in range(15)]

print(f'Unsorted List : {list_of_15}')

number_of_elements = 15

index = [i for i in range(15)]
for i in range(number_of_elements-1):
    for j in range(number_of_elements-i-1):
#         Inversion of bubble sort (since the output needed in descending order)
        if list_of_15[j] < list_of_15[j+1] and i != j:
            list_of_15[j] , list_of_15[j+1] = list_of_15[j+1] , list_of_15[j]
            index[j] , index[j+1] = index[j+1] , index[j]

print(f'Sorted List (Desceding) : {list_of_15}')
print(f'Sorted index of numbers : {index}')
```

```
Unsorted List : [61, 71, 81, 21, 89, 56, 28, 44, 9, 27, 49, 38, 95, 7
1, 14]
Sorted List (Desceding) : [95, 89, 81, 71, 71, 61, 56, 49, 44, 38, 28,
27, 21, 14, 9]
Sorted index of numbers : [12, 4, 2, 1, 13, 0, 5, 10, 7, 11, 6, 9, 3,
14, 8]
```

7. Repeat the previous program for sorting in ascending order. Use numpy array instead of list.

In [7]:

```
import numpy as np

numbers = np.array([random.randint(1,100) for i in range(15)])
print(f'Unsorted Array : {numbers}')

index = [i for i in range(15)]

for i in range(number_of_elements-1):
    for j in range(number_of_elements-i-1):
        if numbers[j] > numbers[j+1] and i != j:
#             Swapping the numbers if numbers at j index is greater than j+1 th ind
            numbers[j] , numbers[j+1] = numbers[j+1] , numbers[j]
#             swapping the indices
            index[j] , index[j+1] = index[j+1] , index[j]

print(f'Sorted Array (Ascending) : {numbers}')
print(f'Sorted index of numbers : {index}')
```

```
Unsorted Array : [25 26 96 47 75 71 86 56 93 72 89 43 10 63 57]
Sorted Array (Ascending) : [10 25 26 43 47 56 71 72 57 63 75 86 89 93
96]
Sorted index of numbers : [12, 0, 1, 11, 3, 7, 5, 9, 14, 13, 4, 6, 10,
8, 2]
```

8. Print a matrix M ∈ R m×n having random values in the given range [-2, 5]. m and n are to be given as a user input.

In [8]:

```
import random

# Execution begins here
m = int(input('Enter m : '))
n = int(input('Enter n : '))

#  Stores the entrie matrix
matrix = np.array([[random.randrange(-2,6) for x in range(n)] for y in range(m)])

print(matrix)
```

```
Enter m : 4
Enter n : 6
[[ 5  2  3  2  0  2]
 [ 3  4  2  0  4  1]
 [ 2  3  4 -1 -2  5]
 [-2  0  3  2  3  4]]
```

9. Program to multiply two random matrices M1 ∈ R m×n , M2 ∈ R n×p (Don't use built-in functions). Compare the result obtained with the built-in function.

In [9]:

```
import random
import numpy as np

# Function to multiply two matrices
def multiply(mat_1, mat_2, row_1, column_1, row_2, column_2):
    result = np.array([[0 for x in range(column_2)] for y in range(row_1)])
    for i in range(row_1):
        for j in range(column_2):
            for k in range(row_2):
                result[i][j] += mat_1[i][k] * mat_2[k][j]
    return result


# Function to check the equality between two matrices
def checkEquality(matrix_1, matrix_2):
#     Checks for the dimensions first
    if len(matrix_1) == len(matrix_2):
        for i in range(len(matrix_1)):
            for j in range(len(matrix_1[i])):
                if matrix_1[i][j] != matrix_2[i][j]:
                    return False
        else:
            return True
    else:
        return False


# execution begins here
# first matrix of 2 x 3
row_1 = 2
column_1 = 3
mat_1 = np.array([[random.randrange(-2,6) for x in range(column_1)] for y in range(
print(f'Matrix 1 : \n{mat_1}')

#second matrix of 3 x 4
row_2 = 3
column_2 = 4
mat_2 = np.array([[random.randrange(-2,6) for x in range(column_2)] for y in range(
print(f'Matrix 2 : \n{mat_2}')

# using User-defined function
func_product = multiply(mat_1, mat_2, row_1, column_1, row_2, column_2)
print(f'From User Defined Function : \n{func_product}')

# Using in-built function
np_product = np.dot(mat_1, mat_2)
print(f'From Built in Function : \n{np_product}')

# Check for the equality of results
if checkEquality(func_product, np_product):
    print('Both are same.')
else:
    print('Both are different')
```

```
Matrix 1 :
[[ 5 -2  3]
 [-1  0  2]]
Matrix 2 :
[[ 4  2  5 -2]
```

```
 [ 5  0 -2  2]
 [-1  0 -1  2]]
From User Defined Function :
[[ 7 10 26 -8]
 [-6 -2 -7  6]]
From Built in Function :
[[ 7 10 26 -8]
 [-6 -2 -7  6]]
Both are same.
```

10. File operations :write

1. Generate a set of n=100 random points X = {xi}, i = 1, … n, xi ∈ R10

In [10]:

```python
import numpy as np
import random

n = 100
x_i =[[random.randint(0,100) for i in range(10)] for j in range(100)]
print(x_i)
```

```
[[15, 81, 77, 46, 25, 43, 97, 34, 46, 65], [57, 79, 91, 62, 90, 33, 5
9, 46, 34, 48], [25, 92, 95, 81, 60, 45, 90, 51, 15, 71], [15, 79, 67,
35, 19, 31, 79, 59, 19, 11], [3, 16, 53, 86, 60, 34, 96, 12, 97, 99],
[95, 90, 41, 56, 67, 14, 51, 34, 62, 73], [11, 98, 0, 57, 81, 68, 100,
2, 16, 53], [28, 35, 42, 69, 78, 42, 59, 5, 5, 22], [74, 24, 56, 56, 5
8, 16, 16, 31, 43, 92], [23, 100, 47, 69, 84, 90, 2, 84, 50, 4], [88,
100, 61, 41, 38, 86, 14, 83, 24, 58], [56, 46, 54, 97, 40, 3, 49, 87,
77, 12], [61, 71, 49, 44, 45, 87, 96, 81, 86, 40], [77, 52, 72, 11, 5,
60, 97, 87, 41, 0], [26, 26, 12, 98, 94, 36, 25, 49, 93, 29], [1, 95,
10, 65, 79, 0, 25, 94, 75, 0], [0, 76, 56, 61, 91, 42, 33, 9, 99, 70],
[89, 8, 76, 10, 52, 3, 76, 82, 8, 1], [47, 99, 62, 61, 85, 76, 4, 83,
90, 7], [80, 4, 83, 42, 92, 64, 4, 26, 38, 77], [90, 4, 58, 73, 92, 4
5, 3, 2, 47, 81], [54, 30, 5, 92, 98, 96, 13, 44, 99, 27], [22, 29, 1,
83, 71, 57, 95, 76, 84, 26], [27, 10, 2, 66, 63, 26, 60, 81, 33, 43],
[97, 32, 14, 66, 82, 81, 51, 1, 74, 68], [78, 97, 1, 49, 15, 63, 1, 6
5, 71, 42], [37, 67, 39, 62, 78, 32, 27, 52, 93, 15], [53, 43, 43, 62,
50, 78, 80, 79, 2, 76], [44, 94, 92, 84, 71, 69, 21, 95, 34, 12], [4,
92, 34, 42, 79, 100, 83, 87, 85, 18], [54, 72, 96, 51, 0, 79, 4, 4, 2,
86], [96, 22, 48, 49, 15, 9, 83, 60, 73, 13], [48, 80, 12, 76, 81, 16,
39, 80, 23, 84], [40, 68, 52, 12, 13, 90, 34, 3, 42, 91], [73, 39, 4,
19, 86, 28, 26, 35, 50, 7], [29, 71, 15, 73, 6, 65, 88, 70, 52, 19],
[69, 24, 68, 40, 85, 2, 45, 94, 67, 67], [81, 34, 27, 45, 93, 99, 68,
26, 97, 43], [34, 35, 27, 62, 95, 51, 84, 51, 86, 75], [100, 86, 84, 2
0, 17, 82, 39, 80, 34, 69], [41, 20, 49, 17, 64, 17, 29, 64, 37, 54],
[4, 34, 28, 57, 37, 14, 28, 85, 100, 71], [36, 16, 30, 24, 97, 44, 92,
75, 8, 21], [21, 3, 99, 71, 17, 25, 22, 18, 78, 83], [71, 13, 43, 28,
60, 27, 42, 56, 100, 9], [79, 35, 71, 4, 88, 83, 19, 74, 84, 49], [76,
36, 18, 73, 57, 41, 94, 38, 54, 32], [88, 18, 70, 11, 31, 39, 33, 90,
60, 82], [67, 37, 26, 10, 0, 64, 5, 79, 2, 27], [48, 92, 21, 94, 51, 7
2, 89, 38, 37, 81], [30, 70, 17, 84, 19, 93, 17, 30, 40, 10], [20, 61,
94, 59, 18, 33, 97, 24, 4, 80], [6, 63, 99, 24, 99, 23, 80, 48, 14, 9
7], [100, 63, 77, 48, 91, 30, 93, 29, 89, 53], [89, 99, 61, 45, 53, 2
9, 72, 31, 90, 39], [54, 40, 96, 68, 9, 82, 22, 40, 10, 71], [36, 57,
37, 79, 62, 80, 19, 93, 23, 91], [67, 9, 17, 80, 88, 75, 70, 41, 100,
50], [98, 71, 16, 76, 86, 20, 6, 64, 50, 92], [75, 8, 27, 98, 50, 74,
43, 59, 88, 91], [7, 18, 77, 52, 29, 81, 74, 18, 70, 96], [89, 95, 95,
28, 54, 39, 1, 8, 66, 87], [94, 20, 63, 84, 18, 30, 9, 55, 64, 67], [3
2, 27, 29, 51, 75, 38, 8, 93, 98, 8], [32, 27, 28, 71, 84, 85, 12, 6,
24, 53], [88, 45, 16, 69, 52, 22, 81, 41, 45, 22], [52, 39, 22, 45, 5
5, 31, 75, 88, 40, 93], [58, 18, 74, 36, 94, 48, 13, 34, 67, 97], [81,
76, 89, 33, 96, 41, 25, 74, 20, 64], [55, 14, 50, 72, 45, 14, 95, 2, 9
5, 65], [85, 67, 96, 89, 64, 15, 27, 99, 57, 0], [21, 78, 48, 74, 7, 1
4, 54, 47, 18, 48], [92, 19, 41, 20, 14, 18, 10, 10, 23, 38], [34, 37,
4, 97, 79, 43, 5, 30, 66, 88], [44, 7, 26, 36, 13, 43, 82, 2, 10, 54],
[83, 55, 71, 1, 77, 70, 23, 59, 49, 32], [54, 77, 55, 8, 23, 94, 50, 9
8, 45, 76], [39, 53, 78, 63, 45, 21, 75, 100, 24, 8], [65, 25, 22, 79,
44, 4, 7, 22, 90, 2], [13, 6, 29, 59, 17, 23, 52, 88, 56, 3], [96, 77,
64, 65, 59, 0, 43, 30, 42, 12], [98, 15, 15, 46, 24, 54, 8, 76, 77, 2
2], [59, 28, 0, 37, 95, 56, 55, 99, 13, 60], [68, 1, 62, 68, 13, 74, 9
1, 46, 68, 72], [18, 48, 85, 61, 58, 49, 6, 43, 74, 16], [68, 59, 1, 8
1, 13, 3, 74, 41, 69, 34], [44, 98, 53, 91, 45, 17, 22, 56, 92, 3], [2
8, 79, 88, 45, 20, 85, 24, 3, 42, 45], [60, 73, 78, 21, 9, 41, 24, 88,
55, 56], [76, 19, 20, 37, 18, 96, 28, 9, 67, 95], [2, 52, 90, 73, 7, 6
7, 20, 52, 44, 80], [78, 40, 38, 93, 81, 100, 4, 45, 27, 32], [77, 39,
90, 45, 28, 40, 57, 78, 1, 56], [91, 15, 12, 77, 53, 47, 2, 35, 61, 9
3], [58, 66, 35, 48, 35, 90, 38, 81, 85, 100], [74, 48, 11, 23, 38, 6
8, 65, 59, 70, 62], [56, 33, 62, 59, 85, 81, 37, 41, 5, 57], [55, 49,
22, 17, 87, 69, 45, 31, 70, 51], [95, 78, 34, 16, 84, 80, 64, 87, 30,
21], [99, 88, 11, 78, 14, 29, 51, 49, 75, 29]]
```

2. Write the points to a csv (https://en.wikipedia.org/wiki/Comma-separated_values (https://en.wikipedia.org/wiki/Comma-separated_values)) file

In [11]:

```
import csv
import pandas as pd

df = pd.DataFrame(x_i)
df.to_csv("points.csv", index=False)
```

11. File operations:read

1. Read the csv (https://en.wikipedia.org/wiki/Comma-separated_values (https://en.wikipedia.org/wiki/Comma-separated_values)) file generated in the previous program to a matrix. Each column of the matrix should represent a vector

In [12]:

```
import csv
import pandas as pd
import numpy as np


df_1=pd.read_csv("points.csv")
df_1=df_1.T
print(df_1)
```

```
   0   1   2   3   4   5    6   7   8    9  ...  90   91  92  93   94  
95  \
0  15  57  25  15   3  95   11  28  74   23  ...   2   78  77  91   58  
74   
1  81  79  92  79  16  90   98  35  24  100  ...  52   40  39  15   66  
48   
2  77  91  95  67  53  41    0  42  56   47  ...  90   38  90  12   35  
11   
3  46  62  81  35  86  56   57  69  56   69  ...  73   93  45  77   48  
23   
4  25  90  60  19  60  67   81  78  58   84  ...   7   81  28  53   35  
38   
5  43  33  45  31  34  14   68  42  16   90  ...  67  100  40  47   90  
68   
6  97  59  90  79  96  51  100  59  16    2  ...  20    4  57   2   38  
65   
7  34  46  51  59  12  34    2   5  31   84  ...  52   45  78  35   81  
59   
8  46  34  15  19  97  62   16   5  43   50  ...  44   27   1  61   85  
70   
9  65  48  71  11  99  73   53  22  92    4  ...  80   32  56  93  100  
62   

   96  97  98  99  
0  56  55  95  99  
1  33  49  78  88  
2  62  22  34  11  
3  59  17  16  78  
4  85  87  84  14  
5  81  69  80  29  
6  37  45  64  51  
7  41  31  87  49  
8   5  70  30  75  
9  57  51  21  29  

[10 rows x 100 columns]
```

2.Compute the following: C = 1 n T ∑ n i=1 (x i − μ)(x i − μ) , where μ = 1 n ∑ n i=1 x i , i = 1, … n , x i = [x i1 , … x i10] T is a column vector

In [13]:

```
import numpy as np

sigma_xi = 0
for i in range(len(matrix)):
    sigma_xi += sum(matrix[i])

mu = sigma_xi/len(matrix)
row_vector = np.array([])

for i in range(len(matrix)):
    row_vector = np.append(row_vector, sum(matrix[i]) - mu)

c = (np.dot(row_vector, row_vector.transpose()))/len(matrix)
print(c)
```

```
3.1875
```

12. Define a class for a complex number a + jb . Define memeber functions to do basic operations conjugate, absolute value, addition, subtraction, multiplication, division and angle. Define two complex numbers c1 , c2 and print the results of the following operations c1 + c2 , c1 − c2 , c1 ∗ c2 , c1/c2 , |c1| , |c2| , ∠c1 , ∠c2 .

In [14]:

```
import math

# Class named complex number
class ComplexNumber:

# Initializes the class
  def __init__(self, real, imaginary):
    self.real = real
    self.imaginary = imaginary

#     Returns the conjugate of given complex number
  def conjugate(self):
    result = ComplexNumber(0,0)
    result.real = self.real
    result.imaginary = (-1)*self.imaginary
    return result

#     Returns the absolute value of given complex number
  def absolute(self):
    return (self.real**2 + self.imaginary**2)**0.5

#     Returns the sum of two given complex numbers
  def addition(self, number):
    result = ComplexNumber(0,0)
    result.real = self.real + number.real
    result.imaginary = self.imaginary + number.imaginary
    return result

#     Returns the difference between two complex numbers
  def subtraction(self, number):
    result = ComplexNumber(0,0)
    result.real = self.real - number.real
    result.imaginary = self.imaginary - number.imaginary
    return result

#     Returns the product of given two complex numbers
  def multiplication(self, number):
    result = ComplexNumber(0,0)
    result.real = self.real*number.real - self.imaginary*number.imaginary
    result.imaginary = self.imaginary*number.real + self.real*number.imaginary
    return result

#     Returns the division of two given complex numbers
  def division(self, number):
    result = ComplexNumber(0, 0)
    denominator = number.real**2 + number.imaginary**2
    result.real = (self.real*number.real + self.imaginary*number.imaginary)/denomin
    result.imaginary = (self.imaginary*number.real - self.real*number.imaginary)/de
    return result

#     Displays the complex number
  def display(self):
    if self.imaginary < 0:
        print(f'{self.real}{self.imaginary}j')
    else:
        print(f'{self.real}+{self.imaginary}j')


# Execution begins here
```

```python
c1 = ComplexNumber(5,-6)
print('First Number : ',end='')
c1.display()
c2 = ComplexNumber(3,8)
print('Second Number : ',end='')
c2.display()

# Conjugate
c3 = ComplexNumber(0,0)
c3 = c1.conjugate()
print('Conjugate of first number, : ',end='')
c3.display()
c3 = ComplexNumber(0,0)
c3 = c2.conjugate()
print('Conjugate of second Number : ',end='')
c3.display()


# Absolute
print(f'Absolute of first number : {c1.absolute()}')
print(f'Absolute of second number : {c2.absolute()}')

# Addition
c3 = ComplexNumber(0,0)
c3 = c1.addition(c2)
print('Addition of the two numbers : ',end='')
c3.display()


# Subtraction
c3 = ComplexNumber(0,0)
c3 = c1.subtraction(c2)
print('Difference between the two numbers : ',end='')
c3.display()


# Multiplication
c3 = ComplexNumber(0,0)
c3 = c1.multiplication(c2)
print('Product of the two numbers : ',end='')
c3.display()


# Division
c3 = ComplexNumber(0,0)
c3 = c1.division(c2)
print('Division of the two numbers : ',end='')
c3.display()


# Angles
print(f'Angle of First number in degrees: {math.degrees(math.atan(c1.imaginary/c1.r
print(f'Angle of Second number in degrees : {math.degrees(math.atan(c2.imaginary/c2
```

```
First Number : 5-6j
Second Number : 3+8j
Conjugate of first number, : 5+6j
Conjugate of second Number : 3-8j
Absolute of first number : 7.810249675906654
Absolute of second number : 8.54400374531753
Addition of the two numbers : 8+2j
```

```
Difference between the two numbers : 2-14j
Product of the two numbers : 63+22j
Division of the two numbers : -0.4520547945205479-0.7945205479452054
j
Angle of First number in degrees: -50.19442890773481
Angle of Second number in degrees : 69.44395478041653
```

13. Plot the function y = 3x + 2 with x ∈ [−10, 10] .

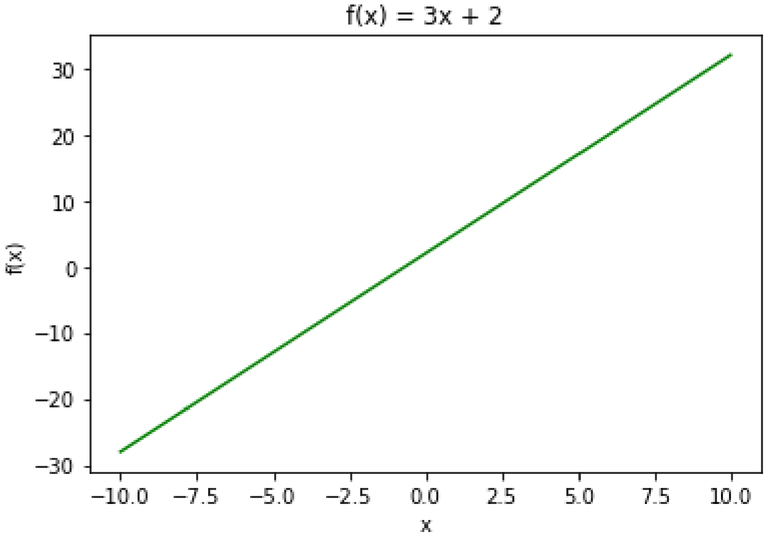
In [15]:

```
import numpy as np
import matplotlib.pyplot as plt

# calculates the function value and returns
def f(x):
    return (3*x + 2)

# Divides the points into 10000
x = np.linspace(-10, 10, 10000)
plt.plot(x, f(x), color = 'green')

plt.xlabel('x')
plt.ylabel('f(x)')
plt.title('f(x) = 3x + 2')
plt.show()
```



14. Scatter plot

1. Generate a set of n = 100 points, X = {x i }, i = 1, … n, x i ∈ R 2 within an ellipse + a 2 b 2[μ x , μ y] = [5, −5] and has a major axis 2a = 10 and minor axis 2b = 5

In [16]:

```
import random
import matplotlib.pyplot as plt


# Ellipse Class
class ellipse:

# Initializes the ellipse class
  def __init__(self, origin_x, origin_y, major_axis, minor_axis):
      self.origin_x = origin_x
      self.origin_y = origin_y
      self.major_axis = major_axis
      self.minor_axis = minor_axis

# Checks whether the point lies inside the ellipse or not
  def isInsideEllipse(self, x_value, y_value):
      if ((x_value - self.origin_x)**2)/((self.major_axis/2)**2) + ((y_value - self
          return True
      else:
          return False


# Execution begins here
# define an ellipse with given constraints
e = ellipse(origin_x=5, origin_y=-5, major_axis=10, minor_axis=5)

points= []

# Loop continues till we get 100 valid points inside the ellipse
while len(points) < 100:
    lower_limit_x = e.origin_x - (e.major_axis/2)
    upper_limit_x = e.origin_x + (e.major_axis/2)
    lower_limit_y = e.origin_y - (e.minor_axis/2)
    upper_limit_y = e.origin_y + (e.minor_axis/2)
    new_point_x = random.uniform(lower_limit_x, upper_limit_x)
    new_point_y = random.uniform(lower_limit_y, upper_limit_y)

#     if generated point lies inside ellipse then append point
    if e.isInsideEllipse(new_point_x, new_point_y) is True:
        points.append([new_point_x, new_point_y])

print(points)
```

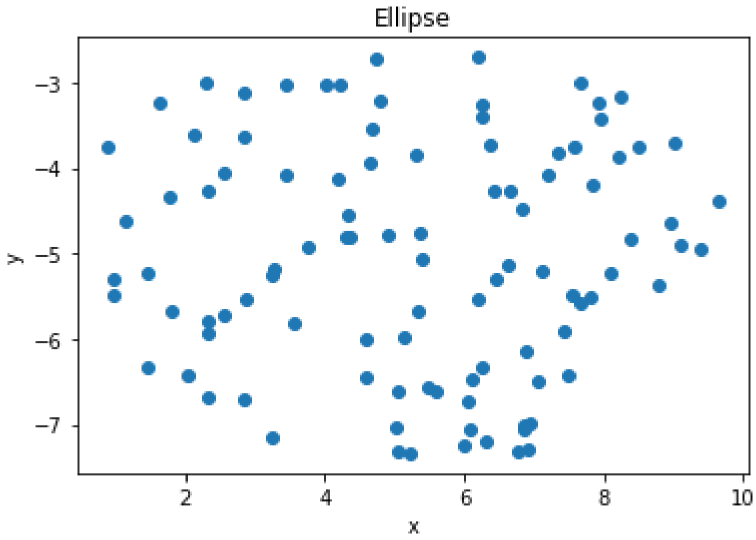
```
[[5.599869729990869, -6.613480023827052], [5.027940971196518, -7.04149
5770499155], [7.116034840140673, -5.217678478784988], [8.1073752548693
3, -5.226345616403846], [4.796727948150301, -3.222289434736764], [8.20
55740280067, -3.863198898188924], [1.633138805372918, -3.2309923274203
776], [7.573649134997496, -3.7547254293387935], [2.8417951481374826, -
3.624474163131506], [6.639499866670004, -5.14095208347622], [1.1499556
032248337, -4.610348366461399], [3.2717502768109274, -5.18432996566230
8], [4.635595331256656, -3.929632147175956], [2.1125158886467688, -3.6
04911193155459], [6.870174257607607, -7.0211554581451], [6.20002331881
4461, -2.704438409513214], [7.357024262015701, -3.815268057566325],
[6.250237975418457, -6.339175445617716], [2.8455735018683925, -6.71012
0743666442], [2.8614424412594177, -5.54609678291662], [6.4156811232655
52, -4.259565486538275], [8.366416585349867, -4.8383687033385225], [2.
3212574142595086, -6.693633192744714], [2.330837622990752, -4.27656982
5648511], [7.673572900332541, -5.588120958733864], [7.806780623992155,
-5.50591875806154], [8.94751833770518, -4.65280354506425], [6.44494344
0269263, -5.298005915408396], [6.245021213338678, -3.39956774323708],
[8.49022700057035, -3.7623315473322414], [4.895656629594191, -4.785778
267079054], [6.185449758598947, -5.537661311387255], [5.04917044430675
9, -7.3054847103143885], [6.2485614745480085, -3.250710258438743], [3.
23993927068269, -5.243037917008575], [0.9688815025449904, -5.494895182
5516215], [6.827256630907771, -4.474006870557985], [2.029030498558453,
-6.423427523725929], [1.8118447869508192, -5.666287062946601], [1.4496
786615833457, -6.319713306587601], [6.103176304753445, -6.478211080581
03], [5.993098227098042, -7.253163281450106], [3.545744553870759, -5.8
20958882752622], [7.425289359379995, -5.902950485097197], [4.317457275
400997, -4.555850072024073], [5.307977711460107, -3.8466616354970036],
[5.356502984614464, -4.7700063464464], [5.229760527788646, -7.33706292
5320409], [3.430026461530473, -3.016473370051189], [0.961436578750626
3, -5.298910095130433], [3.752669012775527, -4.914265076879451], [4.31
2420361560224, -4.815820690074698], [6.096567348248229, -7.06333890305
9592], [7.543596687922114, -5.483310302158889], [7.055847164245251, -
6.494100161320877], [5.05925415410811, -6.604048865193831], [9.6309598
70321972, -4.397720262632886], [7.931026071917522, -3.22630743550111
1], [2.5501935165795286, -4.060138272073152], [2.8252036848058673, -3.
1186938167798486], [7.664489545390529, -3.0130749820044986], [6.956093
941517727, -6.979317205786865], [7.946029948018713, -3.435584231712004
7], [6.882597829243893, -6.134556921947804], [4.728398982920476, -2.72
9079984851367], [6.86257732535934, -7.067062467391977], [7.82443240484
7754, -4.208209876853541], [5.389534610723904, -5.074106111266968],
[4.681101965342708, -3.542134301272962], [1.7725014559762597, -4.33147
3459682834], [2.546111274320002, -5.719251873885527], [4.3526088078453
204, -4.810204218831118], [6.314992141251806, -7.20095360721615], [6.3
77650160369277, -3.7280573212799295], [1.4593750107467351, -5.24035062
8065075], [7.487309234944007, -6.420274627616646], [0.888892758221404
6, -3.7490164360942337], [2.3107574286299917, -5.784288816385132], [6.
055688493692484, -6.733117713860405], [4.197375625087724, -4.138268244
2417575], [5.143790409309792, -5.977803323833319], [8.236194020740207,
-3.170102660379704], [6.916791427970032, -7.289117161468824], [2.29009
18895047964, -3.0097143547666896], [4.21298833840217, -3.0173136958787
64], [7.202065038658843, -4.074529272484794], [5.483358930313832, -6.5
76070932263413], [4.007343901087984, -3.0323411894137733], [4.59736253
8037524, -6.003894845451411], [5.345125288424847, -5.686116508161133],
[3.2268976095542232, -7.156144081823474], [3.4498091706412892, -4.0756
66319996185], [9.398134587528999, -4.945424756867453], [9.013798534225
378, -3.705321440723372], [8.780833824162979, -5.376517037040572], [9.
096445437469413, -4.911949283982571], [6.661549789001814, -4.277828073
849907], [2.3305739358825326, -5.93161447621317], [4.591603117521284,
-6.452054969977368], [6.777816869026443, -7.326660829110662]]
```

2. Scatter plot all the points.

In [17]:

```
import matplotlib.pyplot as plt

x = [points[i][0] for i in range(len(points))]
y = [points[i][1] for i in range(len(points))]
plt.scatter(x,y)
plt.xlabel('x')
plt.ylabel('y')
plt.title('Ellipse')
plt.show()
```


Ellipse
y
x