ICC/Programmation — Série 3

GC/MX, 6 octobre 2023

À faire individuellement ou par petits groupes de deux ou trois.

Exercice 1. Boucle `while` – Mot-clé `break`

Observez le code suivant :

```
i: int = 0
while i < 10:
    print(i)
    if i == 5:
        break
    i += 1
```

a) Sans exécuter ce programme, quel sera son résultat ?

b) Vérifier votre hypothèse en exécutant le programme. Quel est le rôle du mot-clé **`break`** ?

Exercice 2. *Code tracing* : Quelques méthodes de lecture de code (papier-crayon)

Le programme ci-après reprend le code fourni pour l'algorithme « Deux font la paire », de complexité linéaire.

```
from typing import List

liste: List[int] = [-15, -12, -3, -1, 5, 17, 23]
n: int = len(liste)
i: int = 0
j: int = n - 1
found_pair: bool = False
while i < j:
    if liste[i] + liste[j] == 0:
        found_pair = True
        break
    elif liste[i] + liste[j] < 0:
        i += 1
    else:
        j -= 1
if found_pair:
    print("Paire trouvée !")
else:
    print("Pas de paire trouvée...")
```

Le *code tracing* concerne la capacité à lire et à anticiper le résultat d'un programme. Avec des programmes de plus en plus longs et complexes, il est important d'être méthodique dans la lecture de code. Pour cela, il peut être utile de construire un tableau permettant de suivre l'évolution des valeurs des variables importantes du programme. Il faut donc également pouvoir identifier ces variables importantes.

Pour l'algorithme « Deux font la paire », voici un exemple de tableau vous permettant de suivre l'exécution du programme. Recopiez et complétez ce tableau pour suivre l'exécution complète de ce programme.

	i	j	liste[i]	liste[j]	found_pair
Step 0	0	6	-15	23	False
Step 1	0	5	-15	17	False
Step 2					
Step 3					
Step 4					
...					

Table 1 – Tableau d'exécution et de suivi de variables

Exercice 3. Boucle `while` – Nombre mystère

Le code suivant permet d'affecter un entier aléatoire compris entre 0 et 10000 (inclus) à la variable **`random_number`**.

```python
from random import randint

random_number: int = randint(0, 10000)
```

L'objectif de cet exercice est d'implémenter un jeu où l'ordinateur choisit aléatoirement un nombre et l'utilisateur doit le trouver en un minimum d'essais. Pour cela, voici quelques indications qui pourront vous aider dans la construction de ce programme :

- Quelles sont les données manipulées ? Quelles sont les variables nécessaires à l'implémentation de ce programme ?
- L'exercice porte sur la boucle **`while`**, quelle est condition la condition qui va permettre de répéter les instructions dans le corps de la boucle ?
- Quelles sont les instructions répétées dans le corps de la boucle ?

Vous pourrez trouver ci-après un exemple d'interactions dans le terminal :

```
Nombre en 0 et 10000: 5000
C'est plus !
(Quelques interactions intermédiaires...)
Nombre en 0 et 10000: 5831
Trouvé !
Vous avez trouvé en 11 essais !
Le nombre mystère était 5831.
```

Exercice 4. Un petit interpréteur interactif (s02e08)

Note : Si vous n'avez pas encore fait cet exercice (et c'est normal, on n'a pas vu le concept en cours la semaine passée), faites-le lors de cette séance d'exercices.

(a) Depuis la page Moodle du cours, copiez et collez ce code dans un nouveau fichier Python.

(b) Que fait chaque ligne de ce programme ? Comment est-ce que la boucle fonctionne ? Quelle est la condition ? Quand et comment la variable de boucle est-elle modifiée ?

```python
should_continue: bool = True

while should_continue:
    print("Je vais évaluer un calcul pour vous.")

    number1_string: str = input("Tapez le premier nombre: ")
    number1: float = float(number1_string)

    number2_string: str = input("Tapez le second nombre: ")
    number2: float = float(number2_string)

    operation: str = input("Tapez l'opération: ")
    if operation == "+":
        result = number1 + number2
        print(f"{number1} + {number2} = {result}")
    else:
        print(f"Désolé, je ne connais pas l'opération '{operation}'")

    maybe_quit: str = input("Tapez 'q' pour quitter, ou autre chose pour recommencer: ")
    if maybe_quit == "q":
        print("Bye!")
        should_continue = False
```

(c) Modifiez le programme pour qu'il effectue aussi une addition si on tape `"addition"` plutôt que `"+"` comme opération.

(d) Modifiez le programme pour qu'il puisse aussi évaluer des soustractions, multiplications et divisions.

(e) Dans le cas de la division, assurez-vous que le diviseur soit différent de zéro. Sinon, affichez le message suivant **`"Division par zéro, calcul impossible"`**.

Exercice 5. Définition et appel de fonction

Dans l'exercice 7 de la série 2, nous avons complété le programme « Calendrier du cours » pour qu'il prenne en compte les années bissextiles.

a) Reprenez le programme rédigé pour cet exercice.

b) Quel sera l'en-tête d'une fonction appelée **`is_leap_year(...) -> ?`** qui détermine si une année est bissextile ou non ?

c) Implémentez cette fonction en vous aidant de ce que vous avez fait pour l'exercice 7 de la série 2.

d) Appelez cette nouvelle fonction dans votre programme pour obtenir le même comportement que précédemment.

e) *Bonus :* Modifiez ce programme pour prendre en compte le passage à une nouvelle année.

Exercice 6. Fonctions, *default argument values*, et *keyword arguments*

Exercice papier-crayon

Observez le programme suivant.

```
1  def favorite_course(course_name: str = "ICC") -> None:
2      print(f"Mon cours préféré est {course_name}")
3
4  def liked_and_disliked(liked: str = "ICC", disliked: str = "Maths") -> None:
5      print(f"Mon cours préféré est {liked}.")
6      print(f"Mon cours le moins préféré est {disliked}.")
7
8  favorite_course()
9  favorite_course("Physique")
10 liked_and_disliked()
11 liked_and_disliked(disliked="ICC", liked="Maths")
```

a) Sans exécuter ce programme, quel sera son résultat ?

b) Quelle nouveauté se trouve en ligne 1 ? Quel est son effet ?
Aide : Vous pouvez chercher sur Internet ce qu'est un "default argument values" en Python.

c) Quelle nouveauté se trouve en ligne 11 ? Quel est son effet ?
Aide : Vous pouvez chercher sur Internet ce qu'est un "keyword argument" en Python.