Types, opérateurs, mots-clefs et modules

INFORMATIQUE COMMUNE - TP n° 1 - Olivier Reynet

À la fin de ce chapitre, je sais :

- ☞ Utiliser Pyzo pour exécuter un code Python
- ☞ Utiliser le module os pour trouver ou changer le répertoire de travail
- ☞ Utiliser et identifier les types simples (int, float, boolean, complex)
- ☞ Utiliser les opérateurs en lien avec les types numériques et les chaînes de caractères
- ☞ Utiliser les modules os, math et random dans un code simple
- ☞ Utiliser la commande print pour faire afficher des variables formatées dans une chaîne

A Le module `os`, pour savoir où l'on se trouve dans le système de fichier

A1. Répertoire courant de travail

(a) Ouvrir l'IDE Pyzo. À droite se trouve normalement un shell interactif et à gauche un éditeur de texte.

(b) En utilisant le shell interactif, trouver le répertoire de travail courant.

(c) En utilisant la fonction `chdir` du module `os` qui signifie *change directory*, sélectionner un autre répertoire de travail courant.

(d) Dans la partie éditeur de l'IDE, à gauche, inscrire le programme suivant :

```
1 import os
2 current_dir = os.getcwd()
3 print("Current working direxctory is ", current_dir)
```

(e) Sauvegarder ce script sous le nom `tp1.py` dans le répertoire courant.

(f) Lancer l'exécution de ce premier script en choisissant Run > Run file as script [1]

(g) En utilisant le module `os`, modifier le script pour se placer dans le répertoire de l'utilisateur `/home/user`.

(h) En utilisant le module `os` et sa documentation en ligne, lister tous les répertoires de ce répertoire.

B Types et expressions

B1. Types et fonctions du module `math`

1. Cette commande aura le bon goût de sauvegarder le fichier dans le répertoire courant en même temps!

(a) Calculer le sinus de 1. Quel est le type de cette donnée?

(b) Calculer la partie entière de la constante `pi` à l'aide de la fonction `floor` du module `math`. Quel est le type de cette donnée?

B2. **Types issus d'expressions** Trouver et vérifier les types des expressions suivantes :

(a) 3

(b) -3

(c) 3.

(d) 3.5

(e) 0.334

(f) .334

(g) `3 + 3`

(h) `3 + 5.5`

(i) `3 * 3`

(j) `42 / 21`

(k) `5.5 * pi`

(l) `None`

(m) `True`

(n) `False`

(o) `[]`

(p) `[1,2,42]`

(q) `range(42)`

C Opérateurs, types et priorités

Évaluer le type puis la valeur de ces expressions à l'écrit. Les vérifier sur l'ordinateur par la suite.

C1. `10 + 20 * 30`

C2. `100 + 200 / 10 - 3 * 10`

C3. `-1**2`

C4. `(-1)**2`

C5. `3 * 39 % 5`

C6. `39 % 5 * 3`

C7. `name = "Guillaume"; age = 13; expression = name == "Guillaume"or name == "Alix"and age < 8`

C8. `name = "Guillaume"; age = 13; expression = (name == "Guillaume"or name == "Alix")and age < 8`

D Opérateurs sur les chaînes de caractères et boucle for

Pour les questions suivantes, vous aurez besoin d'utiliser les opérateurs + et * sur les chaînes et parfois des boucles `for`. On rappelle ici la syntaxe de la boucle `for` :

```
for i in range(10):
  print(i)
```

D1. Créer une chaîne de caractères `"000001110000011100000111"` qui peut représenter un signal périodique.

D2. Créer la chaîne de caractères `"AACAACGAACAACGTAACAACGAACAACGT"` qui peut représenter une chaîne de nucléotides.

D3. Importer la fonction `choice` du module `random` (et uniquement elle!). À l'aide de cette fonction et d'une boucle for, créer une chaîne de 50000 nucléotides au hasard qui peut représenter le génome d'un virus.

E Module random et affichage de chaînes de caractères

Les questions qui suivent combinent l'utilisation du module `random`, la création de chaînes de caractères et leur affichage avec `print`. On cherche à formater correctement la sortie du programme sur l'écran. La fonction `print` peut être invoquée avec une chaîne de caractères formatée (cf. documentation). Ces chaînes permettent d'insérer des variables dans la chaîne de caractères et donc de rendre l'affichage dynamique. La commande `print` les remplace lors de l'affichage par leur valeur. Une chaîne formatée est précédée d'un `f` et la variable est spécifiée entre accolades avec éventuellement son format : `f"Result = {variable1}, {variable2:format}"`.

Voici une exemple d'utilisation :

```
n = 21
coeff= 3.45667
print(f"number = {n}, coefficient = {coeff:.3f}")
# number = 21, coefficient = 3.457
```

E1. À l'aide de la fonction `randint` du module `random`, simuler le lancer de trois dés dix fois de suite. Les résultats s'afficheront sous la forme `<D1 : 3, D2 : 6, D3 : 1>`.

E2. À l'aide de la fonction `randrange` du module `random`, générer un tirage de loto. Le résultat s'affichera ainsi : `25-14-17-21-5+7`.

E3. À l'aide de la fonction `shuffle` du module `random`, générer un mélange de cartes à partir de la main "78910VDRA", représentation d'une main ordonnée sous la forme d'une chaîne de caractères. Attention, bien lire la documentation, il se peut qu'il y ait un piège!

E4. À l'aide du module `random`, générer des variations du marché aléatoirement. La sortie aura la forme suivante :

```
BRENT 0.90%
ONCE D'OR -2.45%
EUR/USD -5.06%
VIX INDEX 3.17%
BITCOIN/USD 6.09%
```

F Des séquences indiçables, tronçonables et itérables

Les chaînes de caractères `str` en python3 sont des séquences immuables. Au titre de séquence, elles sont :

indiçables on accède directement à un élément donné d'après son indice et les crochets `[]`.

tronçonnables on peut en extraire des tronçons grâce aux crochets `[start:stop:step]`,

itérables on peut parcourir tous ses éléments via une boucle for.

Voici un exemple pour chaque fonctionnalité :

```
s = "CPGE"
print(s[1])          # indexing --> P
print(s[0] + s[2])   # concatenation --> CG
print(s[1:3])        # slicing start stop --> PG
print(s[-1])         # negative indexing --> E
print(s[-3:-1])      # negative slicing --> PG
print(s[::-1])       # reverse string --> EGPC
print(s[0:-1:2])     # slicing start stop step string --> CG
```

F1. Extraire une lettre sur deux de la chaîne de caractères `"L3kedf pppzcyqbthhhguodinol huc5n'ryeztsyrtuj xdbbiongnfc oo`ze"! et l'afficher.

F2. **Palindromes :** écrire un code Python qui permet de savoir si un mot est un palindrome[2].
On peut commencer avec une boucle `while` dont on rappelle ici la syntaxe :

```
ch = "my string"
i = 0
while i < len(ch) :
    print(i)
    i += 1
```

La fonction `len` renvoie la longueur de la chaîne de caractères. Dans un second temps, on s'affranchira de la boucle en utilisant les propriétés des chaînes de caractères.

2. se lit dans les deux sens, comme *rotor* ou *radar* par exemple.