Informatique 2: Travaux Pratiques

Manipulation avancée des objets de base

Le but de cette séance est de se familiariser avec la manipulation avancée des objets de bases en Python. **Procéder par étape et tester votre code pour chaque étape.** Un fichier de test vous est fourni sur Moodle, pour que vous puissiez tester vos fonctions.
Temps mentionné (🕓) à titre strictement indicatif.

Exercices de base

Question 1: (🕓 *5 minutes*) Définir sous la forme d'une fonction lambda, une fonction **triple** renvoyant le triple d'un nombre.

>_ Exemple

triple(3) renvoie **9**

Question 2: (🕓 *5 minutes*) Définir sous la forme d'une fonction lambda, une fonction appelée **est_pair** renvoyant **True** si un nombre est pair et **False** sinon.

>_ Exemple

est_pair(4) renvoie **True**, **est_pair(7)** renvoie **False**

Question 3: (🕓 *5 minutes*) Définir une fonction **multiplicateur** qui prend un entier **n** en paramètre et renvoie une fonction anonyme. Cette fonction prend un entier **x** en paramètre et renvoie cet entier multiplié par **n**.

>_ Exemple

multiplicateur(5) renvoie la fonction $x \mapsto 5x$ et **multiplicateur(5)(3)** renvoie donc 15. **multiplicateur(3)** est équivalente à la fonction **triple** définie ci-dessus.

Question 4: (🕓 *5 minutes*) Générer la liste des entiers impairs de 0 à 10 (non inclus) à partir de la liste renvoyée par **range(10)**.

>_ Exemple

[1, 3, 5, 7, 9]

Question 5: (🕓 *10 minutes*) Générer la liste des triples des entiers de 1 à 10 de deux manières différentes.

>_ Exemple

[3, 6, 9, 12, 15, 18, 21, 24, 27, 30]

Question 6: (🕓 *10 minutes*) Générer et afficher la liste des triples des entiers pairs de 1 à 10 de deux manières différentes.

>_ Exemple

[6, 12, 18, etc].

Question 7: (*10 minutes*) Définir une fonction lambda **produit** qui renvoie le produit de deux entiers passés en paramètre. Puis, afficher le résultat du produit des entiers de 1 à 10.

Comment faire si les entiers sont représentés sous forme de chaîne de caractères ?

Exemple

['1', '2', '3'].

Question 8: (*10 minutes*) Définir une fonction lambda **factorielle** sans boucle **for**.

Question 9: (*10 minutes*) À partir d'une liste d'entiers, générer un dictionnaire qui associe à chaque entier de la liste une valeur booléenne (**True** ou **False**) indiquant si l'entier est pair ou non.

Exemple

Le dictionnaire correspondant à la liste **[1, 4, 3, 7]** est {**1: False, 4: True, 3: False, 7: False**}.

Question 10: (*10 minutes*) Soit **villes** un dictionnaire contenant comme clés des noms de villes et comme valeurs associées le nombre d'habitants de ces villes. Calculer le nombre d'habitants moyen.

villes = {**'Zurich': 409241, 'Genève': 200548, 'Bâle': 171513, 'Lausanne': 138905, 'Berne': 133798**}

Question 11: (*10 minutes*) projet Soit **v** un entier et **l** une liste d'entiers. Définir une fonction **plus_petit_des_plus_grands** qui renvoie la plus petite valeur dans la liste **l** qui soit supérieure a **v**. La fonction renvoie **None** si aucun élément de la liste n'est plus grand que la valeur spécifiée.

Exemple

plus_petit_des_plus_grands([7, 1, 4, 0], 3) renvoie 4 et **plus_petit_des_plus_grands([7, 1, 4, 0], 8)** renvoie **None**.

Question 12: (*20 minutes*) projet Implémenter une fonction **indent** qui prend en paramètre un texte – sous forme d'une chaîne de caractères – et un espacement – sous forme d'une chaîne de caractères également – (égal à une tabulation par défaut) et qui ajoute le caractère d'espacement au début de chaque ligne du texte. Par exemple, **print('zero\n' + indent('one\n' + indent('two\nthree', '\t-'))** affiche :

```
zero
    one
        -two
        -three
```