Exercices de programmation en Python

Christophe Pallier

Fall. 2018

You can get help on these exercises by joining a discussion forum on slack: `https://cogmaster-pcbs.slack.com`

Solutions for these exercises are available on the forum and/or at `http://www.github.com/chrplr/PCBS`

1. *computer-guess-a-number* Read chapter 3 of *Invent your own games with Python* `https://inventwithpython.com/invent4thed/chapter3.html` where the author presents a game where the computer chooses a random number that the user must guess. Study the code. Now, you task is to write another program, where the roles are inverted: the computer tries to guess a number that the user has in mind. The computer proposes a number and the user answeres with '+' (the number he has is mind is larger), '-' (if it is smaller), 'y' (if the guess is correct)

2. *prime numbers*. Write a script that lists all prime numbers between 1 and 10000 (A prime number is a integer that has no divisors except 1 and itself). You may use the following function:

```
def is_divisor(a, b):
""" Args: a, b integers;
Return True if b is a divisor of a, else False"
return a % b == 0
```

3. *multiplication tables* Read `https://pyformat.info/#number_padding` and witre a script that displays the table of multiplication from 1 to 9 in a nice format.

4. Given a list of numbers, print the largest one.

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

6. *Pascal triangle* Write a program that prints the first rows of Pascal's triangle (see `https://www.youtube.com/watch?v=XMriWTvPXHI`). For example:

```
%run triangle-de-Pascal.py
1
1 1
1 2 1
```

```
1 3 3 1
1 4 6 4 1
1 5 10 10 5 1
1 6 15 20 15 6 1
1 7 21 35 35 21 7 1
1 8 28 56 70 56 28 8 1
1 9 36 84 126 126 84 36 9 1
```

Pour résoudre ce problème, une solution consiste à stocker les valeurs de la ligne courante dans une liste Python, et d'écrire une fonction qui étant donnée une liste en argument, calcule et renvoit la ligne suivante dans une nouvelle liste.

4. *string-detector* Lisez le chapitre 8 de *Automate the boring stuff* (`http://automatetheboringstuff.com/chapter8/`). Ecrire un programme qui ouvre et lit un fichier texte, puis affiche toutes les lignes qui contiennent un mot donné, par exemple "Cogmaster".

5. *Rodrego-simulator* (optionnel: seulement pour ceux qui se sentent capables et sont motivés) Ecrire un programme qui simule une machine RodRego à 10 registres (`http://sites.tufts.edu/rodrego/`). Le programme est stocké dans un fichier texte qui est lu, puis executé. Votre programme doit contenir une fonction qui etant donnée les 10 valeurs initiales des registres, et le programme, renvoit les nouvelles valeurs des registres quand l'instruction END est atteinte.

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

7. 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 dictionary)

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

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

10. Two taxi companies propose differents pricing schemes: "A charges rge 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.

11. 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()`)

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.

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/`

9. Write a reverse Polish arithmetic expression evaluator (`https://en.wikipedia.org/wiki/Reverse_Polish_notation`). E.g. `3 4 * 5 -` evaluate to `7`.

10. Experiment 1 consists in a series of trials of two types, 'TypeA' or 'TypeB'.

- Write a function which takes `N`, the total number of trials, and returns a list of labels 'TypeA' and 'typeB', in a random order (hint: use `random.shuffle`).
- Create lists of trials for 20 participants. Each list must be saved in a text file with one column and one line per trial where each line contains a labelcorresponding the trial type).

11. Experiment 2 consists in a series of trials where a written stimulus is presented: the stimulus can be a French word or pseudowords, or an English words or pseudowords (the task is a lexical decision, that is, the participants must decide as quickly as possible if the stimulus is an existing word or not). Create text files listing 100 trials in a random order.

12. 'Kaprekar-numbers' Un nombre de Kaprekar est un nombre dont la représentation décimale du carré peut être découpée en une partie gauche et une partie droite (non nulle) telles que la somme de ces deux parties redonne le nombre initial. Par exemple:

    - 703 est un nombre de Kaprekar en base 10 car 703^2 = 494 209 et que 494 + 209 = 703.
    - 4879 est un nombre de Kaprekar en base 10 car 4879^2 = 23 804 641 et 04641 + 238 = 4879

    Ecrire un programme qui renvoit tous les nombres de Kaprekar entre 1 et N.

13. Posner cuing task:

    1. modifier le script de détection simple (simple-detection-visual-expyriment) pour que la cible apparaisse aléatoirement soit à gauche, soit a droite du centre de l'ecran (qui reste indiqué par une croix).
    2. programmer la tache de détection avec indicage attentionnelle de Posner (`https://en.wikipedia.org/wiki/Posner_cueing_task`). (indice endogène seulement).

14. Zipf length. Utiliser le contenu du fichier `lexical-decision/lexique382-reduced.txt` pour afficher la relation entre longueur des mots frrançais et leur fréquence d'occurence