Programmer en Python
Licence 2, Mathématiques
V. Monbet
TP 1. Prise en main

Dans votre espace personnel, créer un dossier `IOB`. Dans ce dossier créer un sous-dossier `TP1` : c'est là que vous enregistrerez les fichiers relatifs à ce TP.

1 Python en interactif

Lorsque vous ouvrez l'environnement *Spyder* vous obtenez une fenêtre appelée *interpréteur* (ou *console*, ou *terminal*). A partir de là nous pouvons utiliser le langage Python en mode interactif. Les expressions sont rédigées à la suite des chevrons `>>>` puis évaluées avec la commande *Entrée*.

Exercice 1

1. Ouvrir l'editeur de commandes (par exemple IDLE ou spyder) puis réaliser les opérations suivantes dans l'interpréteur.

$$50 + 2 * (12,5 - 4) = \cdots \ , \quad \frac{3}{2} = \cdots, \quad \frac{2}{3} + 1 = \cdots, \quad 2^5 = \cdots$$

2. Evaluer les expressions suivantes `25//3`, `23 % 3`, `abs(1.3)`, `abs(-1.3)`
3. Compléter le tableau ci-dessous

Opération	Opérateur	Opération	Opérateur
Somme		Produit	
Différence		Division numérique	
	\\		%
Puissance		Valeur absolue	

Exercice 2

Les fonctions mathématiques usuelles ne sont pas disponibles immédiatement à l'ouverture de l'interpréteur Python ; il faut charger une librairie (ou module) complémentaire : le module `math`. Nous verrons d'autres modules importants au cours de l'année.

Exécuter les commandes ci-dessous.

```
>>> sin(pi)
>>> from math import *
>>> sin(pi)
```

La commande `import` donne accès à une nouvelle librairie ; le symbole * signifie que toutes les nouvelles commandes issues du module importé sont désormais accessibles.

Exercice 3

Calculer des valeurs (approchées) des expressions suivantes. Donner un résultat avec 2 chiffres puis 4 chiffres après la virgule.

$$\ln(10) = \cdots \ , \quad \cos\left(\tfrac{\pi}{8}\right) = \cdots, \quad \tfrac{\sqrt{2}+2}{2} = \cdots$$

Pour la précision de l'affichage, vous pouvez utiliser l'exemple suivant.

```
>>> "{:.3f}".format(54.334343434)
'54.334'
```

2 Variables et affectations

Exercice 4

Examiner la série de commandes ci-dessous. Prédire le résultat puis confirmer-le à l'aide de l'interpréteur.

```
 >>> x=1
 >>>print(x)
1
>>> x=1
>>> y=2
>>> x=x+y
>>> y=x**y
>>> print(y)
?
>>> print(x,y)
?
```

Exercice 5

Examiner la série de commandes ci-dessous. Prédire le résultat puis confirmer-le à l'aide de l'interpréteur.

```
 >>> x,y=1,2
>>> x=x+y
>>> y=x**y
>>> print(x,y)
?
```

Exercice 6

Remarque. La commande ci-dessous permet d'échanger deux noms de variables.

```
>>> x,y=y,x
```

Écrire une liste d'instructions qui permette d'affecter la valeur $\sin(\pi/4)$ à `x`, $\cos(\pi/4)$ à `y` et de calculer $z = x^2 + y^2$ puis d'afficher la valeur contenue dans la variable z.

Créer les listes `t` et `s` suivantes et utiliser la combinaison d'une boucle `for` et d'une commande `zip` (voir exemples ci-dessous) pour calculer $t - s$.
$t = [1, 1.5, 2, 2.5, 3]$
$s = [0.1, 0.2, 0.3, 0.4, 0.5]$
Vous pourrez vous aider de l'exemple ci-dessous.

```
>>> a = [1, 2, 3]
>>> b = ['a', 'b', 'c']
>>> z = zip(a, b)
>>> z
[(1, 'a'), (2, 'b'), (3, 'c')]
>>> zip(*z)
[(1, 2, 3), ('a', 'b', 'c')]
>>> x = [2 * i + 1 for i in range(3)]
>>> print(x)
[1, 3, 5]
```

3 Premiers programmes Python

L'utilisation de Python en ligne de commande dans l'interpréteur ne permet pas de sauvegarder vos lignes de calcul : ce n'est pas très pratique si vous souhaitez relancer une même série de calculs avec des valeurs différentes pour les variables.

Exercice 7

On souhaite rédiger un programme élémentaire qui calcule le prix d'une commande de bières pour un festival. Les trois valeurs suivantes sont représentées par les variables :

`nbr`	=	entier désignant le nombre de fûts commandés
`prix`	=	prix unitaire d'un fût
`reduc`	=	coefficient (entre 0 et 1) représentant la réduction dont bénéficie le client

Le programme affiche le montant de la facture `m = nbr * prix * reduc`.

1. Ouvrir un nouveau fichier que vous sauvegardez sous le nom de `premiersProgrammes.py`.
2. Affecter les variables comme indiqué ci-dessus dans le cas d'une commande de 27 fûts dont le prix unitaire est de 22,95 euros pour un client bénéficiant de 5 % de réduction.
3. Afficher le résultat dans l'interpréteur. Pour afficher un résultat il faut faire appel à la fonction `print`. Pour afficher une phrase on peut utiliser par exemple la syntaxe : `print('Le montant de la commande est de',m,'euros')`.
   Pour aller plus loin : vous afficherez la valeur tronquée de `m` à deux chiffres après la virgule.

Remarques importante

1. Syntaxe pour les procédures (sans variable en entrée)

   ```
   def Program1():
        instruction 1
        instruction 2
        etc
   ```

   pour appeler le programme appelé Program1

   ```
   Program1()
   ```

2. Vous pouvez afficher des commentaires dans votre script en les rédigeant précédés d'un symbole `#` ; ils ne seront pas exécutés.
3. La fonction `input()` permet d'interagir avec l'utilisateur : elle interrompt le programme jusqu'à ce que l'utilisateur rentre une valeur et appuie sur *Entrée*. Cela nous impose d'en apprendre un peu plus sur les types des variables...

Exercice 8

1. Écrire et tester un programme `triple.py` qui demande à l'utilisateur un nombre et affiche ensuite le triple de ce nombre.
2. Écrire et tester un programme `cercle.py` qui demande à l'utilisateur le rayon du cercle, puis calcule et affiche le diamètre, le périmètre et la surface. L'affichage se fera sur trois lignes, une pour chaque résultat.
3. **Pour aller plus loin**
   Creer une classe `Cercle` ayant pour attribut `rayon`. Et définir les méthodes `perimetre` et `aire` qui renvoie le périmètre et l'aire du cercle.
   Il existe plusieurs façons de répondre à cette question, vous pourrez vous inspirer des exemples de cette page :
   `http://www.courspython.com/classes-et-objets.html`