# Introduction au langage PYTHON

DUT Informatique
IUT de Vélizy

**Ce symbole indique un texte à lire attentivement avant de se lancer dans les exercices qui suivent.**

Vous disposerez de trois sources de cours différentes pour apprendre le python.

1. Le tutoriel python 2.7 en français, dont vous êtes censés acquérir les parties 1 à 5 pour avoir les bases de la programmation en python :

   `https://docs.python.org/fr/2.7/tutorial/`

2. Les vidéos de Dr Algo : cherchez la chaine Dr Algo sur youtube ou cliquez sur

   `https://www.youtube.com/channel/UCAu1QoF2PfYcvIQt5lh0-xw`

3. Pour chercher un point précis comme la liste des méthodes disponibles pour un type : la référence python 2.7 en franglais (une grosse partie non traduite mais il faut vous y habituer)

   `https://docs.python.org/fr/2/library/index.html`

## 1 Premiers programmes et types de base

**Ce symbole indique qu'une vidéo du Dr Algo est liée au sujet abordé – la regarder peut être une bonne idée.**

Introduction au langage python
Un exemple simple de programme en python
Types de base en python
Le type `str` en python

**Ce symbole précise les compétences SACoche liées aux exercices qui suivent.**

TYPE

☐ 0. Dans le mode console : définissez deux variables $a$ et $b$ valant respectivement 10 et 3, calculez ce que valent $a + b$, $a * b$ ou $a/b$. Changez la valeur de $b$ en 3.0 et recommencez. Quelle est la différence ?

☐ 1. Toujours dans le mode console, utilisez la fonction `type` pour demandez le type des objets. Créez des variables entières, des réels, des chaînes de caractères et affichez leur type.

☐ 2. Affichez maintenant une phrase mélangeant une chaîne de caractère et la valeur d'une variable.

À partir de maintenant les exercices seront à traiter en écrivant des scripts (ou programmes) python. N'hésitez pas à revoir les vidéos correspondantes si nécessaire. N'oubliez pas d'ajouter la ligne
`#coding: utf-8`
au début du programme si vous n'utilisez pas que des caractères ascii (accents, etc.)

☐ 3. Écrivez un programme *age.py* qui demande son âge à l'utilisateur et répond en utilisant sa réponse au milieu d'une phrase, comme par exemple en lui donnant l'âge qu'il aura en 2027 (ou sinon, soyez imaginatifs, c'est vous les jeunes :) )

☐ 4. Écrivez `matieres.py` : demandez maintenant deux informations à l'utilisateur, sous forme de chaînes de caractères, et faites une phrase avec les deux informations. Par exemple, on lui demande le nom de ses deux matières préférées, et on répond "moi aussi j'adore (...) et (...) ." (Réponse : programmation et algorithmique). *Attention à bien différencier les fonctions natives* `input` *qui renvoie une valeur et* `raw_input` *qui renvoie une chaîne de caractères.*

☐ 5. Dans `queltype.py`, demandez à l'utilisateur d'entrer une valeur avec `input`, stockez la dans une variable et affichez son type. Rappel : `type(a)` renvoie le type d'une variable a. Vous pouvez l'afficher avec `print`. Testez avec différentes valeurs : `35, 284736537, -35, 2.4, -10.6, "bonjour", "b", [2,4,3], ["bonjour, "au revoir"], ...` . regardez bien le type à chaque fois.

☐ 6. Dans `affichages.py`, affichez (sans utiliser de boucle) :
- la chaîne "aaaaaa..." (250 "a")
- la chaîne "bbbb....a" (250 "b" suivis d'un "a")
- la chaîne "bbbb....aaabbb..." (250 "b" suivis d'un "a" suivis de 100 b)
- La liste [0,1,...1000] (au complet)
- La liste [1000,1001,...1507] (au complet)
- 50 fois "bonjour" en allant à la ligne à chaque fois (utilisez "\n")

## 2 Les boucles

La boucle `while` en python
La boucle `for` en python


Partie 4 du tutoriel :
`https://docs.python.org/fr/2.7/tutorial/controlflow.html`


BOUCLES1, BOUCLES2, BOUCLES ?

☐ 7. Écrire un programme qui affiche 50 fois "bonjour", cette fois-ci en utilisant une boucle `while`.

☐ 8. Même exercice avec une boucle `for`.

☐ 9. Écrire un programme qui demande un mot et un entier, et affiche autant de fois le mot entré. Attention à distinguer `raw_input` et `input`.

☐ 10. Écrire un programme qui affiche un rectangle d'étoiles de p lignes et q colonnes. Par exemple si on entre 3 pour p et 12 pour q, la sortie sera :

```
************
************
************
```

☐ 11. Écrire un programme qui affiche un triangle d'étoiles de taille donnée, par exemple si on entre 6 le programme affiche :

```
*
**
***
****
*****
******
```

☐ 12. Même exercice mais cette fois la sortie est

```
          *
        *   *
      *   *   *
    *   *   *   *
  *   *   *   *   *
*   *   *   *   *   *
```

☐ 13. Écrire un programme qui calcule, étant donné un entier n, la plus grande puissance de n inférieure à un million.

☐ 14. Écrire un programme qui affiche, étant donnée une somme initiale et un taux d'intéret par an en pourcentage, les sommes successives présentes sur le compte en banque chaque année ainsi que le nombre d'années écoulées, jusqu'à arriver à un million d'euros. Attention : divisez par 100.0 et non par 100 de sorte à avoir des résultats en *float*.

☐ 15. La suite de Fibonacci est la suite d'entiers 1 1 2 3 5 8 13 21 34 ... Chaque terme est la somme des deux précédents. Écrire un programme qui affiche les 100 premiers termes de la suite de Fibonacci (indication : il faut toujours garder les deux derniers termes en mémoire)..

## 3 Le type `list`

le type `list` en python


parties 3.1.4 et 5.1 du tutoriel (ignorer pour l'instant 5.1.1, 5.1.2, 5.1.3)
`https://docs.python.org/fr/2.7/tutorial/introduction.html#lists`
`https://docs.python.org/fr/2.7/tutorial/datastructures.html#more-on-lists`

SACoche | LIST, BOUCLES1, BOUCLE ?, CHOIX

☐ 16. Faites un programme `listes1.py` qui crée la liste [3,5,10], l'affiche ; ajoutez ensuite 12 et 17 à la fin affichez à nouveau. Puis changez le 10 en -7.

☐ 17. Dans le même exercice, avec une boucle `for` écrivez le double de chaque élément(devraient s'afficher 6, 10, -14, 24 et 34).

☐ 18. Avec une boucle for, remplacez l'élement d'indice $i$ par sa valeur plus i. Devrait s'afficher 3, 6, -5, 15, 21.

☐ 19. Demandez maintenant 5 entiers à l'utilisateur, mettez-les dans une liste (vide [ ] au départ) puis affichez la somme des ces entiers ainsi que le plus grand de ces entiers.

☐ 20. Même exercice mais on ne sait pas au départ la taille de la liste, on s'arrête quand l'utilisateur entre 0.

☐ 21. Nouveau programme : on demande des entiers à l'utilisateur, que l'on place dans une liste, puis on crée une nouvelle liste qui contient uniquement les entiers pairs de la première liste.

# 4 Le type `dict`

Le type `dict` en python


Partie 5.5 du tutoriel :
`https://docs.python.org/fr/2.7/tutorial/datastructures.html#dictionaries`
Liste des méthodes de `dict`
`https://docs.python.org/2/library/stdtypes.html#mapping-types-dict`

DICT, CHOIX

☐ 22. Dans un programme `tests_dico.py`, créez un objet `ville` de type `dict`. Cet objet associera à un nom la ville d'origine de cette personne, comme `ville["david"] = "rouen"`. Entrez au mons trois autres valeurs de ce type dans le dictionnaire, de votre choix, enfin affichez le dictionnaire dans la console.

☐ 23. Affichez uniquement la *liste* des *clés* de `ville` en appelant la bonne méthode. Affichez ensuite la liste des *valeurs*. Affichez la *longueur* du dictionnaire.

☐ 24. Dans la suite de votre programme, demandez à l'utilisateur d'entrer au clavier un nom. Si le nom est une des clés du dictionnaire, affichez la ville d'origine de la personne, sinon affichez "je ne dispose pas de cette information".

☐ 25. Faites en sorte que dans le cas où le nom donné est inconnu, on demande la ville d'origine et on l'ajoute dans le dictionnaire ("je ne dispose pas de cette informaton, voulez-vous l'ajouter ?").

☐ 26. Enfin, écrivez une boucle qui parcourt le dictionnaire et écrit pour chaque entrée une ligne du type "la ville d'origine de xxx est xxx".

## 5 Le type `set`

Le type `set` en python


Partie 5.4 du tutoriel :
`https://docs.python.org/fr/2.7/tutorial/datastructures.html#sets`
Liste des méthodes de `set`
`https://docs.python.org/2/library/stdtypes.html#set-types-set-frozenset`

CHOIX

☐ 27. Créez un ensemble (`set`) nommé *mammiferes* contenant quelques noms de ces animaux, puis un ensemble carnivores contenant le nom de quelques animaux carnivores (avec au moins un élément en commun, si possible pas uniquement des mammifères).

☐ 28. Affichez l'union, l'intersection de ces ensembles, affichez le premier privé du second. Écrivez une boucle qui parcourt un de ces ensembles et affiche ses éléments (un par ligne)

# 6 Les fonctions

Les fonctions en python


Partie 4.6 du tutoriel (4.7 pour aller plus loin) :
`https://docs.python.org/fr/2.7/tutorial/controlflow.html#defining-functions`

SACoche | FONCTIONS1

☐ 29. Écrivez une fonction qui, étant donnée une liste d'entiers `entiers` donnée en argument, renvoie la liste des entiers pairs de `entiers`. Testez !

☐ 30. Écrivez une fonction qui renvoie un booléen `True/False` qui indique si tous les éléments d'une liste d'entiers sont pairs ou non.

☐ 31. Écrivez une fonction qui prend un entier `n` en paramètre et renvoie la liste des `n` premiers multiples de 7 dans la suite de Fibonacci.

☐ 32. écrire une fonction `diviseurs` qui renvoie dans un tableau la liste des diviseurs d'un entier donné.

☐ 33. écrire une fonction `est_premier`, utilisant `diviseurs`, qui teste si un entier donné est premier.

☐ 34. écrire une fonction `liste_premiers` qui retourne dans un tableau la liste des nombres premiers strictement inférieurs à $n$, en utilisant les fonctions précédentes.

# 7 Exercices supplémentaires pour récapituler tout ça

AVIS À LA POPULATION
À partir de maintenant **et pour toute la vie** les exercices de programmation en python et d'algorithmique sont à écrire *en utilisant des fonctions*.
Une fonction prend des paramètres *et n'utilise pas les fonctions* `input` *ou* `raw_input` *pour les demander*. Faire ceci est passible de châtiments corporels vraiment vicieux.
A la limite **vous pouvez oublier que `input` et `raw_input` existent et ne plus jamais les utiliser.**
De même une fonction, sauf s'il s'agit expréssement d'une fonction d'affichage, n'utilise **jamais** `print` : une fonction `renvoie` des résultats.
En revanche, quand vous voulez testez la fonction, rien ne vous interdit d'afficher les résultats qu'elle renvoie pour voir si cela correspond à vos attentes, mais faites-le *en dehors du corps de la fonction*.

☐ 35. Un nombre entier est dit *parfait* s'il est égal à la somme de ses diviseurs, outre lui-même. Par exemple, 6 est parfait car 6=1+2+3. Écrire une fonction renvoyant les n premiers nombres parfaits (attention, le cinquième est déjà supérieur à 33 millions). On peut bien entendu réutiliser des fonctions déjà écrites auparavant.

☐ 36. Écrire une nouvelle version de la fonction `liste_premiers`, qui utilise cette fois-ci la technique du crible d'Eratosthène : entre 1 et n, on élimine les multiples de 2 (à part 2), puis les multiples de 3 (à part 3), etc, de sorte que les entiers restants à la fin soient premiers. Il peut être intéressant d'utiliser le type `set` ici.

☐ 37. Les trois exercices qui viennent portent sur la conjecture de Collatz et sont à traîter dans le même fichier. En partant d'un entier $a$ de départ, on définit une suite d'entiers en obtenant chaque nouveau terme à partir du précédent soit en divisant le précédent par 2 s'il est pair, et sinon en le multipliant par 3 et ajoutant 1 s'il est impair.
Exemple : a=3, la suite est 3 10 5 16 8 4 2 1 4 2 1 4 2 1 ...
Exemple : a=7, la suite est 7 22 11 34 17 52 26 13 40 20 10 5 16 8 4 2 1 4 2 1 ...
Écrire une fonction qui, à partir d'un entier initial, renvoie la liste des valeurs successives de la suite jusqu'à ce que 1 soit atteint la première fois (le dernier élément de la liste est donc toujours 1).

☐ 38. La conjecture de Collatz (ou problème de Syracuse) affirme que quel que soit l'entier $a$ que l'on choisisse au départ, on finit toujours par arriver à 1. Écrire une fonction qui, en appelant la fonction précédente, va vérifier si la conjecture est vraie pour $a$ de 2 à n, où n est un paramètre de la fonction. Vérifiez pour $n = 10000$. Remarque : si tout se passe bien, la fonction doit juste se terminer et renvoyer 'True' par exemple. Sinon, c'est qu'on sera entré dans une boulce infinie.

☐ 39. On appelle "temps de vol" de l'entier $a$ le nombre d'étapes pour arriver de $a$ jusqu'à 1. Le temps de vol de 1 est 0, le temps de vol de 3 est 7. Écrire une fonction qui renvoie, étant donné un paramètre $n$, un dictionnaire contenant les temps de vol pour chaque entier de 2 à n.

☐ 40. Déterminez quel entier entre 1 et 10000 a le plus grand temps de vol (réponse : 6171, temps de vol 261).

☐ 41. L'altitude maximale de $a$ est la plus grande valeur par laquel passe $a$ au cours de son vol. Déterminez quel entier entre 1 et 10000 a la plus grande altitude maximale (réponse : 27114424, atteinte pour a=9663).

☐ 42. On représente une carte à jouer sous la forme $[v, x]$ où $v$ est la valeur de la carte (de 2, 3, 4,... à 11, 12, 13, 14 pour valet, dame, roi, as, et $c$ est "pi", "co", "ca" ou "tr" pour la couleur. Écrire une fonction qui, étant donné une main de 5 cartes (typiquement, un set), détermine la meilleure figure possible (brelan d'as par les rois, poker de 4 et un as, deux paires de Rois et 6 et un valet, etc.). On peut utiliser des dictionnaires pour les correspondances entre valeurs et chaînes de caractères ( as $\rightarrow$ 14 et 14 $\rightarrow$ as). Et si la main fait plus de 5 cartes, cela fonctionne ?