# Programmation

## Examen intermédiaire blanc

Section MX, 2 novembre 2018

*Veuillez déposer votre carte CAMIPRO sur votre table.*
*Tous les exercices sont à faire* ***individuellement****.*

*Vous avez droit à tout votre matériel et à un accès à internet, mais* ***toute forme de communication avec une autre personne, directe ou indirecte, par quelque moyen ou logiciel que ce soit, est interdite et est, le cas échéant, immédiatement sanctionnée par la note 0****.*

*Pour que le code soit considéré comme complètement correct, il faut qu'il marche comme spécifié dans la consigne* ***même si les données utilisées (qui sont dans le code de base à télécharger) changent****.*

*Si votre code ne compile pas, mettez-le en commentaire avec un double slash au début de chaque ligne (Ctrl-Shift-C ou Cmd-Shift-C). Vous pourrez quand même obtenir des points si votre idée est bonne (mais vous n'obtenez pas de points pour la description textuelle d'une idée sans écrire de code).*

*Dans les exemples de résultats affichés sur la console, les parties soulignées doivent provenir d'une variable ou d'un calcul fait dans votre code, et non d'une chaîne de caractères prédéfinie.*

***Merci d'attendre le feu vert du surveillant pour tourner la page.***

## Exercice 1. Choix multiples

Lesquelles de ces affirmations sont correctes? 20 pts

1. Selon la valeur initiale de **`d1`**, ce code peut afficher soit **`True`**, soit **`False`**.

```
d1: float = # une valeur de type float
i = int(d1)
d2 = float(i)
print(d1 == d2)
```

2. Je peux toujours accéder à la case 0 d'une liste **`my_list`**, peu importe sa taille, sans générer une erreur d'exécution, par exemple comme ceci:

```
print(my_list[0])
```

3. Si, dans la méthode **`__init__`** d'une classe **`A`**, j'écris uniquement **`self.b = 42`**, alors, pour tout objet **`a`** de type **`A`**, cette ligne affichera toujours 42:

```
print(a.b)
```

4. Si **`s`** est un **`int`**, l'exécution de cette ligne générera une erreur:

```
print("valeur de s = " + s)
```

5. Dans une classe, une méthode ne peut pas déclarer de paramètres qui ont le même nom que les champs de la classe (champs = propriétés = variables d'instances).

6. Ce code incrémente la case **`i`** de la liste **`my_list`** de 1:

```
my_list[i += 1]
```

7. Dans le code d'une fonction dont le type de retour n'est pas **`None`**, je dois de toute façon utiliser le mot clé **`return`** au moins une fois pour que le code s'exécute correctement.

8. À la fin d'un bloc **`if`**, je peux rajouter une série de blocs **`elif`** ou un bloc **`else`**, ou même à la fois plusieurs **`elif`** et un **`else`** final.

9. Si j'écris **`a + b`** en Python et que **`b`** est un string, il est converti en nombre pour effectuer l'opération arithmétique.

10. Si j'écris **`a / b`** en Python et que **`a`** et **`b`** sont les deux des **`int`**s, alors le résultat de cette expression est toujours de type **`int`**.

## Exercice 2. Démarrage

(a) Démarrez Visual Studio Code normalement et sélectionnez correctement votre *workspace* à l'ouverture; créez un nouveau fichier Python pour l'examen intermédiaire.

(b) Allez sur la page Moodle du cours et copiez-collez le code de départ dans votre nouveau fichier. Le code, partiellement commenté, doit compiler et afficher deux «sections» vides sur le terminal quand on le lance. 5 pts

## Exercice 3. Fonctions, conditions, boucles

Nous allons écrire du code relatif aux nombres parfait. Un nombre est dit parfait s'il est égal à la somme de ses diviseurs autres que lui-même. Par exemple, les diviseurs de 6 sont 1, 2, 3 et 6. La somme des diviseurs autres que 6 est 1 + 2 + 3 = 6, donc 6 est un nombre parfait.

(a) Dans votre fichier, ajoutez une fonction nommée **`count_divisors_of`**, qui accepte un argument de type **`int`** (dont on fait l'hypothèse qu'il sera toujours strictement positif) et retourne le nombre de diviseurs de son argument. 6 pts

Décommentez la première ligne commentée du code fourni pour vérifier que votre code indique bien que 24 a 8 diviseurs.

(b) De façon similaire, ajoutez une fonction **`divisors_of`**, qui retourne une liste avec tous les diviseurs de son unique paramètre de type **`int`**. 10 pts

Décommentez les deux lignes suivantes du code fourni pour vérifier que votre code indique bien que les diviseurs de 24 sont 1, 2, 3, 4, 6, 8, 12 et 24.

(c) Ajoutez similairement une fonction **`sum_of`**, qui retourne la somme de toutes les valeurs d'une liste de **`int`**s passée en paramètre. 5 pts

Décommentez la ligne suivante du code fourni pour vérifier que votre code indique bien que la somme des diviseurs de 24 est 60.

(d) Ajoutez une fonction **`is_perfect`**, qui indique si oui ou non son unique paramètre de type **`int`** est un nombre parfait. Faites appel aux fonctions **`divisors_of`** et **`sum_of`**. 5 pts

(e) Finalement, ajoutez la fonction **`show_perfect_numbers_up_to`**, qui accepte un argument de type **`int`** et qui trouve (en utilisant des appels à la méthode **`is_perfect`**) tous les nombres parfaits plus petits que ou égaux à cet argument. 11 pts

Ces nombres parfaits (ainsi que le nombre de nombres parfaits trouvés) doivent être affichés sur le terminal selon le format ci-dessous. Décommentez le reste des lignes du code fourni pour cet exercice. Votre code devrait alors afficher ceci:

```
Nombre parfait numéro 1: 6
Nombre parfait numéro 2: 28
Nombre parfait numéro 3: 496
3 nombres parfaits plus petits que 1000 ont été trouvés
```

## Exercice 4. Classes, champs, `__init__`

Nous allons modéliser des personnages et des relations simples entre eux.

(a) Pour modéliser un personnage, créez une nouvelle classe nommée **`Person`**. Elle modélisera une personne en stockant ces données dans des champs avec les noms suivants: 6 pts

- **`id`** pour stocker stocker un **`int`** unique pour identifier le personnage. On fait l'hypothèse que ces identificateurs seront toujours assignés depuis 0 dans l'ordre croissant;
- **`name`** pour stocker le nom du personnage;
- **`age`** pour stocker son âge sous forme de nombre entier;
- **`best_friend`** et **`worst_enemy`**, les deux de type **`Person`**, pour stocker soit `None` (la valeur par défaut), soit une référence vers l'éventuel meilleur ami ou pire ennemi d'un personnage.

Ajoutez la méthode standard **`__init__`** pour donner leur valeur aux *trois premiers de ces champs*. Initialisez les deux derniers à **`None`**.

(b) Ajoutez la méthode standard **`__repr__`** dans la classe **`Person`** pour faire en sorte qu'un personnage soit converti en texte selon cet exemple: 6 pts

**`Luke [0], 19 ans`** *(le 0 entre crochets est l'id du personnage)*

(c) Décommentez, dans le code fourni, les lignes qui définissent les 6 personnages et les relations entre eux. Ajoutez du code qui insère dans une liste tous ces personnages, dans l'ordre dans lequel ils sont créés. Ensuite, affichez, avec une boucle **`for-in`**, chacun de ces personnages sur le terminal.

(d) En-dessous, ajoutez du code — séparé du code de (c) — qui trouve et affiche selon le format suivant tous les personnages qui ont aussi bien un meilleur ami qu'un pire ennemi:

**`Han a aussi bien un meilleur ami (Chewbacca) qu'un pire ennemi (Jabba)`**

(e) Ajoutez du code qui trouve et affiche ainsi les personnages qui sont leur propre pire ennemi:

**`Anakin est son propre pire enemi`** 3 pts

(f) Ajoutez du code qui trouve et affiche selon le format suivant les personnages qui n'ont ni meilleur ami ni pire ennemi: 4 pts

**`Yoda est-il apathique?`**

(g) Ajoutez du code qui trouve tous les personnages qui sont le meilleur ami de leur meilleur ami et les affiche selon ce format: 7 pts

**`Han et Chewbacca sont meilleurs amis symétriquement`**

Attention à ne mentionner qu'une fois chaque paire d'«amis symétriques» et non deux fois avec la seconde fois dans l'ordre inversé de leurs noms.

(h) Calculez et affichez l'âge moyen $\mu$ des personnages sans meilleur ami selon ce format:

**`La moyenne d'âge des personnages sans meilleur ami est de 389.25 ans`** 7 pts

(i) Vérifiez si la valeur de $\mu$ est valable comme médiane, c'est-à-dire si, parmi les personnages sans meilleur ami, il y en a autant dont l'âge est strictement inférieur à $\mu$ qu'il y en a dont l'âge est strictement supérieur à $\mu$. Si c'est le cas, affichez: 10 pts

**`La moyenne est valable comme médiane`**

Sinon, affichez:

**`La moyenne n'est pas valable comme médiane`**

***N'oubliez pas de rendre votre fichier .py sur Moodle.***
***Vérifiez que votre/vos fichier(s) ai(en)t bien été uploadé(s) correctement et qu'il(s) ai(en)t la bonne extension (`.py` et non `.pyc` ou autre).***

**Total:** 115 pts