Université de Rennes

L1 BECV
année 2020/2021

Mémo - Python

Emeline LUIRARD

- Pourquoi Python ? C'est un langage lisible avec une syntaxe orientée présentation. Il est proche de la langue parlée, c'est ce qu'on appelle un langage de haut niveau.
  N.B. *la philosophie de ce langage est résumé dans le* Zen de Python. *Vous pouvez y accéder en important le module* `this`.

💡 Votre devise doit être : **coder compact et simple !** C'est comme cela que vos codes seront les plus lisibles et que vous ferez le moins d'erreurs.

TP 1

⇝ Comment ça marche ?

Thonny, votre environnement de développement Python pour cette année, est constitué

- d'un interpréteur(ou console) : c'est votre boîte de dialogue avec Python. >>> indique que le logiciel attend vos instructions. L'interpréteur peut être utilisé comme une calculatrice : vous entrez un calcul, vous tapez sur la touche "entrée" et Python vous donne le résultat. C'est pratique pour des instructions simples mais dès qu'on veut écrire du code plus long, pouvoir commenter et partager son code, il vaut mieux utiliser l'environnement de développement.
- d'une fenêtre d'éditeur de texte : où l'on peut regrouper les commandes et les sauvegarder dans un `fichier.py`.

💡 On peut naviguer dans l'historique des commandes de l'interpréteur au moyen des flèches haut et bas, cela évite de retaper les commandes si l'on exécute une fonction plusieurs fois de suite par exemple.

Dès que l'on souhaite exécuter plus de deux ou trois tâches consécutives, i.e. dès que l'on sort du mode d'utilisation "supercalculatrice" de Python, on utilise l'éditeur de texte. On enregistre le fichier obtenu avec le suffixe `.py` puis on exécute le fichier.

Pour faciliter la lecture de vos codes pour plus tard, il est possible de les commenter. Une ligne de commentaires commence par `#`. De manière générale, Python ignore tout ce qui, sur une ligne, suit la commande `#`.

1 Objets et types

En Python, tout est un objet. Pour manipuler ces objets, on leur donne un nom. On dit qu'on affecte une variable à une valeur. Par exemple,

```
y=42
mon_prenom="emeline"
```

N.B. *En général, on n'utilise que des minuscules et des "_" pour les noms de variables. Et on essaie de les choisir pour qu'ils aient du sens.*

Référencer une variable permet d'appeler plus facilement un objet et de garder sa valeur en mémoire. Il est possible d'afficher la valeur d'une variable avec `print(votre code)`.

A chaque type d'objets est associé un ensemble de mécanismes. C'est ce qu'on appelle des *méthodes*. Voici une liste non-exhautive de différents types :

- les entiers *int*, ils ont une précision illimité. Par exemple, `5, 17, 42`.
- Les flottants *float*, leur précision est limitée à 15 chiffres significatifs en général. Cela peut amener à des problèmes.
   Pour les nombres décimaux, on utilise un point et non une virgule. Par exemple, `1.2`
- Les chaînes de caractères *str*. On les note entre guillemets.
  Par exemple, `"Bonjour, il fait beau aujourd'hui"`. Le guillemet indique à Python d'y lire comme du texte et non du code. Ainsi `"x"` sera lu comme du texte et non comme la variable x.

1.1 Opérations sur les nombres

Les quatres opérations de base sont `+,*,-,/`. Pour mettre à une puissance, on utilise `**`.

`x**a`	x^a
`a\%b`	reste de la division euclidienne de a par b
`math.sqrt(x)`	$\sqrt{x}$ en pensant à importer le module `math`

1.2 Opérations sur les chaînes de caractères

`"Bonjour"+"Pierre"`	concatène les deux chaînes de caractères
`"A"*20`	concatène 20 fois la chaîne de caractères "A"
`len(chaine)`	donne la longueur d'une chaîne de caractères i.e. le nombre de caractères.

1.3 Opérateurs logiques et booléens

Les opérations logiques de base sont le "et" logique (`and`), le "ou" logique (`or`) et la négation (`not`). Les valeurs logiques (ou booléens) sont `True` et `False`. Voici divers opérateurs de comparaison :

`x==y`	x est égal à y
`x!=y`	x est différent de y
`x<=y`	$x \leq y$

Pour demander des conditions plus complexes, on peut combiner ces opérateurs de comparaison avec les booléens, par exemple, `(x==y) or (x>5)`.

2 Quelques commandes usuelles

`x+=1`	remplace x par $x+1$ (valable aussi pour -,*,/)
`x,y=17,42`	pour affecter deux variables en même temps, toutes les expressions sont évaluées *avant* la première affectation, ex : `x,y=x+2,x`
`x,y=y,x`	du même type que le précédent, pour échanger deux variables
`input("Donner une valeur")`	Demande à l'utilisateur une valeur et renvoie une chaîne de caractères.
`int(x), float(x), str(x)`	change le type de x respectivement en entier, flottant, chaîne de caractères.

3 Bloc d'instructions if/elif/else

Un bloc d'instruction est un ensemble d'instructions indentées du même nombre de caractères. On met **toujours** un " : " avant un bloc d'instructions. Voici un exemple :

```
if nb<30:
        x=0.10*nb
elif nb<50:
        x=0.05*nb
else:
        x=0.01*nb
```

Les conditions sont testées dans l'ordre jusqu'à ce que l'une d'elles soient vérifiées (les suivantes ne sont donc pas testées). Donc attention à bien choisir l'ordre des tests ! Seul le bloc d'instructions ayant la condition réalisée sera lu et exécuté.

TP 2

A venir...

Références

[CBCC16] Alexandre Casamayou-Boucau, Pascal Chauvin, and Guillaume Connan. *Programmation en Python pour les mathématiques - 2e éd.* Dunod, Paris, 2e édition edition, January 2016.

[PL] Thierry Parmentelat and Arnaud Legout. Python 3 : des fondamentaux aux concepts avancés du langage FUN-MOOC.

[Vig18] Vincent Vigon. *python proba stat.* Independently published, October 2018.