Créer et tracer une fonction très simple

Cet exemple vous montrera la syntaxe pour tracer une fonction très simple. On tracera la fonction $f(x) = x^2$.

```
def f(x): #Définition de la fonction f(x)=x**2
    return x**2

x = linspace (-10,10,100) # créer une liste de points dans [-10,10] de 100 valeurs
y = f(x) # calcul les valeurs de y selon f pour tous x

plot(x,y) # affiche y = f(x)
show()# montrer à l'ecran
```

1 Créer et tracer une fonction

Pour cet exercice, on importera les packages `numpy, matplotlib` à l'aide de la commande:

```
from matplotlib.pylab import *
```

Question 1 – Créer une fonction `fH(x)` définie de la manière suivante:

$$fH(x) = \begin{cases} 2 \text{ si } x \in [-\infty, 20] \\ 2 + e^{-2} - e^{0.2x-6} \text{ si } x \in [20, +\infty]. \end{cases}$$

```
def fH(x):
    if x<20:
        return 2
    else:
        return 2+exp(-2)-exp(0.2*x-6)
```

Question 2 – Créer un vecteur x de $n = 1000$ points entre 0 et 34 à l'aide de la commande `linspace` de `numpy`.

```
x = linspace (0, 34, 1000) # Crée 1000 points dans [0,34]
```

Question 3 – Créer un vecteur H de n points à partir de la fonction fH.

```
# Methode classique par utilisation de liste
H = []
for i in range(len(x)):
    H.append(fH(x[i]))

# Methode par utilisation de liste en concaténation
H = [fH(s) for s in x] # Solution en concaténation

# Methode avec un vecteur numpy
H = zeros(len(x)) # Créer une liste de 0 de talle 1000
for i in range(len(x)):
    H[i] = fH(x[i])

# Méthode comme l'exemple. Dans ce cas, on doit modifier un peu le calcul de f pour utiliser
↪  l'écriture vectorielle: H = fH(x)
def fH(x):
    if (x>20).any(): # Permet d'utiliser le calcul vectoriel : H = fH(x)
        return 2+exp(-2)-exp(0.2*x-6)
    return 2 # Ce return est équivalent au else

x = np.linspace (0, 34, 1000)
H = fH(x)
```

Question 4 – Tracer la fonction H en fonction de x à l'aide de la commande `plot` puis `show()`

Question 5 – **Bonus :** vous pouvez essayer de changer les couleurs (`color = "red"`), la taille etc... Vous pouvez également ajouter des labels à l'aide de `xlabel, ylabel, title,...`

```
plot(x,H,color="red",linewidth=2,linestyle="-.")
xlabel("x")
ylabel("y")
title('H(x)')
grid('on')
show()
```

2 Nombres narcissiques

Un nombre narcissique est un nombre où la somme de tous ses chiffres à la puissance 3 est égal à lui même. Par exemple, $153 = 1^3 + 5^3 + 3^3$ est un nombre narcissique.

Question 6 – Ecrire une fonction `SommeCube(n)` qui renvoie la somme de tous les chiffres au cube. Par exemple, `SommeCube(12)` doit renvoyer $1^3 + 2^3 = 9$.

```python
# Version classique en utilisant les str
def SommeCube(n):
    n = str(n) # Passe l'entier (int) en str => Permet de pouvoir parcourir les chiffres
    s = 0 # Créer un sommeur
    for chiffre in n: # parcour les chiffres
        chiffre = int(chiffre) # passe le chiffre (str) en entier (int)
        s += chiffre**3 # somme les chiffres
    return s

# Version condensée en concaténation + sommeur
def SommeCube(n):
    return sum([int(i)**3 for i in str(n)])

# Version 1 ligne (juste pour le fun)
SommeCube = lambda n: sum([int(i)**3 for i in str(n)])

# Version en passant par des divisions euclidiennes:
def SomCube(n):
    s=0
    size=len(str(n))
    i=1
    while i<=size: # parcours tous les chiffres de n
        a=n%10     # récupère le reste de la division euclidienne par 10. Ex : 145%10 = 5
        n=n//10 # Divise par 10. Ex: 145//10 = 14
        s+=a**3 # Somme
        i+=1 # Ajoute une incrémentation à i
    return(s)
```

Question 7 – En faisant une boucle, énumérer tous les nombres narcissiques inférieurs à 1000

```python
for i in range(1000):
    if SommeCube(i)==i: # Si SommeCube(i)==i alors ça renvoie True, sinon False
        print(i)
# Réponse 0, 1, 153, 370, 371, 407
```

3 Palindrome - lire un fichier

Un palindrome est un mot qui peut se lire indifféremment de gauche à droite ou de droite à gauche en gardant le même sens. Par exemple, le mot *kayak* est un palindrome.
Pour cet exercice, vous aurez besoin du fichier dictionnaire `liste_francais.txt` que vous pourrez trouver sur le lien :`https://www.freelang.com/download/misc/liste_francais.zip` ou sur la clé que je vous fournierai.

Question 8 – Ecrire une fonction `inverseMot(mot)` qui renvoie une chaine de caractère à l'envers.

```
def inverseMot(mot):
    ch = '' # Crée une chaine de caractère vide
    for i in range(len(mot)-1,-1,-1): # Parcours du dernier terme de la liste au premier avec
    ↪  un pas de -1 (parcours à l'envers)
        ch+=mot[i] #ajoute à la chaine de caractère la lettre i
    return ch
```

Question 9 – Ecrire une fonction `estPalindrome(mot)` qui va utiliser la fonction `inverseListe` et renvoyer **True** si le mot est un palindrome et **False** si le mot n'est pas un palindrome.

```
def estPalindrome(mot):
    invMot=inverseMot(mot) #inverse le mot
    return invMot==mot # Si le mot est pareil: True, sinon False
```

Question 10 – Essayer cette fonction avec: `estPalindrome("kayak")` et vérifier qu'elle renvoie **True**.

```
print(estPalindrome("kayak"))
```

Question 11 – Créer une fonction `lecture(file)` qui prends en argument un nom de fichier et retourne une liste de mots.
Aide: Pour retirer les saut de ligne "\n", on peut utiliser `line.rstrip('\n')`.

```
def lecture(file):
    L = []
    file = open(file,'r')
    for line in fine: #parcours les lignes du fichier
        L.append(line.rstrip('\n')) #couper le saut de ligne
    return L
```

Question 12 – Faire une boucle avec un compteur permettant de compter combien il y a de palindromes dans la langue française.

```
# Version classique
for mot in lecture("liste_francais.txt"): #Parcours les mots dans le dictionnaire
    if estPalindrome(mot):
        c+=1

# Version abrégée
c = 0
for mot in lecture("liste_francais.txt"): #Parcours les mots dans le dictionnaire
    c+= estPalindrome(mot)
```

Question 13 – Modifier la boucle pour sauvegarder tous les palindromes de la langue française.

```
L = [] # Crée une liste de stockage
for mot in lecture("liste_francais.txt"):
    if estPalindrome(mot):
        L.append(mot)
        c+= 1
```

```
print(L) # ['axa', 'bob', 'CAC', 'elle', 'été', 'ici', 'kanak', 'non', 'pop', 'radar', 'RER',
↪  'sas', 'serres', 'ses', 'SOS', 'tôt']
```