Python

Source : openclassrooms

Plan du cours

1 Premiers pas avec Python

Crée en 1990, le langage Python tire son nom du fait que le créateur de ce langage est un fan des comiques anglais les « Monthy PYTHON ». Pourquoi apprendre Python ? D'abord parce que c'est un langage parfait pour débuter : il fonctionne sur tous les systèmes d'exploitation et vous n'avez pas à utiliser un logiciel spécifique pour voir le résultat de votre code. Par ailleurs, Python est le 4^{e} langage le plus populaire selon l'index TIOBE et son usage est resté stable depuis une dizaine d'années. Pinterest, Instagram et le site du New

York Times ont été développés en Python. Python est aussi un des langages principaux utilisés en *data analysis* (analyse de données) et en *machine learning* (apprentissage par la machine).

Le projet fil-rouge de ce cours est un programme qui ira chercher des citations de San Antonio sur Internet et les fera dire par un personnage de dessin animé au hasard. Voici les étapes que nous allons suivre :

— Installer Python et faire connaissance avec la console
— Trouver comment "enregistrer" une citation et la retrouver plus tard.
— Créer des phrases sous la forme '<personnage> a dit : "<citation>"' et la modifier automatiquement.
— Créer des listes pour stocker plusieurs citations et plusieurs personnages.
— Créer des "dictionnaires" pour attribuer plusieurs citations à un même personnage.
— Enregistrer votre programme dans un fichier externe car il commence à faire plusieurs lignes !
— Interagir avec notre utilisateur : quand il tape "entrée", le programme doit afficher une nouvelle citation. Quand il tape "B", le programme s'arrête.
— Afficher une citation au hasard quand on lance le programme.
— BONUS : Stocker nos citations et nos personnages dans un fichier externe.
— BONUS : Coder un petit robot qui va parcourir le Web à la recherche de citations et de personnages puis les stocker dans un fichier sur votre ordinateur.

1.1 Installer Python

Rendez-vous sur le site officiel de Python (https://www.python.org/) et cliquez sur Downloads. C'est avec la version 3 que nous travaillerons. Sous Windows, enregistrez le dossier à télécharger puis suivez les instructions. Quand l'installation est terminée, vous pouvez vous rendre dans Démarrer > Tous les Programmes. Vous devriez y voir un nouveau dossier Python contenant, notamment, Python et IDLE.

Sous Mac, décompressez le dossier que vous avez téléchargé en double-cliquant dessus, puis cliquez de nouveau sur le document afin de lancer l'installation et laissez-vous guider.

Sous Linux, Python est pré-installé dans la plupart des distributions Linux mais sa version est certainement obsolète.

La console

Petites révisions du cours de première année : la console (ou terminal) est un petit programme qui vous permet d'interagir directement avec votre ordinateur en parlant son lan-

gage. Lorsque vous utilisez un logiciel, chaque action que vous effectuez avec votre souris est traduite en langage informatique puis exécutée par votre ordinateur. La console vous permet d'aller plus vite puisque vous n'utilisez plus d'interface graphique pour traduire vos commandes. Cette console vous permet de vous déplacer dans l'ordinateur, d'en manipuler les fichiers et même de créer de petits programmes qui automatiseront certaines tâches. Lorsque vous interagissez avec la console, vous parlez en langage bash (sur Linux et Mac) ou en DOS (Windows).

Quelques rudiments pour survivre dans le monde déroutant de la console. Et d'abord pourquoi la console ? Au départ, parce qu'on n'avait de toute façon pas le choix ! Les débuts de l'informatique et de la console remontent aux débuts des années 70, à une époque où un écran 2 couleurs était un luxe inimaginable et où la puissance de calcul de ces ordinateurs était cent fois plus faible que celle de n'importe quelle calculatrice d'aujourd'hui. Notez que :

— les commandes sont courtes, abrégées : écrire pwd est moins intuitif que diredans-quelrepertoirejesuis mais ô combien plus rapide !

— les commandes sont intuitives : il s'agit bien souvent d'une abréviation de termes (en anglais, évidemment).

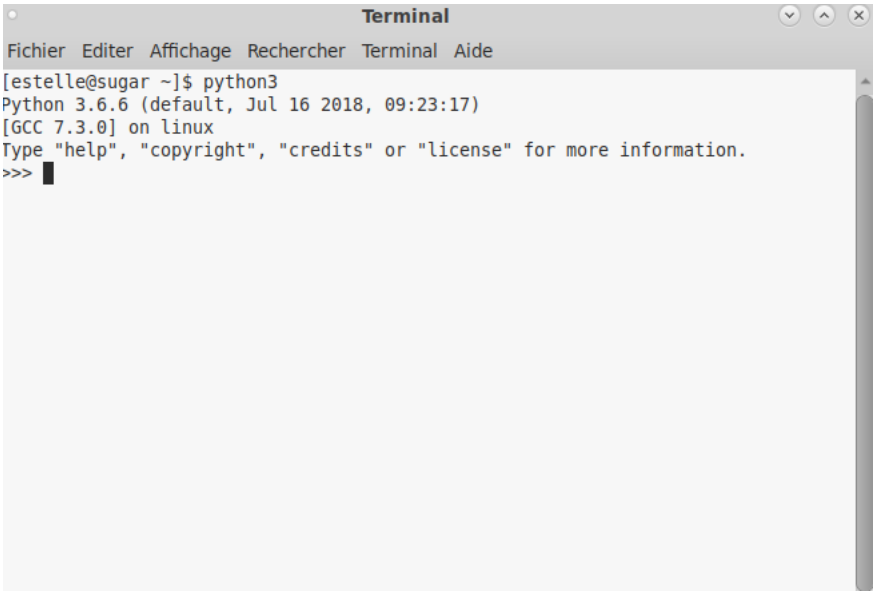
Quand on sait se servir de la console, on va beaucoup plus vite qu'avec l'interface graphique. Vous vous rendrez compte à un moment qu'il y a des choses que seule la console peut faire et qu'il serait de toute façon vraiment inutile de recourir à une interface graphique pour les effectuer. Exemple : en mode graphique, allez dans un répertoire qui contient beaucoup de fichiers en tout genre : des fichiers texte, des images, des vidéos... Vous voudriez savoir combien il y a d'images JPEG dans ce dossier. À la console, en assemblant quelques commandes, on peut obtenir ce résultat sans problème : ls -l | grep jpg | wc -l

L'interpréteur Python

L'interpréteur Python est un programme qui va comprendre les instructions écrites en Python et les "transformer" en langage binaire. Vous n'avez pas envie d'écrire une suite de 0 et de 1 pour que votre programme fonctionne, n'est-ce pas ? Pour lancer l'interpréteur, il suffit d'entrer la commande python3.

Sous Windows, lorsque vous installez Python, vous installez également un autre programme qui s'appelle IDLE. Il s'agit d'un logiciel qui vous permettra d'écrire du code Python et de l'exécuter. Cliquez sur le menu Démarrer, puis sur Python et Idle. Une fenêtre s'ouvre et le programme vous invite à écrire des commandes.

Vous pouvez maintenant coder en Python : pour commencer, tapez "7 + 7". L'intepréteur vous répond : 14. Poursuivons : voyons ensemble comment enregistrer dans la console



une première citation.

1.2 Le vocabulaire de Python

Nous venons de voir qu'on pouvait se servir de l'interpréteur comme calculatrice. Mais il est également possible de lui demander de conserver des informations en mémoire. Chaque information que vous souhaitez réutiliser plus tard s'appelle une variable. Il s'agit d'un concept basique en programmation et que vous retrouverez dans tous les langages.

NB : une variable garde en mémoire une information tant que l'interpréteur Python est ouvert. Si vous quittez le programme puis le redémarrez (en tapant exit() puis python3 par exemple), toutes vos variables seront effacées. Python a la mémoire courte !

Pour définir une variable, il faut taper son nom, un signe égal puis sa valeur entre guillemets (si la valeur est une chaîne de caractères). Définissons une variable correspondant à une citation de San Antonio (notre hypothèse de départ), sachant que le nom de la variable est quote : `quote="Ecoutez-moi, Monsieur Shakespeare: Nous avons beau être ou ne pas être, nous sommes!"`

Retenez que seules les chaînes de caractère sont entourées de guillemets (c'est le cas des citations). D'autres types d'informations (appelées objets en Python) vont être entourés de parenthèses ou de crochets. Pourquoi ne pas avoir la même syntaxe ? Parce que la nature des objets diffère. Exemple : si vous tapez $4 + 4$, vous vous attendez à ce que l'interpréteur vous retourne 8. Et si vous voulez ajouter deux mots les uns à la suite des autres, vous utiliserez certainement le même opérateur, le $+$: "tire-" $+$ "bouchon" et la console vous

retournera... ”tire-bouchon”. Vous ne voulez pas les additionner, comme avec un nombre, mais bien les concaténer.

Quels sont les types d'objets qu'on peut créer avec Python ?

— **des nombres** : il existe deux grands types de nombres en Python : les entiers (*integers*) et les décimaux (*floats*). Tous deux n'ont besoin ni de guillemets, de parenthèses ou de crochets. Notez que les décimales se situent après un point. Ex. : my_float = 1.3

— **des chaînes de caractères** : comme on l'a vu, il s'agit d'un ensemble de signes (lettres, chiffres, ponctuation, caractères spéciaux, ...) délimités par des guillemets ouvrants et des guillemets fermants. NB : vous pouvez également insérer des guillemets à l'intérieur des guillemets à condition de les échapper (en utilisant le \) ou de ne pas utiliser les mêmes. Exs. : quote = ”San Antonio a dit : 'Ecoutez-moi, Monsieur Shakespeare'” ou quote = ”San Antonio a dit : \”Ecoutez-moi, Monsieur Shakespeare\””.

— **des booléens** : un booléen est une information Vraie ou Fausse. Ce type nous sera très utile un peu plus tard lorsque nous commencerons à comparer des valeurs entre elles. Sans surprise, vrai est *True* et faux est *False* (pensez aux majuscules).

— **des listes** : on les reconnaît aisément car elles sont entourées de crochets et comprennent des valeurs séparées par des virgules. Ex. : characters = [”Babar”, ”Betty Boop”, ”Calimero”]. Pour accéder à un élément stocké, il faut écrire le nom de la liste puis sa position entre crochets. Par exemple, si nous souhaitons accéder au premier élément de la liste characters : characters[0]. NB : la position du premier élément d'une liste est le 0.

— **des tuples** : ils servent à créer des structures dont le nombre d'éléments ne bouge pas. On dit qu'ils sont immuables car leur structure ne peut pas changer. On ne peut ajouter ou supprimer un élément. Ils commencent par une parenthèse ouvrante, un ensemble d'objets séparés par des virgules et une parenthèse fermante. Ex. : représenter les coordonnées géographiques d'une ville : paris = (48.856578, 2.351828).

— **des dictionnaires** : un dictionnaire est un ensemble de valeurs auxquelles vous pouvez accéder grâce d'autres objets (des chaînes de caractères par exemple). Ex. :

```
>>> english_french_dict = {
... "un": "one",
... "deux": "two",
... "trois": "three"
... }
```

Nous pouvons utiliser un dictionnaire pour stocker toutes nos citations et tous nos

personnages, par exemple. Pour définir un dictionnaire, il faut commencer par écrire une accolade ouvrante, puis indiquer la première clé, deux points et la valeur associée, et enfin ajouter une accolade fermante. Ex. : my_dictionary = {'key' : 'value'}. Accéder à une valeur se fait de la même manière que pour une liste mais au lieu d'indiquer sa position (0, 1, 2, ...) vous utiliserez sa clé. Ex. : my_dictionary['key'].

Exercice Créez différentes variables pour stocker les éléments suivants :

- group : le nom de votre groupe de musique préféré
- year : l'année de formation du groupe
- still_together : sont-ils encore ensemble
- albums : la liste de leurs albums
- foundin_gmembers : un tuple qui stocke le nom des membres fondateurs
- songs : un dictionnaire qui contient les chansons de leurs albums

```
1 group="les Beatles"
2 year=1960
3 still_together=False
4 albums=["Please Please Me", "Yellow Submarine", "Let It Be"]
5 founding_members=("John Lennon", "Paul McCartney", "George Harrison",
  "Ringo Starr")
6 songs={"Yellow Submarine":["Yellow Submarine", "Only a Northern Song", "All
  Together Now", "Hey Bulldog", "It's All Too Much","All You Need Is Love"]}
7
```

Questions

Question 1

Pourquoi l'inventeur de Python a-t-il choisi ce nom ?

Il aimait beaucoup les serpents.

Il aimait beaucoup le groupe d'humoristes Monty Python.

Question 2

Python est très utilisé dans les domaines suivants :

Big data

Application mobile pour iPhone (iOS)

Web

Robotique

Question 3

Pour installer Python, vous pouvez :

Télécharger un dossier sur www.python.org puis suivre les étapes.

Il n'y a rien à faire, la dernière version de Python est installée par défaut sur tous les ordinateurs.

Ecrire une ligne de commande dans la console (Mac ou Linux).

Question 4

Python 3 est rétrocompatible, c'est à dire que vous pouvez travailler sur des fichiers utilisant la version 2 de Python si vous avez installé la version 3.

Vrai

Faux

Question 5

La console (pas l'interpréteur) sert à...

Naviguer dans un ordinateur en ligne de commande.

Lancer des programmes en ligne de commande.

Coder en Python (vous pouvez écrire $7 + 7$ et la console affichera 14)

Question 6

L'interpréteur Python (pas la console) sert à...

Ecrire du code Python.

Se déplacer dans l'ordinateur en langage Bash ou DOS (je peux écrire ls).

Question 7

Une chaine de caractère est :

Un ensemble de chiffres et de lettres encadrés par des guillemets simples ou doubles. Exemple : "Ceci est une chaîne"

Un ensemble de caractères spéciaux sans guillemets. Exemple : les crochets, les parenthèses, etc.

Un ensemble de chiffres et de lettres à la suite, sans majuscule, espace ou caractère spécial. Exemple : ceci_est_une_chaîne

Question 8

Une variable est :

Un ensemble de chiffres et de lettres encadrés par des guillemets simples ou doubles. Exemple : "Ceci est une variable"

Un ensemble de chiffres et de lettres à la suite, sans majuscule, espace ou caractère spécial, qui sert à sauvegarder une valeur pour la réutiliser ultérieurement. Exemple : ma_variable = "Une valeur"

Question 9

Une liste organise un ensemble de valeurs. On peut retrouver ces valeurs grâce à leur position (leur index) dans la liste. Exemple : ma_liste = [”chien”, ”chat”]

Vrai

Faux

Question 10

Un dictionnaire organise un ensemble de valeurs. On peut les retrouver grâce à leur clé. Exemple : mondict = ”one” : ”un”, ”two” : ”deux”

Vrai

Faux

2 Poser les fondations d'un programme

2.1 Créer son premier script

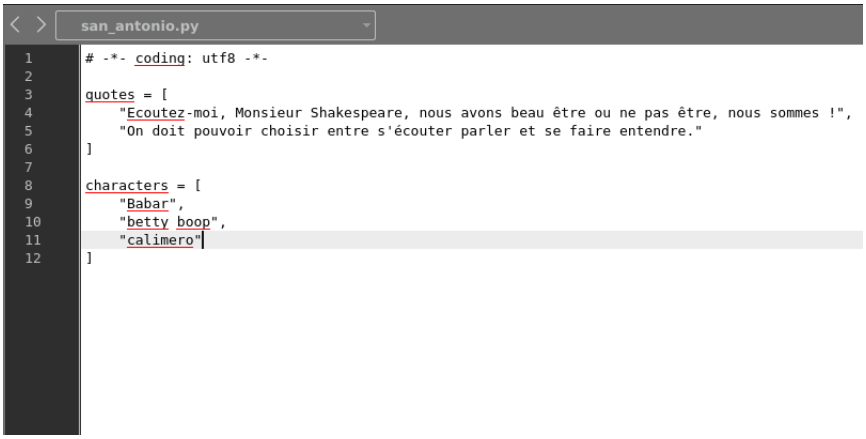
Force est de constater qu'il devient de plus en plus compliqué de travailler avec l'interpréteur. Quand vous le quittez, tout s'efface et il faut donc se souvenir du code créé. De plus, vous ne pouvez pas modifier facilement une variable. Nous allons donc créer un fichier externe qui conservera nos commandes, puis lancer ce fichier avec python. Au lieu d'entrer, à la main, chaque commande, il suffira donc de lancer le programme.

Travailler avec un fichier externe se fait en deux étapes : il faut d'abord créer le document puis l'exécuter.

1. Avant de créer le fichier, pensez à créer un dossier (par exemple ”cours”) dans lequel vous allez l'enregistrer. Pour créer le fichier, ouvrez un éditeur de textes : vous nommerez le fichier san_antonio.py. Pour qu'il ne soit pas vide, écrivez : print(”Hello world !”) et sauvegardez.
2. Pour exécuter le fichier, au terminal allez dans votre dossier ”cours” et lancez python : python3 san_antonio.py [1].

Nous pouvons maintenant sauvegarder dans notre fichier le code dont nous avons besoin pour notre programme. Notez que nous avons besoin d'indiquer dans quel encodage nous souhaitons travailler.

1. Sous Windows, utilisez IDLE. Dans le menu, cliquez sur File > New File puis sauvegardez. Pour lancer votre programme, cliquez sur Run > Run Module. IDLE va exécuter votre programme dans l'interpréteur Python qui était déjà ouvert.



2.2 Comparer les valeurs avec les opérateurs

Intéressons-nous maintenant à l'interaction avec l'utilisateur. La première phrase qui s'affichera sera une citation au hasard. Puis, nous proposerons deux alternatives : si l'utilisateur tape "B", le programme se ferme, sinon une nouvelle citation apparaît. Nous allons commencer par écrire du pseudo-code, c'est-à-dire écrire ce que nous voulons que le programme fasse avec nos propres mots. Il s'agit d'une pratique très courante chez les développeurs. Et c'est l'anglais qui est d'usage (possibilité de partager avec des non-francophones, pas de caractères accentués).

```
# Show random quote

# If user answer is 'B':
# - leave the program

# Else :
# - show another quote
```

On distingue les opérateurs de comparaison des opérateurs mathématiques :

Les opérateurs de comparaison Commençons par découvrir les opérateurs qui nous permettront de comparer user_answer et B. La logique voudrait qu'on utilise le signe = pour comparer deux valeurs. Mais, comme on l'a vu, ce signe est déjà utilisé pour assigner une valeur à une variable. C'est pourquoi nous doublons le signe égal par un autre égal pour signifier la comparaison, comme ceci : == . Les opérateurs de comparaison renvoient un booléen (True ou False) car vous posez une question fermée : c'est vrai ou ça ne l'est pas.

Voici la syntaxe :
Égal : 1 == 1
True
Différent : 1 !=2
True
Supérieur : 1 > 2 ou Inférieur 2 < 1
False
Supérieur ou égal : 1 >= 2 ou Inférieur ou égal : 2 <= 1
False

Les opérateurs mathématiques Ce sont les signes + - * / auxquels s'ajoute le modulo noté % : il s'agit du reste quand une division ne tombe pas juste. Ex. : 14 % 5 –> le modulo est 4.

2.3 Ajouter un peu de logique avec les conditions

Nous savons comment comparer deux valeurs, mais nous ne savons pas encore comment écrire des conditions. Voici la syntaxe :

```
1  if user_answer == "B":
2      # leave the program
```

La condition if `Commentaire:` vous commencez par écrire if, puis vous indiquez la condition à remplir et terminez la ligne par deux points (sans espace avant). Vous indiquez les actions à effectuer juste en-dessous. Afin de différencier ces actions, qui sont à l'intérieur d'une condition, du reste de votre programme, vous ajoutez quatre espaces au-début de la ligne. Nous appelons cela l'indentation et Python est très strict sur le sujet.

else Comment définir ce que se passe si la condition n'est pas remplie ? En utilisant else :
if user_answer == "B" :
    # leave the program
else :
    # show another quote

elif Vous pouvez également proposer d'autres choix grâce au mot-clé elif (contraction de else et if) :

```
if user_answer == "B" :
    # leave the program
elif user_answer == "C" :
    print("Essaie encore !")
else :
    # show another quote
```

pass Pour quitter le programme, on utilise le mot-clé : pass.

2.4 Structurer son programme grâce aux fonctions

Revenons à la première phrase de notre programme : # Show random quote. Il nous faut donc une liste composée d'au moins deux citations :

quotes = ["Ecoutez-moi, Monsieur Shakespeare, nous avons beau être ou ne pas être, nous sommes !", "On doit pouvoir choisir entre s'écouter parler et se faire entendre."]

Afin d'afficher dans le terminal une citation au hasard, nous devons avant choisir une position aléatoirement dans la liste. Il nous faut donc commencer par chercher comment Python peut nous renvoyer un nombre compris entre une valeur minimale (0, le début de la liste) et une valeur maximale (le nombre total d'items - 1). Je vous propose de diviser cette étape en plusieurs sous-étapes :

```
# Show random quote :
    # get a random number
    # get a quote from a list
    # show the quote in the interpreter
```

Fonctions Pour pouvoir stocker cet ensemble d'actions, nous allons utiliser une fonction. Une fonction est un ensemble de commandes regroupées sous un nom unique. Pour définir une fonction, vous utilisez le mot-clé def, suivi d'un nom, de parenthèses et de deux points. Puis vous indentez tout ce qui se trouve à l'intérieur. Exemple :

```
def get_random_quote() :
    # get a random number
    # get a quote from a list
    # show the quote in the interpreter
    pass
```

Ensuite, pour exécuter la fonction, il suffit d'écrire son nom : get_random_quote()

Paramètres Les paramètres sont des variables que vous fournissez à votre fonction et figurent dans les parenthèses. En effet, l'intérêt d'une fonction est de pouvoir être réutilisée dans des contextes différents. Par exemple, notre fonction actuelle affichera une citation au hasard, mais nous voudrons certainement en faire de même pour les personnages. Allons-nous copier / coller la fonction et changer son nom ? Non. Nous allons simplement passer la liste en paramètre.

```
1  quotes = ["Ecoutez-moi, Monsieur Shakespeare, nous avons beau être ou ne pas être, nous sommes !", "On doit pouvoir
   choisir entre s'écouter parler et se faire entendre."]
2
3  def get random item in(my list): # get a random number
4
5      item = my list[0] # get a quote from a list. For the moment, just get the first one.
6
7
8  get random item in(quotes)
9
```

NB : le dièse permet de commenter le code : vous pouvez ainsi laisser des annotations qui vous aideront à mieux comprendre ce que vous écrivez.

Python contient de nombreuses fonctions qui rendent la vie plus simple. Vous voulez connaître le type d'un objet ? Utilisez type(MyObject). Vous voulez afficher une valeur ? Utilisez print(my_value).

Une fonction exécute un ensemble d'actions mais, par défaut, elle ne renvoie pas d'information spécifique. Pour que votre fonction renvoie une certaine valeur, utilisez le mot-clé *return* :

```
1  def get random item in(my list): # get a random number
2
3      item = my list[0] # get a quote from a list
4
5      print(item) # show the quote in the interpreter
6
7      return "program is over" # returned value
8
9
10 print(get random item in(quotes))
11
```

Votre programme san_antonio.py s'est étoffé au fur et au mesure. Exécutons-le au terminal :

```
[estelle@sugar cours]$ python3 san_antonio.py
Ecoutez-moi, Monsieur Shakespeare, nous avons beau être ou ne pas être, nous som
mes !
program is over
[estelle@sugar cours]$
```

2.5 Répéter une action grâce aux boucles

Python nous offre deux boucles principales : `while` et `for`.

La boucle while While est la traduction de “tant que...”. Concrètement, la boucle s’exécutera tant qu’une condition est remplie (donc tant qu’elle renverra la valeur True). Reprenons le cas de notre programme de citations : nous affichons une citation tant que la réponse de l’utilisateur n’est pas B.

while user_answer != ‘B’ :

    print(get_random_item_in(program[’quotes’]))

Voici notre programme complété par cette boucle :

```
1   quotes = ["Ecoutez-moi, Monsieur Shakespeare, nous avons beau être ou ne pas être, nous sommes !", "On doit pouvoir
    choisir entre s'écouter parler et se faire entendre."]
2
3   characters = ["Babar", "betty boop", "calimero"]
4
5
6   def get random item in(my list): # get a random number
7
8       item = my list[0] # get a quote from a list
9
10      return item # return the item
11
12
13  user answer = "A"
14
15
16  while user answer != "B":
17
18      print(get random item in(quotes))
```

Si vous exécutez le programme, vous vous apercevrez que l’interpréteur vous affiche, en boucle, le même message ! En effet, nous sommes dans une boucle infinie : la valeur de user_answer étant A, et la boucle ne se terminant que si user_answer est égal à B, le programme ne se termine jamais. Pour fermer la boucle, il faut utiliser la variable user_answer = ”B” à la fin du programme.

La boucle for Imaginons que nous voulons utiliser une boucle pour mettre en majuscule chaque nom de personnage. Voici la syntaxe :

```
for item in a_list :
    # do something
```

Si on applique cette syntaxe à notre exemple précis, voici ce qu'il faut écrire :

```
for quote in quotes :
    quote.capitalize()
```

Notez que certaines fonctions sont réservées à certains objets : on ne peut mettre en majuscule que des mots, par exemple. C'est ce qui s'appelle des méthodes.

Questions

Question 1

Pour lancer l'interpréteur Python, j'écris dans la console le mot suivant : python (oupython3, selon votre système d'exploitation).

Vrai

Faux

Question 2

Pour lancer un programme mon_fichier.py qui utilise Python, j'utilise la commande suivante :

python mon_fichier.py -o python

python mon_fichier.py

python

python5

Question 3

Comment comparer si deux valeurs sont différentes ?

a != b

a == b

Question 4

Comment comparer que deux valeurs sont égales ?

a = b

a == b

Question 5

Modulo est représenté par un signe pourcentage (%). Il sert à :

Changer le contenu d'une variable.

Enregistrer des échantillons de notes d'instruments de musique.

Réaliser une division et renvoyer l'entier restant.

Question 6

Je souhaite afficher le message "I'm aliiiiiiiiiiiive!" si le contenu de la variable est "Sia". Comment faire ?

A/ if only singer = "Sia" :

    print("I'm aliiiiiiiiiiiive!")

B/ if singer == "Sia" :

    print("I'm aliiiiiiiiiiiive!")

C/ only singer = "Sia" :

    print("I'm aliiiiiiiiiiiive!")

D/ if singer == "Sia"

print("I'm aliiiiiiiiiiiive!")

Question 7

Le mot-clé pass est notamment utilisé :

Pour vérifier que deux valeurs sont identiques.

à l'intérieur d'une fonction pour que celle-ci s'exécute même si elle ne contient aucune "action".

Question 8

Pour créer une nouvelle fonction "message" qui prend deux paramètres ("recipient" et "sender"), j'écris :

A/ def message(recipient, sender) :

    pass

B/ func message(recipient, sender) :

    pass

C/ function message(recipient, sender) :

D/ def message recipient, sender

    pass

Question 9

Pour utiliser une fonction dans votre code, vous suivez ces deux étapes :

Exploration et indexation

Définition et exécution

Vérification et exécution

Question 10

La fonction print() sert notamment à...

Envoyer votre code chez l'imprimeur.

Afficher le message passé entre paramètres dans la console.

Concaténer deux valeurs.

3 Manipuler des objets

3.1 Modifier des chaînes de caractères

Chaque type d'objet dispose de différentes méthodes qui nous permettent d'interagir avec lui. Une méthode est une fonction associée à un type d'objet bien précis. En d'autres termes, une action que seul ce type d'objet peut faire. Étudions seulement les méthodes les plus utiles :

Split a word in several elements and create a new list out of it.

```
>>> “hello world!”.split()
['hello', 'world!']
```

Remove all white spaces at the beginning and the end of a string

```
>>> “       hello world!      “.strip()
“hello world!”
```

First letter in the first word in capital letters

```
>>> "hello world!".capitalize()
"Hello world!"
```

Every word in upper case

```
>>> "hello world!".upper()
"HELLO WORLD!"
```

Every word in lower case

```
>>> "HELLO WORLD".lower()
"hello world!"
```

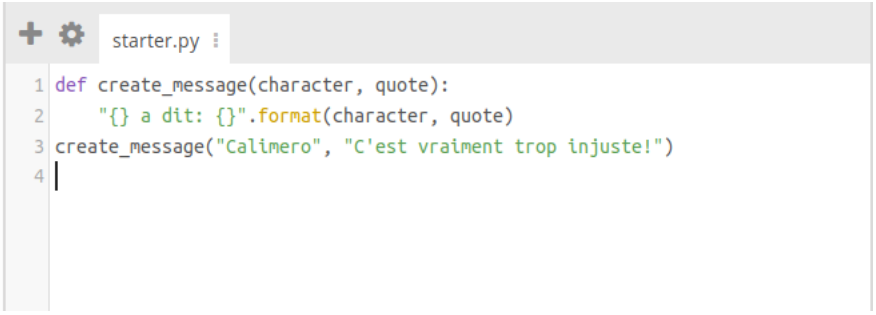
Une dernière méthode, très utilisée, vous permet de remplacer des valeurs à l'intérieur d'une chaîne à la manière d'un texte à trous. Il s'agit de la méthode format(). Chaque variable que nous souhaitons utiliser est entourée d'accolades puis définie en paramètres.

Exemple : ”{character} a dit : {quote}”.format(character=”Babar”, quote=”Tout n'est pas cirrhose dans la vie, comme dit l'alcoolique.”)

Sachez qu'il est plus usuel de ne pas définir de variable et de se contenter de donner les valeurs à remplacer dans le même ordre que la phrase :

”{} a dit : {}”.format(”Babar”, ”Tout n'est pas cirrhose dans la vie, comme dit l'alcoolique.”)

Exercice Créez une fonction create_message qui prend deux paramètres : character et quote. À l'intérieur, utilisez la méthode .format() pour créer une chaîne de caractère sur ce modèle : "{character} a dit : {quote}". Enfin, exécutez la fonction avec le personnage et la citation de votre choix.



3.2 Ranger des données dans des listes

Nous avons vu précédemment comment accéder à la première valeur d'une liste ou d'un tuple : my_list[0] (vous vous souvenez que le premier index d'une liste est toujours 0). Mais comment accéder aux valeurs suivantes ? En indiquant son index entre crochets. Ainsi, pour accéder au quatrième élément d'une liste, on écrira : my_list[3] et pour accéder au dernier élément d'une liste : my_list [-1]. Voici d'autres méthodes (ajouter, modifier ou supprimer un élément).

— Ajouter un élément à la fin de la liste : `characters.append("Mowgli")`
— Ajouter un élément à un certain index. Le premier argument est l'index, le second la valeur à insérer : `characters.insert(4, "Balou")`
— Modifier un élément : y accéder grâce à son index et lui donner une nouvelle valeur : `characters[1] = "La Fée Clochette"`
— Supprimer un élément de la liste sans renvoyer sa valeur : `characters.remove("Mowgli")`

Exercice Créez une liste qui s'appelle characters et qui contient 5 personnages de dessins animés. Puis supprimez un des personnages de la liste. Ajoutez ensuite un personnage à votre liste. Enfin, remplacez "Mowgli" par "Balou".

```
characters = [
    "Tom",
    "Jerry",
    "Mickey",
    "Donald",
    "Betty Boop",
]
characters.remove("Tom")
characters.append("Mowgli")
characters[4]="Balou"
```

3.3 Organiser des données dans un dictionnaire

Nous avons déjà vu comment accéder à la valeur d'un élément stocké dans un dictionnaire : grâce à sa clé.

Reprenons un exemple : program = {"quotes" : ["Ecoutez-moi, Monsieur Shakespeare, nous avons beau être ou ne pas être, nous sommes !", "C'est vraiment trop injuste !"]}

Pour accéder aux éléments d'une liste qui est elle-même dans un dictionnaire, il faut ajouter l'index de l'élément souhaité à la suite, entre crochets : `program["quotes"][0]`. Quelles sont les autres méthodes qu'on peut utiliser ?

- Remplacer ou ajouter une valeur : même méthode que pour une liste : `program["quotes"] = "Une nouvelle citation"`
- Mettre à jour ou ajouter plusieurs valeurs en même temps : `program.update({"characters" : ["Père Noël"], "quotes": ["Une citation unique qui sera sauvegardée"]})`
- Supprimer une clé et renvoyer sa valeur. Vous pouvez utiliser cette même méthode sur une liste :
  program.pop("quotes")
  "quotes"

Questions

Question 1

Pour transformer une chaîne de caractères en liste, j'utilise la méthode suivante :

.list()

.to_list()

.split() *

Question 2

Pour supprimer les espaces blancs au début et à la fin d'une chaîne de caractères, j'utilise

la méthode suivante :

.without_blanks()

.strip() *

.remove_blanks()

Question 3

La méthode format s'utilise de la manière suivante :

A/ "{character} a dit : {quote}".format(character="Babar", quote="Tout n'est pas cirrhose dans la vie, comme dit l'alcoolique.")

B/ "(character) a dit : (quote)".format(character="Babar", quote="Tout n'est pas cirrhose dans la vie, comme dit l'alcoolique.")

C/ "{} a dit : {}".format("Babar", "Tout n'est pas cirrhose dans la vie, comme dit l'alcoolique.")

D/ "[] a dit : []".format("Babar", "Tout n'est pas cirrhose dans la vie, comme dit l'alcoolique.")

Question 4

Pourquoi met-on un nombre entre parenthèses quand nous voulons exécuter une méthode sur cet objet ? Exemple : (1.5).is_integer()

C'est purement décoratif, je peux exécuter cette méthode sans les parenthèses.

Sinon il y aurait trop d'ambiguïté : Python ne saurait pas si la méthode fait partie du nombre (pour former un décimal) ou s'il s'agit bel et bien d'une méthode.

Question 5

Dans books = ["Cent ans de solitude", "Amours aux temps du Choléra", "L'automne du Patriarche"], que représente books ?

une chaîne de caractères.

une liste. *

un dictionnaire.

un tuple.

Question 6

Chaque élément d'une liste est associé à son index, c'est à dire à sa position.

Vrai

Faux

Question 7

Dans friends = "Doctor Who" : "Le tardis", "Red" : "Storybrooke", "Frodon" : "La Terre du Milieu", que représente friends ?

une liste.

un tuple.

un dictionnaire. *

une chaîne de caractères.

Question 8

Chaque élément d'un dictionnaire est associé à une clé.

Vrai

Faux

Question 9

books = ["Cent ans de solitude", "Amours aux temps du choléra", "L'automne du patriarche"]

Pour remplacer "L'automne du patriarche" par "Le Général dans son labyrinthe", j'écris :

books[0] = "Le Général dans son labyrinthe"

books[2] = "Le Général dans son labyrinthe" *

books.replace("L'automne du patriarche", "Le Général dans son labyrinthe")

Question 10

friends = "Doctor Who" : "Le tardis", "Red" : "Storybrooke", "Frodon" : "La Terre du Milieu"

Pour remplacer "Le tardis" par "Gallifrey", j'écris :

friends[0] = "Gallifrey"

friends["Doctor Who"] = "Gallifrey" *

friends.replace('Le tardis', "Gallifrey")

4 Utiliser des ressources externes

Python offre de nombreux modules par défaut qui vous permettent d'aller au-delà des fonctionnalités. Un module est comme une extension : elle vous donne accès à d'autres méthodes spécialisées. Afin de ne pas alourdir Python bien des modules ne sont pas chargés par défaut. Mais ils sont déjà installés sur votre ordinateur : il vous suffit de les activer en utilisant le mot clé `import` suivi du nom du module.

4.1 Importer un module

Prenons le module `random` : après l'avoir importé en utilisant la commande `import random`, la méthode `randint(a, b)` renvoie un nombre compris entre a et b (donnés en paramètre). Par exemple : `random.randint(3, 7)` renverra un nombre entre 3 et 7 (compris).

Nous allons pouvoir mettre à jour notre fonction `get_random_item(my_list)`.

```
1  def get_random_item_in(my_list):
2
3      rand_numb = random.randint(0, len(my_list) - 1)
4
5      item = my_list[rand_numb] # get a quote from a list
6
7      return item # return the item
8
```

Commentaire Voici comment comprendre chaque élément de la fonction :

- 0 : l'index de ma première citation est 0
- len(my_list) - 1 : calcule le nombre d'items dans ma liste (puisque le premier index est 0). my_list[-1] renverra le dernier objet de la liste.
- random.randint(0, len(my_list) - 1) : renvoie un nombre compris entre 0 et la longueur totale de ma liste.
- item = my_list[rand_numb] : trouve l'item qui correspond à l'index déterminé au hasard.
- return item : renvoie la valeur de l'item

Nous avons fini notre programme ou presque. Il ne reste plus qu'à ajouter autant de citations que l'on veut dans les deux premières listes, puis à lancer le programme. Pour éviter une saisie fastidieuse, nous allons récupérer des citations automatiquement depuis internet et les stocker dans un fichier externe.

4.2 Stocker des citations dans un fichier avec le module JSON

L'idéal est de stocker citations et personnages sous forme de dictionnaire : la meilleure manière de le faire est d'utiliser un format de données qui s'appelle JSON (JavaScript Object Notation). Ce format est souvent utilisé pour représenter le contenu d'une page web car il est très facile d'imbriquer des listes à l'intérieur d'autres listes. Avant de l'utiliser, pensez à l'importer : `import json`.

Chaque valeur ayant forcément une clé, voici la manière dont nous allons stocker nos objets : `{"nom_objet": "valeur"}`. Commençons par les citations. Créons un nouveau document quotes.json et remplissons-le :

[{"quote" : "Ecoutez-moi, Monsieur Shakespeare, nous avons beau être ou ne pas être, nous sommes !"},

{"quote" : "C'est vraiment trop injuste !"}]

Faisons de même pour les personnages dans un fichier characters.json :

[{"character" : "Babar"},

{"character" : "Calimero"}]

Je veux maintenant lire le fichier characters.json et le convertir en liste. Voici ce qu'il en

est en pseudo-code :

```
def read_values_from_json() :
    # Create a new empty list
    # open a json file with my objects
    # load all the data contained in this file
    # add each item in my list
    # return my completed list
```

Complétons avec ce qu'on sait déjà faire :

```
def read_values_from_json() :
    values = []
    # open a json file with my objects
    # load all the data contained in this file. data = entries
    for entry in data :
        values.append(entry["character"])
    return values
```

Il reste deux points à traiter. Le premier consiste à charger un document : nous utiliserons la méthode `open('path_to_file')` qui est souvent utilisée avec un bloc `with`. Voici la fonction complétée :

```
def read_values_from_json(key) :
    values = []
    with open("characters.json") as f :
    # load all the data contained in this file. data = entries
    for entry in data :
        values.append(entry["character"])
    return values
```

Il ne reste plus qu'à charger le contenu d'un fichier json et le convertir en objet Python (car pour le moment ce n'est que du texte). Nous utiliserons la méthode `load` :

```
def read_values_from_json(key) :
    values = []
    with open("characters.json") as f :
    data = json.load(f)
    for entry in data :
```

```
            values.append(entry["character"])
        return values
```

4.3 Collecter des citations automatiquement avec Scrapy

La communauté Python est très active et a produit plusieurs milliers de modules. Pip est l'outil de gestion de paquets utilisé pour installer et gérer des modules externes. Il est installé automatiquement quand vous installez Python. Pour installer une librairie, voici la commande : `python3 -m pip install ma_librairie`.

Une des librairies les plus populaires pour récupérer de l'information sur Internet est Scrapy. Il s'agit d'un scraper entièrement personnalisable, en d'autres termes un petit robot que vous pouvez configurer vous-même pour qu'il cherche ce que vous souhaitez. Installons-le : `python3 -m pip install scrapy`.

Le fichier requirements.txt Une bonne pratique est de réunir toutes les librairies externes dans un fichier qui s'appelle requirements.txt et qui se trouve à la racine de votre dossier (au même niveau que san_antonio.py). Ainsi, il vous suffit d'écrire `python3 -m pip install -r requirements.txt` pour que toutes les librairies s'installent d'un seul coup. Pour un point sur le code avant d'utiliser Scrapy, voir les dernières captures de console en annexes.

Utiliser Scrapy Commençons par créer un nouveau document : `characters_scrapper.py`. Dans ce document, je colle le code fourni par Scrapy et je commence à l'adapter. Comment fonctionne un scraper ? Il va sur une url que nous lui donnons puis va trouver les éléments qui correspondent à un sélecteur CSS (Cascading StyleSheet) qui sert à la mise en page d'un site. Petit rappel :

- div : sélecteur d'une balise HTML div
- .myclass : cible toutes les balises ayant cette classe
- #myid : cible la balise ayant cet id
- div .myclass : cible tous les éléments ayant la classe .myclass contenus dans une balise div

Prenons un exemple : rendez-vous sur cette page Wikipedia https://fr.wikipedia.org/wiki/Cat%C3%A9gorie:Personnage_d%27animation qui affiche une liste de personnages que nous voulons récupérer. Afin de trouver le sélecteur, je clique droit sur l'élément que je souhaite cibler et je clique sur inspecter l'élément. Le sélecteur est le nom de la balise

HTML (a dans mon cas). Voici le scraper :

```
import scrapy

class BlogSpider(scrapy.Spider):
    name = 'characterspider'
    start_urls = ['https://fr.wikipedia.org/wiki/Cat%C3%A9gorie:Personnage_d\'animation']

    def parse(self, response):
        for link in response.css('div#mw-pages div.mw-content-ltr li'):
            yield {'character': link.css('a ::text').extract_first()}

```

Pour récupérer les personnages, et les stocker dans un nouveau fichier characters.json , je lance la commande suivante : `scrapy runspider characters.py -o characters.json`. Et là tous mes personnages apparaissent ! Vous voyez maintenant comment récupérer des informations grâce à un scraper. Pour information, ces robots sont utilisés par de nombreuses industries pour agréger les données “publiques” accessibles librement ou pour publier du contenu massivement en l’automatisant. Vous pourriez ainsi, tout à fait illégalement, scraper le site du Monde, copier l’intégralité des articles et les publier sur votre site. Il s’agit d’une technique si répandue que Google a créé un formulaire pour dénoncer les sites de scraping...

Annexes

Voici le fichier `san_antonio.py`.

Étape 1 :

```python
# -*- coding: utf8 -*-
import random

quotes = [
    "Ecoutez-moi, Monsieur Shakespeare, nous avons beau être ou ne pas être, nous sommes !",
    "On doit pouvoir choisir entre s'écouter parler et se faire entendre."
]

characters = [
    "alvin et les Chipmunks",
    "Babar",
    "betty boop",
    "calimero",
    "casper",
    "le chat potté",
    "Kirikou"
]

def message(character, quote):
    n_character = character.capitalize()
    n_quote = quote.capitalize()
    return "{} a dit : {}".format(n_character, n_quote)

def get_random_quote():
    # get a random number
    # get a quote from an array
    # show the quote in the interpreter
    pass

def get_random_item_in(my_list):
    rand_numb = random.randint(0, len(my_list) - 1)
    item = my_list[rand_numb] # get a quote from a list
    return item # return the item

# Programm
user_answer = input('Tapez entrée pour connaître une autre citation ou B pour quitter le programme.')

while user_answer != "B":
    print(message(get_random_item_in(characters), get_random_item_in(quotes)))
    user_answer = input('Tapez entrée pour connaître une autre citation ou B pour quitter le programme.')
```

Étape 2 :

```python
# -*- coding: utf8 -*-
import random
import json

def read_values_from_json(file, key):
    values = []
    with open(file) as f:
        data = json.load(f)
        for entry in data:
            values.append(entry[key])
    return values

def message(character, quote):
    n_character = character.capitalize()
    n_quote = quote.capitalize()
    return "{} a dit : {}".format(n_character, n_quote)

def get_random_item_in(my_list):
    rand_numb = random.randint(0, len(my_list) - 1)
    item = my_list[rand_numb] # get a quote from a list
    return item # return the item

def get_random_quote():
    all_values = read_values_from_json('quotes.json', 'quote')
    return get_random_item_in(all_values)

def get_random_character():
    all_values = read_values_from_json('characters.json', 'character')
    return get_random_item_in(all_values)

# Programm
user_answer = input('Tapez entrée pour connaître une autre citation ou B pour quitter le programme.')

while user_answer != "B":
    print(message(get_random_character(), get_random_quote()))
    user_answer = input('Tapez entrée pour connaître une autre citation ou B pour quitter le programme.')
```

```python
# -*- coding: utf8 -*-
import json
import random

# Give a Json file and return a List
def read_values_from_json(path, key):
    values = []
    with open(path) as f:
        data = json.load(f)
        for entry in data:
            values.append(entry[key])
        return values

# Give a json and return a list
def clean_strings(sentences):
    cleaned = []
    # Store quotes on a list. Create an empty list and add each sentence one by one.
    for sentence in sentences:
        # Clean quotes from whitespace and so on
        clean_sentence = sentence.strip()
        # don't use extend as it adds each letter one by one!
        cleaned.append(clean_sentence)
    return cleaned

# Return a random item in a list
def random_item_in(object_list):
    rand_numb = random.randint(0, len(object_list) - 1)
    return object_list[rand_numb]

# Return a random value from a json file
def random_value(source_path, key):
    all_values = read_values_from_json(source_path, key)
    clean_values = clean_strings(all_values)
    return random_item_in(clean_values)


```

```
35
36
37 ####################
38 ###### QUOTES #######
39 ####################
40
41 # Gather quotes from San Antonio
42
43 def random_quote():
44     return random_value('quotes.json', 'quote')
45
46 ######################
47 #### CHARACTERS ######
48 ######################
49
50 # Gather characters from Wikipedia
51
52 def random_character():
53     return random_value('characters.json', 'character')
54
55
56 ######################
57 #### INTERACTION ######
58 ######################
59
60 # Print a random sentence.
61
62 def print_random_sentence():
63     rand_quote = random_quote()
64     rand_character = random_character()
65     print(">>>> {} a dit : {}".format(rand_character, rand_quote))
66
67 def main_loop():
68     while True:
69         print_random_sentence()
70         message = ('Voulez-vous voir une autre citation ?'
71                    'Pour sortir du programme, tapez [B].')
72         choice = input(message).upper()
73         if choice == 'B':
74             break
75             # This will stop the loop!
76
77 if __name__ == '__main__':
78     main_loop()
```