PROGRAMMER EN LANGAGE INTERPRÉTÉ ORIENTÉ OBJET AVEC « PYTHON » (VARIABLES, TYPES ET OPÉRATEURS)

Objectifs de l'activité pratique :

L'IDLE de Python
Script
Variables, types et opérateurs
Programmation Orientée Objet (POO)
Objet (instance), classe, méthode et attribut
Exercices d'application
QCM

Support d'activité :

Logiciels : Portable python 2.7, Microsoft Office 2003 et Libre Office
Fichiers : Puissance4.py et QCM et exo sur Python-Les variables.htm
Internet
Ce document au format PDF en couleur

DOCUMENT GUIDE

PRÉAMBULE

Actuellement, Python en est à sa version 3. Cependant, la version 2 est encore largement utilisée.
Attention : Python 2 n'est pas compatible avec Python 3 !

Python est un langage multiplateforme, c'est-à-dire disponible sur plusieurs architectures (compatible PC, certains smartphones, ordinateur low cost Raspberry Pi...) et systèmes d'exploitation (Windows, Linux, Mac, Android pour smartphones...).
Le langage Python est gratuit, sous licence libre.
C'est un des langages informatiques les plus populaires avec C, C++, Objective-C, Java, PHP, JavaScript, Delphi, Visual Basic, Ruby et Perl (liste non exhaustive).
Avec Python vous allez pouvoir faire beaucoup de choses :

du calcul scientifique (librairie NumPy),
des graphiques (librairie matplotlib),
du traitement du son,
du traitement d'image (librairie PIL),
des applications avec interface graphique GUI (librairies Tkinter, PyQt, wxPython, PyGTK ...),
des jeux vidéo en temps réel (librairie Pygame),
des applications Web (serveur Web Zope, framework Web Django, Karrigell, framework JavaScript Pyjamas),
interfacer des systèmes de gestion de base de données (librairie MySQLdb ...),
des applications réseau (framework Twisted),
communiquer avec des ports série RS232, Bluetooth... (librairie PySerial),
...

IDE OU IDLE de PYTHON ?

IDE est un environnement de développement intégré (IDE en anglais : Integrated Development Environment).

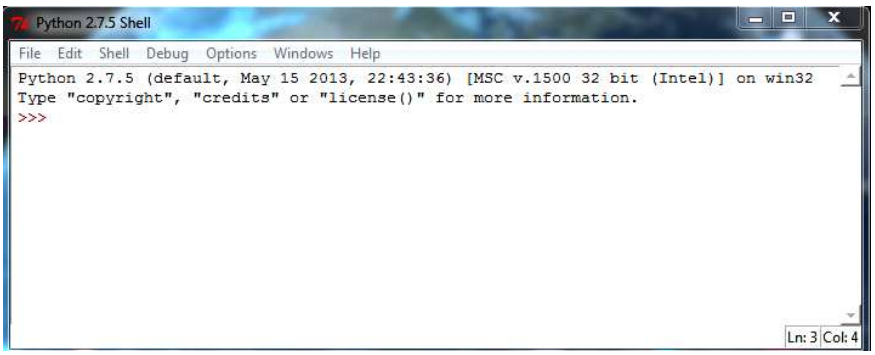
IDLE-Python propose un certain nombre d'outils :

un éditeur de texte (pour écrire le programme)
un interpréteur (pour exécuter le programme)
un débogueur (pour tester le programme)
Il existe d'autres IDLE pour Python : Eclipse/Pydev, NetBeans, Eric...

Question 1 :

Essayer de trouver à l'aide d'internet une réponse à la question suivante : Pourquoi IDLE-Python et pas IDE-Python ?

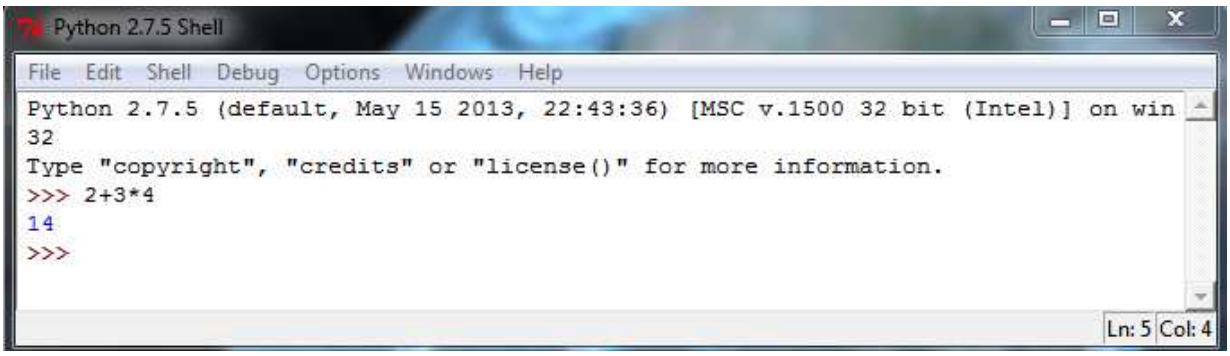
Lancer l'IDLE de Python (sur le bureau dans le dossier programmation et réseau).



L'IDLE vous permet d'entrer et de tester des commandes (**mode CLI**).
Par exemple, taper le code :

>>> **2+4*3**
Puis valider par la touche entrée.

Le résultat s'affiche :



Vous pouvez remarquer que la priorité des opérations est bien respectée.

SCRIPT sous PYTHON

Un programme est une séquence d'instructions.
Dans le cas d'un programme en langage Python, on parle souvent de script Python.

Un script se présente sous la forme d'un fichier texte avec l'extension « **.py** ».

Voici un exemple de script Python :

« **jeu_puissance4.py** »

Copier dans votre dossier personnel le fichier « **Puissance4.py** » se trouvant dans le dossier support puis ouvrir dans l'IDLE.

L'éditeur de texte s'ouvre avec le code source du script (environ 1200 lignes de code, soit plusieurs dizaines d'heures de travail pour un développeur expérimenté).

Cliquer sur **Run** puis **Run module** ou (**touche F5**).

Tester le jeu (1 ou 2 parties pas plus !).

VARIABLES, TYPES ET OPÉRATEURS

Ouvrir un nouveau script (**File/New Window** ou **CTRL+N**)

Remarque :

Penser à sauvegarder vos scripts dans votre dossier personnel.

LE TYPE « INT » (INTEGER OU NOMBRES ENTIERS)

Question 2 :

Tester et vérifier les codes ci-après.

Pour affecter (on dit aussi assigner) la valeur 17 à la variable nommée Age :

```
>>> Age = 17
```

La fonction « **print** » affiche la valeur de la variable :

```
>>> print Age
17
```

La fonction **type()** retourne le type de la variable :

```
>>> print type(Age)
<type 'int'>
```

int est le type des nombres entiers.

```
>>> # ceci est un commentaire
>>> Age = Age + 1
>>> print Age
18

>>> Age = Age - 3
>>> print Age
15

>>> Age = Age * 2
>>> print Age
30

>>> a = 6*3-20
>>> print a
-2

>>> b = 25
>>> c = a + 2*b
>>> print b, c
25 48
```

L'opérateur « **//** » donne la division entière :

```
>>> d = 450//360
>>> print d
1
```

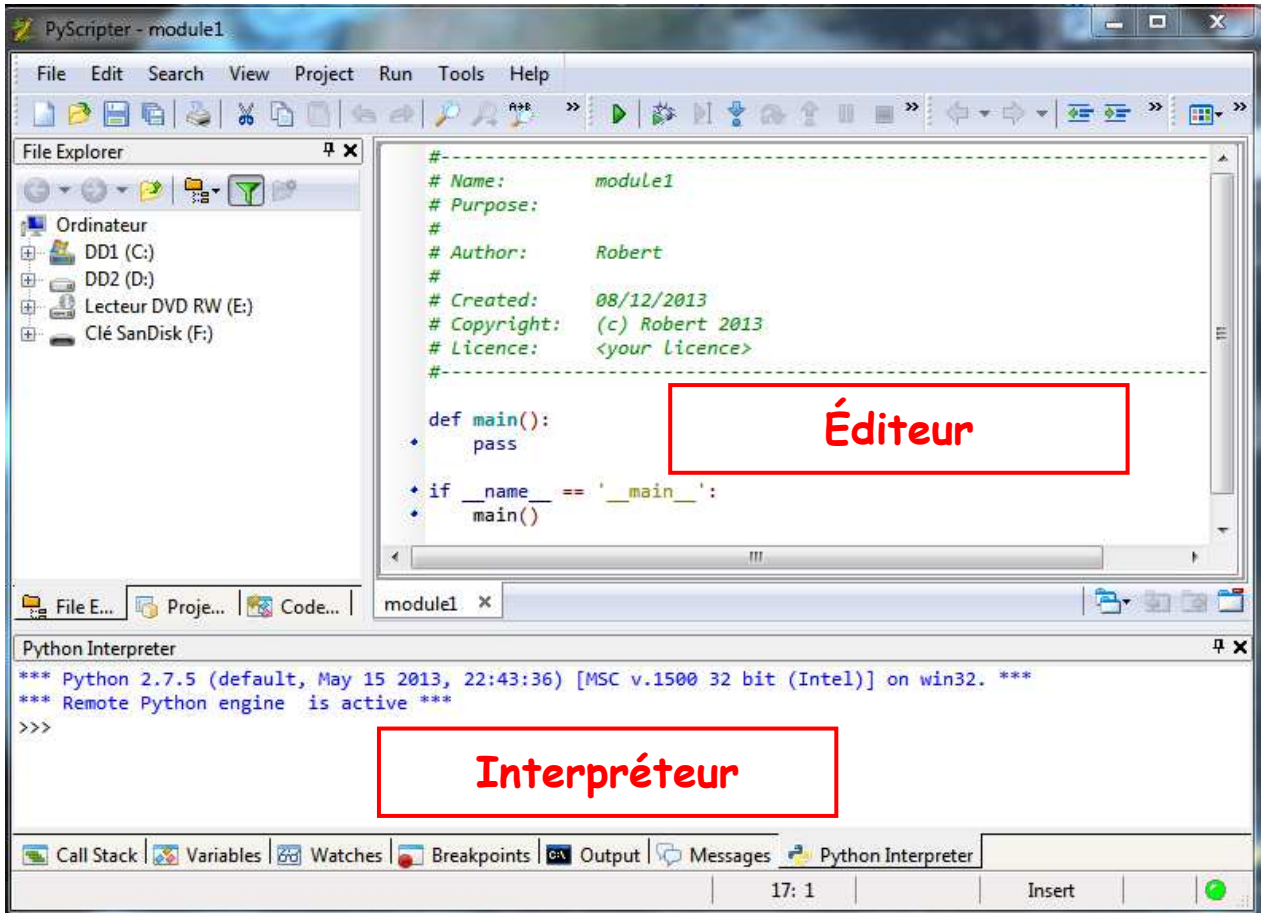
L'opérateur « **%** » donne le reste de la division (**opération modulo**) :

```
>>> reste = 450 % 360
>>> print reste
90
```

L'opérateur « ****** » donne la puissance :

```
>>> Mo = 2**20
>>> print Mo
1048576
```

Vous allez maintenant utiliser « Pyscripter », pour cela lancer Pyscripter (sur le bureau dossier programmation et réseau).



L'interpréteur, on le reconnait facilement. C'est lui qui contient le triple chevron « **>>>** » qui est l'invite de Python (prompt en anglais) et qui signifie que Python attend une commande.

L'éditeur permet l'écriture de scripts.

LE TYPE « FLOAT » (NOMBRES EN VIRGULE FLOTTANTE)

Question 3 :

*Entrer dans l'éditeur les codes ci-après. Tester et vérifier les codes en cliquant sur « **Run** » ou (**CTRL+F9**) pour visualiser le résultat dans l'interpréteur.*

```
b = 17.0                    # le séparateur décimal est un point (et non une virgule)
print b
17.0

print type(b)
<type 'float'>

c = 14.0/3.0
print c
4.66666666667

c = 14.0//3.0               # division entière
print c
4.0
```

Attention : avec des nombres entiers, l'opérateur « / » fait aussi une division entière :

```
c = 14/3
print c
4
```

Notation scientifique :

```
a = -1.784892e4
print a
-17848.92
```

Les fonctions mathématiques :

Pour utiliser les fonctions mathématiques, il faut commencer par importer le module math :

```
import math                 # à tester sur l'IDLE (shell)
```

La fonction « **dir()** » retourne la liste des fonctions et données d'un module :

```
dir(math)

['__doc__', '__name__', '__package__', 'acos', 'acosh', 'asin', 'asinh', 'atan',
'atan2', 'atanh', 'ceil', 'copysign', 'cos', 'cosh', 'degrees', 'e', 'erf',
'erfc', 'exp', 'expm1', 'fabs', 'factorial', 'floor', 'fmod', 'frexp', 'fsum',
'gamma', 'hypot', 'isinf', 'isnan', 'ldexp', 'lgamma', 'log', 'log10', 'log1p',
'modf', 'pi', 'pow', 'radians', 'sin', 'sinh', 'sqrt', 'tan', 'tanh', 'trunc']
```

Pour appeler une fonction d'un module, la syntaxe est la suivante :

module.fonction(arguments)

Pour accéder à une donnée d'un module :

```
module.data
print math.pi                    # donnée pi du module math (nombre pi)
3.14159265359

print math.sin(math.pi/4.0)      # fonction sin() du module math (sinus)
0.707106781187

print math.sqrt(2.0)             # fonction sqrt() du module math (racine carrée)
1.41421356237

print math.exp(-3.0)             # fonction exp() du module math (exponentielle)
0.0497870683679

print math.log(math.e)           # fonction log() du module math (logarithme népérien)
1.0
```

LE TYPE « STR » (STRING : CHAÎNE DE CARACTÈRES)

```
>>> Affiche1 = 'Premiere'        # entre apostrophes
>>> print Affiche1
Premiere

>>> print type(Affiche1)
<type 'str'>

>>> Affiche2 = "SIN"             # on peut aussi utiliser les guillemets
>>> print Affiche2
SIN

>>> print Affiche1,Affiche2      # ne pas oublier la virgule
Premiere SIN
```

La concaténation désigne la mise bout à bout de plusieurs chaînes de caractères.
La concaténation utilise l'opérateur « **+** ».

```
>>> chaine = Affiche1 + Affiche2    # concaténation de deux chaînes de caractères
>>> print chaine
PremiereSIN

>>> chaine = Affiche2 + Affiche1    # concaténation de deux chaînes de caractères
>>> print chaine
SINPremiere
```

```
>>> chaine = Affiche1 + ' ' + Affiche2
>>> print chaine
Premiere SIN

>>> chaine = chaine + ' BAC 2014'
>>> print chaine
Premiere SIN BAC 2014
```

La fonction « **len()** » retourne la longueur (length) de la chaîne de caractères :

```
>>> print len(chaine)
21

>>> print chaine[0]                    # premier caractère (indice 0)
P

>>> print chaine[1]                    # deuxième caractère (indice 1)
r

>>> print chaine[1:4]
rem

>>> print chaine[2:]
emiere SIN BAC 2014

>>> print chaine[-1]                   # dernier caractère (indice -1)
4

>>> print chaine[-6:]
N 2014

>>> chaine = 'Aujourd'hui'
SyntaxError: invalid syntax

>>> chaine  = 'Aujourd\'hui'           # séquence d'échappement \'
>>> print chaine
Aujourd'hui
>>> chaine  = "Aujourd'hui"
>>> print chaine
Aujourd'hui
```

La séquence d'échappement « **\n** » représente un saut ligne :

```
>>> chaine = 'Première ligne\nDeuxième ligne'
>>> print chaine
Première ligne
Deuxième ligne
```

Plus simplement, on peut utiliser les triples guillemets (ou les triples apostrophes) pour encadrer une chaîne définie sur plusieurs lignes :

```
>>> chaine = """Première ligne
Deuxième ligne"""
>>> print chaine
Première ligne
Deuxième ligne
```

On ne peut pas mélanger « les serviettes et les torchons » (ici type str et type int) :

```
>>> chaine = '17.45'
>>> print type(chaine)
<type 'str'>

>>> chaine = chaine + 2
```

La fonction « **float()** » permet de convertir un type str en type float.

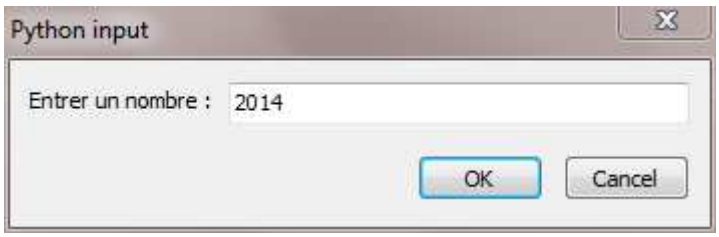
```
>>> nombre = float(chaine)
>>> print nombre
17.45

>>> print type(nombre)
<type 'float'>

>>> nombre = nombre + 2
>>> print nombre
19.45
```

La fonction « **raw_input()** » lance une invite de commande (en anglais : prompt) pour saisir une chaîne de caractères.

```
>>> # saisir une chaîne de caractères et valider avec la touche Entrée
>>> chaine = raw_input("Entrer un nombre : ")
```




```
>>> print chaine
2014

>>> print type(chaine)
<type 'str'>
>>> nombre = float(chaine)                # conversion de type
>>> print nombre**2
<type 'unicode'>
4056196.0
```

LE TYPE « LIST » (LISTE)

Une liste est une structure de données.
Le premier élément d'une liste possède l'indice (l'index) 0.
Dans une liste, on peut avoir des éléments de plusieurs types.

```
>>> InfoPerso = ['Pierre' , 'Dupont' , 17 , 1.75 , 72.5]
>>> # la liste InfoPerso contient 5 éléments de types str, str, int, float et float
>>> print type(InfoPerso)
<type 'list'>

>>> print InfoPerso
['Pierre', 'Dupont', 17, 1.75, 72.5]

>>> print 'Prénom : ',InfoPerso[0]          # premier élément (indice 0)
Prénom :  Pierre

>>> print 'Age : ',InfoPerso[2]             # le troisième élément a l'indice 2
Age :  17

>>> print 'Taille : ',InfoPerso[3]          # le quatrième élément a l'indice 3
Taille :  1.75
```

La fonction « **range()** » crée une liste d'entiers régulièrement espacés :

```
>>> maliste = range(10)
>>> print maliste
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]

>>> print type(maliste)
<type 'list'>
```

```
>>> maliste = range(1,10,2)
>>> print maliste
[1, 3, 5, 7, 9]

>>> print maliste[2]                    # le troisième élément a l'indice 2
5
```

On peut créer une liste de listes, qui s'apparente à un tableau à 2 dimensions (ligne, colonne) :

```
 0  1  2
10 11 12
20 21 22

>>> maliste = [[0,1,2],[10,11,12],[20,21,22]]
>>> print maliste[0]
[0, 1, 2]

>>> print maliste[0][0]
0

>>> print maliste[2][1]                 # élément à la troisième ligne et deuxième colonne
21

>>> maliste[2][1] = 69                  # nouvelle affectation
>>> print maliste
[[0, 1, 2], [10, 11, 12], [20, 69, 22]]
```

LE TYPE « BOOL » (BOOLÉEN)

Deux valeurs sont possibles : True et False :

```
>>> a = True
>>> print type(a)
<type 'bool'>
```

Les opérateurs de comparaison :

Opérateur	Signification	Remarques
<	strictement inférieur	
<=	inférieur ou égal	
>	strictement supérieur	
>=	supérieur ou égal	
==	égal	Attention : deux signes ==
!=	différent	
is	identique	Deux conditions : égal et même type
is not	non identique	

```
>>> b = 10
>>> print b>8
True

>>> print b==5
False

>>> print b!=10
False

>>> print 0<= b <=20
True
```

Les opérateurs logiques : « **and** », « **or** », « **not** » :

```
>>> note=13.0
>>> mentionAB = note>=12.0 and note<14.0     # ou bien : mentionAB = 12.0 <= note < 14.0
>>> print mentionAB
True

>>> print not mentionAB
False

>>> print note==20.0 or note==0.0
False
```

L'opérateur « **in** » s'utilise avec des chaînes (type str) ou des listes (type list) :

```
>>> chaine = 'Bonsoir'
>>> # la sous-chaîne 'soir' fait-elle partie de la chaîne 'Bonsoir' ?
>>> a = 'soir' in chaine
>>> print a
True

>>> print 'b' in chaine
False

>>> maliste = [4,8,15]
>>> # le nombre entier 9 est-il dans la liste ?
>>> print 9 in maliste
False

>>> print 8 in maliste
True

>>> print 14 not in maliste
True
```

LE TYPE « DICT » (DICTIONNAIRE)

Un dictionnaire stocke des données sous la forme clé ⇒ valeur.
Une clé est unique et n'est pas nécessairement un entier (comme c'est le cas de l'indice d'une liste).

```
>>> moyenne = {'sin':18,'math':15.8}          # entre accolades
>>> print type(moyenne)
<type 'dict'>
>>> print moyenne['math']                     # entre crochets
15.8
>>> moyenne['math']=14.3                      # nouvelle affectation
>>> print moyenne
{'sin': 18, 'math': 14.3}
```

Vous avez vu les types les plus courants.
Il en existe bien d'autres :

« long » (nombres entiers de longueur quelconque, par exemple 42849617755620125369541591O2L)
« complex » (nombres complexes, par exemple 1+2.5j)
« tuple » (structure de données)
« set » (structure de données)
« file » (fichiers)
...

PROGRAMMATION ORIENTÉE OBJET (POO)

Python est un langage de programmation orienté objet (comme les langages C++, Java, PHP, Ruby...).
Une **variable** est en fait un **objet** d'une certaine **classe**.

Par exemple, la variable amis est un objet de la classe list.
On dit aussi que la variable amis est une instance de la classe list.
L'instanciation (action d'instancier) est la création d'un objet à partir d'une classe (syntaxe : NouvelObjet = NomdelaClasse(arguments)) :

```
>>> # instanciation de l'objet amis de la classe list
>>> amis = ['Nicolas','Julie']          # ou bien : amis = list(['Nicolas','Julie'])
>>> print type(amis)
<type 'list'>
```

Une classe possède des **fonctions** que l'on appelle **méthodes** et des **données** que l'on appelle **attributs**.

La méthode « **append()** » de la classe list ajoute un nouvel élément en fin de liste :

```
>>> # instanciation d'une liste vide
>>> amis = []                           # ou bien : amis = list()
>>> amis.append('Nicolas')              # synthaxe générale : objet.méthode(arguments)
>>> print amis
['Nicolas']
```

```
>>> amis.append('Julie')                    # ou bien : amis = amis + ['Julie']
>>> print amis
['Nicolas','Julie']

>>> amis.append('Pauline')
>>> print amis
['Nicolas','Julie','Pauline']

>>> amis.sort()                             # la méthode sort() trie les éléments
>>> print amis

['Julie', 'Nicolas', 'Pauline']

>>> amis.reverse()                          # la méthode reverse() inverse la liste des éléments
>>> print amis
['Pauline', 'Nicolas', 'Julie']
```

La méthode « **lower()** » de la classe str retourne la chaîne de caractères en casse minuscule :

```
>>> # la variable chaine est une instance de la classe str
>>> chaine = "BONJOUR"                      # ou bien : chaine = str("BONJOUR")
>>> chaine2 = chaine.lower()# on applique la méthode lower() à l'objet chaine
>>> print chaine2
Bonjour

>>> print chaine
BONJOUR
```

La méthode « **pop()** » de la classe dict supprime une clé :

```
>>> # instanciation de l'objet moyenne de la classe dict
>>> moyenne = {'sport': 17.0, 'anglais': 14.3, 'sin': 18.0}
>>> # ou : moyenne = dict({'sport': 17.0, 'anglais': 14.3, 'sin': 18.0})
>>> moyenne.pop('anglais')
14.3
>>> print moyenne
{'sport': 17.0, 'sin': 18.0}

>>> print moyenne.keys()                    # la méthode keys() retourne la liste des clés
['sport', 'sin']

>>> print moyenne.values()                  # la méthode values() retourne la liste des valeurs
[17.0, 18.0]
```

QCM ET EXERCICES D'APPLICATION

Copier puis coller le fichier « **QCM et exo sur Python-Les variables.htm** » dans votre dossier personnel.
Ouvrir le fichier « **QCM et exo sur Python-Les variables.htm** » à l'aide de « **FireFox** ».

Question :

*Compléter le formulaire puis sauvegarder votre travail sous le nom : « **AP_Python1_VOS NOMS.htm** » dans votre dossier personnel puis glisser une copie du fichier dans votre groupe de partage.*