**Lien vers le tutoriel :** [**https://tutorial.djangogirls.org/fr**](https://tutorial.djangogirls.org/fr)

**Liens utiles :** [**https://docs.python.org/3/tutorial/datastructures.html**](https://docs.python.org/3/tutorial/datastructures.html)

**Commandes utilisées :**

* Cd changer répertoire
* Ls  lister
* Rm remove
* La fonction str convertit des choses en chaînes de caractères
* La fonction int convertit des choses en entiers (chiffres)
* print(name) Python va "imprimer" le contenu de la variable sur l'écran, sans les guillemets
* elif (sinon si) et else (sinon)

« Vous avez peut-être remarqué que parfois, on appelle la fonction avec . en la plaçant après la chaîne de caractères (comme "Ola".upper()) alors qu'à d'autres moment, on appelle d'abord la fonction puis la chaîne de caractères entre parenthèses ? Il s'avère que dans certains cas, les fonctions appartiennent à des objets (c'est le cas de upper()) et qu'elles ne peuvent être appliquées qu'à des chaînes de caractères. Dans ce cas, on appelle la fonction une méthode. D'autres fois, les fonctions n’appartiennent à rien de particulier et peuvent être utilisées sur différents types d'objets (c'est le cas de len()). C'est pour ça que nous passons "Ola" comme argument à la fonction len. »

**Résumé 1 :**

OK, assez parlé de chaînes de caractères. Jusque-là, nous avons découvert :

le prompt - taper des commandes (du code) dans le prompt Python donne des réponses dans Python

les nombres et les chaînes de caractères - dans Python, les nombres sont utilisés pour faire des calculs, et les chaînes de caractères pour manipuler du texte

opérateurs - comme + et \* qui combinent des valeurs pour en obtenir de nouvelles

les fonctions - comme upper() et len() qui effectuent des actions sur les objets.

**Variables**

Il existe un concept super important en programmation : les variables. Une variable, c'est juste un nom pour quelque chose que l'on aimerait utiliser plus tard. Les programmeurs⋅euses utilisent des variables pour stocker des données, rendre leur code plus lisible, et pour ne pas avoir à se rappeler de ce que sont les choses.

Disons que nous aimerions créer une variable appelée name :

>>> name = "Ola"

**NameError.**

Python vous donne cette erreur quand vous essayez d'utiliser une variable qui n'a pas encore été définie. Si vous rencontrez cette erreur par la suite, vérifiez dans votre code que vous n'avez pas fait une faute de frappe dans une variable.

**Les listes**

Python possède tout un tas d'autres types d'objets. Nous allons maintenant vous présenter un type appelé listes. Les listes sont exactement ce que vous pensez qu’elles sont : des objets qui sont des listes d’autres objets :)

Allez-y, créez une liste :

>>> []

[]

**Dictionnaires**

Un dictionnaire est un peu comme une liste. Cependant, nous utilisons des clefs plutôt que des indices pour accéder aux valeurs. Une clef peut être n'importe quelle chaîne de caractère ou n'importe quel nombre. La syntaxe pour définir un dictionnaire vide est la suivante :

>>> {}

{}

Dictionnaire ou liste ?

* Vous avez besoin d'une suite ordonnée d'éléments ? Utilisez une liste.
* Vous avez besoin d'associer des valeurs à des clefs, de manière à pouvoir les retrouver efficacement (par clef) par la suite ? Utilisez un dictionnaire.

**Résumé 2 :**

C'est super ! Vous savez plein de choses sur la programmation maintenant. Dans cette partie, vous avez appris :

* les erreurs - vous savez maintenant comment lire et comprendre les erreurs qui apparaissent quand Python ne comprend pas l'une de vos commandes
* les variables - des noms pour les objets qui vous permettent de coder plus facilement et de rendre votre code plus lisible
* les listes - des listes d'objets stockés dans un ordre particulier
* les dictionnaires - des objets stockés sous forme de paires clef-valeur

**Comparer**

Vous vous demandez probablement pourquoi nous avons mis deux signes == côte à côte pour savoir si deux nombres étaient égaux ? On utilise déjà = pour assigner des valeurs aux variables. Du coup, il faut toujours, oui toujours, mettre deux == si vous voulez savoir si deux choses sont égales. Nous pouvons aussi dire que certaines choses ne sont pas égales à d'autres Pour cela, nous utilisons le symbole !=, comme illustré dans l'exemple ci-dessus.

**Booléen**

Au passage, vous venez de découvrir un nouveau type d'objets en Python. On l'appelle Booléen. C'est probablement le type le plus simple qui existe.

Il n'y a que deux objets Booléens : - True (vrai) - False (faux)

Pour que Python comprenne qu'il s'agit d'un Boléen, il faut toujours l'écrire True (première lettre en majuscule, les autres en minuscule). true, TRUE, tRUE ne marchent pas -- seul True est correct. (Et c'est aussi vrai pour False.)

Les Booléens aussi peuvent être des variables ! regardez :

>>> a = True

>>> a

True

Vous pouvez aussi faire ça :

>>> a = 2 > 5

>>> a

False

**Résumé 3**

Avec ces trois derniers exercices, vous avez appris :

Comment comparer des choses - en Python, vous pouvez comparer des choses avec >, >=, ==, <=, < et avec les opérateurs and, or

Booléen - un type d'objet qui n'a que deux valeurs possibles : True et False

Comment sauvegarder des fichiers - stocker votre code dans des fichiers pour pouvoir écrire des programmes plus longs.

if...elif...else - des instructions que vous permettent de n'exécuter du code que dans certaines conditions.

**Vos propres fonctions !**

Vous vous souvenez des fonctions comme len() que vous pouvez exécuter en Python ? Et bien, bonne nouvelle : vous allez apprendre à écrire vos propres fonctions!

Une fonction est un ensemble d'instructions que Python va exécuter. Chaque fonction en Python commence par le mot-clef def. On lui donne un nom, et elle peut avoir des paramètres. Commençons par quelque chose de facile. Remplacer le code de python\_intro.py par ceci :

def hi():

print('Hi there!')

print('How are you?')

hi()