ICC/Programmation — Série 2

GC/MX, 29 septembre 2023

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

Note : Il y a beaucoup d'exercices. C'est voulu : il s'agit de pouvoir pratiquer un maximum les concepts vus en cours. Il ne faut donc pas se décourager si vous n'avez pas fini la série durant la séance d'exercices, mais pensez à essayer de les faire pour plus tard.

Exercice 1. Expressions booléennes

Partie «papier-crayon»

Nous avons vu dans le cours plusieurs opérations dont le résultat est une valeur booléenne. Cet exercice sera idéalement réalisé sur papier, puis vos réponses pourront être vérifiées avec Python.

Sans exécuter ou tester le programme sur ordinateur, indiquez si le résultat des expressions booléennes suivantes est **`True`** ou **`False`**. L'objectif de cet exercice est d'y voir plus clair dans la priorité des opérateurs.

```
a: bool = 4 + 2 < 1
b: bool = not True and not False or False
c: bool = "epfl" != "EPFL" and not False
d: bool = "genève" > "lausanne"
```

Partie sur Visual Studio Code

a) Ouvrez Visual Studio Code ainsi que votre **workspace** s'il ne s'est pas ouvert tout seul.

b) Créez un nouveau fichier qui s'appelle **`s02e01.py`**. *Note : Prenez l'habitude de créer un nouveau fichier à chaque fois, même si cela n'est pas précisé dans la donnée de l'exercice.*

c) En utilisant la fonction **`print()`**, vérifiez vos réponses.

d) Si vous avez fait des erreurs, essayez de comprendre pourquoi. Si vous avez encore des doutes, appelez l'enseignant ou un·e assistant·e.

Exercice 2. Fonction `input()` et I/O

I/O (ou IO) signifie *Input/Output*, soit «entrées/sorties». Pour cet exercice, nous allons lire et écrire des lignes de texte sur la console, c'est-à-dire la «zone des résultats» où s'affiche le résultat des **`print()`**.

Dans le code suivant, la fonction **`input()`** aura pour effet d'arrêter l'exécution du programme et d'attendre que l'utilisateur tape quelque chose dans le terminal, suivi de la touche Entrée. Le texte saisi à ce moment sera alors affecté à la variable **`first_name`**.

```
print("Bonjour, veuillez saisir votre prénom: ")
first_name: str = input()
print(f"Bonjour {first_name} !")
```

a) Recopiez ce code et testez le.

b) Après la seconde ligne, insérez l'instruction suivante :

```
last_name: str = input("Veuillez saisir votre nom de famille: ")
```

c) Notez la différence de syntaxe, que pensez-vous que cette instruction va faire ?

d) Modifier la dernière ligne de votre programme pour afficher les valeurs contenues dans les variables **`first_name`** et **`last_name`**.

Exercice 3. Structures conditionnelles et utilisation de l'opérateur "modulo"

Une année n est bissextile si elle respecte l'un ou l'autre des deux critères suivants :
- n est divisible par 4 sans être divisible par 100 ;
- n est divisible par 400.

a) Observez le code suivant, recopiez-le puis exécutez-le.

```
mod_1: int = 5 % 2
mod_2: int = 18 % 7
mod_3: int = 24  % 8
print(mod_1)
print(mod_2)
print(mod_3)
```

À votre avis, quel est le rôle de l'opérateur `%` (qui se lit «modulo») ? En cas de doute, demandez à l'enseignant ou à un·e assistant·e.

b) Reformulez la définition d'une année bissextile en mettant en évidence les mots SI, ALORS, ainsi que d'éventuelles opérations conduisant à une valeur booléenne.

c) En utilisant l'opérateur modulo, écrivez un programme qui détermine puis affiche si les années 2023 et 2024 sont bissextiles ou non. Vous pourrez utiliser une seule variable et exécuter plusieurs fois le programme avec des valeurs différentes pour cette variable.

Exercice 4. I/O et conversion de types

Nous allons progressivement améliorer le programme écrit pour l'exercice 3.

Avant cela, nous allons voir qu'il est possible de changer les types de certaines variables grâce aux fonctions **`int()`**, **`float()`**, **`str()`**.

```
var_a: str = "42"
var_b: str = "13.9"
var_c: int = 50
var_d: float = 1.4

var_a_as_int: int = int(var_a)
var_b_as_float: float = float(var_b)
var_c_as_str: str = str(var_c)
var_d_as_str: str = str(var_d)

print(type(var_a_as_int))
print(type(var_b_as_float))
print(type(var_c_as_str))
print(type(var_d_as_str))
```

a) Dans un nouveau fichier, écrivez la ligne suivante :

```
year = input("Saisir une année: ")
```

b) Reprenez désormais votre programme écrit pour l'exercice 3. Faites attention à bien faire en sorte que les noms de variables soient cohérents entre ces deux exercices. Vous pouvez désormais l'améliorer en demandant à l'utilisateur de saisir une année plutôt que de la définir «en dur» dans le programme.

c) Testez votre programme. Que se passe-t-il ?

d) Notez que la dernière ligne affichée en console indique le message suivant :
`TypeError: not all arguments converted during string formatting`.
Ce message est un peu cryptique, mais le premier mot de ce message (**`TypeError`**) nous donne une indication importante : notre programme semble avoir un problème avec les types de données que nous manipulons.

e) Entre l'exercice 3 et l'exercice 4, une variable a changé la façon dont une valeur lui était assignée. Laquelle ? Quelles sont ces deux façons ?

f) Puisque le message d'erreur indique un problème avec les types de données, utilisez la fonction **`type()`** pour afficher les types des variables utilisées. Comment pouvez-vous corriger l'erreur introduite ? Quelle conversion de type faut-il réaliser ?

Exercice 5. Boucles, manipulation de chaînes de caractères, et variables

(a) Créez un nouveau fichier Python et insérez-y cette ligne telle quelle sans la modifier :

```
line: str = input("Veuillez taper quelque chose: ")
```

Complétez ensuite le code : à l'aide d'une boucle **`for`** et en utilisant les fonctions **`len()`** et **`range()`**, affichez chaque caractère de la chaîne de caractères **`line`** individuellement avec la fonction **`print()`**. Utilisez pour cela le fait que, si **`my_string`** est une chaîne de caractères, alors **`my_string[i]`** est le caractère à la position i de cette chaîne de caractères.

(b) Modifiez votre code en simplifiant la boucle : utilisez pour cela le fait que la boucle **`for`** peut aussi marcher directement sur une valeur de type **`str`** plutôt que sur une valeur retournée par la fonction **`range()`**. Par exemple : `for c in my_string` va itérer sur tous les caractères de la chaîne de caractères **`my_string`**.

(c) Écrivez une boucle qui compte le nombre de caractères en majuscule et en minuscule de la chaîne de caractères **`line`** et qui affiche ces deux nombres.

Pour cela, vous pouvez utiliser les méthodes **`isupper()`** et **`islower()`**. Elles vous renvoient un `bool` qui indique si la chaîne de caractères sur lequel elles sont appelées est en majuscule ou minuscule, respectivement.

Exercice 6. Boucles et manipulation de chaînes de caractères, suite

Considérez le programme suivant :

```
course_title: str = "information, calcul, communication"
for index, character in enumerate(course_title):
    print(f"course_title[{index}] = {character}")
```

a) Anticipez le résultat de ce programme sans l'exécuter. Puis vérifiez votre réponse en exécutant ce programme.

b) Créez une variable **`capitalized_course_title`** qui contient, pour le moment, la chaîne de caractères vide `""`. Nous souhaitons que cette nouvelle variable contienne, à la fin de l'exécution du programme, la chaîne de caractères `"Information, Calcul, Communication"`.

i) Comment construire, lettre après lettre, le contenu de la variable **`capitalized_course_title`** pour que celui-ci soit identique au contenu de la variable **`course_title`** ?

ii) Quelles conditions permettraient de ne capitaliser que les premières lettres de chaque mot ? Vous pouvez prendre en compte l'indice des lettres (en particulier pour la première lettre) ou les caractères précédant ceux devant être écrits en majuscule.

iii) Implémentez ces conditions pour construire, lettre après lettre, le contenu souhaité de la variable **`capitalized_course_title`**. Vous pourrez utiliser la *méthode* **`.upper()`** vue dans la Série 1.

c) Affichez, après la dernière itération de la boucle, le contenu de la variable **`capitalized_course_title`**.

La suite des exercices ci-après.

Exercice 7. Calendrier du cours – Version finale

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

```python
day: int = int(input("Jour: "))
month: int = int(input("Mois: "))
year: int = int(input("Année: "))

nb_days_in_month: int = 0
new_month: bool = True

for i in range(14):
    print(f"Le cours numéro {i+1} aura lieu le {day}.{month}.{year}.")
    if new_month:
        if month in (1, 3, 5, 7, 8, 10, 12):
            nb_days_in_month = 31
        elif month in (4, 6, 9, 11):
            nb_days_in_month = 30
        else:
            nb_days_in_month = 28
    new_month = False
    if day + 7 > nb_days_in_month:
        month += 1
        new_month = True
        day = (day + 7) - nb_days_in_month
    else:
        day += 7
```

b) Complétez ce programme pour qu'il puisse également prendre en compte les années bissextiles. Pour cela, vous pourrez vous aider de l'exercice 3.

Exercice 8. Un petit interpréteur interactif

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

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

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

(d) 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"`**.