exercices2

November 2, 2018

1 Exercices 2

Given a list of numbers, print the largest one.

```
In [66]: def maximum(liste):
             maxi = -float('Inf')  # or numpy.inf
             for n in liste:
                 if n > maxi:
                     maxi = n
             return maxi

         print(maximum([1, -6, 78, 34, 22]))
```

```
78
```

Given a list of numbers, print them separated by a space (e.g. `[1, 2, 4]` -> `1 2 3`).

```
In [67]: l = [1, 2, 4]

         print(" ".join([str(n) for n in l]))
         # or
         for n in l:
             print(n, end=' ')
```

```
1 2 4
1 2 4
```

Given a list of words, print how many different words are in that list (hint: use a dictionary or a set)

```
In [68]: l = ['Alan', 'Joe', 'Alan', 'Jim']

         print(len(set(l)))  # shortest solution, using a set
         # or
         n = 0
         d = {}
         for e in l:
```

```
            if not e in d.keys():
                d[e] = 1
                n = n + 1
        print(n)
```

```
3
3
```

Given a list of words, count the number of times each word appears in the list. Eg. `[Jim, Alan, Jim, Joe]` -> `Jim:2, Alan:1, Joe:1` (hint: use a dictionary)

```
In [69]: d = {}
         for e in l:
             if not e in d.keys():
                 d[e] = 1
             else:
                 d[e] += 1
         print(d)
```

```
{'Alan': 2, 'Joe': 1, 'Jim': 1}
```

Write a script that prints the first 10 lines of a file (or the whole file is it is less than 10 lines long).

```
In [70]: f = open('aga.txt', 'r', encoding='utf-8')
         for l in f.readlines()[:10]:
             print(l, end='')
         f.close()
```

```
sdfsd Cogmaster
qsfqsfqs
qsfdqds
sqf vdsv
Cogmaster qsdf
```

Write a script that prints the last 10 lines of a file (or the whole file is it is less than 10 lines long).

```
In [71]: f = open('aga.txt', 'r', encoding='utf-8')
         all_lines = f.readlines()
         for l in all_lines[-10:]:
             print(l, end='')
         f.close()
```

```
sdfsd Cogmaster
qsfqsfqs
```

```
qsfdqds
sqf vdsv
Cogmaster qsdf
```

Two taxi companies propose differents pricing schemes: "A charges 4.80 plus 1.15 by km travelled; B 3.20 plus 1.20 by km travelled. Write a script that finds which company is the cheapest as a function of the distance to travel.

```
In [72]: def selectAorB(distance):
             priceA = 4.8 + 1.15 * distance
             priceB = 3.2 + 1.20 * distance
             if priceA < priceB:
                 return 'Take A!'
             else:
                 return 'Take B!'

         for d in range(1, 50):
             print(f"{d} km -> " + selectAorB(d))
```

```
1 km -> Take B!
2 km -> Take B!
3 km -> Take B!
4 km -> Take B!
5 km -> Take B!
6 km -> Take B!
7 km -> Take B!
8 km -> Take B!
9 km -> Take B!
10 km -> Take B!
11 km -> Take B!
12 km -> Take B!
13 km -> Take B!
14 km -> Take B!
15 km -> Take B!
16 km -> Take B!
17 km -> Take B!
18 km -> Take B!
19 km -> Take B!
20 km -> Take B!
21 km -> Take B!
22 km -> Take B!
23 km -> Take B!
24 km -> Take B!
25 km -> Take B!
26 km -> Take B!
27 km -> Take B!
28 km -> Take B!
```

```
29 km -> Take B!
30 km -> Take B!
31 km -> Take B!
32 km -> Take A!
33 km -> Take A!
34 km -> Take A!
35 km -> Take A!
36 km -> Take A!
37 km -> Take A!
38 km -> Take A!
39 km -> Take A!
40 km -> Take A!
41 km -> Take A!
42 km -> Take A!
43 km -> Take A!
44 km -> Take A!
45 km -> Take A!
46 km -> Take A!
47 km -> Take A!
48 km -> Take A!
49 km -> Take A!
```

Computing descriptive statistics from a detection experiment

In a signal detection experiment, a faint stimulus (e.g. a faint sound or a faint visual target) is presented or not at each trial and the participant must indicate whether he has perceived it or not. There are four possible outcomes for each trial:

1. A *hit* is when the participant correctly detects the target.
2. A *miss* is when the target was there but the participant did not detect it.
3. A *false alarm* is when the participant reports the presence of the target when it was not actually there.
4. A *correct rejection* is when the participant correctly reports that the target was not present.

One defines;

- The *hit rate* , equal to #hits / (#hits + #misses)
- The *False alarm rate*, equal to #false alarms / (#false alarms + # correct rejections)

Let us first suppose that the data from a participant is represented as a string. This string represents a series of trials, each trial being represented by two characters indicating the trial type (1=target present, 0=target absent) and the participant's response (Y=target perceived, N=No target perceived). For example:

```
---
data = "0Y,0N,1Y,1Y,0N,0N,0Y,1Y,1Y"
---
```

Write a function that, given such a string, returns the Hit rate and the False rate (hint: use the function `split()`)

```
In [73]: def hit_fa(data):
             hit, miss, cr, fa = 0, 0, 0, 0
             trials = data.split(',')
             for t in trials:
                 if t == '1Y':
                     hit += 1
                 elif t == '0Y':
                     fa += 1
                 elif t == '1N':
                     miss += 1
                 elif t =='0N':
                     cr += 1
                 else:
                     print('Wrong coding:' + t)
             hit_rate = (hit / (hit + miss))
             fa_rate = (fa / (fa + cr))
             return (hit_rate, fa_rate)

         print(hit_fa("0Y,0N,1Y,1Y,0N,0N,0Y,1Y,1Y"))
```

```
(1.0, 0.4)
```

Now, the results from different participants are stored in different files `subj*.dat` (download the files from https://github.com/chrplr/PCBS/tree/master/exercices2/subjdat.zip') Write a script that computes the hit rates and false alarms for each subject, andisplays the group averages and standard deviations.

```
In [74]: import glob
         files = glob.glob('subj*.dat')
         for f in files:
             with open(f, 'r') as sub:
                 print(f, "%.02f, %.02f" % hit_fa(sub.read()))
```

```
subj18.dat 0.67, 0.34
subj08.dat 0.53, 0.14
subj41.dat 0.63, 0.19
subj22.dat 0.78, 0.04
subj16.dat 0.68, 0.32
subj40.dat 0.53, 0.09
subj38.dat 0.76, 0.41
subj29.dat 0.88, 0.34
subj30.dat 0.86, 0.11
subj35.dat 0.54, 0.06
subj14.dat 0.57, 0.31
subj44.dat 0.56, 0.38
subj31.dat 0.81, 0.21
subj20.dat 0.69, 0.38
```

```
subj19.dat 0.87, 0.28
subj25.dat 0.81, 0.19
subj15.dat 0.77, 0.09
subj49.dat 0.58, 0.10
subj17.dat 0.89, 0.05
subj12.dat 0.72, 0.13
subj36.dat 0.78, 0.03
subj28.dat 0.65, 0.12
subj02.dat 0.45, 0.36
subj26.dat 0.69, 0.35
subj50.dat 0.63, 0.23
subj06.dat 0.66, 0.31
subj37.dat 0.74, 0.02
subj21.dat 0.83, 0.39
subj32.dat 0.90, 0.24
subj11.dat 0.52, 0.32
subj10.dat 0.74, 0.36
subj33.dat 0.83, 0.31
subj13.dat 0.76, 0.29
subj34.dat 0.68, 0.24
subj27.dat 0.76, 0.37
subj39.dat 0.90, 0.11
subj42.dat 0.63, 0.45
subj48.dat 0.62, 0.45
subj04.dat 0.44, 0.36
subj47.dat 0.71, 0.17
subj09.dat 0.56, 0.21
subj07.dat 0.82, 0.21
subj01.dat 0.67, 0.29
subj46.dat 0.72, 0.28
subj43.dat 0.67, 0.25
subj24.dat 0.82, 0.34
subj23.dat 0.85, 0.24
subj05.dat 0.74, 0.36
subj03.dat 0.70, 0.16
subj45.dat 0.87, 0.30
```

Use `matplotlib.pyplot.plot` to display each participant as a dot on a graphics with False alarm rate on the X-axis and Hit Rate on the Y-axis. Read the section on reading comma separated value ('.csv") files from http://automatetheboringstuff.com/chapter14/

```
In [75]: import glob
         files = glob.glob('subj*.dat')
         scores = []
         for f in files:
             with open(f, 'r') as sub:
                 scores.append(hit_fa(sub.read()))
```

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

sc = np.array(scores)
plt.scatter(sc[:,1], sc[:,0])
plt.ylim(0, 1)
plt.xlim(0, 1)
plt.show()
```