1. Data :

|  |  |  |
| --- | --- | --- |
| 12.5 | 11.4 | 15.7 |
| 13.1 | 12.9 | 14.1 |

1. Compute mean and the standard deviation.
2. Compute Q1, Q2, and Q3.
3. Make errorbar for mean and standard deviation.
4. Make a boxplot.

*# ----------------------------  
# 1*a = np.array([12.5, 11.4, 15.7,  
 13.1, 12.9, 14.1,  
 ])  
  
*# a)*mean = np.mean(a)  
std = np.std(a, ddof=1)  
n = len(a)  
print(**"Mean "**, mean)  
print(**"Standard Deviation "**, std)  
  
*# b)*Q = np.percentile(a, [25, 50, 75])  
print(**"Percentile "**, Q)  
  
*# c)*fig = plt.figure();  
plt.clf()  
ax = fig.add\_subplot(1, 1, 1)  
ax.plot(np.ones(n), a, **'o'**)  
ax.errorbar(1.2, mean, std, capsize=5)  
ax.plot(1.2, mean, **'sk'**)  
ax.boxplot(a, positions=[0.8])  
ax.set\_xlim([0, 2])  
ax.set\_ylim([10, 17])  
ax.set\_title(**"Problem 1"**)  
fig.tight\_layout()  
*# plt.show()*

1. Derive a polynomial for Write a python function for computing the function with interface: “def mycos(x,n):”. Compute for x = π, n = 20.

*# -----------------------------  
# 2*print(**'========= PROBLEM 2 =========='**)  
  
  
**def** mycos(x, n):  
 total = 0  
 **for** i **in** range(n):  
 *total = total + (-1)\*\*i \* (8\*x)\*\*(2\*i) / fact(2\*i)* **return** total  
  
  
x = 3 / 8 \* np.pi  
n = 20  
  
value = mycos(x, n)  
realValue = np.cos(8 \* x)  
print(**"mycos "**, value)  
print(**"numpy cos "**, realValue)

1. A = , b =
2. Solve A x = b, x = ?
3. Compute det(A).
4. Compute 2.

*# ----------------------------  
# 3*A = np.array([  
 [3, 1, 0, 0],  
 [2, 3, 2, 0],  
 [1, 2, 4, 1],  
 [0, 1, 1, 2]  
])  
  
b = np.array([  
 [1],  
 [1],  
 [1],  
 [1]  
])  
  
x = np.linalg.solve(A, b)  
print(x)  
  
*# b)*det\_A = np.linalg.det(A)  
print(**"determinant A"**, det\_A)  
  
norm\_b = np.linalg.norm(b)  
print(**"||b2|| "**, norm\_b)

1. Solve the following initial values,
2. Write RK6.
3. Solve for x E [0,1] and = 0.1
4. Plot x vs y

import numpy as np

from scipy.integrate import \*

def Gauss(f, a, b):

zeta = np.array([0.774597 , 0, -0.774597])

w = np.array([0.555556,0.888889, 0.555556])

x = (b+a)/2 + (b-a)/2 \* zeta

area = (b-a)/2 \* np.sum(w \* f(x))

return area

f = lambda x: (2+ np.sin(x))

a = 0

b = np.pi / 2

area\_scipy = quad(f, a, b)[0]

area\_gauss = Gauss(f, a, b)

print("Area scipy ", area\_scipy)

print("Area gauss ", area\_gauss)