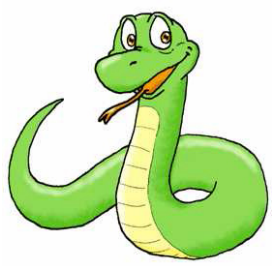
ECT1



TP 13 : Tracé de fonctions et de suites

Exercice 1 : Révision

On considère la fonction f définie par $f(x)=x+e^x$

1- Montrer que l'équation $f(x)=0$ admet une unique solution $\alpha \in \mathbb{R}$. De plus $\alpha \in [-1, 0]$.

2- Écrire un script `Python` permettant d'encadrer α dans un intervalle d'amplitude 10^{-2}

Nous allons travailler avec des vecteurs donc tous les exercices débutent par:
```
import numpy as np
```

Pour faire des représentations graphiques, on doit par importer la bibliothèque matplotlib.pyplot avec la commande `import matplotlib.pyplot as plt`

Exercice 2 : La commande np.arange

1- Évaluer à la console `np.arange(1,5)`

2- Évaluer à la console `3*np.arange(1,5)`

3- Évaluer à la console `np.arange(1,5,2)`

4- Évaluer à la console `np.arange(1,5,0.1)`

5- Évaluer à la console `np.arange(1,5,-1)`

6- Évaluer à la console `np.arange(5,1,-1)`

7- Évaluer à la console `np.arange(5)`

Bilan 1: la commande np.arange
np.arange(début , fin , pas)

La commande `np.arange()` a les mêmes arguments que la commande range() mais renvoie un vecteur. Ce qui permet, à la différence de `range()`, d'utiliser les opérations de bases

Si `début` non spécifié alors le début est 0 par défaut
Si `pas` non spécifié alors le pas est 1 par défaut

Contrairement à `range()`, `np.arange()` accepte des arguments qui ne sont pas des entiers

Exercice 3 : la commande np.linspace

1- Évaluer à la console `np.linspace(3,12,5)`

2- Évaluer à la console `np.linspace(3,12,10)`

3- Évaluer à la console `np.linspace(3,12,50)`

4- Évaluer à la console `np.linspace(3,12)`

Bilan 2: la commande np.linpace
np.linspace(a, b, n)

`np.linspace(a,b,n)` renvoie un vecteur contenant n valeurs comprises entre a et b (**a et b inclus**) régulièrement espacées

`n` un argument optionnel dont la valeur par défaut est 50.

Remarques :
Les opérateurs `np.linspace()` et `np.arange()` représentent deux réponses différentes au problème de créer des vecteurs lignes de valeurs équidistantes.

- L'opérateur `np.linspace()` met l'accent sur le nombre d'éléments du vecteur. Le pas utilisé s'en déduit.

- L'opérateur `np.arange()` met quant à lui l'accent sur le pas utilisé. Les éléments s'en déduisent.

Exercice 4 : Premier graphique

Nous allons travailler avec des vecteurs donc tous les exercices débutent par:
`import numpy as np`

Pour faire des représentations graphiques, on doit par importer la bibliothèque matplotlib.pyplot avec la commande `import matplotlib.pyplot as plt`

Considérons une fonction f définie sur Df. En mathématique, on appelle graphe d'une telle fonction et on note Cf l'ensemble : $Cf=\{ (x,f(x)),\ x \in Df \}$
Formellement, la courbe de f correspond au tracé de l'ensemble infini de ces points. Concrètement, on ne peut tracer à la main chacun de ces points. On se contente donc d'obtenir une allure de la courbe à l'aide d'une étude : sens de variations, calcul des tangentes aux points d'intérêt, calcul des asymptotes, intersection avec les axes.
L'approche Python est différente de l'approche mathématique consistant à obtenir l'allure de la courbe. C'est en effet l'approche « point par point » qui est retenue. Et même si seul un nombre fini de points peut être représenté, la puissance de calcul permet que ce nombre soit suffisamment grand pour fournir une très bonne approximation de la courbe souhaitée. Le nuage de points obtenu est alors complété en reliant les paires de points successifs, ce qui permet d'obtenir un tracé continu.
On utilisera ensuite la commande `plt.plot()` pour faire le tracé

1- Écrire le script suivant et l'exécuter . Que se passe t-il ?

```
1  import numpy as np
2  import matplotlib.pyplot as plt
3  X=np.array([-2,-1,0,1,2])
4  Y=np.array([3,-1,1,0,1])
5  plt.plot(X,Y)
```

2- Écrire le script suivant et l'exécuter . Que se passe t-il ?

```
import numpy as np
import matplotlib.pyplot as plt
X=np.array([-2,-1,0,1,2])
Y=np.array([3,-1,1,0,1])
plt.plot(X,Y)
plt.show()
```

A noter que la fonction `plt.plot()` seule ne permet pas l'affichage de la courbe à l'écran. Il faut lui ajouter à la suite la fonction `plt.show()` qui remplit ce rôle.

Pour tracer des fonctions, Python procède par interpolation de points (tracé de segments entre deux points donnés par l'utilisateur et déterminés par leur abscisse et leur ordonnée).

Remarque 1 : la syntaxe `plt.plot()` accepte en entrée des types plus généraux que des vecteurs. Par exemple, X pourrait être de type range, ou une liste et de même pour Y. Mais dans le programme officiel, les listes ne sont pas mentionnées, donc on appliquera ces commandes à des vecteurs.

Remarque 2 : dans les menus de la figures , vous pouvez modifier allure, echelle… et revenir à l'écran précédent en appuyant sur les flèches situées en haut de l'écran figure 1

3- Écrire le script suivant et l'exécuter . Que se passe t-il ? Ne pas oublier de fermer la figure 1

```
import numpy as np
import matplotlib.pyplot as plt
X=np.array([-2,-1,0,1,2])
Y=np.array([3,-1,1,0,1])
plt.plot(Y)
plt.show()
```

Exercice 5 : Mise en forme du graphique précédent

1- Autour du repère

Écrire le script suivant et l'exécuter . Que se passe t-il ?

```
import numpy as np
import matplotlib.pyplot as plt
X=np.array([-2,-1,0,1,2])
Y=np.array([3,-1,1,0,1])
plt.plot(X,Y,label='graphique 1')
plt.legend()
plt.grid()
plt.show()
```

Il existe une multitude d'améliorations. Voici une liste non exhaustive

Commande	Signification
`plt.xlim(`xmin,xmax`)`	restreint la représentation graphique à l'intervalle [xmin, xmax] en abscisse
`plt.ylim(`ymin, ymax`)`	restreint la représentation graphique à l'intervalle [ymin, ymax] en ordonnée
`plt.xlabel('`nom axe abscisses`')`	donne un titre à l'axe des abscisses
`plt.ylabel('`nom axe ordonnées`')`	donne un titre à l'axe des ordonnées
`plt.title('`nom du graphique`')`	donne un titre au graphique
`plt.legend()`	affiche la légende définie par `label=`'nom' dans `plt.plot()`
`plt.axis('equal')`	pour que le repère soit orthonormé
`plt.grid()`	ajoute une grille au graphique
`plt.figure()`	crée une nouvelle figure

Remarque: extrait du BO

Représentations graphiques de fonctions, de suites. On pourra utiliser les ***commandes non exigibles*** *xlim, ylim, axis, grid, legend….*

2- Couleur et style

Écrire le script suivant et l'exécuter . Que se passe t-il ?

```
import numpy as np
import matplotlib.pyplot as plt
X=np.array([-2,-1,0,1,2])
Y=np.array([3,-1,1,0,1])
plt.plot(X,Y,'r-.',label='graphique 1')
plt.legend()
plt.grid()
plt.show()
```

Bilan 4: `commande plt.plot et plt.show`
`plt.plot(X,Y,'option',label='nom)`

Considérons des points du plan dont on connaît les abscisses et les ordonnées.
Disons $A_1(x_1, y_1)$, $A_2(x_2, y_2)$, . . . , $A_n(x_n, y_n)$.
On implémente ces points en Python en mettant x_1, . . . ,x_n dans un vecteur X et y_1, . . . , y_n dans un vecteur Y
Il est impératif que X et Y ont le même nombre d'éléments

- `plt.plot(X,Y)` relie les points A_1, A_2, . . . , A_n (dans cet ordre).
- `plt.show()` affiche la représentation graphique. Cette commande doit être mise à la toute fin (après avoir appelé différents plt.plot et ajouté d'éventuelles options).

Les options est une chaîne de caractères (donc entre guillemets ou apostrophes) qui regroupe la couleur de la courbe, le marqueur de point et le style de liaison entre les points.
Voir ci-dessous les tableaux des différentes options qui sont très simples à mémoriser

Enfin si un nom est donnée ne pas oublier la commande `plt.legend()`

Couleur	Syntaxe
Rouge	'r'
Vert	'g'
Bleu	'b'
Cyan	'c'
Magenta	'm'
Jaune	'y'
Noir	'k'

Style	Syntaxe
Pointillé	' _ '
Lignes en pointillés	' : '
Traits/points	' -. '
Cercle	' o '
Triangle	' v '
Carré	' s '
Signe +	' + '
Croix	' x '

Par défaut , si vous ne spécifiiez pas de couleur, la couleur sera bleu

Si on fait appelle plusieurs fois à la commande `plt.plot()`, les courbes successives seront tracées dans la même fenêtre. Si on veut changer de fenêtre, on utilise la commande `plt.figure()`.

Exercice 6 : première fonction

1- Écrire le script suivant et l'exécuter . Que se passe t-il ?

```
1  import matplotlib.pyplot as plt
2  X=np.array([-2,-1,0,1,2])
3  Y=X**2
4  plt.plot(X,Y,'m', label='fonction carré')
5  plt.legend()
6  plt.grid()
7  plt.show()
```

Comme Python relie des points, il est nécessaire d'en avoir beaucoup pour que le résultat ressemble à une courbe

2- Écrire le script suivant et l'exécuter . Que se passe t-il ?

```
import numpy as np
import matplotlib.pyplot as plt
plt.figure()
def f(x):
    return np.log(x-1)

X=np.arange(-5,5,0.01)
Y=f(X)
plt.plot(X,Y,'k',label="Cf")
plt.legend()
plt.show()
```

Comment tracer une fonction *f* sur [*a*,*b*] avec `plt.plot(X,Y..)`

Ne pas oublier d'importer les bibliothèques numpy et matplotlib.pylot

Comment définir le vecteur abscisse X ?
`X=np.arange(a, b, p)` où p le pas est petit ,par exemple 0.01
ou `X=np.linspace(a, b, n)` où n le nombre de points est important, par exemple 100

Comment définir le vecteur ordonnée Y ?

- via les fonctions usuelles : en effet toutes les fonctions usuelles peuvent être appliquées à un vecteur (donc en particulier au vecteur d'abscisses X) et renvoient alors le vecteur des images correspondantes.

- via une fonction que vous avez crée au préalable, et qui peut renvoyer un vecteur.

Programme type pour tracer une fonction *f* sur [*a*,*b*]

```
import numpy as np
import matplotlib.pyplot as plt
plt.figure()
def f(x):
    return ..............

X=.............. np.arrange ou np.linspace
Y=f(X)
plt.plot(X,Y,'…….',label="……..")  couleur + nom du graphique (optionnel)
plt.legend()  (obligatoire si label)
plt.show()
```

<u>Exercice 7</u> : fonctions

Tracer les fonctions suivantes sur les intervalles données

1- f définie par $f(x)=\frac{3x+1}{x-2}$ sur [-3, 7]

2- g définie par $g(x)=\sqrt{x+1}$ sur [-1, 10]

3- h définie par $h(x)=e^x$ sur [-5, 5]

4- i définie par $i(x)=3\ln(x)-7$ sur]0, 10]

Exercice 8 : Tracé de suites

Le style pour les suites est obligatoire

Comment tracer une suite avec `plt.plot(X,Y,'+')`

Ne pas oublier d'importer les bibliothèques numpy et matplotlib.pylot

Comment définir le vecteur abscisse X ?
Si la suite debute à u_0, inutile de définir X
`X=np.arange(`début, fin+1`)` où p le pas est omis puisque par défaut il vaut 1

Comment définir le vecteur ordonnée Y ?
Il faut construire la suite dans le vecteur Y qu'on notera u

dans les options ne pas oublier de mettre un style comme 'x' sinon les points vont être reliés.

1- Soit (u_n) la suite définie par $u_0 = \frac{3}{4}$ et $\forall\, n \in \mathbb{N}$, $u_{n+1} = \frac{{u_n}^2}{2} + \frac{4}{7}u_n$

Écrire un programme Python qui représente les n premiers termes de la suite

2- Soit (u_n) la suite définie par $u_1 = 1$ et $\forall\, n \in \mathbb{N}^*$, $u_{n+1} = 1 + \dfrac{2}{u_n}$

Écrire un programme Python qui représente les n premiers termes de la suite