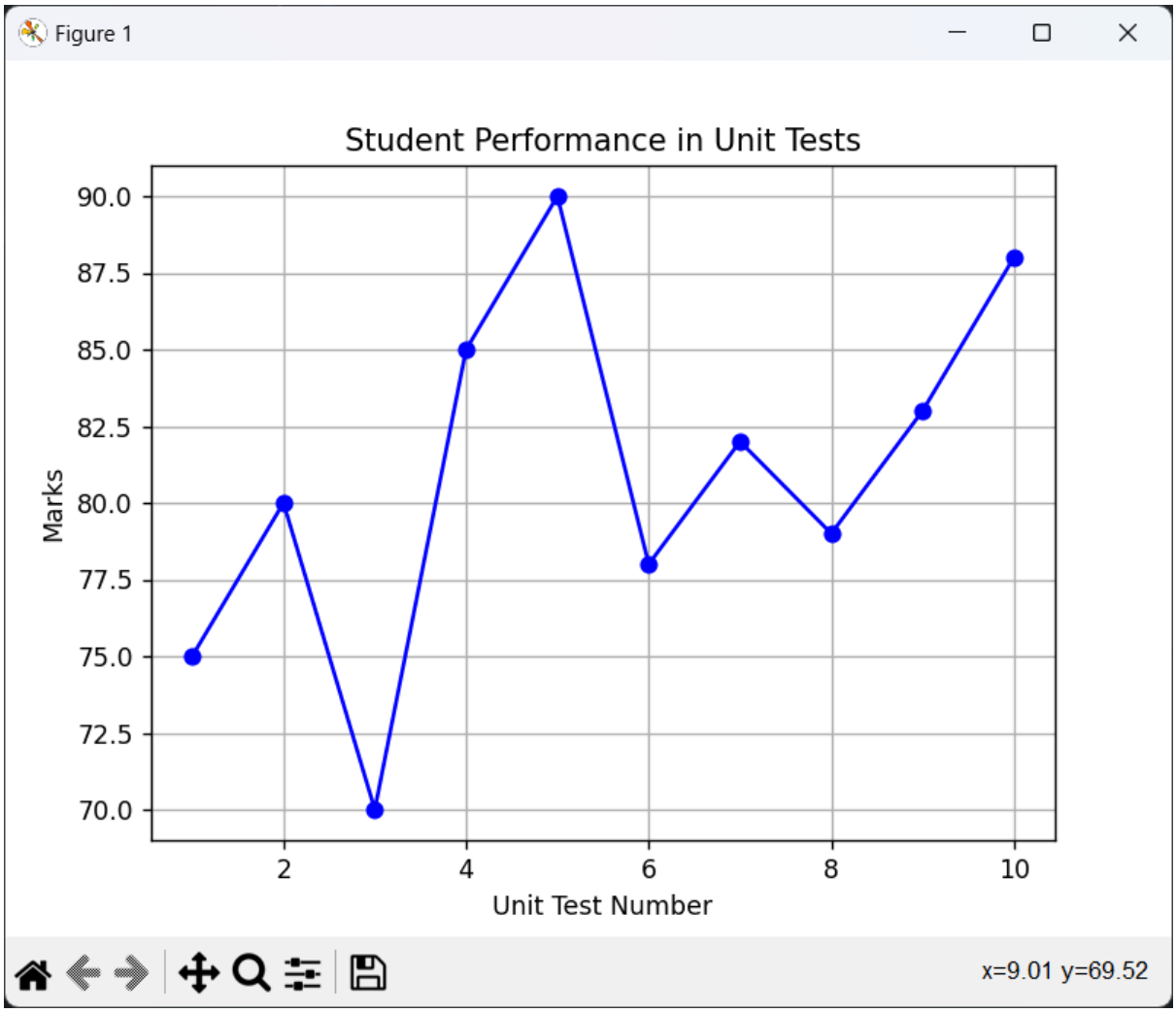
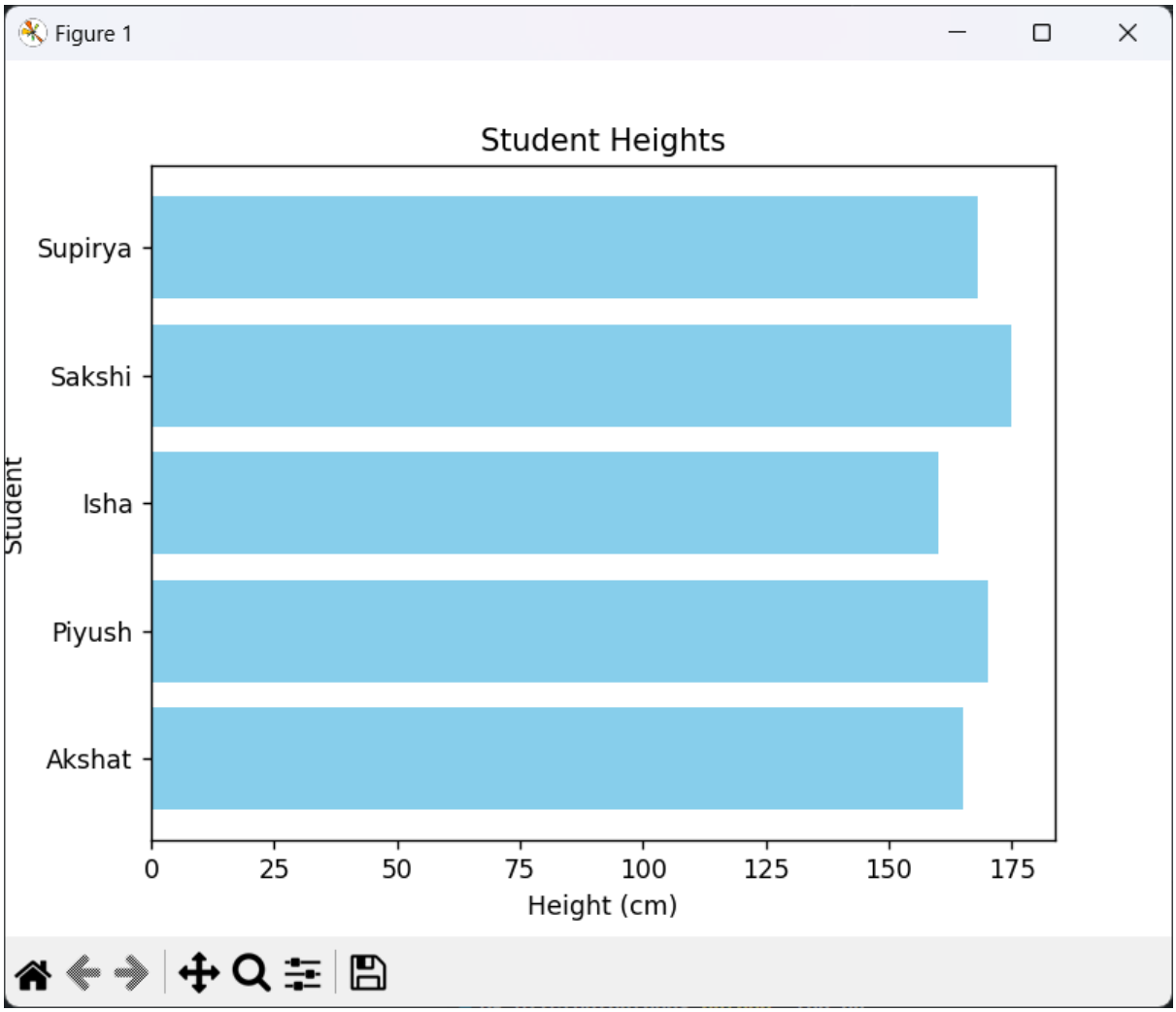
Q9. Marks is a list that stores marks of a student in 10-unit test. Write a program to plot Line chart for the student's performance in these 10 tests.

```
import matplotlib.pyplot as plt
marks = [75, 80, 70, 85, 90, 78, 82, 79, 83, 88]
unit_tests = list(range(1, 11))
plt.plot(unit_tests, marks, marker='o', color='b', linestyle='-')
plt.xlabel('Unit Test Number')
plt.ylabel('Marks')
plt.title('Student Performance in Unit Tests')
plt.grid(True)
plt.show()
```



Q10. Write a program to plot a horizontal bar chart from the height of some students.

```
import matplotlib.pyplot as plt
student_heights = { 'Akshat': 165, 'Piyush': 170, 'Isha': 160, 'Sakshi': 175, 'Supirya': 168 }
names = list(student_heights.keys())
heights = list(student_heights.values())
plt.barh(names, heights, color='skyblue')
plt.xlabel('Height (cm)')
plt.ylabel('Student')
plt.title('Student Heights')
plt.show()
```



Q11. Write a program to implement ANNOVA.

```
from scipy import stats

group1 = [11, 16, 21, 26, 31]

group2 = [13, 19, 23, 29, 33]

group3 = [9, 15, 17, 22, 27]

f_statistics, p_value = stats.f_oneway(group1, group2, group3)

print("F-Statisc:", f_statistics)

print("P-Value:", p_value)

alpha = 0.05

if p_value < alpha: print("Reject null hypothesis: There is a significant difference between the group means")

else: print("Fail to rejext null hypothesis: There is no signifiant difference between the group means.")
```

```
PS D:\stud\python> python q11.py
F-Statisc: 0.6372605919987138
P-Value: 0.5457255495477551
Fail to rejext null hypothesis: There is no signifiant difference between the group means.
```

```
# without library
group1 = [11, 16, 21, 26, 31]
group2 = [13, 19, 23, 29, 33]
group3 = [9, 15, 17, 22, 27]
mean_group1 = sum(group1) / len(group1)
mean_group2 = sum(group2) / len(group2)
mean_group3 = sum(group3) / len(group3)

overall_mean = (sum(group1) + sum(group2) + sum(group3)) / (len(group1) + len(group2) +
len(group3))
SSB = len(group1) * (mean_group1 - overall_mean) ** 2 + \
   len(group1) * (mean_group1 - overall_mean) ** 2 + \
   len(group1) * (mean_group1 - overall_mean) ** 2
SSW = sum([(x - mean_group1) ** 2 for x in group1]) + \
   sum([(x - mean_group2) ** 2 for x in group2]) + \
   sum([(x - mean_group3) ** 2 for x in group3])
df_between = 3-1
df_within = len(group1) + len(group2) + len(group3) - 3
F_statistics = (SSB / df_between) / (SSW / df_within)
critical_F_value = 3.354
print("F-Statistics:", F_statistics)

if F_statistics > critical_F_value:
   print("Reject null hypothesis: There is a significant difference between the group means")
else:
   print("Fail to rejext null hypothesis: There is no signifiant difference between the group means.")
```

```
PS D:\stud\python> python q11.py
F-Statistics: 0.005223447475333682
Fail to rejext null hypothesis: There is no signifiant difference between the group means.
PS D:\stud\python>
```

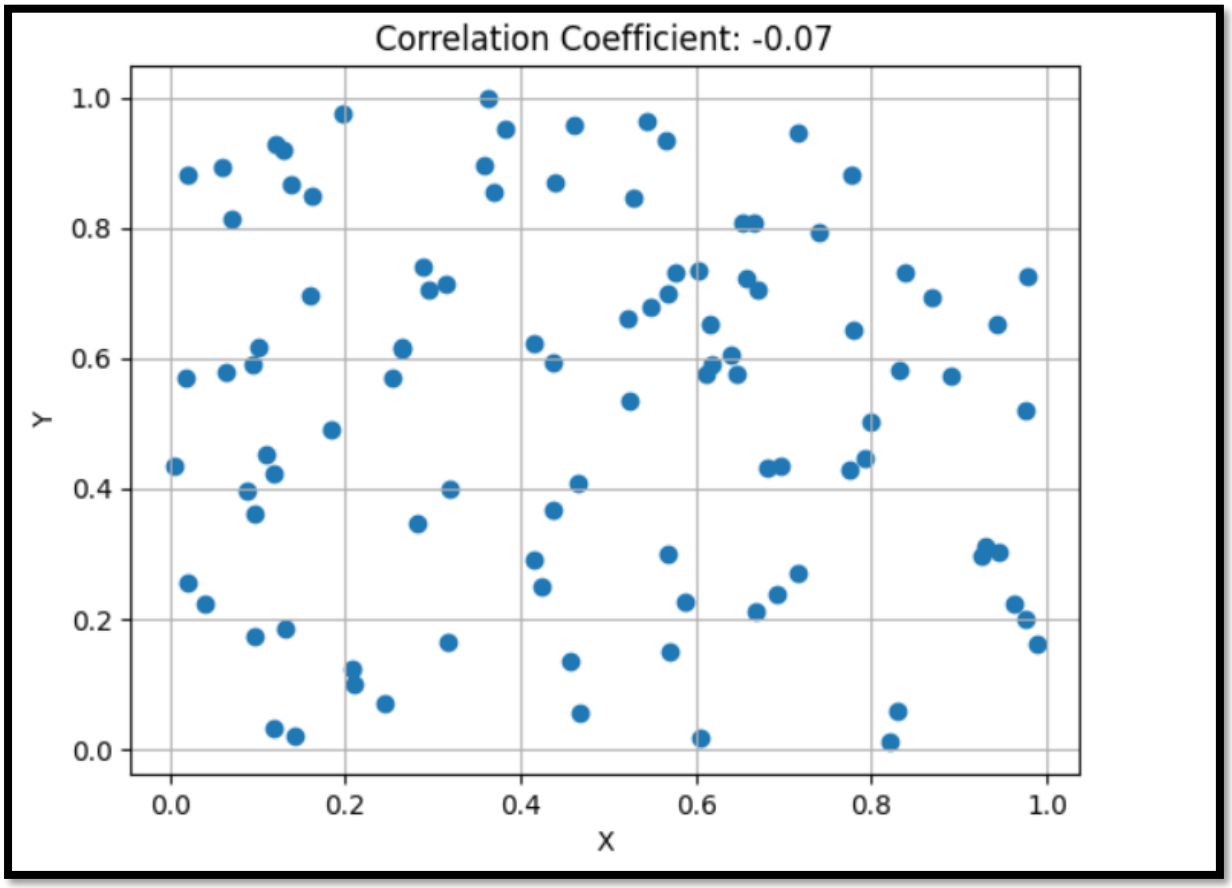
Q12. Write a program to show correlation between two randomly generated numbers.

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

np.random.seed(0)
x=np.random.rand(100)
y=np.random.rand(100)

correlation_coeffient = np.corrcoef(x,y)[0,1]

plt.scatter(x,y)
plt.title(f"Correlation Coefficient: {correlation_coeffient}")
plt.xlabel('X')
plt.ylabel('Y')
plt.grid(True)
plt.show()
```

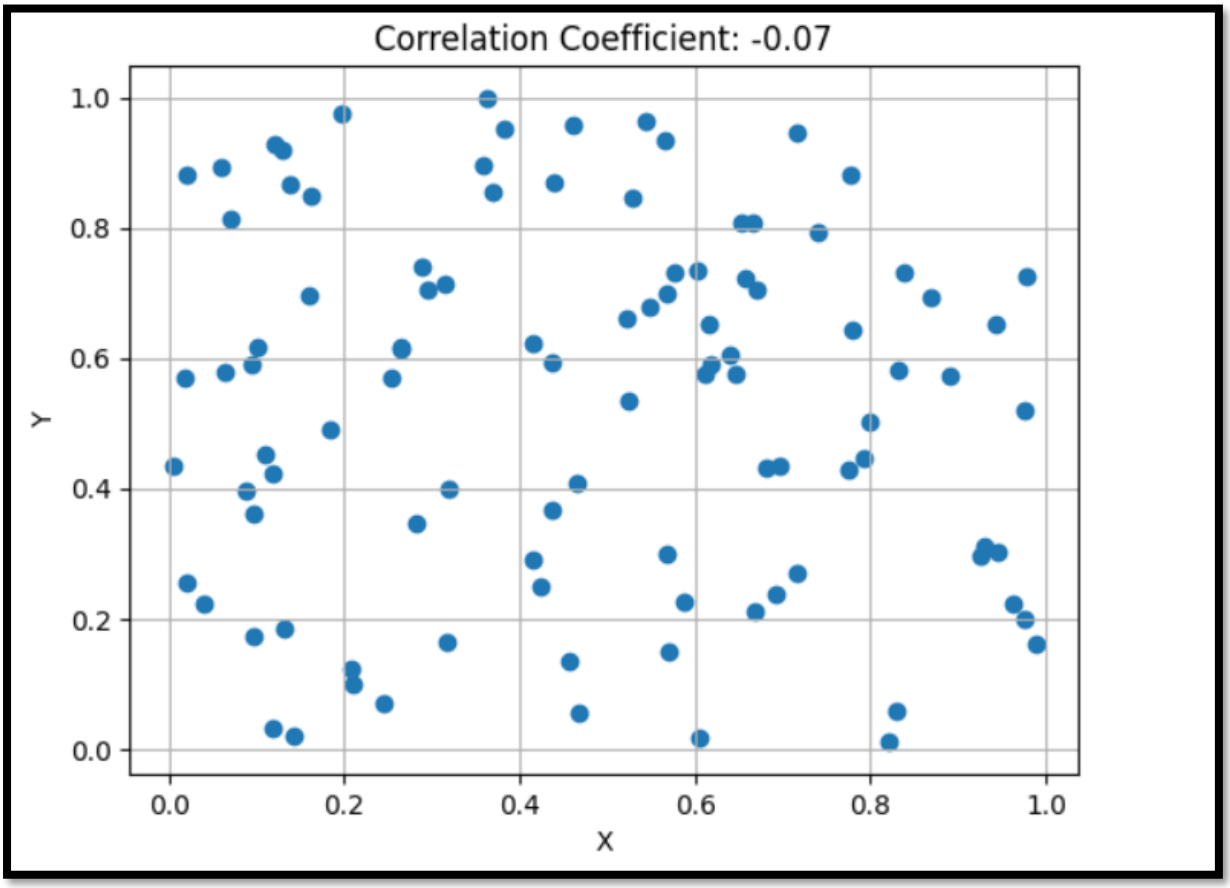


```
# without library
import numpy as np
import matplotlib.pyplot as plt

np.random.seed(0)
x=np.random.rand(100)
y=np.random.rand(100)
mean_x = sum(x) / len(x)
mean_y = sum(y) / len(y)

covariance = sum((x[i] - mean_x) * (y[i] - mean_y) for i in range(len(x))) / (len(x) - 1)
std_dev_x = (sum((xi - mean_x) ** 2 for xi in x) / (len(x) - 1)) ** 0.5
std_dev_y = (sum((yi - mean_y) ** 2 for yi in y) / (len(y) - 1)) ** 0.5
correlation_coeffient = covariance / (std_dev_x * std_dev_y)

plt.scatter(x,y)
plt.title(f"Correlation Coefficient: {correlation_coeffient}")
plt.xlabel('X')
plt.ylabel('Y')
plt.grid(True)
plt.show()
```



Q13. Write a program to implement Covariance.

```
import random


random.seed(0)
x = [random.random() for _ in range(100)]
y = [random.random() for _ in range(100)]


mean_x = sum(x) / len(x)
mean_y = sum(y) / len(y)
covaraince = sum((x[i] - mean_x) * (y[i] - mean_y) for i in range(len(x))) / (len(x) - 1)
print("Covariance:", covaraince)
```

```
PS D:\stud\python> python q13.py
Covariance: -0.0019005603132405055
PS D:\stud\python>
```