6.1.2 Boucle `for`

Fiche d'exercices

1. Exercice 1

On donne une liste d'acteurs. Utilisez cette liste pour produire la sortie suivante:

```
Tahar a eu le César du meilleur acteur
Omar a eu le César du meilleur acteur
Guillaume a eu le César du meilleur acteur
Swann a eu le César du meilleur acteur
Alex a eu le César du meilleur acteur
Roschdy a eu le César du meilleur acteur
```

In [ ]:
```
liste_acteurs = ['Tahar', 'Omar', 'Guillaume', 'Swann', 'Alex', 'Roschdy']
```

2. Exercice 2

Produire la sortie:

```
IIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIII
NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
OOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOO
```

(il y a 80 caractères par ligne)

In [ ]:
```

```

3. Exercice 3

Dans l'extrait de code suivant:

- `chaine` est une variable initialisée avec un `str` vide : `""`;
- on veut qu'en sortie de programme cette variable contienne la valeur `'aeiouy'`.

L'idée est d'ajouter une par une les lettres à la variable `chaine`.

Compléter le code.

In [ ]:
```
chaine = ""
```

```
for  in ['a', 'e', 'i', 'o', 'u', 'y']
```

Cette variable `chaine` est appelée un **accumulateur**.

4. Exercice 4

En Python, la fonction `ord` renvoie le code Unicode d'un caractère et la fonction `chr` le contraire: elle renvoie le caractère correspondant à un code Unicode.

Par exemple:

In [ ]:
```
ord('a')
```

In [ ]:
```
chr(97)
```

Voici une liste contenant les codes Unicode des lettres d'un mot secret... À vous d'écrire un programme où en sortie, la variable `mot_secret` contiendra la chaîne de caractères de ce mot.

In [ ]:
```
mystere = [111, 107, 44, 32, 98, 105, 101, 110, 32, 106, 111, 117, 233]
mot_secret = ""
```

5. Exercice 5

Dans la cellule suivante, modifier les paramètres de `range` pour obtenir successivement:

1. `[0, 1, 2, 3, 4]`
2. `[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 ,11, 12]`
3. `[0, 1 ,2, 3, ..., 100]`
4. `[1, 2, 3, 4, ...,  50]`
5. `[10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20]`
6. `[0, 5, 10, 15, 20, 25, 30]`
7. `[10, 12, 14, 16, 18]`
8. `[20, 18, 16, 14, 12, 10, 8, 6, 4, 2]`

In [ ]:
```
list(range())
```

6. Exercice 6

On souhaite calculer la somme des 1000 premiers nombres entiers naturels, c'est-à-dire:

$1+2+3+4+5+ \dots +999+1000$

Écrire un programme avec une variable `somme` **accumulateur** (comme à l'exercice 3) qui contiendra la valeur souhaitée en fin de programme.

In [ ]:

7. Exercice 7

Calculer $1\times 2 \times 3 \times \dots 99 \times 100$.

In [ ]:

8. Exercice 8

Des valeurs sont données dans la liste `nombres`. Écrire un programme qui calcule leur moyenne.

In [ ]:
```
nombres = [15, 8, 12, 19, 10, 17]
```

9. Exercice 9

Même exercice que le précédent, cette fois-ci:

- en demandant le nombre de valeurs à saisir
- en saisissant au clavier les valeurs une à une

Ne pas hésiter à consulter la page https://cgouygou.github.io/1NSI/T12_Divers/5Trucs/Trucs/

In [ ]: