TP # 2

Bibliothèque numpy, fonctions

numpy : une bibliothèque pour faire des maths

Certaines commandes ne sont pas directement accessibles dans Python. Pour les obtenir, il faut importer la *bibliothèque* dans laquelle elles se trouvent.
Il existe de nombreuses bibliothèques, répondant à de nombreux besoins :

- `numpy` : calculs mathématiques.
- `matplotlib.pyplot` : représentations graphiques de fonctions, de suites.
- `Pygame` : créer des jeux 2D et 3D.
- Etc...

Importer la bibliothèque numpy

- Méthode 1 : `import numpy as np` (importer numpy sous le nom : np)
- Méthode 2 : `from numpy import *` (depuis numpy, importer : tout)

On peut ainsi importer la bibliothèque `numpy` en tapant l'une ou l'autre de ces instructions dans la console, ou bien en ajoutant l'instruction au début d'un script tapé dans l'éditeur de texte. Une fois `numpy` importée, on peut utiliser les instructions qu'elle contient.

Fonctions usuelles et constantes mathématiques (depuis numpy)

exp	`np.exp`	$\sqrt{\cdot}$	`np.sqrt` (square root)
ln	`np.log` (et non `ln`!)	$\lvert\cdot\rvert$	`np.abs` (absolute value)
cos	`np.cos`	$\lfloor\cdot\rfloor$	`np.floor` (partie entière)
sin	`np.sin`	π	`np.pi`
tan	`np.tan`	e	`np.e`

Attention !

Si on importe la bibliothèque `numpy` à l'aide de la Méthode 2 (non recommandée par le programme), les commandes n'ont pas besoin d'être précédées de : "`np.`"

Exemple : Les deux scripts suivants sont valides et affichent la même valeur.

```
from numpy import *
a=exp(2*pi)
print(a)
```

```
import numpy as np
a=np.exp(2*np.pi)
print(a)
```

Exercice 1

1. Dans la console, tapez l'instruction : `from numpy import *`

Tapez ensuite les commandes suivantes. Constater les erreurs.

```
>>> exp(0)      >>> cos(2*pi)      >>> abs(-5)
>>> np.log(2)   >>> np.sqrt(4)
```

2. Réinitialiser la console : `Exécuter > Moteur Python > Réinitialiser`.
Dans la console, tapez maintenant l'instruction : `import numpy as np`
Tapez ensuite les commandes suivantes. Constater les erreurs.

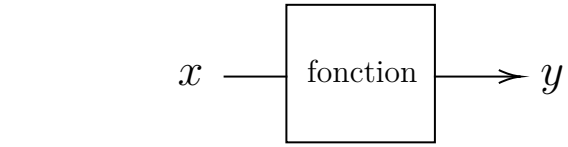
```
>>> np.exp(0)      >>> np.cos(2*np.pi)      >>> np.abs(-5)
>>> log(2)   >>> sqrt(4)
```

Exercice 2

Dans l'éditeur de texte, écrire un script qui stocke dans une variable nommée `X` la valeur $\frac{\sqrt{3\pi}}{2\cos(1)}$, puis affiche cette valeur.

Fonctions en Python

En programmation, une fonction est une "procédure" à laquelle on fournit un ou plusieurs paramètres (**argument(s) d'entrée**), qui effectue des opérations à partir de ces paramètres, et renvoie un résultat (**argument(s) de sortie**).



argument d'entrée — série d'instructions — argument de sortie

Une fonction se définit de préférence dans l'éditeur de texte
(même s'il est possible de définir une nouvelle fonction directement dans la console...)

Définir une fonction

```
def fonction(x) :  # nom de la fonction et argument d'entree
    instructions # calculs pour definir y a partir de x
    return(y) # la fonction "renvoie" la valeur y
```

⚠ Attention !

- Ne pas oublier les 'deux points' (`:`)
- L'indentation (=le décalage par rapport à la marge de gauche) est essentielle ! Les calculs effectués au sein de la définition de la fonction peuvent contenir plusieurs lignes d'instructions (avant le `return`), mais toutes ces lignes doivent alors être indentées de la même façon. Une indentation est "proposée" par l'éditeur de texte.
- Pour qu'une fonction "renvoie" une valeur (celle du ou des arguments de sortie), on utilisera toujours l'instruction `return` (et non pas `print` !)

Notons qu'un script définissant une fonction n'affiche rien de particulier !
En revanche, une fois le script exécuté, la fonction "`fonction`' est définie et donc utilisable en Python comme n'importe quelle instruction usuelle ! On peut ainsi *appeler* la fonction dans la console (ou ailleurs, d'ailleurs !), c'est à dire l'évaluer pour un certain argument d'entrée, et on voit le résultat s'afficher.

Exercice 3

Dans l'éditeur de texte, définir la fonction `increment` suivante :

```
def increment(a):  # l'argument d'entree est a
    b=a+1  # a partir du a "fourni", on définit b
    return(b)  # la fonction renvoie la valeur de b
```

Exécuter ce script et constater que rien ne s'affiche...
Tester alors la fonction en l'appelant dans la console :

```
>>> increment(2)    >>> increment(-5)    >>> increment(7.3)
```

Exercice 4

1. Définir en Python une fonction nommée `P` qui prend en entrée un réel `x` et renvoie la valeur : $x^3 - 5x + 2$.

2. On appelle *racine* de P tout réel x tel que $P(x) = 0$.
Vérifier grâce à Python que 2, $\sqrt{2} - 1$ et $-1 - \sqrt{2}$ sont des racines de P.

Exercice 5

Définir en Python une fonction `gaussienne` qui à $x \in \mathbb{R}$ associe la valeur $\dfrac{e^{-x^2}}{2}$.
On n'oubliera pas, avant de définir la fonction, d'importer la bibliothèque `numpy` !

Tester quelques valeurs dans la console, pour s'assurer que la fonction fonctionne.

Une fonction peut avoir **plusieurs** arguments d'entrée, et/ou arguments de sortie.
Il suffit (dans le `def(...)` pour l'entrée, dans le `return(...)` pour la sortie) de **séparer les différentes variables par des virgules**.

Exercice 6

Définir une fonction `pythagore` de sorte que `pythagore`$(x, y) = x^2 + y^2$

Exercice 7

Définir la fonction `couple` suivante :

```
def couple(N):
    a=2/N
    b=1/N+1/(2*N)+1/(3*N)+1/(6*N)
    return(a,b)
```

Appeler cette fonction dans la console pour différente valeurs de `N`.
Comprendre pourquoi on observe ce que l'on observe...

Pour s'exercer un peu

Exercice 8

On rappelle que le volume d'une sphère de rayon R est $V = \frac{4}{3}\pi R^3$.

Définir une fonction `volume_sphere` qui prend en entrée un réel `R` et renvoie le volume de la sphère de rayon `R`.

Exercice 9

Définir une fonction `conversion` qui prend en entrée un triplet `(H,M,S)` donnant une durée en nombre d'heures, de minutes, de secondes, et qui renvoie une valeur `T` exprimant cette même durée uniquement en seconde.

Par exemple, on veut que `conversion(1,1,2)` renvoie la valeur `3662`.

Exercice 10

1. Définir, à la suite de la fonction précédente, une fonction `anticonversion` qui prend en entrée une durée `T` donnée en seconde, et qui renvoie un triplet `(H,M,S)` comptant le nombre d'heures, minutes, secondes.

Par exemple, on veut que `anticonversion(63)` renvoie la valeur `(0,1,3)`.

(*Rappel : on dispose de `//` pour calculer le quotient dans une div. euclidienne...*)

2. Prévoir (puis vérifier) la valeur affichée par l'instruction `anticonversion(conversion(2,3,2))` dans la console.