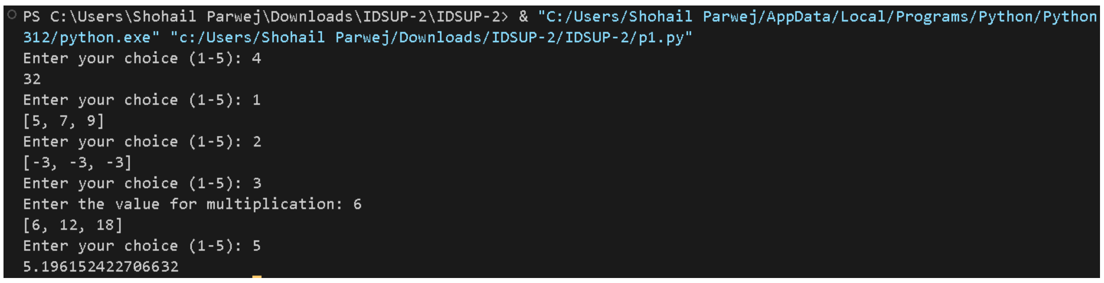
1. Write a menu-driven program to perform Addition, Subtraction, Scalar Multiplication, Dot Product and Length of vectors.

Code :-

```
#Author : Shohail Parwej

#Regd.No. : 2141016146

import math

def addition(lst, lst1):

    assert len(lst) == len(lst1)

    lst2 = []

    for i in range(len(lst)):

        lst2.append(lst[i] + lst1[i])

    print(lst2)

def subtraction(lst, lst1):

    assert len(lst) == len(lst1)

    lst2 = []

    for i in range(len(lst)):

        lst2.append(lst[i] - lst1[i])

    print(lst2)

def multiplication(lst, incval):

    lst2 = []

    for i in range(len(lst)):

        lst2.append(incval * lst[i])

    print(lst2)

def dotproduct(lst, lst1):

    assert len(lst) == len(lst1)

    lst2 = []

    for i in range(len(lst)):

        lst2.append(lst[i] * lst1[i])

    print(sum(lst2))

def calculation(lst, lst1):

    assert len(lst) == len(lst1)
    lst2 = []
    for i in range(len(lst)):
        lst2.append((lst[i] - lst1[i]) ** 2)
    res = sum(lst2)
    print(math.sqrt(res))

# Sample list initialization
lst = [1, 2, 3]
lst1 = [4, 5, 6]

while True:
    choice = int(input("Enter your choice (1-5): "))
    if choice == 1:
        addition(lst, lst1)
    elif choice == 2:
        subtraction(lst, lst1)
    elif choice == 3:
        incval = int(input("Enter the value for multiplication: "))
        multiplication(lst, incval)
    elif choice == 4:
        dotproduct(lst, lst1)
    elif choice == 5:
        calculation(lst, lst1)
    else:
        print('Wrong Input')
```

Output :



```
PS C:\Users\Shohail Parwej\Downloads\IDSUP-2\IDSUP-2> & "C:/Users/Shohail Parwej/AppData/Local/Programs/Python/Python
312/python.exe" "c:/Users/Shohail Parwej/Downloads/IDSUP-2/IDSUP-2/p1.py"
Enter your choice (1-5): 4
32
Enter your choice (1-5): 1
[5, 7, 9]
Enter your choice (1-5): 2
[-3, -3, -3]
Enter your choice (1-5): 3
Enter the value for multiplication: 6
[6, 12, 18]
Enter your choice (1-5): 5
5.196152422706632
```

2. . Write a program that takes the order of the matrix and creates a matrix in the following manner: The (ij)th entry of the matrix should be the sum of i and j. Eg: The 0 th row and 0 th column should have the value (0+0) i.e. 0 and the 0 th row and first column should have value (0+1) i.e. 1 and so on.

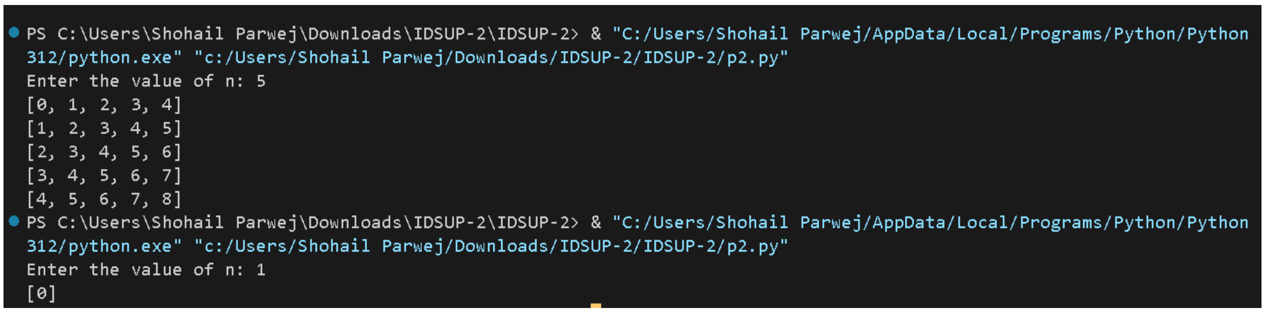
Code

#Author : Shohail Parwej

#Regd.No. : 2141016146

```python
lst = []

def creatematrix(n):
    for i in range(n):
        lst1 = []
        for j in range(n):
            lst1.append(i + j)
        lst.append(lst1)

creatematrix(int(input("Enter the value of n: ")))
for row in lst:
    print(row)
```

output



```
PS C:\Users\Shohail Parwej\Downloads\IDSUP-2\IDSUP-2> & "C:/Users/Shohail Parwej/AppData/Local/Programs/Python/Python
312/python.exe" "c:/Users/Shohail Parwej/Downloads/IDSUP-2/IDSUP-2/p2.py"
Enter the value of n: 5
[0, 1, 2, 3, 4]
[1, 2, 3, 4, 5]
[2, 3, 4, 5, 6]
[3, 4, 5, 6, 7]
[4, 5, 6, 7, 8]
PS C:\Users\Shohail Parwej\Downloads\IDSUP-2\IDSUP-2> & "C:/Users/Shohail Parwej/AppData/Local/Programs/Python/Python
312/python.exe" "c:/Users/Shohail Parwej/Downloads/IDSUP-2/IDSUP-2/p2.py"
Enter the value of n: 1
[0]
```

3. . Write two functions that extract the rows and columns of a matrix A.

Code

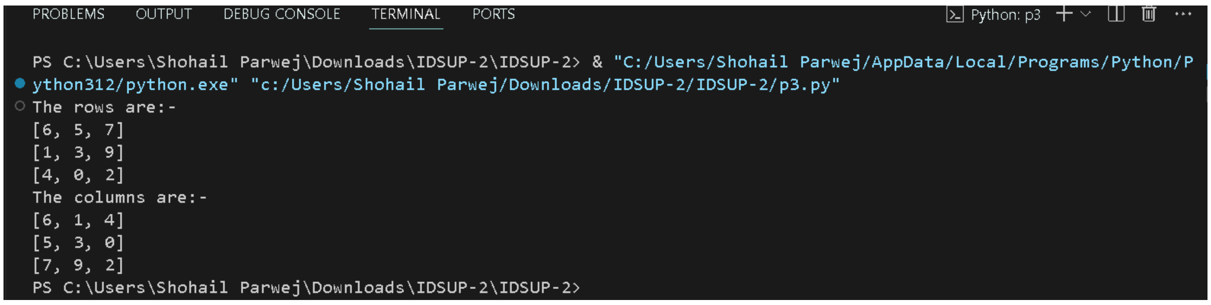
#Author : Shohail Parwej

#Regd.No. : 2141016146

```python
def extractrows(lst):
    for i in range(len(lst)):
        print(lst[i])
```

```python
def extractcolumns(lst):
    for i in range(len(lst[0])):
        column = [row[i] for row in lst]
        print(column)
#Sample List initialisation
lst = [
    [6, 5, 7],
    [1, 3, 9],
    [4, 0, 2]
]

print("The rows are:- ")
extractrows(lst)
print("The columns are:- ")
extractcolumns(lst)
```

Output



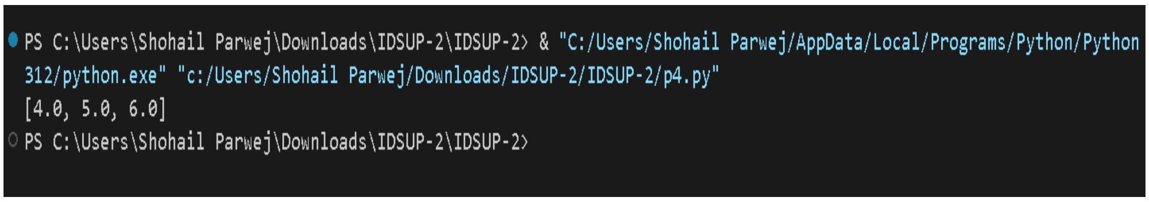
```
PS C:\Users\Shohail Parwej\Downloads\IDSUP-2\IDSUP-2> & "C:/Users/Shohail Parwej/AppData/Local/Programs/Python/P
ython312/python.exe" "c:/Users/Shohail Parwej/Downloads/IDSUP-2/IDSUP-2/p3.py"
The rows are:-
[6, 5, 7]
[1, 3, 9]
[4, 0, 2]
The columns are:-
[6, 1, 4]
[5, 3, 0]
[7, 9, 2]
PS C:\Users\Shohail Parwej\Downloads\IDSUP-2\IDSUP-2>
```

4. Write a function to compute the component-wise mean of a list of vectors. Assert the condition that the vectors must be of same length.

Code

```python
#Author : Shohail Parwej
#Regd.No. : 2141016146
def component_wise_mean(vectors):
    assert vectors, "No vectors provided"
    vector_length = len(vectors[0])
    assert all(len(vec) == vector_length for vec in vectors), "Vectors have different lengths"
    num_vectors = len(vectors)
    mean_vector = [sum(component[i] for component in vectors) / num_vectors for i in range(vector_length)]
    return mean_vector

#Sample vectors initialisation
vectors = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]
print(component_wise_mean(vectors))
```

Output



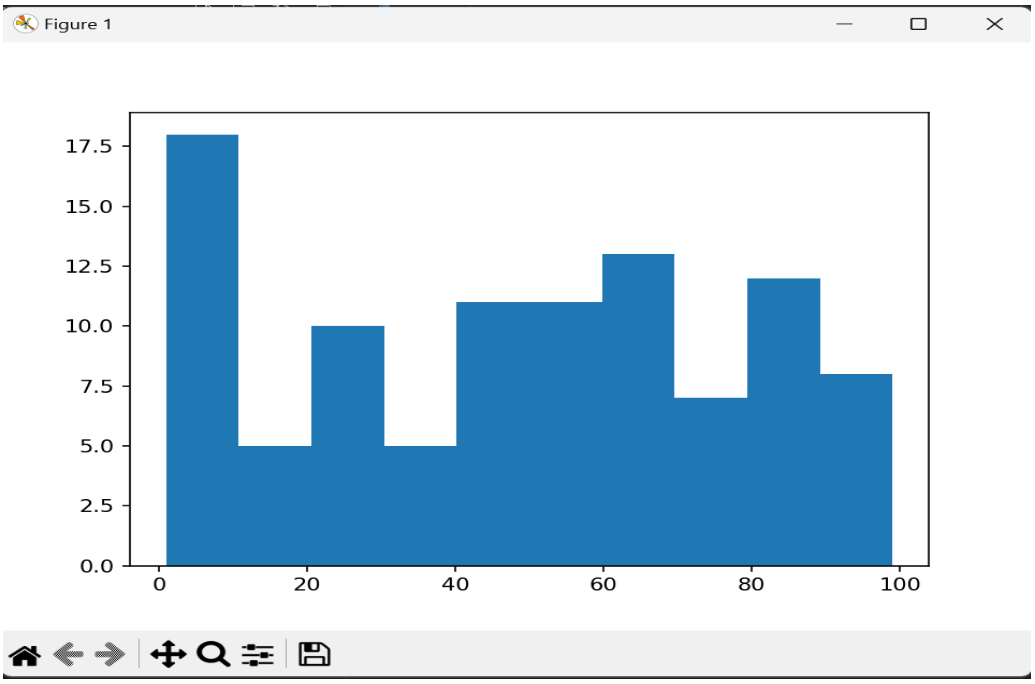
```
PS C:\Users\Shohail Parwej\Downloads\IDSUP-2\IDSUP-2> & "C:/Users/Shohail Parwej/AppData/Local/Programs/Python/Python312/python.exe" "c:/Users/Shohail Parwej/Downloads/IDSUP-2/IDSUP-2/p4.py"
[4.0, 5.0, 6.0]
PS C:\Users\Shohail Parwej\Downloads\IDSUP-2\IDSUP-2>
```

5. Generate a list of 100 random integers between 1 and 100 and plot a histogram of the same .

Code :

```
#Author : Shohail Parwej
#Regd.No. : 2141016146
import matplotlib.pyplot as plt
import random
data = [random.randint(1, 100) for _ in range(100)]
plt.hist(data)
plt.show()
```

Output :



6. Write a program to find median of a given list of integers. Combine both odd and even number of terms.

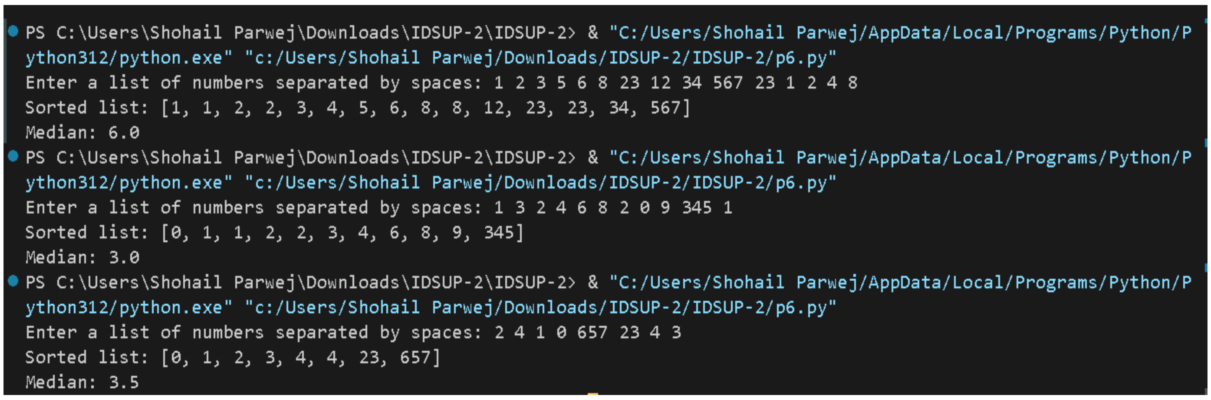
Code

#Author : Shohail Parwej

#Regd.No. : 2141016146

```
lst = input("Enter a list of numbers separated by spaces: ")
lst = list(map(int, lst.split()))
n = len(lst)
sorted_lst = sorted(lst)
print("Sorted list:", end=" ")
print(sorted_lst)  # Sort the list
print("Median:", end=" ")

if n % 2 != 0:
    # If the length of the list is odd, print the middle element
    print(float(sorted_lst[n // 2]))
else:
    # If the length of the list is even, calculate the average of the two
middle elements and print it
    print(float((sorted_lst[int((n - 1) / 2)] + sorted_lst[int(n / 2)]) /
2.0))
```

Output :



```
PS C:\Users\Shohail Parwej\Downloads\IDSUP-2\IDSUP-2> & "C:/Users/Shohail Parwej/AppData/Local/Programs/Python/Python312/python.exe" "c:/Users/Shohail Parwej/Downloads/IDSUP-2/IDSUP-2/p6.py"
Enter a list of numbers separated by spaces: 1 2 3 5 6 8 23 12 34 567 23 1 2 4 8
Sorted list: [1, 1, 2, 2, 3, 4, 5, 6, 8, 8, 12, 23, 23, 34, 567]
Median: 6.0
PS C:\Users\Shohail Parwej\Downloads\IDSUP-2\IDSUP-2> & "C:/Users/Shohail Parwej/AppData/Local/Programs/Python/Python312/python.exe" "c:/Users/Shohail Parwej/Downloads/IDSUP-2/IDSUP-2/p6.py"
Enter a list of numbers separated by spaces: 1 3 2 4 6 8 2 0 9 345 1
Sorted list: [0, 1, 1, 2, 2, 3, 4, 6, 8, 9, 345]
Median: 3.0
PS C:\Users\Shohail Parwej\Downloads\IDSUP-2\IDSUP-2> & "C:/Users/Shohail Parwej/AppData/Local/Programs/Python/Python312/python.exe" "c:/Users/Shohail Parwej/Downloads/IDSUP-2/IDSUP-2/p6.py"
Enter a list of numbers separated by spaces: 2 4 1 0 657 23 4 3
Sorted list: [0, 1, 2, 3, 4, 4, 23, 657]
Median: 3.5
```

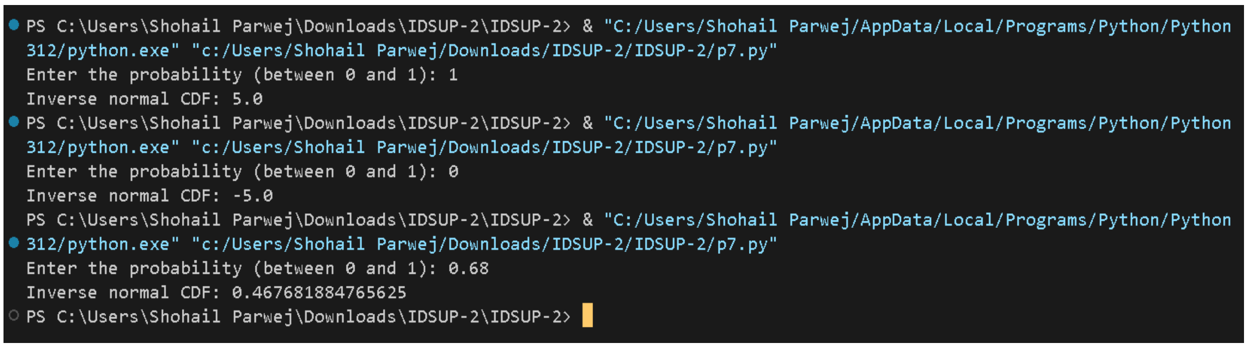
7. We have defined the function normal cdf. Write a program to invert normal cdf to find the value corresponding to a specified probability.

Code

```
#Author : Shohail Parwej

#Regd.No. : 2141016146

import math

def normal_cdf(x, mu=0, sigma=1):

   return (1 + math.erf((x - mu) / math.sqrt(2) / sigma)) / 2

def invert_normal_cdf(p, mu=0, sigma=1, tolerance=0.00001, max_iterations=1000):

   if mu != 0 or sigma != 1:

      raise ValueError("Non-standard parameters are not supported yet.")

   low_z = -10.0

   high_z = 10.0


   # Perform binary search for the inverse CDF

   for _ in range(max_iterations):

      mid_z = (low_z + high_z) / 2

      mid_p = normal_cdf(mid_z)

      if abs(mid_p - p) < tolerance:

         return mid_z

      if mid_p < p:

         low_z = mid_z
```

```
 else:
      high_z = mid_z
  raise ValueError("Failed to converge within maximum iterations.")

p = float(input("Enter the probability (between 0 and 1): "))
print("Inverse normal CDF:", invert_normal_cdf(p))
```

Output :



```
● PS C:\Users\Shohail Parwej\Downloads\IDSUP-2\IDSUP-2> & "C:/Users/Shohail Parwej/AppData/Local/Programs/Python/Python
  312/python.exe" "c:/Users/Shohail Parwej/Downloads/IDSUP-2/IDSUP-2/p7.py"
  Enter the probability (between 0 and 1): 1
  Inverse normal CDF: 5.0
● PS C:\Users\Shohail Parwej\Downloads\IDSUP-2\IDSUP-2> & "C:/Users/Shohail Parwej/AppData/Local/Programs/Python/Python
  312/python.exe" "c:/Users/Shohail Parwej/Downloads/IDSUP-2/IDSUP-2/p7.py"
  Enter the probability (between 0 and 1): 0
  Inverse normal CDF: -5.0
  PS C:\Users\Shohail Parwej\Downloads\IDSUP-2\IDSUP-2> & "C:/Users/Shohail Parwej/AppData/Local/Programs/Python/Python
● 312/python.exe" "c:/Users/Shohail Parwej/Downloads/IDSUP-2/IDSUP-2/p7.py"
  Enter the probability (between 0 and 1): 0.68
  Inverse normal CDF: 0.467681884765625
○ PS C:\Users\Shohail Parwej\Downloads\IDSUP-2\IDSUP-2>
```

8. Plot the Normal PDFs using various value of μ and σ as mentioned below:

μ	σ
0	1
0	2
0	0.5
-1	1

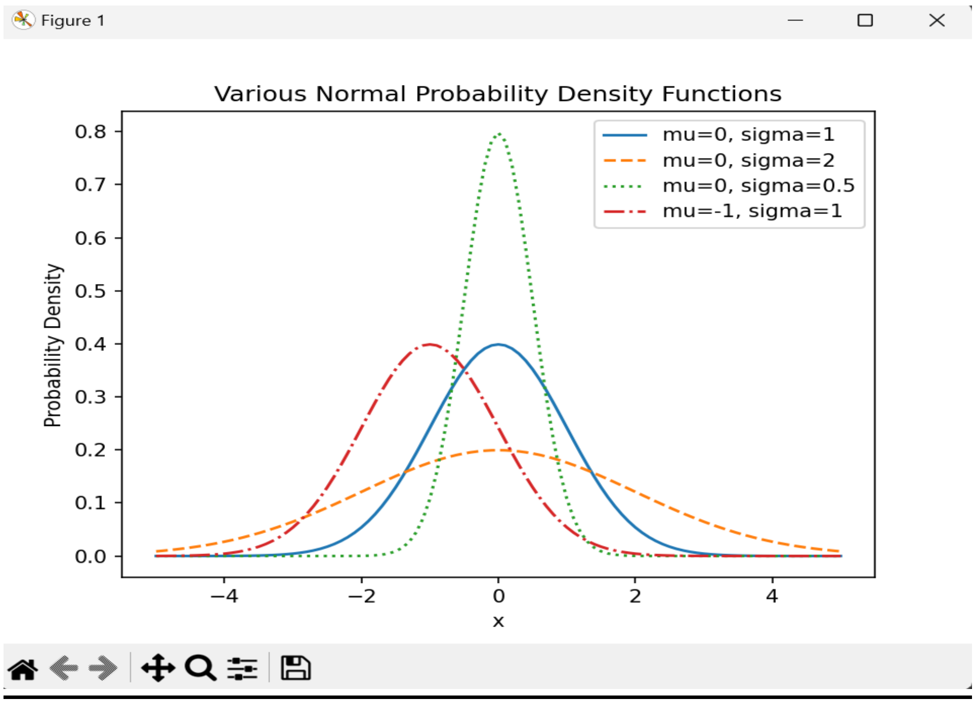
Use different line styles for each plot and compare the graphs thus obtained. You can use any range for x-axis.

Code

```
#Author : Shohail Parwej
#Regd.No. : 2141016146
import math
import matplotlib.pyplot as plt
SQRT_TWO_PI = math.sqrt(2 * math.pi)
```

```
def normal_pdf(x: float, mu: float = 0, sigma: float = 1) -> float:
    exponent = -(x - mu) ** 2 / (2 * sigma ** 2)
    return math.exp(exponent) / (SQRT_TWO_PI * sigma)

xs = [x / 10.0 for x in range(-50, 51)]

plt.plot(xs, [normal_pdf(x, sigma=1) for x in xs], '-', label='mu=0, sigma=1')
plt.plot(xs, [normal_pdf(x, sigma=2) for x in xs], '--', label='mu=0, sigma=2')
plt.plot(xs, [normal_pdf(x, sigma=0.5) for x in xs], ':', label='mu=0, sigma=0.5')
plt.plot(xs, [normal_pdf(x, mu=-1) for x in xs], '-.', label='mu=-1, sigma=1')
plt.legend()
plt.xlabel('x')
plt.ylabel('Probability Density')
plt.title('Various Normal Probability Density Functions')
plt.show()
```

<u>Output</u> :



9. Do the same as above question for Normal CDFs using the same values of μ and σ .

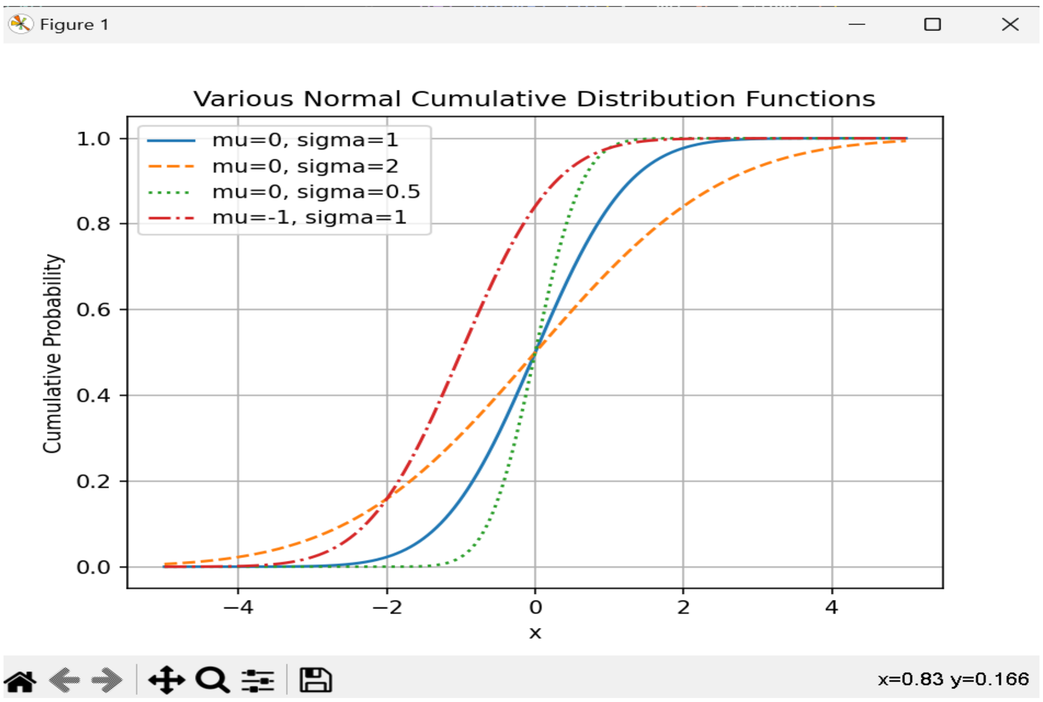
Code

#Author : Shohail Parwej

#Regd.No. : 2141016146

```
import matplotlib.pyplot as plt
import math

def normal_cdf(x, mu=0, sigma=1):
    return (1 + math.erf((x - mu) / math.sqrt(2) / sigma)) / 2

xs = [x / 100.0 for x in range(-500, 501)]

plt.plot(xs, [normal_cdf(x, sigma=1) for x in xs], '-', label='mu=0, sigma=1')
plt.plot(xs, [normal_cdf(x, sigma=2) for x in xs], '--', label='mu=0,
sigma=2')
plt.plot(xs, [normal_cdf(x, sigma=0.5) for x in xs], ':', label='mu=0,
sigma=0.5')
plt.plot(xs, [normal_cdf(x, mu=-1) for x in xs], '-.', label='mu=-1, sigma=1')
plt.legend()
plt.xlabel('x')
plt.ylabel('Cumulative Probability')
plt.title('Various Normal Cumulative Distribution Functions')
plt.grid(True)
plt.show()
```

Output



10. Python Program to find all Numbers in a Range (given by user) which are Perfect Squares and Sum of all Digits in the Number is Less than 10.

Answer :-

A random variable is a mathematical concept used to model uncertain outcomes in probability theory and statistics. It represents the possible values of an experiment or process, each associated with a probability of occurrence. Random variables can be discrete, taking on a countable number of distinct values, or continuous, taking on any value within a range. They are fundamental in analyzing and predicting the behavior of stochastic processes, such as coin flips, dice rolls, or the outcomes of experiments in science and engineering.

E.g. :- A very simple random variable equals 1 if a coin flip turns up heads and 0 if the flip turns up tails.

11. What are independent events? Give two examples of the same.

Answer :-

Independent events in probability theory are events where the occurrence of one event does not affect the occurrence of another.

Mathematically, we say that two events E and F are independent if the probability that they both happen is the product of the probabilities that each one happens:

$P(E,F) = P(E)P(F)$.

E.g. :-

1. Tossing a fair coin: The outcome of one coin toss (e.g., getting heads) does not affect the outcome of another coin toss. So, each toss of a fair coin is an independent event.
2. Rolling a fair six-sided die: If you roll a fair die twice, the outcome of the first roll (e.g., rolling a 3) does not influence the outcome of the second roll. Each roll of the die is an independent event.

12. Using the Binomial(n, p) distribution plot a histogram to show the actual binomial samples. Use a line chart to show the normal approximation. Plot both in the same graph. Take n=100, p=0.75 and number of points should be 100.

<u>Code :</u>

```
#Author : Shohail Parwej

#Regd.No.: 2141016146
```

```
import random

from collections import Counter

import matplotlib.pyplot as plt

import math


def normal_cdf(x, mu=0, sigma=1):

   return (1 + math.erf((x - mu) / math.sqrt(2) / sigma)) / 2

def bernoulli_trial(p):

   return 1 if random.random() < p else 0

def binomial(n, p):

   return sum(bernoulli_trial(p) for _ in range(n))

def binomial_histogram(p, n, num_points):

   data = [binomial(n, p) for _ in range(num_points)]

   histogram = Counter(data)


   plt.bar([x - 0.4 for x in histogram.keys()],

         [v / num_points for v in histogram.values()], 0.8,

         color='0.75')

   mu = p * n

   sigma = math.sqrt(n * p * (1 - p))

   xs = range(min(data), max(data) + 1)

   ys = [normal_cdf(i + 0.5, mu, sigma) - normal_cdf(i - 0.5, mu, sigma) for i in xs]

   plt.plot(xs, ys)

   plt.title("Binomial Distribution vs. Normal Approximation")

   plt.show()


binomial_histogram(0.75, 100, 100)
```

Output :-

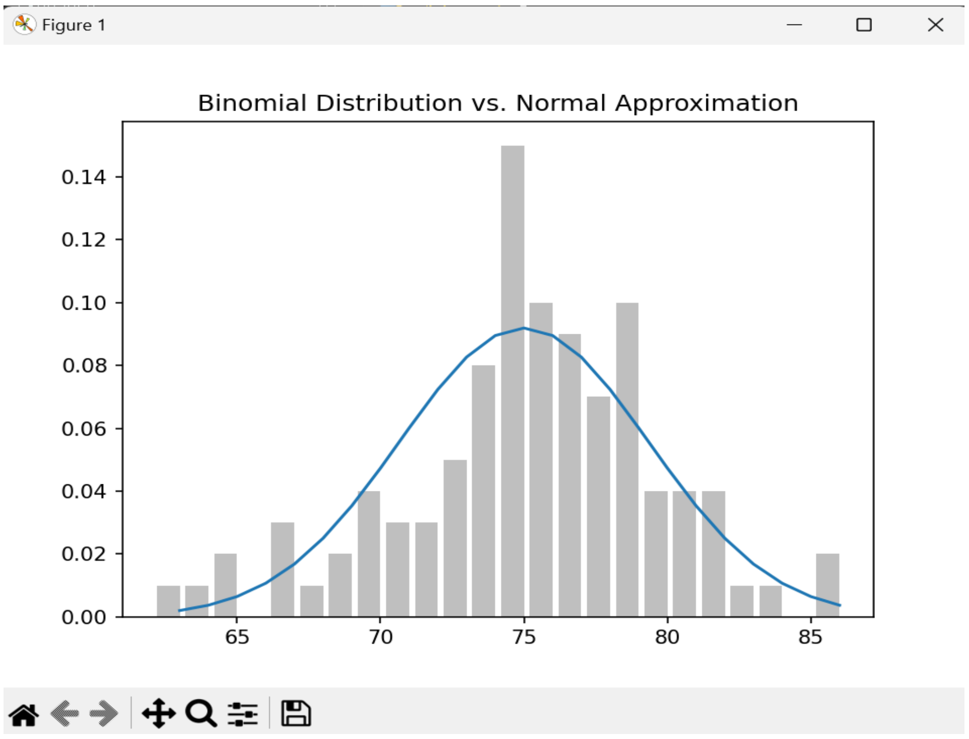


Figure 1
Binomial Distribution vs. Normal Approximation
0.14
0.12
0.10
0.08
0.06
0.04
0.02
0.00
65
70
75
80
85

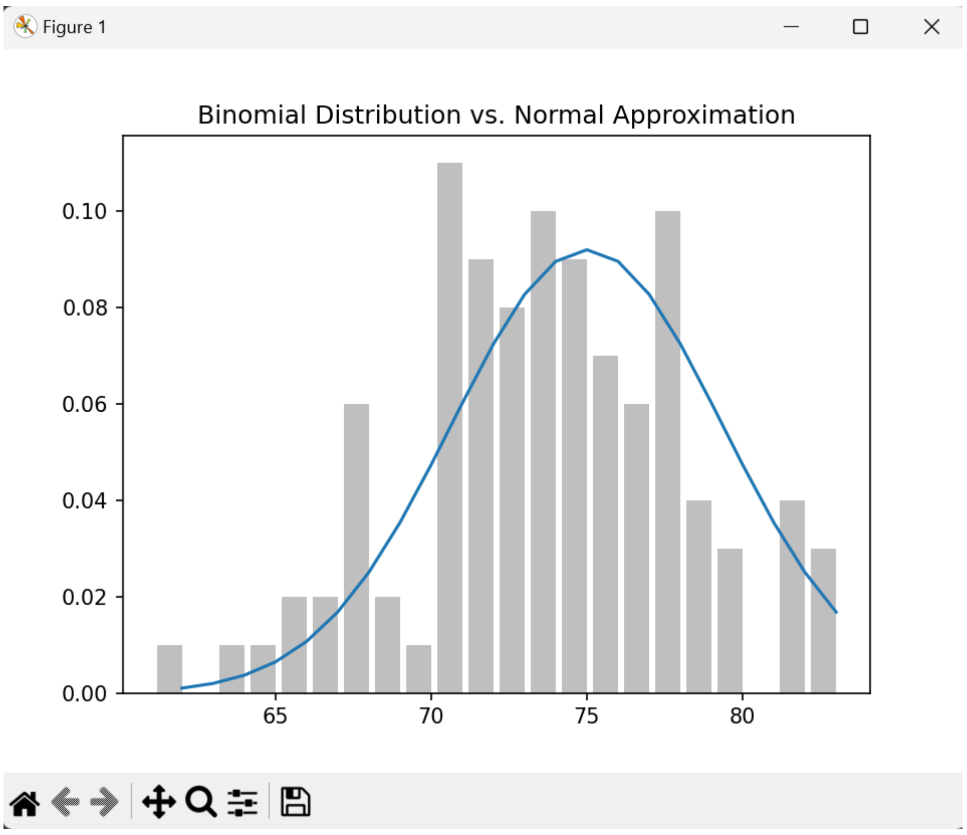


Figure 1
Binomial Distribution vs. Normal Approximation
0.10
0.08
0.06
0.04
0.02
0.00
65
70
75
80