TP 2 : Fonctions en Python

Dans ce TP, on fera essentiellement des fonctions en Python.

Exercice 1 *Avant de commencer (pour rappel)*

Démarrer le logiciel Spyder puis, dans le menu « Fichier » créer un nouveau fichier puis le sauvegarder avec un nom sous la forme TP2_votrenom. Se souvenir des touches de raccourci. Dans ce TP, on écrira la quasi intégralité du code dans l'éditeur de texte sauf pour tester les fonctions créées.

I Rappels sur les représentations graphiques

Exercice 2 *Fonctions réciproques*

1. Charger les modules `matplotlib.pyplot` et `numpy` sous les noms `pypl` et `np`.
2. Revoir la manière de faire des représentations graphiques dans l'éditeur de texte (on pourra ouvrir le fichier `.py` du TP précédent avec la fiche de TP). Représenter la fonction sh sur $[-2, 2]$ (on pourra charger des modules). On utilisera un code de la forme :
   ```
   abscisse = np.linspace(···)
   ordonneesh = [···]
   pypl.plot(abscisse,ordonneesh)
   ```
3. Ajouter les lignes, à la suite de votre code précédent :
   (a) `pypl.grid()`
   (b) `pypl.axhline(color='black')`
   (c) `pypl.axvline(color='black')`
   (d) `pypl.legend(['fonction sh'],loc='upper left')`
   (e) `pypl.axis('equal')`
4. Pourquoi sh réalise-t-elle une bijection de $\mathbb{R}$ sur $\mathbb{R}$? Exécuter la commande `pypl.plot(ordonneesh,abscisse)`. Qu'a-t-on représenté ?
5. Représenter également sur le graphique précédent, la fonction $x \mapsto \ln(x+\sqrt{x^2+1})$ sur $[-2, 2]$. Que peut-on conjecturer ?
6. *Question facultative :* Démontrer la conjecture précédente.

On va désormais écrire le script précédent sous la forme d'une fonction Python. On rappelle qu'avec les notations précédentes, `pypl.clf()` permet d'effacer le graphe des fonctions précédentes.

Exercice 3 *Une fonction pour faire des représentations graphiques*

Écrire une fonction Python `graph_sh_recip(a,b)` qui prend en argument les bornes a et b d'un intervalle et qui renvoie la représentation graphique de la fonction sh et de sa réciproque sur l'intervalle $[a, b]$. À l'intérieur de cette fonction, on commencera par effacer le graphe avant toute représentation et on n'utilisera pas de légende. Testez ensuite votre fonction Python.

II Calcul numérique

On a montré en TD que l'unique solution de l'équation d'inconnue $x \in \mathbb{R}$:

$$\arctan(x) + \arctan(2x) = \frac{\pi}{4}$$

est $\dfrac{-3+\sqrt{17}}{4}$. Les exercices suivants permettent d'avoir une approximation de l'équation.

Exercice 4 *Résolution graphique*

1. Définir une fonction Python $g : x \mapsto \arctan(x) + \arctan(2x)$ (vous aurez peut-être besoin de modules supplémentaires).
2. Représenter graphiquement la fonction g puis déterminer graphiquement une valeur approchée de la solution de l'équation d'inconnue $x \in \mathbb{R}$: $\arctan(x) + \arctan(2x) = \frac{\pi}{4}$ (on pourra tracer la droite $y = \frac{\pi}{4}$).
3. Calculer une valeur approchée de $\dfrac{-3+\sqrt{17}}{4}$ et vérifier que votre approximation précédente est bonne.

Exercice 5 *Résolution numérique par Python*

1. Importer le module `scipy.optimize` sous le nom `op`.
2. Créer une fonction Python $P : x \mapsto x^2 + x - 1$ et puis exécuter la commande `>>> op.fsolve(P,0)` puis `>>> op.fsolve(P,-1)`. Interpréter ces résultats.
3. Avec cette fonction, déterminer une valeur approchée de la solution de l'équation d'inconnue $x \in \mathbb{R}$: $\arctan(x) + \arctan(2x) = \frac{\pi}{4}$.

III Interaction avec l'utilisateur

On rappelle que pour afficher des « mots » ou plus précisément des chaînes de caractères, on peut utiliser, dans l'éditeur de texte

```
x = 'texte a afficher'
print(x)
```

ou, à l'intérieur d'une fonction, on peut aussi utiliser `return x`.

Exercice 6 *Un programme d'affichage*

1. Créer une variable `mon_prenom` qui contient une chaîne de caractères avec votre prénom.
2. Faire une fonction Python `bienvenue(nom)` (où nom est une chaîne de caractères) qui affiche

   'Bonjour nom' .

   où nom variera selon la chaîne de caractères entrée.

Exercice 7 *Un programme de bienvenue*

1. On donne le programme suivant :

   ```
   def test():
       a = input('Entrer un objet')
       return a
   ```

   Tester ce programme et expliquer son fonctionnement. Taper dans la console `>>> b = test()`, entrer un objet puis déterminer le type de `b`.
2. Tester dans la console `>>>'age :'+25` puis `>>>'age :'+'25'` et enfin `>>>'age :'+str(25)`. En utilisant l'aide de Python pour la fonction `str`, expliquer ce qu'a retourné chaque commande.
3. Construire un programme `bienvenue()` qui ne prend pas d'argument, mais qui demande le prénom et l'âge de l'utilisateur (avec la fonction `input`), puis renvoie suivant ces données un message de bienvenue du style :

   'Bonjour Damien, vous avez 18 ans'

IV Manipulation d'heures

Exercice 8 *Conversions*

1. Déterminer, avec Python combien font 4127 secondes en heures, minutes et secondes (on pourra utilisera `//` avec 3600, puis 60) puis combien font $2h3m25s$ en secondes.
2. Créer une fonction Python `conversion_heures_secondes(h,m,s)` qui transforme les heures, minutes, secondes en secondes et résultat sous la forme

   '1h1m1s font 3661 secondes'

3. Créer une fonction Python `conversion_secondes_heures(s)` qui transforme les secondes en heures, minutes, secondes et renvoie le résultat sous la forme

   '3661 secondes font 1h1m1s'

Exercice 9 *Somme d'heures*

Créer une fonction Python `sommes_heures(h1,m1,s1,h2,m2,s2)` qui renvoie la somme des heures entrées et renvoie le résultat sous la forme

'1h2m4s + 2h5m8s = 3h7m12s'

Exercice 10 *Voyages-SNCF*

1. Entrer la commande `from time import *` puis tester la commande `>>>localtime()` puis tester également `>>>localtime()[0]`, `>>>localtime()[5]` par exemple.
2. Créer une fonction Python `heure_train(h,m,s)` où lorsque vous entrez l'heure de départ d'un train sous la forme `>>> heure_train(16,11,0)`, la fonction retourne

   'Votre train part dans 1h2m25s'

3. Modifier la fonction précédente pour qu'elle affiche un message `'Votre train est parti.'` lorsque vous avez entré une heure dépassée.

V Compléments

Exercice 11 *Une étude de suite*

Créer une fonction Python `serie(alpha,n)` qui pour un entier $n \in \mathbb{N}^*$ et un réel α, retourne la valeur de

$$S_n = 1 + \frac{1}{2^\alpha} + \cdots + \frac{1}{n^\alpha} = \sum_{k=1}^{n} \frac{1}{k^\alpha}.$$

Déterminer la comportement de la suite (S_n) (ceci dépend de α). On pourra faire des représentations graphiques.

Exercice 12 *Racines de l'unité*

Créer une fonction Python `racines_unite_graphe(n)` qui représente, pour $n \in \mathbb{N}^*$, dans le plan complexe, les points M_k d'affixe $z_k = e^{\frac{2ik\pi}{n}}$ pour $k = 0, \ldots, n-1$. On utilisera le module `cmath` ainsi que l'aide.

Exercice 13 *Résolution des équations du second degré à coefficients complexes*

Créer une fonction Python `racines_trinome_complexes(a,b,c)` où, pour trois complexes a, b, c données, la fonction retourne les racines de $aX^2 + bX + c$.