Rappel de syntaxe en python

Caroline Koudoro-Parfait

Crédits : Hugo Chevroton

Doctorant ObTIC, STIH, SCAI

caroline.parfait@sorbonne-universite.fr

28 février 2022

Variable et opérateur

age_de_Paul = 20

Cette ligne *déclare* la *variable* age_de_Paul et lui *affecte* la valeur 20. Elle lui assigne ainsi le *type* entier.

Les variables peuvent être manipulées à l'aide des opérateurs suivants :

- + : addition
- - : soustraction
- / : division
- // : division entière (renvoie le quotient)
- % : division (renvoie le reste)

Variable et opérateur

Attention, l'effet des opérateurs varie en fonction du *type de variables* auxquels il s'applique. Par exemple :

variable = *5* + *3* : variable vaut 8 et est un entier.

variable = "*Salut*" + " *!*" : variable vaut "Salut" et est une chaîne de caractère.

variable = "*5*" + "*3*" : variable vaut "53" et est une chaîne de caractère.

variable = "*Salut*" + *3* : cette ligne ne fonctionne pas car additionner une chaîne de caractère n'a pas de sens.

Si ... alors ... sinon (if ... then ... else)

La structure *Si ... alors ... sinon*
permet d'effectuer des actions qu'à certaines conditions.

Jean veut passer son permis.

Si Jean a 18 ans ou plus, alors il peut passer son permis
sinon il doit apprendre.

Si Jean a plus de 18 ans alors :
    il peut passer son permis
sinon :
    il doit attendre

Syntaxe Python

```
if age_de_jean => 18:
    il peut passer son permis
else :
    il doit attendre
```

```
if la condition est vrai :
    action 1
else :
    action 2
```

Construction des conditions : Et, ou, non

Condition :
Jean n'a pas moins de 18 ou (il a 17 ans et il est émancipé).

Syntaxe :
(not age_de_jean < 18) or (age_de_jean == 17 and jean_est_émancipé)

Boucle pour tout (for)

La *Boucle pour tout* permet d'exécuter une action d'après un ensemble d'élément.

Toutes les heures entre midi et 15h, je vérifie mes mails.

Pour toutes les heures dans l'ensemble 12h, 13h, 14h :
je vérifie mes mails

Syntaxe

```
for heure in [12,13,14]:
    je vérifie mes mails

for heure in range(12,15) :
    je vérifie mes mails


somme = 0
for chiffre in [1, 2, 3]:
    somme += chiffre
```

Après exécution, somme vaut 6

Boucle tant que ... (while)

La *Boucle tant que* permet d'exécuter une action jusqu'à ce qu'une condition soit remplie.

Tant que la maison brûle, les pompiers lancent de l'eau.
Le feu est alors éteint.

Syntaxe :

```
while la maison brûle :
    les pompiers lancent de l'eau
Le feu est éteint.
```

Les fonctions

Les fonctions permettent de diviser et d'organiser le code. Exemple de la définition de la fonction *addition*, qui prend deux nombres en paramètre et renvoie leur somme.

Syntaxe :

```
def fonction_addition ( valeur1, valeur2) :
    return valeur1 + valeur2
```

Vocabulaire :

```
def nom_de_la_fonction ( argument(s) ) :
    return élément retourner par la fonction
```

Petit point sur les listes

Une liste est une structure de données permettant d'accéder à un objet par son index.

Pour illustrer, supposons qu'une liste représente un rue.

Chaque habitant vit dans une maison avec un numéro.

Ce numéro est l'index de la liste grâce auquel on peut accéder à l'habitant.

Syntaxe :

liste =[] :définition d'une liste vide
liste =["Annie", "Paul"] :définition avec quelques éléments
liste[0] vaut "Annie", liste[1] vaut "Paul".

Petit point sur les dictionnaires

Un dictionnaire est une structure de donnée permettant d'accéder à un objet par une clef.

Dans un dictionnaire de langue, on utilise les *mots* comme clef afin d'accéder à leur *définition*, qui sont ici les objets.

On remarquera que dans un dictionnaire, chaque mots (chaque clef) est unique.
Par contre, une même définition Peut correspondre à plusieurs mot.

Syntaxe :

```
dico_age ={"Annie" :20, "Paul" :18}
```

dico_age["Annie"] vaut 20.

Manipulation de fichiers

3 modes d'ouverture d'un fichier :

En lecture : suffixe , "r" pour "read"

En écriture : suffixe , "w" pour "writting"

En écriture en fin de fichier : suffixe , "a" pour "append"

Fonction d'ouverture d'un fichier

```
fichier = open( chemin_de_votre_fichier , suffixe)
```

Fermeture d'un fichier

```
fichier.close()
```

Manipulation d’un fichier en lecture

Fonction d’ouverture d’un fichier en lecture

```
fichier = open( chemin_de_votre_fichier , "r")
```

Lecture d’une ligne

```
ligne = fichier.readline()
```

Manipulation d'un fichier en écriture

Fonction d'ouverture d'un fichier en écriture

fichier = open(chemin_de_votre_fichier , "w")
(écriture, remplace le texte déjà présent)

fichier = open(chemin_de_votre_fichier , "a")(écriture en fin)

Écriture dans un fichier :

fichier.write("Bonjour monde")

Caractère spéciaux :

- **"\n"** : permet d'aller à la ligne en écriture
- **"\t"** : permet d'écrire une tabulation

Manipulation de chaînes de caractères

Joignez tous les éléments d'une liste dans une chaîne, en utilisant l’espace comme séparateur avec la fonction .join() :

```
liste=["Bonjour","je","suis","heureux.se"]
string= " ".join(liste)
print(string)
Bonjour je suis heureux.se
```

Séparation d’une chaîne de caractère en plusieurs mots (liste) en s’appuyant sur les espaces avec la fonction .split() - (ligne et mot sont de type : *chaîne de caractère* ou *string*)

```
liste_de_mots = string.split(‘ ’)
print(liste_de_mots)
['Bonjour', 'je', 'suis', 'heureux.se']
```