Informatique – TP1 – Premiers pas en Python

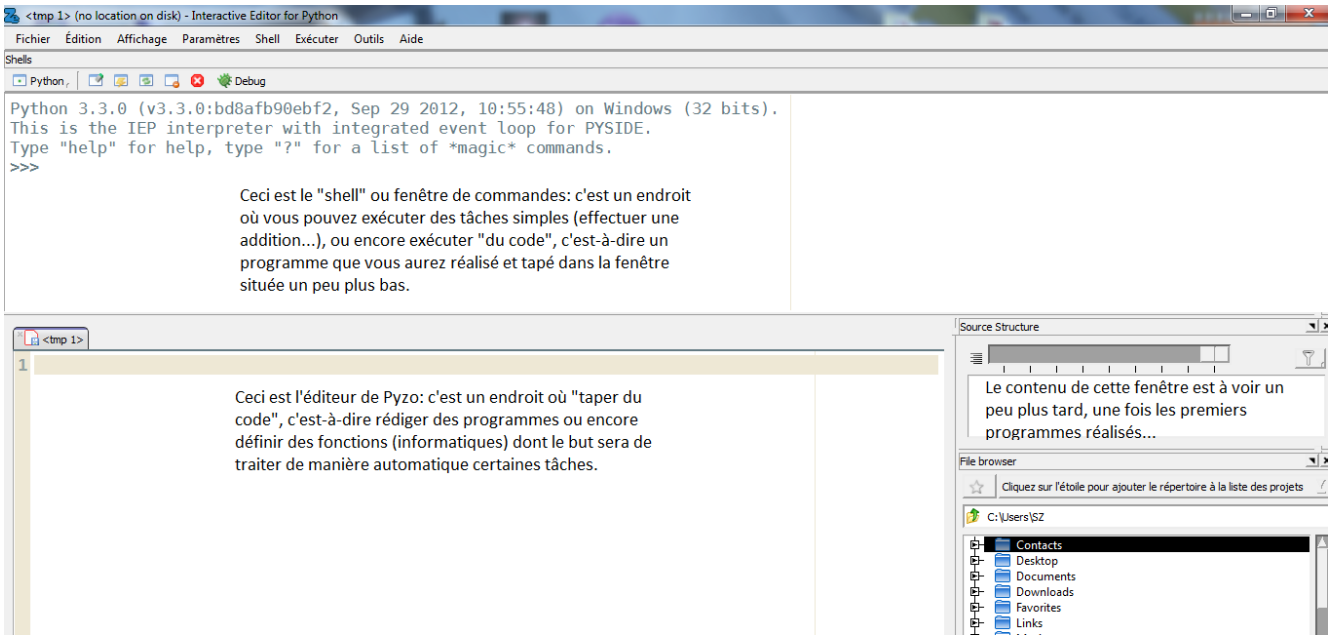
L'objectif principal de ce premier TP est de présenter sommairement Python. C'est un langage de programmation informatique. C'est par son intermédiaire que nous (re)verrons les bases de l'algorithmique cette année.

Découvrir Pyzo

Python est un langage de programmation; ce n'est donc ni un logiciel ni un programme. Cette année, nous utiliserons Pyzo comme **environnement de travail** pour commencer à se familiariser avec Python. Vous pourrez le télécharger (gratuitement!) à l'adresse suivante :

`http://www.pyzo.org/`

A l'ouverture, s'ouvre un mini-guide décrivant l'environnement dans lequel vous allez travailler. Vous arrivez ensuite sur la fenêtre d'accueil suivante :



Très grossièrement, on peut faire le parallèle avec votre calculatrice, qui vous sert à effectuer des opérations de base et également à faire tourner des programmes (résolution d'une équation du second degré). Les opérations de base sont ici directement effectuées dans **le shell**, et les programmes rédigés dans **l'éditeur** (de code).

Opérations basiques dans le shell

L'objectif de cette première activité est de vous assurer que la syntaxe de Python ne diffère pas de celle que vous connaissez déjà avec votre calculatrice. Le but est de taper dans le shell l'instruction donnée dans la colonne de gauche, et d'observer et d'être critique face au résultat obtenu. N'oubliez pas de taper sur Entrée pour valider une instruction; observez que vous pouvez utiliser les flèches directionnelles ↑ et ↓ du clavier pour retrouver l'historique des instructions déjà tapées.

Instruction à taper dans le shell	Résultat (et commentaires pour les cases colorées)
5 + 9	
2 * 8	
2 ** 8	
17 / 5	
17 // 5	
17 % 5	
2 * pi	
a + b	
'a' + 'b'	
3 * 'MPSI'	
4 * 12	
4 * '12'	
	'123456123456123456123456123456123456123456123456'

Remarque. Une des instructions tapées plus haut a sans doute produit un message d'erreur. Pour en comprendre l'origine, et la corriger, exécutez dans le shell la commande suivante:

```
from math import *
```

Puis exécutez à nouveau l'instruction évoquée plus haut...

*Explication: l'instruction "from math import *" permet d'importer tout le contenu d'une bibliothèque (ou package) nommé math, et d'utiliser les fonctions (trigonométriques, racine carrée...) et constantes (π, e) contenues dans cette bibliothèque. Grossièrement, cette instruction permet donc de transformer une calculatrice collège en une calculatrice scientifique.*

Il existe de nombreuses autres bibliothèques pour réaliser des tâches spécifiques en Python: certaines sont consacrées aux graphiques, aux dessins en 3D, au calcul matriciel, aux animations et même... aux jeux vidéo! Nous en utiliserons quelques-unes cette année (certaines sont explicitement au programme des Concours).

Création de plages d'entiers

Comme vous le savez peut-être déjà, on utilise souvent les **boucles** lorsque l'on fait de la programmation. Nous aurons l'occasion d'y revenir, mais en attendant rappelons qu'une boucle fait intervenir une certaine variable (ou compteur) qui va parcourir une certaine plage de valeurs entières. La commande range permet de générer ces plages, comme l'illustrent les exemples ci-dessous.

Instruction à taper dans le shell	Résultat
list (range(8))	
list (range(4,9))	
list (range(4,9,3))	
	[0,1,2,3,4,5,6,7,8,9,10]
	[3,5,7,9,11,13]
	[9,19,29,39,49,59,69,79,89,99]

Remarque : Les instructions précédentes ont permis de créer des listes d'entiers. Une autre commande utile pour les listes est la commande len (abréviation de length) qui permet de connaître la longueur d'une liste.

Ex 1 Combien y a-t-il de nombres pairs compris (au sens large) entre 0 et 100?

Ex 2 Combien y a-t-il de multiples de 17 compris entre 170 et 2023?

Différents types de données

Comme vous l'avez déjà observé dans le premier exercice, le langage Python fait intervenir des objets de natures ou de types différents. Le résultat de l'opération 2 * 12 n'est pas celui de l'opération 2 * '12'. Pour chaque type de donnée, un certain nombre d'opérations seront définies : il est par exemple permis de faire la somme de deux mots (ex: 'Bon' + 'jour!'), mais pas la multiplication ou la division par 3 ... La commande type permet de connaître (comme son nom l'indique) le type d'un objet et de savoir quelles opérations il sera légitime de lui appliquer.

Instruction à taper dans le shell	Résultat (et commentaires pour les cases colorées)
type(5)	
type(2.5)	
type(2,5)	
type(1/3)	
type(1//3)	
type(5.00)	
type('5')	
type('Bonjour')	
type(Bonjour)	
type(3 >2)	
type(3 =2)	
type(3 ==2)	
type(list (range(10)))	

Les différents types rencontrés sont:

- ➤
- ➤
- ➤
- ➤
- ➤

On peut également convertir une expression d'un type donné dans un autre type. Il suffit alors d'utiliser respectivement les fonctions list, str, float et int qui transforment dans la mesure du possible en un objet de type liste, chaîne de caractères, flottant et entier.

Instruction à taper dans le shell	Résultat
float (5)	
int (2.5)	
int (-2.5)	
str (2.5)	
list ('toto')	
float ('toto')	
str(list ('toto'))	
str (3 >2)	

Affecter une valeur à une variable

Pour donner une valeur à une variable, rien de plus simple en Python. L'instruction `x = 2` permet à la fois de créer une variable `x` et de lui affecter la valeur 2. Et rien n'empêche de travailler avec d'autres types que des nombres.

Instruction à taper dans le shell	Résultat
`x =3`	
`x ** 2`	
`type( x )`	
`type( x / 2 )`	
`y ='Abc'`	
`y +y`	
`4 * y`	
`type( y )`	
`x ==y`	
`x ==3`	

Ex 3 Créer une variable `a` et lui affecter la valeur 2; créer une variable `b` et lui affecter la valeur 3. Puis proposer plusieurs méthodes pour échanger les valeurs de `a` et `b`.

Le type booléen

Une variable de type booléen ne peut prendre que deux valeurs True et False. Comme vous le savez encore, lorsque l'on dispose d'un booléen (d'une assertion) P, on peut définir sa négation; et si l'on a deux booléens, on peut définir leur conjonction [P et Q] et leur disjonction [P ou Q]. Les syntaxes respectives de ces opérateurs en Python sont les suivantes :

- **not**(P) désigne l'assertion non P;
- P **and** Q désigne l'assertion [P et Q];
- P **or** Q désigne l'assertion [P ou Q].

Instruction à taper dans le shell	Résultat (et commentaires pour les cases colorées)
`3 == 2 +1`	
`9 ==3 ** 2`	
`-1 >0`	
`k ==3`	
`( 1.5 ** 2 ) ** (1/2) ==( 1.5 ** (1/2)) ** 2`	
`(1 +2 ** (-54)) - 1 ==0`	
`(1 - 1 ) +2 ** (-54) ==0`	
`5 in list ( range(8) )`	
`a +b ==ab`	
`'a' +'b' =='ab'`	

Remarque : Attention au = et au ==. Le premier sert à affecter une valeur à une variable, le second sert à comparer deux expressions.

Utilisation de l'éditeur

Si la fenêtre de commande est utile pour les opérations élémentaires, l'éditeur se révèle indispensable dès qu'il s'agit d'effectuer des tâches plus complexes. Et pour rentrer directement dans le vif du sujet, mentionnons deux commandes indispensables en programmation:

- la commande print: cette première instruction sert à afficher un texte, un nombre, une liste ... Son intérêt est évident: imaginez que vous ayez dans votre calculatrice un programme qui calcule les solutions d'une équation mais qui ne vous les donne pas!
- la commande input: celle-ci sert à interagir avec l'utilisateur, et à récupérer des données saisies au clavier.

Exemple : Voici un premier programme basique, qui consiste simplement à faire afficher le texte

```
Hello world
```

Pour réaliser ce programme, il suffit de taper dans **l'éditeur** la commande :

```
print( 'Hello world' )
```

Si rien ne se passe, essayez de trouver un moyen de débloquer la situation...

Exemple : Après la commande print, la commande input. Pour débuter, un nouvel exemple simple :

```
print( 'Tapez un mot au clavier' )
Mot =input()
print( 'Vous avez écrit ',Mot)
```

Ex 4 En vous inspirant de l'exemple précédent, écrire un programme qui demande à l'utilisateur son prénom, et qui renvoie comme message Bonjour prénom !.

Ex 5 Modifiez le programme précédent pour qu'il affiche aussi la phrase: Vous avez un bien joli prénom de xxx lettres (où xxx désigne effectivement le nombre de lettres du prénom).

Ex 6 Écrire un programme qui demande à l'utilisateur un nombre N et qui retourne comme résultat son double.

Ex 7 Cherchez l'erreur... Recopier puis exécuter le script ci-dessous (dont le but est d'afficher les carrés des entiers de 1 à 8).

```
for nombre in range(1.0, 8.0):
        print(nombre *nombre)
```

Ex 8 Cherchez l'erreur (bis)... Recopier puis exécuter le script ci-dessous; puis utiliser le message renvoyé par Pyzo pour corriger les éventuelles erreurs contenues dans ce code.

```
print('Bonjour!')
print(' Choisissez un entier naturel')
NB1 =int(input())
NB2 ='2021'
NB3 =NB2 +NB1
print(NB(3))
print('Au revoir!)
```