Piscine Python

D00

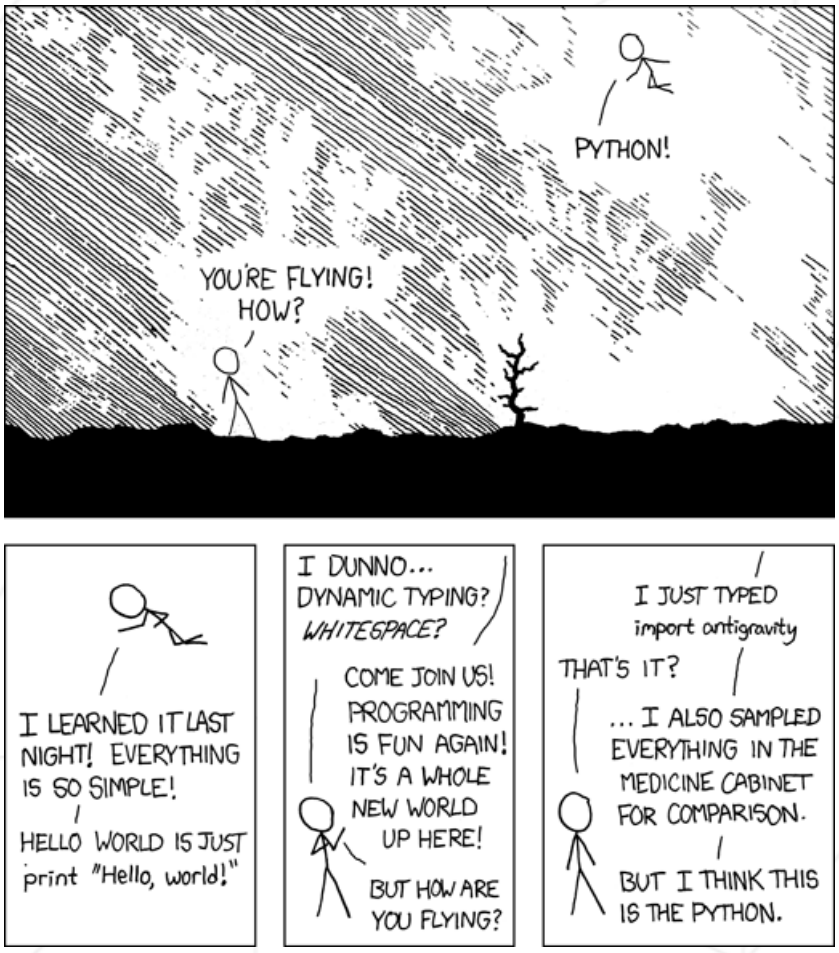
pedago@le-101.fr

Résumé: Aujourd'hui, nous allons découvrir une première partie des bases syntaxiques et sémantiques de Python

Table des matières

Chapitre I

Préambule



Source : https ://xkcd.com/353/

Chapitre II

Consignes

- Seule cette page servira de référence : ne vous fiez pas aux bruits de couloir.
- La version de Python utilisée pour cette piscine est la 3.7.
- Aucun import autorisé, à l'exception de ceux explicitement mentionnés dans la section 'Fonctions Autorisées' du cartouche de chaque exercice.
- Les exercices sont très précisément ordonnés du plus simple au plus complexe. En aucun cas un exercice complexe ne pourra être pris en compte si un exercice plus simple n'est pas parfaitement réussi.
- Vous devez suivre la procédure de rendu pour tous vos exercices : seul le travail présent sur votre dépot GIT sera évalué en soutenance.
- Vos exercices seront évalués par vos camarades de piscine, et parfois peut-être par des tuteurs. Effectuez vos évaluations avec sérieux et rigueur !
- Vous ne devez laisser dans votre répertoire aucun autre fichier que ceux explicitement spécifiés par les énoncés des exercices.
- Lorsque vous devez obtenir une sortie précise dans vos programmes, il est bien entendu interdit d'afficher une sortie précalculée au lieu de réaliser l'exercice correctement.
- Vous avez une question ? Demandez à votre voisin(e) de droite. Sinon, essayez avec votre voisin(e) de gauche.
- Votre manuel de référence s'appelle Google / Internet / la doc Python ....
- N'hésitez pas à utiliser le slack mis à votre disposition pour discuter de la piscine !
- Lisez attentivement les exemples. Ils pourraient bien requérir des choses qui ne sont pas autrement précisées dans le sujet...
- Ce qui n'est pas demandé n'est pas à faire (sauf dans les Bonus)

Chapitre III

Introduction

Aujourd'hui vous allez découvrir le langage python.

Nous travaillerons avec la version 3.7, si elle n'est pas déjà installée sur votre poste, ouvrez un terminal et tapez simplement la commande `brew install python3`.

Pour réaliser avec aisance vos rendus de piscine, il vous faudra maîtriser l'utilisation basique d'un terminal et d'un repo git. En effet, pour faire évaluer vos exercices, vous devrez les push sur notre fameuse **Vogsphere**.

Pour ceux qui auraient besoin d'un rappel à ce sujet, n'hésitez pas à vous référer aux vidéos et à la **cheat sheet** fournie dans les ressources du day avant de commencer vos exercices.

Pour tester vos bouts de code à la volée n'hésitez pas à utiliser la **console interactive python**. Vous pouvez la lancer en tapant simplement `python3` dans votre terminal. Vous pouvez aussi installer `ipython` en tapant la commande `pip install ipython` pour ceux qui préfèreraient une version plus user-friendly. Vous pouvez ensuite le lancer avec la commande `python -m IPython`.

Si vous ne savez pas encore quoi taper dans votre console, on vous recommande chaudement la fonction `help()` ;)

Chapitre IV

Exercice 00 : Forty two

101	Exercice : 00
Parce que 42	
Dossier de rendu : $ex00/$	
Fichiers à rendre : `forty_two.py`	
Fonctions Autorisées : `n/a`	

- Créez un script nommé `forty_two.py`.
- Dans ce script, déclarez une variable `addition`, assignez-lui comme valeur l'expression `4 + 2`.
- Déclarez une autre variable `concatenation`, assignez lui comme valeur l'expression `"4" + "2"`.
- Faites maintenant en sorte que votre programme affiche la valeur de `addition`, un retour à la ligne, puis la valeur de `concatenation`.
- Admirez la sortie de votre programme.

Chapitre V

Exercice 01 : You're my type

101	Exercice : 01
You're just my type of variable	
Dossier de rendu : $ex01/$	
Fichiers à rendre : `types.py`	
Fonctions Autorisées : `n/a`	

Créez un script nommé `types.py` dans lequel vous définirez une fonction `my_var`. Dans cette fonction, déclarez 9 variables de types différents et imprimez-les sur la sortie standard avec leur type.
La sortie de votre script doit ressembler exactement à ça :

```
$> python3 types.py
42 est de type <class 'int'>
42 est de type <class 'str'>
quarante-deux est de type <class 'str'>
42.0 est de type <class 'float'>
True est de type <class 'bool'>
[42] est de type <class 'list'>
{42: 42} est de type <class 'dict'>
(42,) est de type <class 'tuple'>
set() est de type <class 'set'>
$>
```

Il est bien entendu **interdit** d'écrire explicitement les types de vos variables dans les prints votre code.

C'est le moment idéal pour commencer à vous renseigner un peu sur l'utilité de ces différents types de variables.

Chapitre VI

Exercice 02 : Top 5

1E 101	Exercice : 02
Pour notre playlist	
Dossier de rendu : $ex02/$	
Fichiers à rendre : `top.py`	
Fonctions Autorisées : `n/a`	

Si vous avez suivi nos conseils, vous connaissez maintenant les variables de type `list` et `string`.

Ça tombe bien, cet exercice consiste simplement à :

- Déclarer une liste de 5 chaînes de caractères, chacune étant le nom d'un de vos 5 morceaux de musique favoris.
- Afficher la liste telle quelle sur la sortie standard.

Chapitre VII

Exercice 03 : En boucle

101	Exercice : 03
	Le classement
Dossier de rendu : $ex03/$	
Fichiers à rendre : `loop.py`	
Fonctions Autorisées : `n/a`	

Pour rendre votre top de l'exercice 02 un peu plus clair, on veut l'afficher comme sur l'exemple suivant :

```
$> python3 loop.py
My favorite title 1 is Emerald Rush.
My favorite title 2 is Raingurl.
My favorite title 3 is Crazy In Love.
My favorite title 4 is Gare du Nord.
My favorite title 5 is The Message.
$>
```

Vous devez obtenir ce résultat en itérant sur la liste que vous aurez reprise de votre ex02, et à l'aide d'une boucle `for`.

`enumerate`

Chapitre VIII

Exercice 04 : Pass me the args please

101	Exercice : 04
Arguments	
Dossier de rendu : $ex04/$	
Fichiers à rendre : `args.py`	
Fonctions Autorisées : `import sys`	

On va maintenant apprendre à écrire un script un peu plus interractif, qui a un comportement variable en fonction des arguments qui lui sont passés.

Votre script `args.py` doit afficher en sortie les arguments qui lui sont passés et leur nombre total, en excluant le script lui-même de la liste et du décompte final, exactement comme dans l'exemple ci-dessous :

```
$> python3 args.py Hello "Guten tag" 42 text.txt
Argument 1 is : Hello
Argument 2 is : Guten tag
Argument 3 is : 42
Argument 4 is : text.txt
I have 4 arguments
$>
```

Tous types d'arguments pourront être testés lors de la soutenance.

Dans le cas où le script est lancé sans qu'on lui passe aucun argument, il doit se rebeller et afficher uniquement un message d'erreur `Hey, give me at least one argument!`

`argv`

Chapitre IX

Exercice 05 : Pioneers

101	Exercice : 05
Un peu d'histoire	
Dossier de rendu : $ex05/$	
Fichiers à rendre : `pioneers_dict.py`	
Fonctions Autorisées : `n/a`	

Voici une belle liste de `tuples` de deux éléments, qui associent les noms de grands pionniers et pionnières de la tech à leur année de naissance :

```
dates = [
    ('Alan Turing' , '1912'),
    ('Ada Lovelace' , '1815'),
    ('Tim Berners-Lee' , '1955'),
    ('Grace Hopper' , '1906'),
    ('Denis Ritchie' , '1941'),
    ('Richard Stallman' , '1953'),
    ('Guido van Rossum' , '1956'),
    ('Radia Perlman' , '1951'),
    ('Larry Page' , '1973'),
    ('Mary Kenneth Keller' , '1913'),
    ('Dorothy Vaughan' , '1910'),
    ('Annie Easley' , '1933'),
    ('Ian Murdock' , '1973'),
    ('Carol Shaw' , '1955'),
    ('Aaron Swartz' , '1986'),
    ('Hedy Lamarr' , '1914'),
    ('Katherine Johnson' , '1918'),
    ('Jimmy Wales' , '1966'),
    ('Jean E. Sammet' , '1928'),
    ('Charles Babbage' , '1791'),
    ('Margaret Hamilton', '1936'),
    ('Vint Cerf' , '1943'),
]
```

Il vous faut d'abord copier cette liste telle quelle dans votre programme afin de pouvoir l'exploiter.

Votre mission consiste alors à transformer ces données en un joli dictionnaire, qui aura des dates comme clefs et les noms correspondants comme valeurs.

Vous devez ensuite afficher ce dictionnaire selon le formatage suivant :

```
1933 : Annie Easley
1986 : Aaron Swartz
1913 : Mary Kenneth Keller
1973 : Larry Page, Ian Murdock
1791 : Charles Babbage
...
```

L'ordre final ne sera pas necessairement identique a l'exemple, et c'est un comportement normal. Savez-vous pourquoi ?

Chapitre X

Exercice 06 : Colors

101	Exercice : 06
Colors	
Dossier de rendu : $ex06/$	
Fichiers à rendre : `colors.py, colors.txt`	
Fonctions Autorisées : `n/a`	

Vous avez dans les ressoures en annexe de ce sujet un fichier `colors.txt` contenant 170 noms de couleurs séparés par des virgule.

Concevez un script Python nommé `colors.py` dont le rôle est d'ouvrir le fichier `colors.txt`, lire les couleurs qu'il contient, et les afficher sur la sortie standard, une par ligne et sans virgule.

`open, read`

Chapitre XI

Exercice 07 : Bonjour Albert

101	Exercice : 07
Parce qu'Albert est un script bien élevé	
Dossier de rendu : $ex07/$	
Fichiers à rendre : `albert_says_hi.py`	
Fonctions Autorisées : `n/a`	

Vous avez déjà écrit un script qui engueule l'utilisateur lorsqu'on ne lui passe aucun argument (cf. ex05). Ici on va faire un programme un peu plus poli, qui prends les devants et demande explicitement un input.

Votre script sera nommé `albert_says_hi.py`
Au lancement, il demandera à l'utilisateur : `Hello, my name is Albert, what's yours?` (la question sera suivie d'un retour à la ligne).
L'utilisateur pourra alors taper son nom, et une fois le nom envoyé, le programme répondra `Hi X, nice to meet you!` (où "X" est bien entendu remplacé par le nom que l'utilisateur aura donné...)

`input`

Chapitre XII

Exercice 08 : What a useful bot

101	Exercice : 08
Birth Year Bot	
Dossier de rendu : *ex*08/	
Fichiers à rendre : `birth_year.py`	
Fonctions Autorisées : `n/a`	

- Écrivez un script qui, lorsqu'on lui donne un nom de personnalité historique de la tech, renvoie son année de naissance.
- Votre programme doit connaitre les personnalités de l'ex04, vous pouvez réutiliser la liste fournie.
- Au lancement, le bot doit se présenter et demander à l'utilisateur de lui donner le nom de la personnalité dont il souhaite connaitre l'année de naissance.
- Il doit répondre par une phrase qui reprend le nom et affiche l'année de naissance associée.
- Si le bot ne connait pas la personnalité demandée il doit renvoyer un message d'erreur adapté.
- Votre programme doit être indifférent à la casse de l'input utilisateur. Dans sa réponse le nom doit cependant être écrit comme dans la liste fournie à l'ex04.

`casefold()`

Chapitre XIII

Bonus

101	Exercice : 09
An even more useful bot	
Dossier de rendu : $ex09/$	
Fichiers à rendre : `birth_year_v2.py`	
Fonctions Autorisées : `Tout ce que vous voulez`	

La même chose que l'exercice précédent. Mais en mieux.

Attention, pour que ces bonus soient pris en compte il faudra que l'ex08 soit parfaitement réalisé.

Quelques idées :

- Rendez le programme plus exhaustif en rajoutant vos petits favoris dans la liste de personnalités.
- Affinez les messages d'erreur.
- Lorsque la personnalité n'est pas connue, au lieu de quitter après le message d'erreur, relancez l'utilisateur en lui proposant de demander un autre nom.
- Affichez un usage avec par exemple la liste des personnalités disponibles.
- Prévoyez une façon cohérente de quitter le script en toutes circonstances.
- Gratifiez votre programme d'un humour désopilant.
- Donnez directement l'âge (ou une approximation de l'âge) de la personne plutôt que son année de naissance.

Surprenez-nous, les possibilités sont multiples.