LECTURE 2 (PART 1) - MARCH 14, 2024

statistics LIBRARY

THIS LIBRARY CONTAINS THE MOST COMMON FUNCTIONS USED IN DESCRIPTIVE STATISTICS: IT IS ALREADY IN Anaconda.

TO IMPORT THE LIBRARY

```
import statistics as stat
```

LIBRARY NAME — NICKNAME

[]:
```
import statistics as stat

# List of marks of a student
marks = [26, 28, 27, 30]

print('Mean =', stat.mean(marks))
print('Variance (pop) =', stat.pvariance(marks))
print('Variance (samp) =', stat.variance(marks))
print('Mode =', stat.mode(marks))
print('Median =', stat.median(marks))
print('Quantiles =', stat.quantiles(marks))
```

WE CAN COMPUTE THE MINIMUM AND MAXIMUM VALUES USING THE FUNCTIONS min AND max (THEY ARE BUILT-IN FUNCTIONS NOT IN statistics)

[]:
```
print('Minimum =', min(marks))
print('Maximum =', max(marks))
```

random LIBRARY

IN SIMULATIONS WE OFTEN NEED TO GENERATE (PSEUDO) RANDOM NUMBERS. TO DO SO, WE CAN IMPORT THE random LIBRARY: IT IS ALREADY IN Anaconda.

EXAMPLE GENERATE 10 RANDOM NUMBERS IN THE RANGE 1-6

```
[]: import random

    for k in range(10):
        print(random.randrange(1,7))
```

WE ARE NOT OBLIGED TO GIVE A NICKNAME

IT STARTS FROM 0 BY DEFAULT

K = 0, 1, ..., 9

INCLUDED

EXCLUDED

IF WE RE-EXECUTE THE for WE WILL GET DIFFERENT NUMBERS

```
[]: for k in range(10):
        print(random.randrange(1,7))
```

TO ASSURE REPRODUCIBILITY WE USE random.seed()

```
[]: random.seed(32)

    for k in range(10):
        print(random.randrange(1,7))
```

YOU CAN CHOOSE A NUMBER

```
[]: random.seed(32)

    for k in range(10):
        print(random.randrange(1,7))
```

WE GET SAME RESULTS!

EXAMPLE SHOW THE LAW OF LARGE NUMBERS ROLLING A DIE 6000000 TIMES USING randrange: IT SAMPLES FROM A UNIFORM DISTRIBUTION ON $\{1, \ldots, 6\}$ SO WE SHOULD GET RESULTS CLOSE TO $1/6 = 0.1\overline{6}$.

[ ]:
```
import random

frequency1 = 0
frequency2 = 0
frequency3 = 0
frequency4 = 0
frequency5 = 0
frequency6 = 0
N = 6000000

# N die rolls
for roll in range(N):
    face = random.randrange(1,7)

    # Increment the appropriate face counter
    if face == 1:
        frequency1 += 1
    elif face == 2:
        frequency2 += 1
    elif face == 3:
        frequency3 += 1
```

```
    elif face == 4:
        frequency4 += 1
    elif face == 5:
        frequency5 += 1
    elif face == 6:
        frequency6 += 1

print('Face 1:', frequency1 / N)
print('Face 2:', frequency2 / N)
print('Face 3:', frequency3 / N)
print('Face 4:', frequency4 / N)
print('Face 5:', frequency5 / N)
print('Face 6:', frequency6 / N)
```

WE WANT TO REPRESENT THE RELATIVE FREQUENCY DISTRIBUTION IN A **BAR CHART**.

TO DO GRAPHS AND PLOTS WE REFER TO THE LIBRARY **Matplotlib**.

[]:
```
# Create a list of relative frequencies
rfreq = [frequency1 / N, frequency2 / N, frequency3 / N,
         frequency4 / N, frequency5 / N, frequency6 / N]

# Create a list of labels
labels = ['Face 1', 'Face 2', 'Face 3', 'Face 4',
          'Face 5', 'Face 6']
```

WE NEED TO IMPORT pyplot FROM Matplotlib

[]:
```
import matplotlib.pyplot as plt
```
← NICKNAME

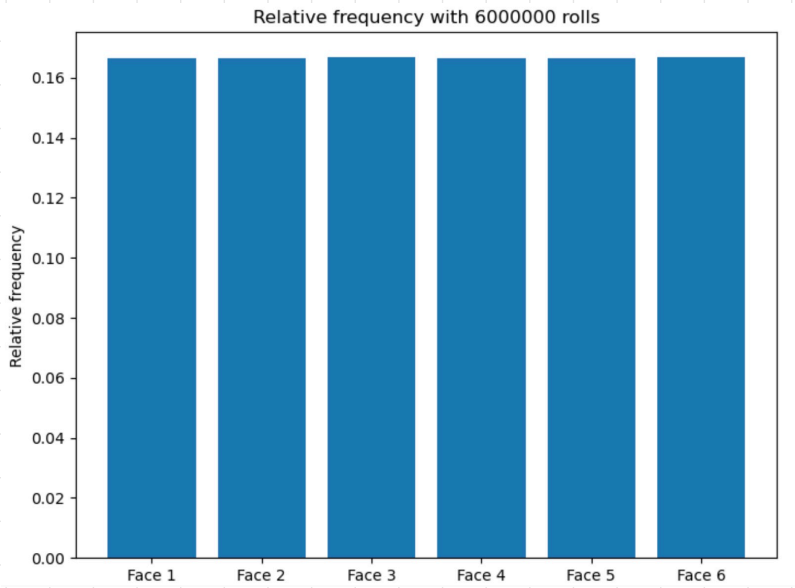
```
fig = plt.figure()
```
← CREATES A FIGURE

```
ax = fig.add_axes([0, 0, 1, 1])
```
← ADDS AXES

```
ax.set_ylabel('Relative frequency')
```
← LABEL ON y AXIS

```
ax.set_title('Relative frequency with ' +
             str(N) + ' rolls')
```
↖ TITLE

```
ax.bar(labels, rfreq)
plt.show()
```
← PLOTS AND SHOWS THE BAR CHART



LISTS AND TUPLES

LISTS AND TUPLES ARE EXAMPLES OF PYTHON'S NATIVE COLLECTIONS (NO LIBRARIES ARE NEEDED FOR THEM).

- LISTS TYPICALLY CONTAIN HOMOGENEUS DATA AND ARE MODIFIABLE → DELIMITED BY [ ]
- TUPLES TYPICALLY CONTAIN HETEROGENEUS DATA AND ARE NOT MODIFIABLE → DELIMITED BY ( )

LISTS

```
                    0  1  2  3   <- INDICES
[]: l = [8, 9, 6, 7]
    l[1] = -5
    l
```

```
[]: # Visit the list iterating its elements
    for element in l:
        print(element)
```

```
[]: # Visit the list through the indices
    for k in range(len(l)):
        print(l[k])
```

```
[]: # Create an empty list and fill it
    a = []
    for i in range(10):   <- i = 0, 1, ..., 9
        a.append(i**2)
```

TUPLES

```
[]: t = (10, 'hello', 3.3)
```
0 1 2 ← INDICES

```
[]: t[0] = 11
```
← ERROR! A TUPLE CANNOT BE MODIFIED

```
[]: # Unpacking a tuple
    first, second, third = t
    first
    second
    third
```

```
[]: # Packing a new tuple
    s = (second, first, third)
```

REMARK: A TUPLE, LIKE A LIST, CAN BE VISITED EITHER ITERATING ITS ELEMENTS OR THROUGH THE INDICES.