In [769…
```
import numpy as np
import pandas as pd
import random
from scipy import stats
from statistics import median,median
from statistics import mean as mn, stdev

random.seed(0)
salaries = np.array([round(random.random()*1000000, -3) for _ in range(100)], dtype
# don't know what happend i just change 8 bytes to 4bytes
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[769], line 9
      6 from statistics import mean as mn, stdev
      8 random.seed(0)
----> 9 salaries = np.array([round(random.random()*1000000, -3) for _ in range(10
0)], dtype = 'int32')

TypeError: 'numpy.int32' object is not callable
```

In [583…
```
print(salaries)
```

```
[844000 758000 421000 259000 511000 405000 784000 303000 477000 583000
 908000 505000 282000 756000 618000 251000 910000 983000 810000 902000
 310000 730000 899000 684000 472000 101000 434000 611000 913000 967000
 477000 865000 260000 805000 549000  14000 720000 399000 825000 668000
   1000 494000 868000 244000 325000 870000 191000 568000 239000 968000
 803000 448000  80000 320000 508000 933000 109000 551000 707000 547000
 814000 540000 964000 603000 588000 445000 596000 385000 576000 290000
 189000 187000 613000 657000 477000  90000 758000 877000 923000 842000
 898000 923000 541000 391000 705000 276000 812000 849000 895000 590000
 950000 580000 451000 660000 996000 917000 793000  82000 613000 486000]
```

In [585…
```
np.average(salaries)
```

Out[585…
```
585690.0
```

In [587…
```
np.mean(salaries)
```

Out[587…
```
585690.0
```

In [589…
```
np.median(salaries)
```

Out[589…
```
589000.0
```

In [591…
```
stats.mode(salaries)
```

Out[591…
```
ModeResult(mode=477000, count=3)
```

In [593…
```
np.ptp(salaries)
```

Out[593…
```
995000
```

In [595…
```
salaries.shape
```

Out[595…
```
(100,)
```

In [597…
```
salaries.ndim
```

Out[597…
```
1
```

In [599…
```
salaries.dtype
```

Out[599…
```
dtype('int32')
```

In [601…
```
salaries.size
```

Out[601…
```
100
```

In [603…
```
a = np.array( [[[1,2],[3,4]]])
```

In [605…
```
print(a)
```

```
[[[1 2]
  [3 4]]]
```

In [607…
```
a[0,1]
```

Out[607…
```
array([3, 4])
```

In [609…
```
b=[[5,6, 27, 35,43, 54, [4,5,6]],[1,2,3,4,5,6,[1,2,3],[1,2,3,4,[6,64,3,2,["2","5","
```

In [611…
```
print(b)
```

```
[[5, 6, 27, 35, 43, 54, [4, 5, 6]], [1, 2, 3, 4, 5, 6, [1, 2, 3], [1, 2, 3, 4, [6, 6
4, 3, 2, ['2', '5', '10']]]]]
```

In [613…
```
b[1][7][4][4][2]
```

Out[613…
```
'10'
```

In [665…
```
c=np.ones(a.shape)
c
```

Out[665…
```
array([[[1., 1.],
        [1., 1.]]])
```

# Exercise 1

In [616…
```
# start of activity
# Execise 1

mean = sum(salaries)/len(salaries)
mean # sum of all item in the array over the length of it
```

Out[616… 585690.0

In [618…
```
sorted_salaries = sorted(salaries)

n = len(sorted_salaries)

if n % 2 == 1:#if only theres one middle
    median = sorted_salaries[n // 2]
else: #if there is 2 middle, (middle1 + middle2) /2
    median = (sorted_salaries[n // 2 - 1] + sorted_salaries[n // 2]) / 2

median
```

Out[618… 589000.0

In [620…
```
salaries_dict = {} # store the elements in this dictionary for Speedful
for salary in salaries:
    if salary in salaries_dict:
        salaries_dict[salary] += 1 # if the values is already seen increment the ke
    else:
        salaries_dict[salary] = 1
```

In [622…
```
mode = max(frequency_dict, key=salaries_dict.get)
count = salaries_dict[mode]
```

In [624…
```
print(mode,count)
```

```
477000 3
```

In [626…
```
# Sample variance

# using list comprehension loop for values in the array
sd_squared = [(x - mean) ** 2 for x in salaries]
sv = sum(sd_squared) / (len(salaries) - 1)
sv
```

Out[626… 70664054444.44444

In [628…
```
# Sample standard deviation

sd = sv**.5 # just the sqrt sample variance
sd
```

Out[628… 265827.11382484

# Exercise 2

In [631…
```
# range = max(value) - min(value)
range = max(salaries) - min(salaries)
range
```

Out[631… 995000

In [633…
```
# first lets convert it in list
salaries_list = salaries.tolist()
```

In [635…
```
# Coefficient of variation Interquartile range

salaries_mean = mn(salaries_list)
salaries_sd = stdev(salaries_list) # stdev is a function for getting a standard dev
cv = (salaries_sd / salaries_mean) * 100
```

In [637…
```
print(cv)
```

```
45.38699889443903
```

In [643…
```
# Quartile coefficient of dispersion

q1 = np.percentile(salaries_list, 25) # getting the first quartile
q3 = np.percentile(salaries_list, 75) # getting the third quartile
```

In [645…
```
Quartile_Coef = (q3 - q1) / (q3 + q1)
Quartile_Coef
```

Out[645…
```
0.338660110633067
```

# Exercise 3

In [674…
```
diabetes = pd.read_csv("diabetes.csv")
diabetes
```

Out[674…

| | Pregnancies | Glucose | BloodPressure | SkinThickness | Insulin | BMI | DiabetesPedigreeFu |
|---|---|---|---|---|---|---|---|
| **0** | 6 | 148 | 72 | 35 | 0 | 33.6 | |
| **1** | 1 | 85 | 66 | 29 | 0 | 26.6 | |
| **2** | 8 | 183 | 64 | 0 | 0 | 23.3 | |
| **3** | 1 | 89 | 66 | 23 | 94 | 28.1 | |
| **4** | 0 | 137 | 40 | 35 | 168 | 43.1 | |
| **...** | ... | ... | ... | ... | ... | ... | |
| **763** | 10 | 101 | 76 | 48 | 180 | 32.9 | |
| **764** | 2 | 122 | 70 | 27 | 0 | 36.8 | |
| **765** | 5 | 121 | 72 | 23 | 112 | 26.2 | |
| **766** | 1 | 126 | 60 | 0 | 0 | 30.1 | |
| **767** | 1 | 93 | 70 | 31 | 0 | 30.4 | |

768 rows × 9 columns

In [680…
```
# identify column names

diabetes.columns
```

Out[680…
```
Index(['Pregnancies', 'Glucose', 'BloodPressure', 'SkinThickness', 'Insulin',
       'BMI', 'DiabetesPedigreeFunction', 'Age', 'Outcome'],
      dtype='object')
```

In [686…
```
# identify datatypes
diabetes.dtypes
```

Out[686…
```
Pregnancies                 int64
Glucose                     int64
BloodPressure               int64
SkinThickness               int64
Insulin                     int64
BMI                       float64
DiabetesPedigreeFunction  float64
Age                         int64
Outcome                     int64
dtype: object
```

In [702…
```
#3. Display the total number of record
diabetes.count()
```

Out[702…
```
Pregnancies                 768
Glucose                     768
BloodPressure               768
SkinThickness               768
Insulin                     768
BMI                         768
DiabetesPedigreeFunction    768
Age                         768
Outcome                     768
dtype: int64
```

In [692…
```
# Display the first 20

diabetes.head(20)
```

Out[692...

| | Pregnancies | Glucose | BloodPressure | SkinThickness | Insulin | BMI | DiabetesPedigreeFur |
|---|---|---|---|---|---|---|---|
| **0** | 6 | 148 | 72 | 35 | 0 | 33.6 | |
| **1** | 1 | 85 | 66 | 29 | 0 | 26.6 | |
| **2** | 8 | 183 | 64 | 0 | 0 | 23.3 | |
| **3** | 1 | 89 | 66 | 23 | 94 | 28.1 | |
| **4** | 0 | 137 | 40 | 35 | 168 | 43.1 | |
| **5** | 5 | 116 | 74 | 0 | 0 | 25.6 | |
| **6** | 3 | 78 | 50 | 32 | 88 | 31.0 | |
| **7** | 10 | 115 | 0 | 0 | 0 | 35.3 | |
| **8** | 2 | 197 | 70 | 45 | 543 | 30.5 | |
| **9** | 8 | 125 | 96 | 0 | 0 | 0.0 | |
| **10** | 4 | 110 | 92 | 0 | 0 | 37.6 | |
| **11** | 10 | 168 | 74 | 0 | 0 | 38.0 | |
| **12** | 10 | 139 | 80 | 0 | 0 | 27.1 | |
| **13** | 1 | 189 | 60 | 23 | 846 | 30.1 | |
| **14** | 5 | 166 | 72 | 19 | 175 | 25.8 | |
| **15** | 7 | 100 | 0 | 0 | 0 | 30.0 | |
| **16** | 0 | 118 | 84 | 47 | 230 | 45.8 | |
| **17** | 7 | 107 | 74 | 0 | 0 | 29.6 | |
| **18** | 1 | 103 | 30 | 38 | 83 | 43.3 | |
| **19** | 1 | 115 | 70 | 30 | 96 | 34.6 | |

In [696...

```
# display the last 20 records
diabetes.tail(20)
```

Out[696...

| | Pregnancies | Glucose | BloodPressure | SkinThickness | Insulin | BMI | DiabetesPedigreeFu |
|---|---|---|---|---|---|---|---|
| **748** | 3 | 187 | 70 | 22 | 200 | 36.4 | |
| **749** | 6 | 162 | 62 | 0 | 0 | 24.3 | |
| **750** | 4 | 136 | 70 | 0 | 0 | 31.2 | |
| **751** | 1 | 121 | 78 | 39 | 74 | 39.0 | |
| **752** | 3 | 108 | 62 | 24 | 0 | 26.0 | |
| **753** | 0 | 181 | 88 | 44 | 510 | 43.3 | |
| **754** | 8 | 154 | 78 | 32 | 0 | 32.4 | |
| **755** | 1 | 128 | 88 | 39 | 110 | 36.5 | |
| **756** | 7 | 137 | 90 | 41 | 0 | 32.0 | |
| **757** | 0 | 123 | 72 | 0 | 0 | 36.3 | |
| **758** | 1 | 106 | 76 | 0 | 0 | 37.5 | |
| **759** | 6 | 190 | 92 | 0 | 0 | 35.5 | |
| **760** | 2 | 88 | 58 | 26 | 16 | 28.4 | |
| **761** | 9 | 170 | 74 | 31 | 0 | 44.0 | |
| **762** | 9 | 89 | 62 | 0 | 0 | 22.5 | |
| **763** | 10 | 101 | 76 | 48 | 180 | 32.9 | |
| **764** | 2 | 122 | 70 | 27 | 0 | 36.8 | |
| **765** | 5 | 121 | 72 | 23 | 112 | 26.2 | |
| **766** | 1 | 126 | 60 | 0 | 0 | 30.1 | |
| **767** | 1 | 93 | 70 | 31 | 0 | 30.4 | |

In [706...

```
# Change the Outcome column to Diagnosis
diabetes.rename(columns = {'Outcome':'Diagnosis'})
```

Out[706…

| | Pregnancies | Glucose | BloodPressure | SkinThickness | Insulin | BMI | DiabetesPedigreeFu |
|---|---|---|---|---|---|---|---|
| **0** | 6 | 148 | 72 | 35 | 0 | 33.6 | |
| **1** | 1 | 85 | 66 | 29 | 0 | 26.6 | |
| **2** | 8 | 183 | 64 | 0 | 0 | 23.3 | |
| **3** | 1 | 89 | 66 | 23 | 94 | 28.1 | |
| **4** | 0 | 137 | 40 | 35 | 168 | 43.1 | |
| **...** | ... | ... | ... | ... | ... | ... | |
| **763** | 10 | 101 | 76 | 48 | 180 | 32.9 | |
| **764** | 2 | 122 | 70 | 27 | 0 | 36.8 | |
| **765** | 5 | 121 | 72 | 23 | 112 | 26.2 | |
| **766** | 1 | 126 | 60 | 0 | 0 | 30.1 | |
| **767** | 1 | 93 | 70 | 31 | 0 | 30.4 | |

768 rows × 9 columns

In [710…

```
# Create a new column Classification that display "Diabetes" if the value of outcom
diabetes = diabetes.assign(
    new_col = lambda x: ['Diabetes' if outcome == 1 else 'No Diabetes' for outcome
)
```

In [712…

```
diabetes
```

Out[712…

| | Pregnancies | Glucose | BloodPressure | SkinThickness | Insulin | BMI | DiabetesPedigreeFu |
|---|---|---|---|---|---|---|---|
| **0** | 6 | 148 | 72 | 35 | 0 | 33.6 | |
| **1** | 1 | 85 | 66 | 29 | 0 | 26.6 | |
| **2** | 8 | 183 | 64 | 0 | 0 | 23.3 | |
| **3** | 1 | 89 | 66 | 23 | 94 | 28.1 | |
| **4** | 0 | 137 | 40 | 35 | 168 | 43.1 | |
| **…** | … | … | … | … | … | … | |
| **763** | 10 | 101 | 76 | 48 | 180 | 32.9 | |
| **764** | 2 | 122 | 70 | 27 | 0 | 36.8 | |
| **765** | 5 | 121 | 72 | 23 | 112 | 26.2 | |
| **766** | 1 | 126 | 60 | 0 | 0 | 30.1 | |
| **767** | 1 | 93 | 70 | 31 | 0 | 30.4 | |

768 rows × 10 columns

In [718…

```
# Create a new dataframe "withDiabetes" that gathers data with diabetes
withDiabetes = diabetes[diabetes['Outcome'] == 1]
withDiabetes
```

Out[718...

| | Pregnancies | Glucose | BloodPressure | SkinThickness | Insulin | BMI | DiabetesPedigreeFu |
|---|---|---|---|---|---|---|---|
| **0** | 6 | 148 | 72 | 35 | 0 | 33.6 | |
| **2** | 8 | 183 | 64 | 0 | 0 | 23.3 | |
| **4** | 0 | 137 | 40 | 35 | 168 | 43.1 | |
| **6** | 3 | 78 | 50 | 32 | 88 | 31.0 | |
| **8** | 2 | 197 | 70 | 45 | 543 | 30.5 | |
| **...** | ... | ... | ... | ... | ... | ... | |
| **755** | 1 | 128 | 88 | 39 | 110 | 36.5 | |
| **757** | 0 | 123 | 72 | 0 | 0 | 36.3 | |
| **759** | 6 | 190 | 92 | 0 | 0 | 35.5 | |
| **761** | 9 | 170 | 74 | 31 | 0 | 44.0 | |
| **766** | 1 | 126 | 60 | 0 | 0 | 30.1 | |

268 rows × 10 columns

In [722...

```
# Create a new dataframe "noDiabetes" thats gathers data with no diabetes
noDiabetes = diabetes[diabetes['Outcome'] == 0]
noDiabetes
```

Out[722...

| | Pregnancies | Glucose | BloodPressure | SkinThickness | Insulin | BMI | DiabetesPedigreeFu |
|---|---|---|---|---|---|---|---|
| **1** | 1 | 85 | 66 | 29 | 0 | 26.6 | |
| **3** | 1 | 89 | 66 | 23 | 94 | 28.1 | |
| **5** | 5 | 116 | 74 | 0 | 0 | 25.6 | |
| **7** | 10 | 115 | 0 | 0 | 0 | 35.3 | |
| **10** | 4 | 110 | 92 | 0 | 0 | 37.6 | |
| **...** | ... | ... | ... | ... | ... | ... | |
| **762** | 9 | 89 | 62 | 0 | 0 | 22.5 | |
| **763** | 10 | 101 | 76 | 48 | 180 | 32.9 | |
| **764** | 2 | 122 | 70 | 27 | 0 | 36.8 | |
| **765** | 5 | 121 | 72 | 23 | 112 | 26.2 | |
| **767** | 1 | 93 | 70 | 31 | 0 | 30.4 | |

500 rows × 10 columns

In [724...

```
# Create a new dataframe "Pedia" that gathers data with age 0 to 19
Pedia = diabetes[diabetes['Age'].between(0, 19)]
Pedia
```

Out[724...

| | Pregnancies | Glucose | BloodPressure | SkinThickness | Insulin | BMI | DiabetesPedigreeFunct |
|---|---|---|---|---|---|---|---|

In [726...

```
# Create a new dataframe "Adult" that gathers data with age greater than 19
Adult = diabetes[diabetes['Age'] >= 19]
Adult
```

Out[726…

| | Pregnancies | Glucose | BloodPressure | SkinThickness | Insulin | BMI | DiabetesPedigreeFu |
|---|---|---|---|---|---|---|---|
| **0** | 6 | 148 | 72 | 35 | 0 | 33.6 | |
| **1** | 1 | 85 | 66 | 29 | 0 | 26.6 | |
| **2** | 8 | 183 | 64 | 0 | 0 | 23.3 | |
| **3** | 1 | 89 | 66 | 23 | 94 | 28.1 | |
| **4** | 0 | 137 | 40 | 35 | 168 | 43.1 | |
| **...** | ... | ... | ... | ... | ... | ... | |
| **763** | 10 | 101 | 76 | 48 | 180 | 32.9 | |
| **764** | 2 | 122 | 70 | 27 | 0 | 36.8 | |
| **765** | 5 | 121 | 72 | 23 | 112 | 26.2 | |
| **766** | 1 | 126 | 60 | 0 | 0 | 30.1 | |
| **767** | 1 | 93 | 70 | 31 | 0 | 30.4 | |

768 rows × 10 columns

In [728…

```
# Use numpy to get the average age and glucose value.
average_age = np.mean(diabetes["Age"])
average_glucose = np.mean(diabetes["Glucose"])
print(f"Average Age: {average_age} and Average Glucose {average_glucose}")
```

```
Average Age: 33.240885416666664 and Average Glucose 120.89453125
```

In [744…

```
# Use numpy to get the median age and glucose value.
median_age = np.median(diabetes["Age"])
glucose_value = np.median(diabetes["Glucose"])
print(f"Median Age: {median_age} and Glucose Value: {glucose_value}")
```

```
Median Age: 29.0 and Glucose Value: 117.0
```

In [754…

```
# Use numpy to get the middle values of glucose and age.
sorted_age = np.sort(diabetes.Age)
sorted_glucose = np.sort(diabetes.Glucose)

middle_age = np.median(sorted_age)
middle_glucose = np.median(sorted_glucose)
print(f"Middle Values of Glucose: {middle_age} and Middle Values of Age: {middle_gl
```

```
Middle Values of Glucose: 29.0 and Middle Values of Age: 117.0
```

In [762…

```
# Use numpy to get the standard deviation of the skinthickness.
np.std(diabetes.SkinThickness)
```

Out[762…

```
15.941828626496978
```

In my conclusion I in this activity I learn how to use numpy, and other libraries in python. In the fisrt activity we only use our knowledge for implementing the formulas but i use the lib for only checking, and I watch some tutorial how to use numpy, collection, scipy and math lib. I enjoy using the numpy libraries because it make the activity easier.

In [ ]: