Python 3

Variables et types (en construction)

– Une variable est un espace de stockage en mémoire, identifié par un nom, qui contient une valeur. En Python, le typage est dynamique : le type d'une variable est déterminé par la valeur qu'elle contient et peut changer au cours de l'exécution. L'opérateur d'affectation est `=`.

– Il existe plusieurs types de données simples, dits primitifs, qui constituent les briques de base :
  ◦ Le type `int` (integer) représente les nombres entiers, positifs ou négatifs.
  ◦ Le type `float` (floating-point) désigne les nombres à virgule flottante.
  ◦ Le type `bool` (boolean) ne peut prendre que deux valeurs : `True` ou `False`.
  ◦ Le type `str` (string) correspond aux chaînes de caractères, délimitées par des apostrophes ou des guillemets.

  La fonction `type(variable)` permet de connaître le type d'une variable.

– La conversion de type, ou *cast*, consiste à transformer une donnée d'un type vers un autre, lorsque cela est possible. On utilise pour ce faire des fonctions éponymes :
  ◦ `int(x)` convertit `x` en entier. La conversion d'un `float` tronque la partie décimale.
  ◦ `float(x)` convertit `x` en nombre à virgule flottante.
  ◦ `str(x)` convertit `x` en chaîne de caractères.
  ◦ `bool(x)` convertit `x` en booléen. La valeur est `False` pour `0`, `0.0`, une chaîne vide `""` ou un objet vide ; `True` pour toute autre valeur.

– Les types composés, ou structures de données, permettent de regrouper plusieurs valeurs :
  ◦ La liste (`list`) est une collection ordonnée et modifiable d'éléments. Syntaxe : `[elt1, elt2]`.
  ◦ Le n-uplet (`tuple`) est une collection ordonnée mais non modifiable (immuable). Syntaxe : `(elt1, elt2)`.
  ◦ Le dictionnaire (`dict`) est une collection non ordonnée de paires clé-valeur. Syntaxe : `'cle1': valeur1`.
  ◦ L'ensemble (`set`) est une collection non ordonnée d'éléments uniques. Syntaxe : `elt1, elt2`.

Les fonctions (en construction)

– L'objectif principal des fonctions est la modularité des programmes. Elles permettent de décomposer un problème complexe en sous-tâches plus simples, d'éviter la répétition de code et d'améliorer la lisibilité ainsi que la maintenance du code.

– En Python, une fonction se définit avec le mot-clé `def`, suivi du nom de la fonction, de ses paramètres entre parenthèses, et de deux-points. Le corps de la fonction est indenté. L'instruction `return` permet de renvoyer une ou plusieurs valeurs.

```python
def additionner(a, b): # Fonction qui retourne la somme de deux nombres
    # a et b sont les parametres
    resultat = a + b
    return resultat
```

Algorithmique (en construction)

- L'instruction conditionnelle permet d'exécuter des blocs de code différents en fonction de la validité d'une condition. Elle se structure avec les mots-clés `if`, `elif` (contraction de *else if*) pour les conditions alternatives, et `else` pour le cas par défaut.
- La boucle définie, ou boucle `for`, est utilisée pour itérer sur les éléments d'une séquence (liste, chaîne de caractères, etc.) ou d'un itérable. Le nombre d'itérations est connu à l'avance. On l'associe souvent à la fonction `range(n)` qui génère une séquence d'entiers de 0 à n-1.
- La boucle indéfinie, ou boucle conditionnelle `while`, répète un bloc d'instructions tant qu'une condition spécifiée reste vraie. Le nombre d'itérations n'est pas nécessairement connu à l'avance. Il est fondamental de s'assurer que la condition de la boucle deviendra fausse à un moment donné, afin d'éviter une boucle infinie.