TP n° 3 : Découverte de Python

1 Langage compilé vs interprété

Pour commencer, et suivant la tradition, nous allons créer un premier programme qui affiche « *Hello world !* » à l'écran. Il est plus simple d'écrire ce programme dans un langage compréhensible par les humains (nous !) puis de le traduire en un code exécutable par la machine. Nous allons utiliser le langage « C » puis le langage Python pour bien comprendre la différence entre un langage compilé et un langage interprété.

1. Commencer par créer un répertoire `~/info/tp03`. Vous vous souvenez comment faire ?
2. Copier le fichier `~/mpsi3/informatique/tp03/hello.c` ainsi que le fichier `~/mpsi3/tp03/hello.py` dans votre répertoire `~/info/tp03` et y aller (`cd`) dans un terminal.
3. La commande `cat hello.c` permet d'afficher le contenu du programme. Avec un peu d'imagination, vous pouvez comprendre ce que ce programme réalise.
4. Pour compiler le programme, écrit en C, on va utiliser le compilateur GCC (*GNU Compiler Collection*) :

```
gcc hello.c -o hello.exe
```

   On devrait alors obtenir, dans le répertoire courant, un programme exécutable `hello.exe`.
5. Vérifier que vous avez bien les droits d'exécution pour `hello.exe` (cf. TP n° 02 !).
6. Lancer le programme en écrivant directement `hello.exe` et en validant. Que voyez-vous ? .....
7. Pour lire le contenu du programme, on peut utiliser la commande suivante : `hexdump hello.exe`. Qu'en pensez-vous ?
8. Pour en donner une représentation un peu plus compréhensible, on peut « *désassembler* » le programme. Essayer `objdump -d hello.exe`. La première colonne est une liste d'adresses mémoire, écrites en hexadécimal : ce sont les adresses où seront chargées les instructions du programme. La seconde colonne montre ces instructions sous forme hexadécimale. Enfin, les dernières colonnes proposent une représentation sous forme plus compréhensible pour nous.
9. Python désigne aussi bien le langage que le *programme* qui permet *d'interpréter* un fichier Python (qui se termine par `.py`). Essayer la commande `python hello.py`. Que voyez-vous ? .....
10. On envoie les fichiers `hello.exe` et `hello.py` sur un autre ordinateur d'un autre type. Est-ce que l'on pourra exécuter ces programmes sur cet autre ordinateur en faisant :
    - `hello.exe` .....
    - `python hello.py` .....

    Expliquer.
11. Quels sont les avantages et les inconvénients d'un langage interprété par rapport à un langage compilé ? .....
    .....
    .....
    .....
    .....
    .....
    .....
    .....

2 Dialogue avec Python

Python [1], avec sa syntaxe élégante, son typage dynamique et étant interprété, est un langage de programmation puissant et facile à apprendre. Avant d'écrire des programmes plus conséquents, nous allons découvrir le langage en « dialoguant » avec l'interpréteur Python. C'est comme lorsque l'on dialoguait avec le terminal, qui n'est finalement rien d'autre qu'un interpréteur Bash.

12. Dans un terminal, entrer la commande `ipython` (*Interactive Python*). Une boîte de dialogue s'ouvre et attend vos requêtes. Essayer :

Python
```
In [1]: 42
Out[1]: 42

In [2]: 4 + 2
Out[2]: 6
```

On peut afficher le type d'une expression avec la fonction `type` :

Python
```
In [3]: type(42)
Out[3]: int

In [4]: type(42.0)
Out[4]: float

In [5]: type(42j)
Out[5]: complex

In [6]: type("42")
Out[6]: str
```

Nous allons maintenant étudier rapidement les types de base et les opérations associées.

1. Au fait, le nom du langage provient de l'émission de la BBC « Monty Python's Flying Circus » et n'a rien à voir avec les reptiles.

3 Types et opérations

3.1 Entiers

En Python, les entiers sont écrits en base 10 avec autant de chiffres que l'on souhaite (on parle, par abus de langage, de « précision arbitraire »). Les opérations `+`, `-`, `*` s'utilisent comme sur une calculatrice avec les priorités habituelles.

13. Calculer `42 * 123456789 - (3 + -42) * 12345 * 67890`

⚠ Les opérateurs *doivent* être entourés d'espaces [a]. On ne met pas d'espace à l'intérieur des parenthèses, crochets ou accolades. Par exemple, on écrit `(3 + 4)` et pas `( 3+4 )`.

a. Selon la PEP8 (`https://www.python.org/dev/peps/pep-0008/#whitespace-in-expressions-and-statements`) on groupe les opérateurs mathématiques ayant la priorité la plus haute pour distinguer les groupes, mais je trouve cette règle difficile à appliquer correctement. Voir par exemple la discussion ici : `https://caml.inria.fr/resources/doc/guides/guidelines.fr.html`.

L'exponentiation s'obtient avec le symbole `**`.

14. Peut-on calculer `42 ** 100000` ? .....

15. Que fait la fonction `abs` ? On pourra consulter l'aide, accessible en ajoutant le caractère « ? » juste après le nom de la fonction inconnue, par exemple : `abs?`.

Le quotient et le reste de la division euclidienne s'obtiennent respectivement avec les symboles `//` et `%`. En informatique, on dit plutôt *division entière* et *modulo*.

16. Calculer le quotient et le reste de la division euclidienne de 42 par 3.

17. Prévoir, puis calculer le quotient et le reste de la division euclidienne de 42 par −5. Qu'en pensez-vous ? .....

18. Que se passe-t-il si on essaie [2] de diviser par 0 ?

2. Pour une fois, vous avez le droit d'essayer !

Théorème 3.1

Pour tous entiers relatifs $(n, m) \in (\mathbb{Z}, \mathbb{Z}^*)$, avec $m \neq 0$, il existe un unique couple $(q, r) \in \mathbb{Z}^2$ tel que $n = qm + r$ avec r compris entre 0 inclus et m exclu.

⚠ Ce théorème est différent de celui que l'on utilise usuellement dans le cours de mathématiques.

19. Que vaut l'expression `1 - 2 - 3` ? .....
20. Que vaut l'expression `2 ** 1 ** 3` ? .....

✔ Ne pas hésiter à parenthéser lorsque cela vous semble nécessaire.

3.2 Flottants

En informatique, on ne peut pas représenter tous les réels[3] On peut cependant représenter un sous-ensemble des rationnels sous la forme de nombres à virgules (fixe ou flottante). Les nombres à virgule flottante sont appelés plus simplement « *flottants* ». On utilise le symbole « `.` » à la place de la virgule.

21. Essayer en Python :
    - `4.2`
    - `4.`
    - `42e-24`
    - `-0.42`
    - `.2`
    - `-42e1`

Comme pour les entiers, on dispose des opérateurs `+`, `-`, `*`, `**` et de la fonction `abs`.

22. Le vérifier. Que se passe-t-il si un des deux opérandes est un entier ?

3. Pourquoi ?

⚠ En mathématique, un nombre entier est aussi un nombre réel. Ce n'est pas le cas en informatique !

✔ La plupart du temps, lorsque Python requiert un flottant et qu'on lui donne un entier, il effectue la conversion automatiquement.

Les flottants disposent également d'un opérateur de division flottante « `/` », à ne pas confondre avec la division entière `//`.

23. Essayer :
    - `42. / 2.0`
    - `42. / 2`
    - `42 / 2`
    - `42 // 2`

Le *module* `math` contient de nombreuses constantes et fonctions utiles. Pour l'utiliser il faut commencer par l'importer :

Python
```
import math
```

24. Prévoir la valeur des expressions suivantes, *puis* vérifier avec Python :
    - `math.pi` .....
    - `math.sin(math.pi)` .....
    - `math.log(math.e)` .....
    - `math.log2(1.)` .....
    - `math.log2(1)` .....

Il y a bien d'autres fonctions disponibles dans le module math. On peut en obtenir la liste en écrivant `dir(math)` et en utilisant ensuite l'aide si on a un doute sur une fonction.

25. Que font les fonctions `math.floor` et `math.ceil` ?

On peut convertir un entier en flottant avec la fonction `float` et un flottant en entier avec la fonction `int`.

26. Quel est l'arrondi pratiqué par la fonction `int` ?

3.3 Booléens

Les booléens comportent deux valeurs : vrai (`True`) et faux (`False`). Les opérateurs sur les booléens correspondent aux connecteurs logiques que l'on utilise en mathématiques :

- la négation (*non logique*) : `not`
- la conjonction (*et logique*) : `and`
- la disjonction (*ou logique*) : `or`

27. Compléter le tableau suivant :

p	q	not p	p and q	p or q
True	True			
True	False			
False	True			
False	False			

✔ Les opérateurs `and` et `or` sont dits *paresseux* : ils ne calculent que lorsque cela est nécessaire.

28. Prévoir, essayer puis expliquer : `0 != 0 and 1 / 0 == 2` .....
.....

29. Que vaut :
    - `not True or True` .....
    - `(not True) or True` .....
    - `not (True or True)` .....
    - `True or False and False` .....
    - `(True or False) and False` .....
    - `True or (False and False)` .....

✔ Le test d'égalité s'écrit avec l'opérateur `==` (deux symboles « = »).

On dispose également de la différence (`!=`) et des opérateurs de comparaison (`<`, `<=`, `>`, `>=`).

30. Écrire une expression équivalente à `0 != 0` sans utiliser l'opérateur `!=`.
31. Que vaut `not (not True) and 4 <= 2 or (0 != 0 and True)` ?

4 Variables

32. Initialiser une variable[4] `foo` avec l'entier 1. Quel est son type ? .....
33. Incrémenter (c'est-à-dire ajouter 1 à) cette variable.
34. Multiplier le contenu de cette variable par π. Quel est maintenant son type ? ..
35. Initialiser une variable `bar` avec la valeur contenue dans `foo`.
36. Modifier le contenu de la variable `foo` par sa partie entière inférieure. Quel est son type ? .....
37. Est-il possible d'échanger le contenu des variables `foo` et `bar` ? .....

5 Exercices

Exercice 1 Écrire une expression qui permet de déterminer le dernier chiffre de 2018^{2018} sans afficher ce nombre en entier. .....

Exercice 2 Quelle est la valeur des expressions suivantes, lorsqu'elle existe ?

`1.5 - 1` .....
`(1 - 2).` .....
`.5 - .3` .....
`4 / (9 - 3 ** 2)` .....
`float(7 // 2)` .....

4. Les termes `foo` et `bar` sont souvent utilisés comme pantonymes en informatique.

Exercice 3 On suppose que les variables `x`, `y`, `z` sont définies dans l'environnement. Donner (puis vérifier !) une suite d'instructions permettant d'obtenir, à la fin :

- le contenu de `x` dans `z` ;
- le contenu de `z` dans `y` ;
- le contenu de `y` dans `x`.

Exercice 4 Les expressions :

- `8.5 / 2.5`
- `int(8.5) / int(2.5)`
- `int(8.5 / 2.5)`

sont-elles équivalentes ? .....

.....

.....

.....

Exercice 5

Les expressions `int(a / b)` et `(a // b)` sont-elles équivalentes pour toutes les valeurs entières de `a` et de `b` ? Le démontrer ou proposer un contre-exemple. .....

.....

.....

Exercice 6 Écrire des expressions booléennes traduisant les conditions suivantes. Les nombres mentionnés sont tous des entiers :

- L'entier n est divisible par 5. .....
- Les entiers m et n sont tels que l'un est multiple de l'autre. .....

  .....
- Les trois entiers m, n et p sont de même signe. .....

  .....
- n est le plus petit multiple de 7 supérieur à 10^{100}. .....

  .....
- Les trois entiers m, n et p sont distincts deux à deux. .....

  .....

Exercice 7 Écrire des expressions booléennes traduisant les conditions suivantes. Les nombres mentionnés sont tous des flottants :

- Le point de coordonnées (x, y) est à l'intérieur du cercle de centre (z, t) et de rayon r. .....
- Les points de coordonnées (x, y) et (z, t) sont situés sur une même droite parallèle à l'un des axes du repère. .....

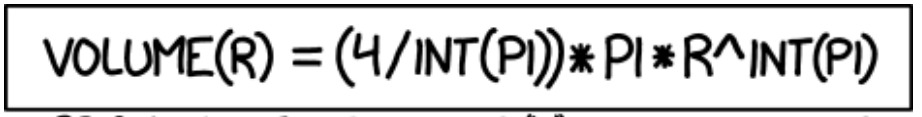
  .....
- Les points de coordonnées (x, y) et (z, t) sont les sommets opposés d'un carré dont les côtés sont parallèles aux axes du repère. .....

  .....
- Il existe un triangle dont les côtés mesurent respectivement a, b et c. .....

  .....

6 Bibliographie

Ce TP suit de très près la présentation de l'excellent manuel[5] *Informatique pour tous en classes prépartoires aux grandes écoles*, de Wack, Conchon, Courant, de Falco, Dowek, Filliâtre et Gonnord aux éditions Eyrolles.



https://xkcd.com/1275/

5. Téléchargeable sur le site du cours.