Python 1 dans Visual Studio Code

«Open folder...» -> dans un répertoire vide...

Ouvrir un Terminal dans Visual Studio Code.

Taper
```
python
```

Calculs de base : + - * /

Exposants: ** [^ = ou exclusif, mais inutile]

Modulo: % vous donne le nombre entier qui reste : utile pour identifier des nombres pairs (ou impairs)

Variables. Ce sont des contenants. On y place du contenu avec le symbole =

```
a = 60
b = 15
c = a + b
```

Convention pour les noms de variables:
- Idéalement toujours en minuscule : url
- peut avoir des nombres : url2
- jamais de caractères accentués
- majuscules juste quand on a un nom complexe: choix, premierChoix, nomPrenom
- ne JAMAIS commencer par une majuscule un nom de variable -> ce sera une erreur

Quitter python dans le terminal:
```
exit()
```

Votre premier script dans un nouveau document

On va écrire un premier script.
«New file» -> donnez-lui le nom que vous voulez... mais terminez par **.py** (document python)

Première ligne:
```
# ©2020 - Jean-Hugues Roy
# coding: utf-8
```

Commentaires :
lignes dont on ne tient pas compte. -> commande-backslash
Mettez-en dans vos scripts pour expliquer ce que vous tentez de faire

Entrez:
```
a = 635
b = 27354
c = b/a
c
```

Dans Terminal, si vous n'y êtes pas déjà, rendez-vous dans le même répertoire que celui dans lequel vous avez sauvé votre script.

Pour faire rouler un script, commande:
```
python nomduscript.py
```

Ne produit rien.
Changez
```
c
```
par
```
print(c)
```

print() est une fonction. On va en voir plusieurs.

7 grands types de variables (il y en a 23, mais voici les principales):

Nombres entiers.
```
a = 4
print(a)
print(type(a))
```

Nombres décimaux (à virgule flottante).
```
b = 5.88
print(b)
print(type(b))
```

Valeur booléenne.
```
c = True (ou False)
print(c)
print(type(c))
```

Texte (qu'on appelle aussi des « chaînes de caractères »).
Guillemets anglais ou apostrophe.
```
d = "Je prends mon temps parce que je suis rendu vieux"
d = 'Je prends mon temps parce que je suis rendu vieux'
print(d)
print(type(d))
```

Habitude de prendre guillemets anglais pcq on a bcp d'apostrophes dans la langue française.
```
d = "J'prends mon temps"
```
Et si on a les deux? -> notion de caractère d'échappement
```
d = "J'prends mon temps: \"Chu rendu vieux.\""
```

On peut « additionner » texte.
```
d = "http://"
e = "uqam.ca"
d = d + e
```

Attention: des nombres peuvent être du texte.
```
d = "2018"
```

None.
Il arrive qu'on essaie de mettre quelque chose dans des variables et que ça ne fonctionne pas.
Dilemme existentiel, mais en informatique, le néant existe.
```
e = None
print(e)
print(type(e))
```

Listes. Variable «iterable»: dans laquelle on peut faire des itérations.
Type très courant. Plusieurs valeurs, séparés par des virgules, entre crochets.
```
f = [1,2,3,4,5,6,7,8,9]
```
On peut additionner les listes.
Elles peuvent se multiplier par des nb entiers
Les listes peuvent contenir tous les types de variables. Incluant des listes.
```
f = [1,'deux',3.33,[4,5,6],None,False]
```

Dictionnaires.
Autre type courant. Plusieurs paires de clé:valeur, séparées par virgules, entre des accolades.
```
g = {"année":1896,"ville": "Athènes", "pays":
["Grèce","France","Italie"]}
h = {"année":1900,"ville": "Paris"}
```

Écrire:
```
type(a)
```
Et b, c, d, etc. Pour confirmer le type de ces variables.

Manipuler texte.

Créer variable
```
url = 'http://ici….'
```

len()
Fonction pour trouver longueur d'une chaîne de caractères.
```
print(len(url))
```

Notion d'index.
bit.ly/jhroy0

En informatique, le premier élément d'une chaîne de caractères ou d'une liste ou d'un dict -> **0**
Le dernier -> **-1**

Accéder à des éléments dans une chaîne de caractères en utilisant les crochets:
```
print(url[0])
print(url[-1])
```

Accéder à des intervalles -> le dernier nombre est exclu:
```
print(url[10:20])
```

Accéder aux x premiers (ici, les 3 premiers -> éléments 0, 1, 2):
```
print(url[:3])
```

Accéder aux x derniers:
```
print(url[-10:])
```

Pour prendre des éléments à partir d'une chaîne de caractères?
```
print(url['//':])
```
Ceci ne marche pas pcq l'index est toujours un nombre.

.find()
Méthode pour trouver l'index d'un caractère, ou de caractères, donné.s
```
print(url.find('//'))
print(url.find('.ca'))
```

Résultat est un nombre.
Trouver dernier « / »?

.rfind()

Méthode pour trouver l'index d'un caractère, ou de caractères, donné.s, mais à partir de la fin.

```
print(url.rfind(‘/‘))
```

Donc, comment extraire juste la fin de l'URL?

```
print(url[url.rfind('/')+1:])
```

input()

Fonction pour demander à l'utilisateur d'entrer du texte.

```
nom = input("Votre nom? ")
print("Bonjour " + nom + « !!! »)
```

.format()

Méthode pour imprimer des variables.

```
print("Votre nom? ")
nom = input()
print("Votre prénom? ")
prenom = input()
print("Bonjour {} {}".format(prenom,nom))
```

.upper() / .lower() / .capitalize()

Méthodes pour changer la casse des textes.

.strip()

Méthode pour retrancher des caractères inutiles.
D'abord des espaces.

Créez variable texte:

```
t = ‘                        Tu vas te faire mal à un moment donné
‘
print(t + ‘.’)
```

Éliminez les espaces.

```
z = t.strip()
print(z + ‘.’)
```

Fonctionne avec d'autres caractères.

```
prix = ‘5,000$’
prix2 = prix.strip(‘$’)
```

Mais essayez avec la virgule.
Marche pas -> .strip() ne retranche que les caractères au début et à la fin d'une chaîne. Et les retranche aussi, ne les remplace pas par autre chose.

.replace()

Méthode pour remplacer des caractères.
Donc:

```
prix3 = prix2.replace(‘,’,’’)
print(prix3)
```

Changements de type.

Ici, prix3 est un texte:

```
print(type(prix3))
```

```
prix4 = int(prix3)
print(prix4)
print(type(prix4))
```

Tous les changements de type qui ont du bon sens sont possibles.
Pour retransformer prix4 en chaîne de caractères:

```
prix5 = str(prix4)
```

Listes

Créez un nouveau document.
Donnez-lui le nom qui vous plaît.

Créez votre **première liste**.

```
l1 = [22,34,40,55,66,73,82,91]
```

Les listes ressemblent beaucoup aux chaînes de caractères.

len()

Fonction pour compter le nombre d'éléments dans une liste.

```
print(len(l1))
```

index

Pareil comme les caractères.

```
print(l1[0])
```

range()

Fonction qui crée étendue de nombres.
Créez une 2e liste (l2).
Commencer à zéro n'est pas nécessaire...

```
l2 = range(0,100)
print(l2)
```

L'étendue est créée.
Mais faut en changer le type.

```
l2 = list(range(0,100))
print(l2)
```

Vous voyez que le dernier nb n'est pas inclus.
Sens inverse?

reversed()

```
l3 = list(reversed(range(200,2000)))
print(l3)
```

pas ou **step**

Possible de créer une étendue avec des intervalles.
On a quoi, ici?

```
l4 = list(range(1896,2024,4))
```

```
print(l4)
```

Liste de mots.
Créez 5e liste avec des chaînes :

```
l5 = ['Canadiens de Montréal', 'Black Hawks de Chicago', 'Red Wings de Détroit', 'Maple Leafs de Toronto', 'Rangers de New York', 'Bruins de Boston']
```

Listes de listes.
Créez une 6e liste composée de quelques-unes des listes précédentes.

```
l6 = [l1,l2,l4,l5]
print(l6)
```

Quand on a ces structures complexes, on peut accéder à des sous-éléments ainsi:

```
print(l6[2])
print(l6[2][10])
```

Pour faire afficher seulement « Canadiens de Montréal », on entrerait quoi?

```
print(l6[3][0])
```

min() et max()

Fonctions pour afficher plus petit ou plus grand élément d'une liste.

```
print(min(l4))
```

Fonctionne aussi avec texte.

```
print(max(l5))
```

.append()

Méthode pour ajouter éléments à liste.
Créer d'abord une liste vide.

```
l7 = []
print(l7)
l7.append('Quelque chose')
print(l7)
```

Boucles

Allez maintenant chercher une liste ici:
http://bit.ly/jhroycinema

Liste de tous les films québécois dans Wikipédia.

Téléchargez-le et placez-le dans votre répertoire.

Écrivez l'entête

```
# encoding:utf-8
```

C'est une immense liste.
Donnez-lui un nom.
Donnez toujours à vos variables (contenants) des noms qui en décrivent bien le contenu.

On va donc l'appeler, au pluriel:

```
films
```

Faites, pour vérifier:
```
print(films)
```

-> Combien de films dans cette liste?
```
print(len(films))
```

-> Quel est le 1001e film dans cette liste?
```
print(films[1000])
```

En informatique, on va très souvent faire des boucles pour traiter des masses d'information un item à la fois. Tous les langages informatiques ont des boucles.

En python, voici la syntaxe.

for variable in liste:
    on fait quelque chose

L'indentation est importante. Sinon, ça ne fonctionne pas.
Utilisez tabulateur, ou 2 espaces ou plus.

Donc, ici, on va écrire:

```
for film in films:
    print(film)
```

Compteur.

Ajoutez une variable **avant la boucle** et donnez-lui la valeur de zéro.
```
n = 0
```

Dans la boucle, augmentez-en la valeur de 1.
```
n = n + 1
```

Mais les pros écrivent ça ainsi:
```
n += 1
```

```
-=
```
va faire la même chose que:
```
n = n - 1.
```

Et imprimez cette variable en même temps que chaque film:
```
print(n,film)
```

.split()

Méthode pour briser des chaînes de caractères en fonction d'un ou de plusieurs caractères donnés.
Par exemple, ici, chaque film contient trois informations intéressantes.
On va briser la variable **film** en trois en fonction du point-virgule.
```
film = film.split(';')
```

La variable film va être transformée en une liste avec autant d'éléments qu'il y a d'occurrences du caractère recherché +1.

->EXERCICE

Écrivez pour chaque film un **print()** avec deux phrases complètes disant «Le 565e film de la liste est Eldorado. Il a été réalisé par Charles Binamé en 1995. » Utilisez la méthode **.format()**.

-> EXERCICE

Extrayez le nombre de mots de cette chaîne de caractères:
« Idées, mercredi 4 mars 2015 1259 mots, p. A7 ».

Boucle dans un range().

Il est aussi possible de faire des boucles avec un intervalle.
Ici, la syntaxe sera:

```
for i in range(1000,10000):
  print(i/5)
```

Boucle dans une boucle.

Le meilleur exemple, est pour générer les codes des médecins.

```
for annee in range(1930,2021):
   for code in range(1001,2000):
         permis = str(annee)[2:] + str(code)[1:]
         print(len(permis),permis)
```