SNT

Séance 1 sur Python

Compléter la fiche au fur et à mesure puis faire les exercices

I Edupython

Pour ouvrir le logiciel edupython, ouvrez la suite MCNL (logo carré bleu et blanc) et cherchez dans la barre de recherche "edupython".

Il y a deux zones où on peut programmer en python:

- Dans la console en bas, vous verrez les signes ">>>", vous pouvez écrire à cet endroit une commande python et l'executer en tapant sur Entrée. De nouveaux signes ">>>" apparaissent pour la commande suivante (on ne peut pas revenir en arrière et modifier les commandes précédentes).
- Dans la zone de script en haut à droite, on peut écrire un vrai programme sur plusieurs lignes, l'enregistrer dans un fichier, etc. Pour l'executer on clique sur le bouton triangulaire vert au dessus, et le résultat de l'execution (si quelque chose est affiché par le programme ou si il y a une erreur) est affiché dans la console en bas.

On va commencer par utiliser la console et ensuite dans la partie II on utilisera la zone de script.

A Calculs

Compléter le tableau suivant en tapant les instructions au fur et à mesure.

>>>	5+3	2-9	7+3*4	(7+3)*4	3**2	20/3	20.5/3	20,5/3	20//3	20%3
résultat	8									

Par exemple : si on tape 5+3 puis sur la touche « entrée », on obtient 8.

```
>>> 5+3
8
```

1 Vérifier que l'ordre des opérations est bien respecté : 7+3*4 ≠ (7+3)*4.

2 Vérifier sur des exemples que a**b donne bien a^b.

3 Vérifier sur des exemples que a/b donne bien le quotient (arrondi) de a par b quand a et b sont des nombres entiers ou décimaux.

4 Avez-vous compris pourquoi 20,5/3 donne le couple (20, 1.6666666666666667) ?
Indication : en anglais 20.5 signifie 20,5 et 20,5 est le couple (20,5).

5 Vérifier sur des exemples que a//b et a%b donnent respectivement le quotient et le reste de la division euclidienne de deux entiers a et b ?

20	3
2	6

Le reste sera très utile dans vos futurs programmes.

En effet, pour savoir si :

- un nombre entier a est pair, il suffira de voir si le reste a%2 vaut .....
- un nombre entier a est impair, il suffira de voir si le reste a%2 vaut ..... .

B Variables, affectations

Tapez les trois instructions ci-contre.

1 n, msg et pi sont des **variables**.

2 Que signifie le signe égal ici ?

```
>>> n=7
>>> msg="Quoi de neuf ?"
>>> pi=3.14159
```

Les 3 instructions d'affectations ci-dessus ont eu pour effet chacune de réaliser plusieurs opérations dans la mémoire de l'ordinateur :

- créer et mémoriser un nom de variable (comme ici n) ;
- lui attribuer un type bien déterminé qui peut être un entier, un nombre décimal, une chaîne de caractères... (pour la variable n, c'est un entier) ;
- créer et mémoriser une valeur de la variable (pour n, ce sera 7 ici) ;
- établir un lien (par un système interne de pointeurs) entre le nom de la variable et l'emplacement mémoire de la valeur correspondante.

Les trois noms de variables sont des références, mémorisées dans une zone particulière de la mémoire que l'on appelle espace de noms, alors que les valeurs correspondantes sont situées ailleurs, dans des emplacements parfois fort éloignés les uns des autres.

Remarque : il n'y a pas besoin comme vous le voyez de déclarer les variables avec Python, on dit que c'est un typage dynamique. Ici, Python a déclaré n comme un entier (**integer**), msg comme une chaîne de caractères (**string**) et pi comme un décimal (**float**).

Compléter le tableau suivant :

- Dans la 2ème ligne et 2ème colonne, on vous demande d'écrire ce que Python affiche lorsque vous mettez print(n) puis entrée.
- Dans la dernière ligne et 2ème colonne, on vous demande d'écrire ce que Python affiche lorsque vous mettez type(n) puis entrée.

>>>	n	msg	pi
Résultat avec la fonction **print()**			
Résultat avec la fonction **type()**			

Récapitulons :

- **print(n) affiche la valeur de la variable n**

- **type(n) affiche le type de la variable n (« int » pour entier, « float » pour nombre décimal, « str » pour une chaîne de caractères)**

Tester les deux instructions suivantes :

```
>>> point_de_vie=15
>>> print(point_de_vie)
```

Qu'affiche Python ?

Écrire à présent l'instruction suivante :

```
>>> print("point_de_vie")
```

Qu'affiche Python ?

Récapitulons :
- **print(nom_de_variable)** affiche le contenu de la variable nom_de_variable.
- **print("nom_de_variable")** affiche bêtement la chaîne de caractères qui se trouve entre les guillemets.

conclusion : bien faire la distinction entre print(a) et print("a")

La fonction print() permet également d'afficher plusieurs choses, il suffit de les séparer par une virgule. Tester l'instruction suivante :

```
>>> print("point_de_vie =",point_de_vie)
```

Qu'affiche Python ?

C La fonction *input()*

La fonction input() permet à l'utilisateur de rentrer une valeur qu'elle transformera automatiquement comme une chaîne de caractères.

Tester l'instruction ci-dessous. La variable *nom* va contenir la réponse entrée au clavier par l'utilisateur.

```
>>> nom=input("Quel est votre nom ?")
Quel est votre nom ?
```

Afin de vérifier que la variable *nom* contient bien ce qui a été rentré au clavier, écrire l'instruction :

```
>>> print(nom)
```

Qu'affiche Python ?

Cette fois on souhaite rentrer un entier. Tester l'instruction ci-dessous en tapant au clavier un entier.

```
>>> age=input("Quel est votre âge ?")
Quel est votre âge ?
```

Afin de vérifier que la variable *age* contient bien ce qui a été rentré au clavier, écrire l'instruction :

```
>>> print(age)
```

Qu'affiche Python ?

Maintenant, tester l'instruction :

```
>>> print(age+age)
```

Qu'affiche Python ?

Pour comprendre ce qu'il s'est passé, taper l'instruction :

```
>>> type(age)
```

Qu'affiche Python ?

À retenir : « **input » renvoie toujours une chaîne de caractères !**

Explication de l'affichage de print(age+age) : age est donc une chaîne de caractères. Le signe plus en Python entre deux chaînes de caractères est une **concaténation**. Tester les instructions :

```
>>> a="Une conca"
>>> b="téna"
>>> c="tion"
>>> print(a+b+c)
```

Qu'affiche Python ?

Question : Du coup, comment faire pour demander à l'utilisateur un entier ou un décimal ?

Réponse : **on utilise les fonctions int() ou float() pour convertir une chaine de caractères en entier ou en décimal.**

Ainsi, pour demander à l'utilisateur un entier, on peut écrire :

```
>>> age=int(input("Quel est votre âge ?"))
Quel est votre âge ?
```

Puis tester les instructions suivantes :

```
>>> type(age)
```

Qu'affiche Python ?

Puis tester les instructions suivantes :

```
>>> print(age+age)
```

Qu'affiche Python ?

Attention : Python ne pourra pas convertir n'importe quelle chaîne de caractères en entier ou en décimal.

Exemple : tester encore l'instruction mais rentrer au clavier 3.2 par exemple.

```
>>> age=int(input("Quel est votre âge ?"))
Quel est votre âge ?
```

Python génère alors une erreur.

II Expressions booléennes (vrai/faux)

En anglais, « True » signifie « vrai » et « False » signifie « faux ».
Une variable qui ne peut contenir que True ou False est de type booléen.

Contrairement au signe « = » qui signifie « prend la valeur », **l'opérateur « == » signifie « égal à »**.
Tester l'instruction suivante :

```
>>> print(4==5)
```

Qu'affiche Python ?

Tester maintenant :

```
>>> print(7==7)
```

Qu'affiche Python ?

Pour créer un programme dans la zone de script :
Écrire le code qui peut être sur plusieurs lignes dans la zone en haut à droite d'edupython, et cliquer sur le triangle vert pour l'executer. Vous pouvez aussi enregistrer votre programme comme un fichier (fichier -> enregistrer sous).
Écrire donc ce programme puis l'exécuter.

`a=4`
`b=7`
`print(a==b)`
`a=7`
`print(a==b)`
	Affichage après exécution :

Modifier le programme précédent et le tester.

`a=4`
`b=7`
`print(a!=b)`
`a=7`
`print(a!=b)`
	L'opérateur « != » signifie « différent de ».
	Affichage après exécution :

Modifier le programme précédent et le tester.

	L'opérateur « < » signifie « strictement inférieur à ». Affichage après exécution :

`a=4`
`b=7`
`print(a<b)`
`a=7`
`print(a<b)`	

Modifier le programme précédent et le tester.

`a=4`
`b=7`
`print(a<=b)`
`a=7`
`print(a<=b)`
	L'opérateur « <= » signifie « inférieur ou égal à ».
	Affichage après exécution :

De même l'opérateur « > » signifie « strictement supérieur à » et « >= » signifie « supérieur ou égal à ».

III Instructions conditionnelles

En anglais, « if » signifie « si » et « else » signifie « sinon ».

Exercice 1 : Répondre aux questions suivantes sans taper le programme ! Après avoir répondu aux questions, on tapera le programme et on testera les réponses.

1 Selon vous, que va afficher le programme suivant ?

```
note=int(input("rentrer une note:"))
if note<8:
    print("ajourné")
else:
    if note<10:
        print("repêchage")
    else:
        print("reçu")
```

si note=7 : ……

si note=8 : ……

si note=9 : ……

si note=10 : ……

si note=11 : ……

Afin de préciser le début et la fin d'un bloc d'instructions après un « if » par exemple, dans Python, on décale les instructions après avoir mis deux points. On dit alors que l'on a indenté (=décalé) des instructions.

2 Maintenant, écrire ce programme puis tester vos réponses précédentes.

Remarquer qu'un « else » est toujours au même niveau que le « if » qui lui est associé.

Remarque : on peut très bien employer un « if » sans « else ».

Attention au décalage ! Une instruction décalée ne joue pas le même rôle qu'une instruction non décalée !

```
a=5
b=2
if a<b:
    print(a)
    print("a")
```

Que va afficher le programme suivant selon-vous ?

Vérifier votre réponse en l'exécutant :

```
a=5
b=2
if a<b:
    print(a)
print("a")
```

Que va afficher le programme suivant selon-vous ?

Vérifier votre réponse en l'exécutant :

Exercice 2 : créer un programme qui demande à l'utilisateur deux valeurs entières a et b, et qui donne la plus petite de ces deux valeurs. Par exemple si a=3 et b=2, votre programme devra afficher « La valeur la plus petite est 2 » et non pas « La valeur la plus petite est b ».
Tester bien votre programme en prenant également des nombres égaux.

Exercice 3 : créer un programme qui demande à l'utilisateur trois valeurs décimales x, y, z et qui affiche la plus grande valeur (*on pourra utiliser une variable* max qui prendra la plus grande valeur entre x et y, puis il suffira de comparer max à z). Par exemple si x=3 , y=2.5 et z=7, votre programme devra afficher « La valeur la plus grande est 7.0 ».
Tester bien votre programme en prenant également des nombres égaux.

Exercice 4 : créer un programme qui demande à l'utilisateur deux valeurs décimales a et b et qui affiche les solutions de l'équation $ax+b=0$.

Rappel : si $a\neq 0$ alors l'équation $ax+b=0$ admet comme solution $-\frac{b}{a}$.

En effet, $ax+b=0 \Leftrightarrow ax=-b \Leftrightarrow x=\frac{-b}{a}$.

Exemple : $3x+2=0$ **a pour solution** $-\frac{2}{3}\approx -0{,}6667$.

Si $a=0$ et $b\neq 0$ alors l'équation n'a pas de solution.
Exemple : $0x+2=0 \Leftrightarrow 2=0$ **donc pas de solution.**
Enfin, si $a=0$ et $b=0$ alors l'équation a une infinité de solutions.
En effet l'équation $0x+0=0$ **est vraie pour n'importe quel nombre réel x.**

- Si a=3 et b=2, votre programme devra afficher :
  La solution est x=-0.6666666666666666
- Si a=0 et b=2, votre programme devra afficher :
  L'équation n'a pas de solution
- Si a=0 et b=0, votre programme devra afficher :
  Tous les nombres réels sont solutions