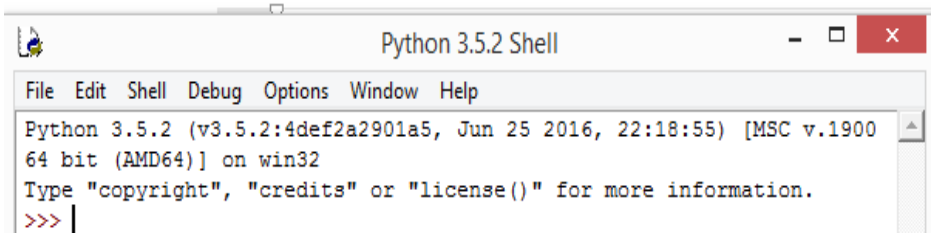
Niveau Zombie

Python: Initiation

Fiche n°1

Si vous lancez directement Python vous trouverez un écran noir pas très attrayant mais ce n'est pas cela qui nous intéressera. Cherchez IDLE dans votre menu démarrer et vous verrez apparaitre le Shell blanc de python (ci-dessous).



Cette interface nous permettra de lancer notre programme et de débugger en cas d'erreur. Pour commencer votre premier programme, cliquez sur ***File*** puis ***File New***.

Un genre de traitement de texte vierge s'ouvre et c'est là que nous pourrons taper notre code.

Premier code :

```
nom=input('Quel est votre nom? ')
age=input('Quel est votre age? ')
naissance=2017-int(age)
print('Bienvenue {0}, vous êtes donc né en {1}'.format(nom,naissance))
```

Quelques explications avant de le lancer notre programme. Input permet de demander une information à l'utilisateur.

La première ligne demande donc le nom et python l'attribue à la variable nom.

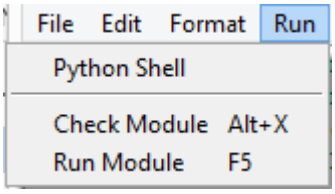
La deuxième ligne demande l'âge et l'attribue à la variable âge.

La troisième créé une variable nommée naissance qui soustrait l'âge à l'année en cours.

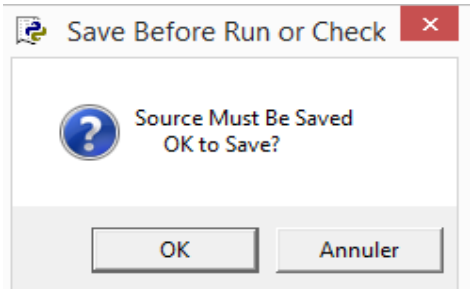
Remarque : **int** permet de transformer l'information renseignée par l'utilisateur en entier (integer)

La dernière affiche alors le message où {0} fait référence à la première variable indiquée dans le format en fin de phrase, c'est-à-dire nom et {1} fait référence à la seconde donc naissance.

Lançons à présent le programme pour cela deux possibilités, soit via la touche **F5** de votre clavier, soit en cliquant sur ***Run*** dans la barre supérieure.



Python vous demande alors de sauvegarder votre programme avant de pouvoir le tester.
Sélectionnez le dossier de votre choix, donnez un nom au fichier sans oublier de finir par **.py**



Maintenant que nous savons interagir avec l'utilisateur, nous allons voir comment faire réaliser à python des tâches qui peuvent paraitre répétitives, c'est ce que l'on appelle les boucles.

LES BOUCLES :

```
for i in range(10):
    print('Ca fait {0} tour que je fais'.format(i))
```

Nous choisissons un nom pour une variable muette qui représentera le nombre d'itérations à faire réaliser à notre boucle, ici i.

Nous souhaitons faire écrire 10 lignes à python chacune indiquant la valeur de i. Lançons le programme :

```
Ca fait 0 tour que je fais
Ca fait 1 tour que je fais
Ca fait 2 tour que je fais
Ca fait 3 tour que je fais
Ca fait 4 tour que je fais
Ca fait 5 tour que je fais
Ca fait 6 tour que je fais
Ca fait 7 tour que je fais
Ca fait 8 tour que je fais
Ca fait 9 tour que je fais
```

Python écrit bien 10 lignes mais il s'arrête à i = 9 car sans indication contraire python commence toujours à compter à 0.

Nous avons vu la boucle **FOR** (POUR) mais il ne s'agit pas là de la seule boucle possible, nous avons également accès aux boucles **WHILE** (TANT QUE).

Tapez le texte ci-contre et executez-le.

```
i=1
while i<10 :
    print('Ca fait {0} tour que je fais'.format(i))
```

Pour l'arrêter appuyez simultannéement sur les touches CTRL et C. Vous êtes tombés dans le piège de la boucle infinie, c'est-à-dire que si on ne l'arrête pas manuellement elle ne s'arrêterait jamais. En effet, ici i vaut 1 dès le début mais on ne le fait jamais évoluer. Si on veut faire évoluer i, il faut alors le faire augmenter de un à chaque tour, pour cela on peut rajouter :

```
i=i+1
```

Si vous êtes habitué d'un autre langage, on peut aussi utiliser la syntaxe suivante pour obtenir la même chose :

```
i+=1
```

On relance alors le programme et on remarque que ce coup-ci la valeur évolue bien en commençant à 1 mais s'arrête à nouveau à 9. En effet, la condition de boucle est que i doit rester inférieur à 10 (strictement) donc 10 ne satisfait pas la condition de boucle. Pour s'arrêter à 10 il faudra alors modifier la condition de la boucle par :

```
while i<11 :
```

On se rend compte que les conditions sont primordiales, nous allons donc les étudier plus précisément.

LES CONDITIONS

Classiquement on parle des conditions IF – ELSE (Si – Sinon). Prenons l'exemple suivant :

```
valeur=input('Entrez un nombre ')
if int(valeur)<10 :
    print('il est plus petit que 10')
else :
    print('il est supérieur à 10')
```

Comme précédemment on demande à l'utilisateur d'entrer une valeur et on va « analyser » cette valeur, vérifier si elle est inférieure à 10 ou non.

Les tests sont « classiques », on peut vérifier si deux valeurs sont strictement supérieures, supérieures, strictement inférieures, inférieures, égales, différentes… Mais le souci habituel est la syntaxe, voici un petit tableau récapitulatif :

<	Strictement Inférieur
>	Strictement Supérieur
<=	Inférieur
>=	Supérieur
==	Egal
!=	Différent

Les conditions ne peuvent être réalisées sans comparaisons et sans variable, mais il faut être vigilant car il existe plusieurs sortes de variables.

LES VARIABLES SIMPLES

A la différence d'autres langages, python ne demande pas une déclaration du type de variable, c'est-à-dire que python « comprendra » par lui-même si la variable est un nombre, un mot ou une liste. Voyons quelques exemples :

```
mot='physique'
print('La seconde lettre est : {0}'.format(mot[1]))
```

Ici python reconnait grâce aux ' ' que la variable mot est une suite de caractères, on peut alors faire afficher le caractère en seconde position.

Ici on définit deux variables, chiffre et nombre, le premier est entier (int) et le second un décimal (float). On additionne les deux et naturellement python va transformer le résultat de l'addition en un décimal. On peut le transformer en entier en plaçant int avant le résultat.

```
chiffre=1
nombre=16.0
print(chiffre+nombre)
print(int(chiffre+nombre))
print(float(chiffre+nombre))
```

```
liste=['maison','champignon','tortue']
print(len(liste))
print(liste[0])
print(liste[1][4])
```

A présent nous créons une liste composées de trois mots mais on pourrait y mélanger les types de variables. La commande ***len*** permet de connaitre le nombre variables dans la liste.

Encore une fois la première variable est la numéro 0. Si on veut afficher la cinquième lettre du second mot de la liste, on pourra alors utiliser la dernière ligne du code.

Reprenons la liste précédente et comme une liste n'est pas une variable figée, on peut y rajouter des termes en fin de liste grâce à la fonction append.

```
liste=['maison','champignon','tortue']
print(len(liste))
liste.append('grue')
print(len(liste))
print(liste)
```

DEFIS *Réaliser un jeu du pendu.*

Réaliser un jeu de mastermind avec 4 cases et 5 couleurs (Rouge-Vert-Bleu-Jaune-Orange).