HW1 ASSIGNMENT

Q 2.1)

a) Write a code for the mean and find the following values:

   μ(X) = ?

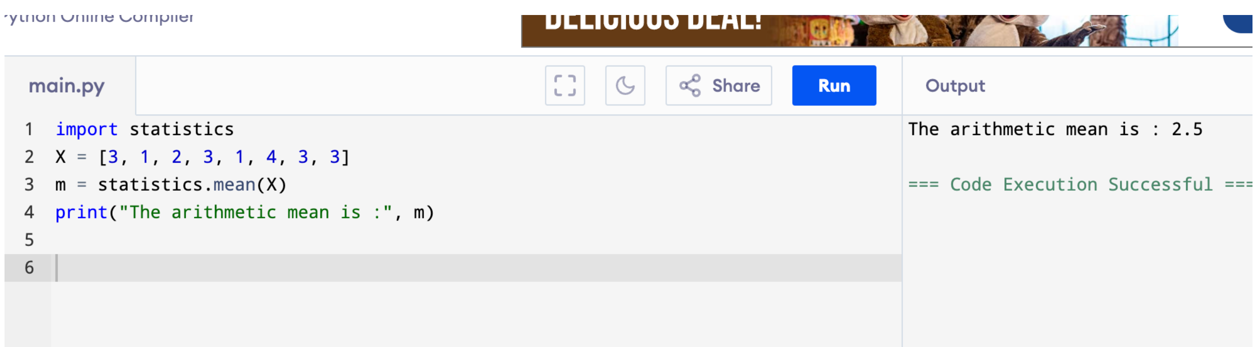
   μ(Y) = ?

   μ(Z) = ?

➔ Code:-

μ(X) = 2.5 μ(Y) = 2.1 6 μ(Z) = 2.5

```
import statistics
X = [3, 1, 2, 3, 1, 4, 3, 3]
m = statistics.mean(X)
print("The arithmetic mean is :", m)
```




```
def amean(X):
  mX = 0
  for ele in X:
    mX += ele
  return mX/len(X)
```

b) Write a code for the harmonic mean and find the following values:

   μh(X) = ?

   μh(Y) = ?

   μh(Z) = ?

➔ Code:-

```
from scipy.stats import hmean


harmonic_mean_X = hmean(X)

harmonic_mean_Y = hmean(Y)

harmonic_mean_Z = hmean(Z)


print("Harmonic Mean (X):", harmonic_mean_X)

print("Harmonic Mean (Y):", harmonic_mean_Y)

print("Harmonic Mean (Z):", harmonic_mean_Z)
```

$\mu_h(X) = 1.9591836735$ $\mu_h(Y) = 1.5319148936$ $\mu_h(Z) = 1.92$

```
def hmean(X):
  mX = 0
  for ele in X:
    mX += 1/ele
  mX = mX/len(X)
  return 1/mX
```

c) Write a code for the geometric mean and find the following values:
   $\mu g(X) = ?$
   $\mu g(Y) = ?$
   $\mu g(Z) = ?$

➔ Code:-

```
from scipy.stats import gmean


geometric_mean_X = gmean(X)

geometric_mean_Y = gmean(Y)
```

geometric_mean_Z = gmean(Z)

print("Geometric Mean (X):", geometric_mean_X)

print("Geometric Mean (Y):", geometric_mean_Y)

print("Geometric Mean (Z):", geometric_mean_Z)

$\mu_g(X) = 2.2461919979$ $\mu_g(Y) = 1.8171205928$ $\mu_g(Z) = 2.2133638394$

```
def gmean(X):
  mX = 1
  for ele in X:
    mX *= ele
  mX = mX**(1/len(X))
return mX
```

d) Write a code for the arithmetic-geometric mean and find the following values:
   μag(X) = ?
   μag(Y) = ?
   μag(Z) = ?

➔ Code:-

$\mu_{ag}(X) = 2.37139789655$ $\mu_{ag}(Y) = 1.9880506265$ $\mu_{ag}(Z) = 2.35450047778$

Iterative version:-

```
def myagmean(X):
  am = amean(X)
  gm = gmean(X)
  while abs(am - gm) > 0.000001:
    tam = (am + gm)/2
    gm = (am * gm)**(1/2)
    am = tam
  return (am + gm)/2
```

Recursive version:-

call `agmean(amean(X), gmean(X))` initially

```
def agmean(am, gm):
  if abs(am - gm) < 0.000001:
    return (am + gm)/2
  else:
    tam = (am + gm)/2
    tgm = (am * gm)**(1/2)
   return agmean(tam, tgm)
```

e) Write a code for the arithmetic-geometric mean and find the following values:

$\mu ag(X) = ?$

$\mu ag(Y) = ?$

$\mu ag(Z) = ?$

➔ Code:-

$\mu_{ahg}(X) = 2.224062384$ $\mu_{ahg}(Y) = 1.8202847281$ $\mu_{ahg}(Z) = 2.1983271599$

Iterative version:-

```
def myaghmean(X):
  am = amean(X)
  gm = gmean(X)
  hm = hmean(X)
  while max(abs(am - gm),abs(am - hm),abs(gm - hm)) > 0.00001:
    tam = (am + gm + hm)/3
    tgm = (am * gm * hm)**(1/3)
    hm = 3/(1/am + 1/gm + 1/hm)
    am = tam
    gm = tgm
  return (am + gm + hm)/3
```

Recursive version

call `aghmean(amean(X), gmean(X)), hmean(X))` initially

```
def aghmean(am, gm, hm):
  if max(abs(am - gm),abs(am - hm),abs(gm - hm)) < 0.00001:
    return (am + gm + hm)/3
  else:
    tam = (am + gm + hm)/3
    tgm = (am * gm * hm)**(1/3)
```

```
    thm = 3/(1/am + 1/gm + 1/hm)
  return aghmean(tam, tgm, thm)
```

f) Write a code for the median and find the following values:

   $\mu i(X) = ?$

   $\mu i(Y) = ?$

   $\mu i(Z) = ?$

➔ Code:-

$\mu_i(X) = 3$ $\mu_i(Y) = 2$ $\mu_i(Z) = 2.5$

```
print(statistics.median(X))
```

```
def mymedian(X):
  n = len(X)
  X.sort()
  if n % 2 == 0:
    return (X[n//2-1] + X[n//2])/2
  else:
    return X[n//2]
```

g) Write a code for the mode and find the following values:

   $\mu o(X) = ?$

   $\mu o(Y) = ?$

   $\mu o(Z) = ?$

➔ Code:-

from scipy.stats import mode

$\mu_o(X) = 3$ $\mu_o(Y) = 1$ $\mu_o(Z) = 1$

```
statistics.mode(X)
```

```
from scipy import stats
stats.mode(x)
```

```
hx=[2,1,4,1]
np.argmax(hx)
```

h) Write a code for the midpoint and find the following values:
   μm(X) = ?
   μm(Y) = ?
   μm(Z) = ?

➔ Code:-

$\mu_m(X) = 2.5$ $\mu_m(Y) = 2.5$ $\mu_m(Z) = 2.5$

```
(max(X)+min(X))/2
```

i) Write a code for the p% trimmed midpoint and find the following values:
   μm(X25∼75%) = ?
   μm(Y25∼75%) = ?
   μm(Z25∼75%) = ?

➔ Code:-

$\mu_m(X_{25\sim75\%}) = 2.5$ $\mu_m(Y_{25\sim75\%}) = 2.0$ $\mu_m(Z_{25\sim75\%}) = 2.5$

```
import numpy as np
import math
def trimmedmidpoint(X, p):
  n = len(X)
  X.sort()
  s = math.floor(n*p/100+1)
  e = math.floor(n*(100-p)/100)
  S = np.array(X)
  return S[(s-1):e]
```

```
tX = trimP(X,25)
print((max(tX)+min(tX))/2)
```

```
import math
def trimmedmidpoint(X, p):
  n = len(X)
  X.sort()
  s = math.floor(n*p/100+1)
  e = math.floor(n*(100-p)/100)
  return (X[s-1]+X[e-1])/2
print(trimmedmidpoint(X,25))
```

j) Write a code for the exclusive quartiles and find the following values:
   xQ1(X) = ?
   xQ1(Y) = ?
   xQ1(Z) = ?
   xQ3(X) = ?
   xQ3(Y) = ?
   xQ3(Z) = ?

➔ Code:-

$_xQ_1(X) = 1.5$ $_xQ_1(Y) = 1.0$ $_xQ_1(Z) = 1.5$
$_xQ_3(X) = 3.0$ $_xQ_3(Y) = 3.0$ $_xQ_3(Z) = 3.5$

```
def Quartile_x(X):
  n = len(X)
  X.sort()
  S = np.array(X)
  e = math.floor(n/2)
  s = math.floor((n+1)/2)
  return(mymedian(S[:e]), mymedian(S[s:n]))
```

k) Write a code for the inclusive quartiles and find the following values:
   iQ1(X) = ?

iQ1(Y) = ?
iQ1(Z) = ?
iQ3(X) = ?
iQ3(Y) = ?
iQ3(Z) = ?

➔ Code:-

$_iQ_1(X) = 1.5$ $_iQ_1(Y) = 1.0$ $_iQ_1(Z) = 1.5$
$_iQ_3(X) = 3.0$ $_iQ_3(Y) = 3.0$ $_iQ_3(Z) = 3.5$

```
def Quartile i(X):
  n = len(X)
  X.sort()
  S = np.array(X)
  e = math.ceil(n/2)
  s = math.floor(n/2)
  return(mymedian(S[:e]), mymedian(S[s:n]))
```

I) Write a code for the entity proportional quartiles and find the following values:
eQ1(X) = ?
eQ1(Y) = ?
eQ1(Z) = ?
eQ3(X) = ?
eQ3(Y) = ?
eQ3(Z) = ?

➔ Code:-

$_eQ_1(X) = 1.5$ $_eQ_1(Y) = 1.0$ $_eQ_1(Z) = 1.5$
$_eQ_3(X) = 3.0$ $_eQ_3(Y) = 3.0$ $_eQ_3(Z) = 3.5$

```
def Quartile ep(X):
  n = len(X)
  X.sort()
  q1 = (4 - (n+2)%4)/4 * X[math.floor((n+2)/4)-1]
     + ((n+2)%4)/4 * X[math.ceil((n+2)/4)-1]
  q3 = ((n+2)%4)/4 * X[math.floor((3*n+2)/4)-1]
     + (4-(n+2)%4)/4 * X[math.ceil((3*n+2)/4)-1]
  return(q1, q3)
```

m) Write a code for the scale proportional quartiles and find the following values:
   ${}_sQ_1(X)$ = ?
   ${}_sQ_1(Y)$ = ?
   ${}_sQ_1(Z)$ = ?
   ${}_sQ_3(X)$ = ?
   ${}_sQ_3(Y)$ = ?
   ${}_sQ_3(Z)$ = ?

➔ Code:-

${}_sQ_1(X) = 1.75$ ${}_sQ_1(Y) = 1.0$ ${}_sQ_1(Z) = 1.75$
${}_sQ_3(X) = 3.0$ ${}_sQ_3(Y) = 3.0$ ${}_sQ_3(Z) = 3.2$

```
def Quartile ep(X):
  n = len(X)
  X.sort()
  q1 = (4 - (n-1)%4)/4 * X[math.floor((n-1)/4)]
     + ((n-1)%4)/4 * X[math.ceil((n-1)/4)]
  q3 = ((n-1)%4)/4 * X[math.floor((3*n-3)/4)]
     + (4-(n-1)%4)/4 * X[math.ceil((3*n-3)/4)]
  return(q1, q3)
```

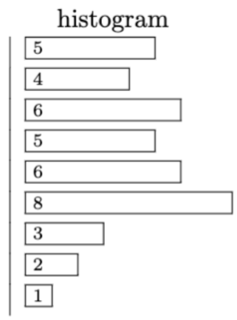
OR

```
import numpy as np
print(np.quantile(X, .25), np.quantile(X, .75))
```

Q 2.2)

During the Peloponnesian war in 431 BC, attackers besieging Plataea wanted to find the height of the wall in order to build ladders needed. The height of the wall was estimated by counting the number of bricks. 40 soldiers reported their estimates of the number of bricks. Frequency of soldiers estimated each number of bricks is given as follows:

X	$fr(X)$
103	5
104	4
105	6
106	5
107	6
108	8
109	3
110	2
111	1



The Data

The data consists of:

- X: Number of bricks
- fr(X): Frequency of estimates for each number of bricks

We first calculate the actual dataset by "expanding" the frequencies into a list of values.

```
import numpy as np

from scipy.stats import hmean, gmean, mode


# Frequency distribution

X = np.array([103, 104, 105, 106, 107, 108, 109, 110, 111])  # Brick counts

frequencies = np.array([5, 4, 6, 5, 6, 8, 3, 2, 1])        # Frequencies


# Expand the data

data = np.repeat(X, frequencies)
```

a) Write a code for the arithmetic mean and find the value, μ(B).

```
# Arithmetic mean

arithmetic_mean = np.mean(data)

print("Arithmetic Mean (μ(B)):", arithmetic_mean)
```

$$\frac{5 \times 103 + 4 \times 104 + 6 \times 105 + 5 \times 106 + 6 \times 107 + 8 \times 108 + 3 \times 109 + 2 \times 110 + 111}{40}$$

$\mu_w(B, f\, r(B)) = 106.375$

b) Write a code for the median and find the value, $\mu_i(B)$.

```
# Median

median = np.median(data)

print("Median (μ_i(B)):", median)
```

c) Write a code for the mode and find the value, $\mu_o(B)$.

```
# Mode

mode_value = mode(data).mode[0]

print("Mode (μ_o(B)):", mode_value)
```

d) Write a code for the harmonic mean and find the value, $\mu_h(B)$.

```
# Harmonic mean

harmonic_mean = hmean(data)

print("Harmonic Mean (μ_h(B)):", harmonic_mean)
```

e) Write a code for the geometric mean and find the value, μ_g(B).

```
# Geometric mean

geometric_mean = gmean(data)

print("Geometric Mean (μ_g(B)):", geometric_mean)
```

μ_{ag} (B) = 106.3642304

f) Write a code for the arithmetic-geometric mean and find the value, μ_{ag}(B).

The arithmetic-geometric mean is computed iteratively:

1. Start with $a_0 = \text{arithmetic mean}$ and $g_0 = \text{geometric mean}$.
2. Repeat until $|a_n - g_n|$ is small:

$$a_{n+1} = \frac{a_n + g_n}{2}, \quad g_{n+1} = \sqrt{a_n \cdot g_n}$$

```
def arithmetic_geometric_mean(a, g, tol=1e-9):

  while abs(a - g) > tol:

    a, g = (a + g) / 2, np.sqrt(a * g)

  return a


# Initial values

a0 = arithmetic_mean

g0 = geometric_mean


# Compute the arithmetic-geometric mean

ag_mean = arithmetic_geometric_mean(a0, g0)

print("Arithmetic-Geometric Mean (μ_ag(B)):", ag_mean)
```