**PYTHON**

<https://openclassrooms.com/fr/courses/4262331-demarrez-votre-projet-avec-python>

**Partie 1 - Faites connaissance avec Python**

* Pourquoi apprendre Python ?
* Installez Python !
* Découvrez le vocabulaire de Python
* Quiz : Découvrez les bases de Python

**Partie 2 - Posez les fondations de votre programme**

* Créez votre premier script
* Comparez des valeurs avec les opérateurs
* Ajoutez un peu de logique avec les conditions
* Structurez votre programme en utilisant les fonctions
* Répétez une action grâce aux boucles
* Quiz : Les fondations

**Partie 3 - Manipulez des objets**

* Modifiez des chaînes de caractères
* Modifiez des nombres
* Rangez des données dans des listes
* Organisez vos données dans un dictionnaire
* Quiz : Manipulez des objets

**Partie 4 - Utilisez des ressources externes**

* Installez votre premier module
* [BONUS] Stockez des citations dans un fichier avec le module json
* [BONUS] Collectez des citations automatiquement avec Scrapy

Liens :

* Tuto : <https://docs.python.org/3.9/tutorial/index.html>
* Doc : <https://docs.python.org/3.9/index.html>
* w3schools : <https://www.w3schools.com/python/> | <https://www.w3schools.com/python/python_intro.asp>

**Partie 1 - FAITES CONNAISSANCE AVEC Python**

**1\_ POURQUOI APPRENDRE PYTHON ? (**(créateur : [Guido van Rossum](https://en.wikipedia.org/wiki/Guido_van_Rossum" \t "_blank), amateur de Monthy …)

Utilisation : création programmes ordi, sites web et même jeux (Pinterest, Instagram et site New York Times

* un des langages principaux utilisés en data analysis (analyse de données) et machine learning
* langage de référence pour apprendre la robotique
* dans de nombreux projets libres, Guido van Rossum fervent contributeur à des projets Open Source

Compatibilité : tous les OS (Windows, Mac et Linux)

Communauté : sur [Meetup](https://www.meetup.com/fr-FR/topics/python/)

Projet (du cours) : faire dire à un personnage de dessin animé des citations de [San Antonio](https://fr.wikipedia.org/wiki/San-Antonio_(s%C3%A9rie)) sur Internet

**2\_ INSTALLEZ PYTHON !**  <https://www.python.org/downloads/>

Installation : Cliquez sur Download Python 3.X.X (X étant les dernières versions en date)

* **Mac :** 
  + double-clic sur le téléchargement puis lancer l’instal.
  + Taper  brew install python3  depuis la console
* **Windows**: Installer le téléchargement => Démarrer > Tous les Programmes > Python (Python, IDLE)
* **Linux**: Utilisez APT-GET pour l'installer ou mettre à jour Python  sudo apt-get install python3

em : pré-installé souvent sous Linux mais sa version certainement obsolète..

Premiers pas

#### Mac/Linux : depuis « terminal »/ « console »

#### (cours : [La Console, ça se mange ?](https://openclassrooms.com/courses/reprenez-le-controle-a-l-aide-de-linux/la-console-ca-se-mange) / [Reprenez le contrôle à l’aide de Linux !](https://openclassrooms.com/courses/reprenez-le-controle-a-l-aide-de-linux) )

#### Windows : « invite de commande »

Interpréteur Python traduit les instructions écrites en Python en binaire.

#### Mac et Linux : taper « python3 »

#### Windows : [IDLE](https://fr.wikipedia.org/wiki/IDLE_(Python)) est un logiciel qui permet d'écrire du Python et l'exécuter. [Découvrez-en plus](https://fr.wikipedia.org/wiki/IDLE_(Python)).

**3\_VOCABULAIRE PYTHON !**

Variables :

Bergson, philosophe du langage, "Nous ne voyons pas les choses mêmes; nous nous bornons, le plus souvent, à lire des étiquettes collées sur elles."

* déclaration >>> varname = "valeur"
* lecture >>> varname
* modification >>> vaname = "newval"
* suppression >>> del newval

Types d’objets : (cours :  [Découvrez la programmation orientée objet en Python](https://openclassrooms.com/courses/decouvrez-la-programmation-orientee-objet-avec-python))

#### nombres : entiers (integers) et les décimaux (floats) >>> my\_integer = 1 / >>> my\_float = 1.3

#### chaînes de caractères : (strings) : >>> quote = "San Antonio a dit : 'Ecoutez-moi, '"

* + échapper un caractère = antislash (“\”) avant >>> quote = "San Antonio a dit : \"Ecoutez-moi\""
  + sur plusieurs lignes triple guillemets en début et fin de chaine.

#### booléens (booleans) : True / False  (pensez aux majuscules !)

#### listes : succession de plusieurs objets

#### déclaration : >>> liste = ["valeur1", "valeur2", "valeur3"]

#### liste de variables : >>> var1 = "chaine1" / var2 = "chaine2" / liste = [chaine1, chaine2]

#### lecture : >>> liste[0]

#### tuples : créer des structures dont le nombre d'éléments ne peut pas changer (ni ajout, ni suppression)

* + déclaration >>> ("valeur1", "valeur2", "valeur3")

#### dictionnaires : « structure de données qui permet d'associer des objets deux à deux »(1 clé # 1valeur)

#### déclaration : >>> dico = { ‘cle1’: ‘valeur1’, ’cle2’: ‘valeur2’, ‘cle3’: ‘valeur3’}

#### lecture >>> dico[‘valeur1’]

#### Remarque : type d’un objet ? >>> type(objet)

**Partie 2 - POSEZ LES FONDATIONS DE VOTRE PROGRAMME**

**4\_PREMIER SCRIPT**

Fichier  externe : enregistrer du code python : filename.py

#### Linux/Mac

* + créer : **touch filename.py**
  + éditer : **subl filename.py**
  + coder : print("Yo ! T’as pas un 06 ?") et sauvegarder.
  + exécuter **python3 san\_antonio.py**

#### Windows : utilisez IDLE

* + Créer : **File > New File**, sauvegarde ctrl + s
  + Exécution : **Run > Run Module**.

Code

|  |
| --- |
| 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"  ] |

charset : ajouter # -\*- coding: utf8 -\*- en tête du fichier pour prendre en compte les caractères accentués.

commentaire : **# commentaire**

Repository : GitHub [sur ce lien](https://github.com/OpenClassrooms-Student-Center/demarrez_votre_projet_avec_python/tree/P2C1).

**5\_OPERATEURS**

Algorithme : 0/ affiche une citation au hasard 1/ enter affiche une nouvelle citation ou ‘B’ ferme le programme.

Comparaison : renvoient un booléen ‘True’ ou ‘False’ 🡪 **== | != | >= | <**

Calcul : renvoient le résultat du calcul 🡪 **+ | - | \* | / | %** (reste de la division)

Repository : GitHub [en cliquant ici](https://github.com/OpenClassrooms-Student-Center/demarrez_votre_projet_avec_python/tree/P2C2)

**6\_CONDITIONS**

|  |  |  |
| --- | --- | --- |
| If | If then | If else if then |
| if user\_answer == "B":  # leave the program | if user\_answer == "B":  # leave the program  else:  # show another quote | if user\_answer == "B":  pass  elif user\_answer == "C":  print("C pas la bonne réponse ! Et G pas d’humour, je C...")  else:  # show another quote |

Indentation : 4 espaces (Python très strict sur le nombre d’espace)

Remarque : pass  ne sert pas vraiment à quitter un programme # nop

Repository : GitHub [en cliquant ici](https://github.com/OpenClassrooms-Student-Center/demarrez_votre_projet_avec_python/tree/P2C3)

**7\_CONDITIONS**

Objectif : créer une fonction show\_random\_quote

Déclaration : **def fctname():**

|  |
| --- |
| def get\_random\_quote():  # get a random number  # get a quote from an array  # show the quote in the interpreter  pass |

Exécution : **get\_random\_quote()**

Paramètres :

|  |
| --- |
| def get\_random\_item\_in(my\_list):  # TODO: get a random item  my\_list[0] # get an item from a list. For the moment, just get the first one.  # TODO: show the quote  get\_random\_item\_in(quotes) |

Fonctions standards : **type(variable) | print(valeur) | input(mon\_message)**

Ex : user\_answer = input('Tapez entrée pour connaître une autre citation ou B pour quitter le programme.')

Retourner une valeur : **return valeur**

|  |
| --- |
| def get\_random\_item\_in(my\_list):  # TODO: get a random number  item = my\_list[0] # get a quote from a list  print(item) # show the quote in the interpreter  return "program is over" # returned value  print(get\_random\_item\_in(quotes)) |

Repository : GitHub [en cliquant ici](https://github.com/OpenClassrooms-Student-Center/demarrez_votre_projet_avec_python/tree/P2C4)

**8\_BOUCLES**

|  |  |
| --- | --- |
| While | For |
| **while condition**  **instruction** | **for item in a\_list:**  **# do something** |
| while user\_answer != "B":  print(get\_random\_item\_in(quotes)) | for quote in quotes:  quote.capitalize() |

Repository : GitHub [en cliquant ici](https://github.com/OpenClassrooms-Student-Center/demarrez_votre_projet_avec_python/tree/P3C4)

**Partie 3 - MANIPULEZ DES OBJETS**

Vocabulaire : méthode = fonction dédiée à un type d’objets (ex : capitalize() pour des chaines de caractères)

**9\_ CHAÎNES DE CARACTÈRES**

* **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!”

* **capitalize()** first letter in the first word in capital letters

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

* **upper()** every word in upper case

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

* **lower()** every word in lower case

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

* **format()** *formats the specified value(s) and insert them inside the string's placeholder*

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

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

Documentation : <https://docs.python.org/3.5/library/stdtypes.html#text-sequence-type-str>

**10\_ NOMBRES**

* **is\_integer()** Is an integer?

>>> (2.5).is\_integer()=> False

Documentation : <https://docs.python.org/3.5/library/stdtypes.html#numeric-types-int-float-complex>

**11\_ LISTES** (tuple)

* **index()** *donner position d’un élément*

>>> characters.index("Babar") => 1

* **append()** ajouter un élément à la fin de la liste

>>> characters.append("Mowgli")

* **insert()** ajouter un élément à un certain index

>>> characters.insert(4, "Balou")

* **=** modifier un élément : y accéder grâce à son index et lui donner une nouvelle valeur

>>> characters[1] = "La Fée Clochette"

* **pop()** supprimer le dernier élément de la liste et ***renvoyer sa valeur***.

>>> characters.pop() # removes the last item and returns it => 'Mowgli'

vous pouvez passer en paramètre l'index de l'élément à supprimer

>>> characters.pop(4) # removes the item at index 4 and returns it | 'Balou'

* **remove(param)**supprimer un élément de la liste ***sans renvoyer sa valeur. (1er*** élément à cette valeur supprimée)

>>> characters.remove("Mowgli")

Remarque : [Difference between del, remove and pop on lists](https://stackoverflow.com/questions/11520492/difference-between-del-remove-and-pop-on-lists)

* **len()** connaitre le nombre d’éléments dans une liste

>>> len(characters)

* [-1] accéder au dernier élément d’une liste

>>> characters[-1]

Documentation : <https://docs.python.org/3.5/tutorial/datastructures.html>

**12\_ DIXTIONNAIRES**

|  |
| --- |
| >>> program = {"quotes": ["q1","q2"], "characters": ["c1","c2","c3","c4","c5"]}  >>> program["characters"] => ["c1","c2","c3","c4","c5"]  >>> program["characters"][0] =>"c1" |

* **=**  remplacer ou ajouter une valeur : même méthode que pour une liste.

>>> program["characters"] = "un nouveau nom"

* **update()** mettre à jour ou ajouter plusieurs valeurs en même temps

>>> program.update({"characters" : ["c1new", "c2new"], "quotes": ["qnew"]})

* **pop()** supprimer une clé et renvoyer sa valeur. Vous pouvez utiliser cette même méthode sur une liste !

>>> program.pop("quotes") => "quotes"

Documentation : <https://docs.python.org/3.5/library/stdtypes.html#mapping-types-dict>

Remarque : cours Data Science : [Découvrez les librairies Python pour la Data Science](https://openclassrooms.com/fr/courses/4452741-decouvrez-les-librairies-python-pour-la-data-science)

**Partie 4 - UTILISEZ DES RESSOURCES EXTERNES**

**13\_INSTALLEZ VOTRE PREMIER MODULE**

Définition : module est une extension qui donne accès à d’autres méthodes spécialisées.

Importation : >>> import module\_name

|  |
| --- |
| >>> import turtle  >>> turtle.forward(100) |

Remarque : Turtle est une librairie de dessin graphique. [La documentation est ici](https://docs.python.org/3/library/turtle.html).

|  |
| --- |
| from turtle import \*  color('red', 'yellow')  begin\_fill()  while True:  forward(200)  left(170)  if abs(pos()) < 1:  break  end\_fill()  done() |

utilisation : selon la méthode d’importation

* **Méthode 1 (conseillée) :**

import module\_name  = importe tous les objets utilisés par ce module

module\_name.method() = manière la plus commune et la plus explicite d’utiliser les modules.

* **Méthode 2 (déconseillée)** : pour ne pas écrire le nom du module puis la méthode.

from module import \* = importe tous les objets utilisés par ce module

method()  = sans la précéder du nom du module.

module random : import random

* + **randint(a, b)** . renvoie un nombre au hasard compris entre a et b (donnés en paramètre)

|  |
| --- |
| 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 |

Notes :

* 0 l’index de ma première citation est 0
* len(object\_list) – 1 renvoie le nombre d’items dans ma liste
* random.randint(0, len(my\_list) - 1) renvoie un nombre entre 0 et la longueur totale
* item = object\_list[rand\_numb] trouve l’item qui correspond à l’index au hasard
* return item renvoie la valeur de l’item

Repository : GitHub [en cliquant ici](https://github.com/OpenClassrooms-Student-Center/demarrez_votre_projet_avec_python/tree/P4C1)

Pour aller plus loin : [Découvrez le concept de Programmation Orientée objet](https://openclassrooms.com/courses/decouvrez-la-programmation-orientee-objet-avec-python)

**13\_DONNES A PARTIR D’UN FICHIER JSON**

JSON : JavaScript Object Notation (très facile d’imbriquer des listes à l’intérieur d’autres listes)

* + Syntaxe : {"nom\_objet": "valeur"} #dictionnaire de python
  + Import : import json

*quotes.json*

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

characters.json

|  |
| --- |
| [{"character": "alvin et les Chimpunks"},  {"character": "Babar"}] |

pseudo-code :

|  |  |
| --- | --- |
| pseudo-code | 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 | 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 |

accès fichier :

* charger un document : open('path\_to\_file') (mais ne ferme pas le fichier)
* bloc with : with file as variable (le bloc  with  ouvre le fichier et le ferme à la fin)

documentation : la [documentation](https://docs.python.org/3/tutorial/inputoutput.html#methods-of-file-objects). 😉

chargement de données :

* [load()](https://docs.python.org/3/library/json.html#json.load) transforme un fichier JSON en objet Python.

pour aller plus loin : passer le nom du fichier en paramètre (pour lire le fichier personnages puis citations)

|  |
| --- |
| 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 |

fonction qui renverra un personnage au hasard :

|  |
| --- |
| # Return a random value from a json file  def random\_character():  all\_values = read\_values\_from\_json()  return get\_random\_item\_in(all\_values) |

Repository : GitHub [en cliquant ici](https://github.com/OpenClassrooms-Student-Center/demarrez_votre_projet_avec_python/tree/P4C2)

**14\_AUTOMATIQUEMENT AVEC Scrapy**

Modules externes :Pip est l’outil de gestion de paquets utilisé pour installer et gérer des modules externes.

* liste des modules externes : [PyPI](https://pypi.python.org/pypi?%3Aaction=browse) <https://pypi.org/>
* installation : python3 -m pip install ma\_librairie

[Scrapy](https://scrapy.org/) : une des librairies les plus populaires pour récupérer de l’information sur Internet [page d’accueil](https://scrapy.org/)

= scraper entièrement personnalisable

* installation : >>> python3 -m pip install scrapy

requirement.txt fichier pour réunir toutes les librairies externes à la racine du dossier

* installation : python3 -m pip install -r requirements.txt
* versions : BeautifulSoup==3.2.0

projet

|  |
| --- |
| # -\*- 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)  #####################  ###### QUOTES #######  #####################  # Gather quotes from San Antonio  def random\_quote():  return random\_value('quotes.json', 'quote')  ######################  #### CHARACTERS ######  ######################  # Gather characters from Wikipedia  def random\_character():  return random\_value('characters.json', 'character')  ######################  #### INTERACTION ######  ######################  # Print a random sentence.  def print\_random\_sentence():  rand\_quote = random\_quote()  rand\_character = random\_character()  print(">>>> {} a dit : {}".format(rand\_character, rand\_quote))  def main\_loop():  while True:  print\_random\_sentence()  message = ('Voulez-vous voir une autre citation ?'  'Pour sortir du programme, tapez [B].')  choice = input(message).upper()  if choice == 'B':  break  # This will stop the loop!  if \_\_name\_\_ == '\_\_main\_\_':  main\_loop() |

question : \_\_name\_\_ == '\_\_main\_\_': qu’est ce que cette ligne ?

Utiliser Scrapy (objectif : comment lire une documentation)

scrapy possède un [robot d’exemple](https://doc.scrapy.org/en/1.2/intro/overview.html) <https://doc.scrapy.org/en/1.2/intro/overview.html>

characters\_scrapper.py . coller le code fourni par Scrapy

* fonctionnement d’un scraper : va trouver les éléments correspondant à un sélecteur CSS sur une URL

|  |
| --- |
| 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()} |

* utilisation : $ scrapy runspider characters.py -o characters.json
* code final : [voir le code final du Scraper](https://github.com/OpenClassrooms-Student-Center/demarrez_votre_projet_avec_python/tree/master)

FYI : scrapers très utilisés

* pour agréger les données “publiques” accessibles librement
* ou pour publier du contenu massivement en l’automatisant.

exp1 : scraper le site du Monde, copier l’intégralité des articles et les publier sur votre site.

Rem : technique si répandue que Google a créé un formulaire pour dénoncer les sites de scraping !   
[Lire l’article de MyPoseo sur Google Scraper Report](http://fr.myposeo.com/blog/google-scraper-report-formulaire-denoncer-les-sites-scraping/).

* technique d’acquisition assez répandue

exp2 : Airbnb a ainsi utilisé un scraper pour augmenter sa base d’utilisateurs   
([lire l’analyse de GrowthHackers](https://growthhackers.com/growth-studies/airbnb)) !

Repository : GitHub [ce lien](https://github.com/OpenClassrooms-Student-Center/demarrez_votre_projet_avec_python/tree/master)

<https://github.com/OpenClassrooms-Student-Center/demarrez_votre_projet_avec_python/tree/master>