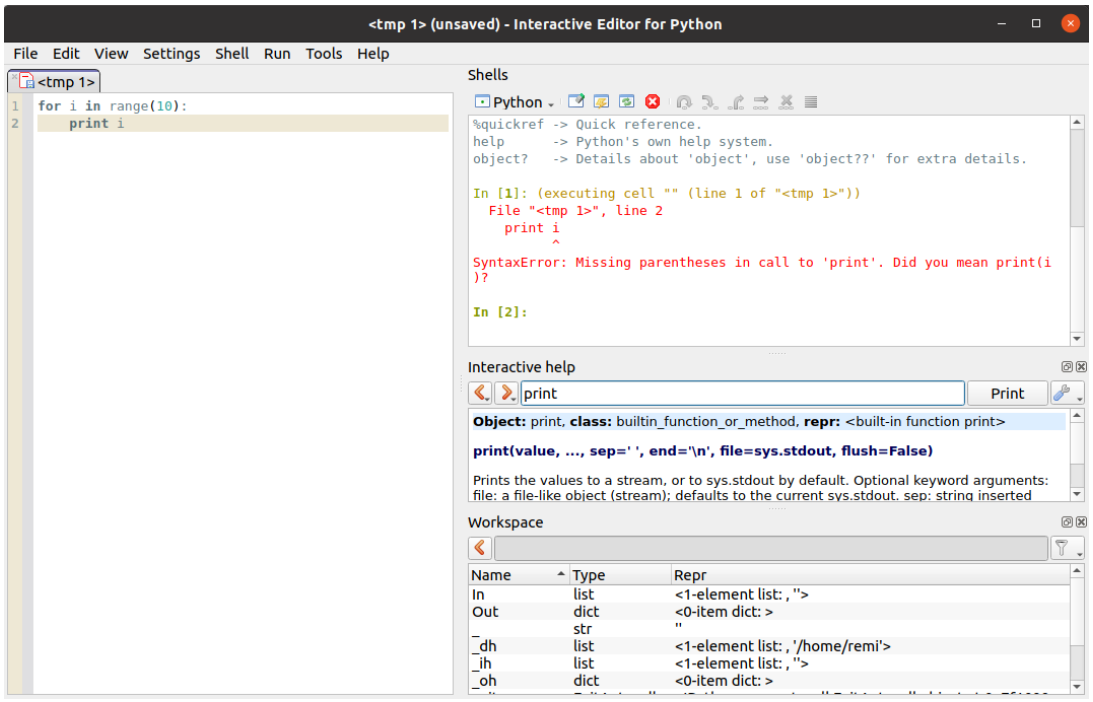
Chapitre 1.1 : Types, variables, imports

I Pyzo

Pyzo est un environnement de développement pour écrire et exécuter des scripts Python. Il se présente à peu près comme suit :



1 – Fenêtre Pyzo

Il y a deux parties principales :

- la partie en haut à droite est la console Python. On peut écrire du code Python dedans (après le `In [...]:`). Le code est interprété dès qu'il est complet. On peut aussi y lire les messages d'erreur (en rouge).
- la partie à gauche sert à rédiger des scripts Python plus complexes. Une fois le script écrit, on peut l'exécuter (en utilisant le menu `Run`, ou bien en pressant `F5`, ou encore `Ctrl + Entrée`).

Les deux autres fenêtres à droite sont moins importantes et peuvent être modifiées en utilisant le menu `Tools`.

Pour n'exécuter qu'une partie du script, on peut taper `##` avant et après la partie en question : cela délimite une cellule. En plaçant le curseur dans la cellule et en utilisant les touches `Ctrl + Entrée` on exécute seulement la cellule.

II Types, variables

En Python, les objets ont des types qui caractérisent les attributs de l'objet ainsi que les fonctions que l'on peut appliquer dessus. Il n'y a pas besoin de préciser un type (en général) :

l'interpréteur détermine lui-même à la volée le type lors de l'exécution. On dit que le typage est **dynamique**.

Les trois types simples sont

- `int` : les entiers qui représentent les éléments de $\mathbb{Z}$;
- `float` : les flottants qui représentent les éléments de $\mathbb{R}$;
- `bool` : les booléens.

Pour tester le type d'un objet, on dispose de la fonction `type()`.

II.1 Les nombres

On dispose sous Python des opérations usuelles (`+`, `-`, `*`, `/`), mais aussi de l'exponentiation `**`, de la division entière `//` et du modulo `%` qui renvoient respectivement le quotient et le reste de la division euclidienne de deux `int`.
Il faut faire attention aux types : les `int` sont automatiquement convertis en `float` si besoin. On peut toutefois forcer la conversion à l'aide des fonctions `float()` et `int()` (qui tronque).

Les règles de priorité sont les mêmes qu'en maths : puissances, modulos, multiplications et divisions, additions et soustractions. En cas d'ambiguïté, les opérations sont effectuées de gauche à droite (sauf pour les puissances). On peut bien sûr ajouter des parenthèses en cas de doute.

II.2 Les booléens

Les booléens n'ont que deux valeurs : `True` et `False`. On les utilise principalement pour les conditions dans les instructions conditionnelles.
On obtient notamment un booléen en résultat d'une comparaison entre deux nombres à l'aide d'un des opérateurs `<`, `>`, `<=`, `>=`, `==` et `!=`.
On dispose de trois opérations sur les booléens : `not`, `or` et `and`, cette dernière étant prioritaire.
L'évaluation des expressions booléennes est paresseuse : si `a` est faux, alors dans `a and b`, `b` ne sera pas évalué.

II.3 Variables

Pour affecter une valeur à une variable, on utilise `=` : le nom de la variable est à gauche et la valeur à droite. Lorsque la valeur est une expression, elle est d'abord évaluée et son résultat est affecté à la variable. On peut aussi affecter plusieurs variables d'un coup en utilisant `,`.
Pour vérifier la valeur stockée dans une variable, il suffit de taper son nom dans la console puis **Entrée**. On peut aussi utiliser la fonction `print()` qui permet d'afficher.

III Importer un module

Certaines opérations mathématiques utiles (par exemple les fonctions trigonométriques, l'exponentielle, etc...) ne sont pas d'office utilisable sous Python : elles sont codées dans le **module** `math` et doivent être **importées**. Par exemple, pour la fonction `cos`, on peut :

- importer le module `math` avec `import math` : on peut alors utiliser la fonction avec `math.cos` ;
- importer le module `math` avec un alias avec `import math as m` : on peut alors utiliser la fonction avec `m.cos` ;
- importer la fonction avec `from math import cos` : on peut alors utiliser la fonction avec `cos` ;

- importer toutes les fonctions avec `from math import *` : on peut alors utiliser la fonction avec `cos` et toutes les autres fonctions aussi ;

IV Exercices

Exercice 1 Préciser le type et la valeur stockée dans chaque variable à la fin de l'exécution des séquences d'instructions :

1. `x = 40 + 2`
2. `a = 21 * 2.0`
3. `y = 1+1 == 2`
4. `c = int(-3.14)`
5. `x = 2**3.0 + 4`
6. `b = 2>=3 or 2**4*5**2//20==20`
7. `d = True or 4>3 and 3>4`
8. `e = 8.6 + 2`
9. `e = int(8.6)+2`
10. `f = float(2)**3`
11. `b = 2**3 != 9`
12. `x = 3%2 != 1 and 10/0 == 1`
13. `x = 10/0 == 1 and 3%2 != 1`
14. .
```
x = 2
y = 3
a = x*y
x = y
y = a
s = a + x + y
```
15. .
```
x, y = 2, 3
x = y
y = x
```
16. .
```
x = 2
y = 3
x, y = y, x
```

■