Révisions ITC

Ce document reprend les principales fonctions vues en cours ou TP. Vous devez comprendre et maîtriser le principe de chacun de ces algorithmes et savoir les adapter en fonction des situations rencontrées. La liste n'est pas exhaustive et vous pourrez la compléter par vous-même par d'autres fonctions.

1 Quelques fonctions sur les entiers et/ou flottants

- Calcul de la somme $\sum_{i=1}^{n} i$:

Code Python
```python
def somme(n) :
    s = 0
    for i in range(1, n+1) :
        s += i      #ou s = s+i
    return s
```

- Indique si n est premier :

Code Python
```python
def estPremier(n) :
    if n <= 1 :
        return False
    for d in range(2, n) :
        if n % d == 0 :
            return False
    return True
```

- Pour $n \geqslant 0$, renvoie u_n où : $\begin{cases} u_0 = 4 \\ \forall n \geqslant 0 : u_{n+1} = u_n^2 - u_n + 1 \end{cases}$

Code Python
```python
def calcul_u(n) :
    u = 4
    for i in range(1, n+1) :
        u = u*u - u + 1
    return u
```

- Pour $n \geqslant 0$, renvoie F_n où : $\begin{cases} F_0 = 0, F_1 = 1 \\ \forall n \geqslant 0 : F_{n+2} = F_{n+1} + F_n + 1 \end{cases}$

Code Python
```python
def fibo(n) :
    x, y = 0, 1
    for i in range(1) :
        tmp = y
        y = y + x
        x = tmp
    return x
```

- Pour $n \geqslant 2$, renvoie le plus grand entier $d < n$ tel que d divise n :

Code Python
```python
def plusGrandDiviseur(n) :
    for d in range(n-1, 0, -1) :
        if n % d == 0 :
            return d
```

- Renvoie la somme des chiffres d'un nombre :

Code Python
```python
# Version 1
def sommeChiffres(n) :
    s = 0
    while n > 0 :
        s = s + (n%10)
        n = n//10
    return s
```

Code Python
```python
# Version 2
def sommeChiffres(n) :
    s = 0
    for c in str(n) :
        s = s + int(c)
    return s
```

2 Algorithmes classiques sur les listes

- Calcul de la somme des éléments d'une liste :

Code Python
```python
# Version 1
def somme(L) :
    s = 0
    for k in L :
        s += k
    return s
```

Code Python
```python
# Version 2
def somme(L) :
    s = 0
    for k in range(len(L)) :
        s += L[k]
    return s
```

- Détermination du maximum ou du minimum :
On suppose que la liste étudiée n'est pas vide.

Code Python
```python
def maximum(L) :
    M = L[0]
    for k in L :
        if k > M :
            M = k
    return M
```

Code Python
```python
def minimum(L) :
    m = L[0]
    for k in L :
        if k < m :
            m = k
    return m
```

- Détermination de l'indice du maximum ou du minimum :
On suppose que la liste étudiée n'est pas vide.

Code Python
```python
def indiceMaximum(L) :
    M = L[0]
    iMax = 0
    for k in range(len(L)) :
        if L[k] > M :
            M = L[k]
            iMax = k
    return iMax
```

Code Python
```python
def indiceMinimum(L) :
    m = L[0]
    iMin = 0
    for k in range(len(L)) :
        if L[k] < m :
            m = L[k]
            iMin = k
    return iMin
```

- Indique si e est un élément d'une liste ou d'une chaîne de caractére :

Code Python
```python
def appartient(L, e) :
    for a in L :
        if a == e :
            return True
    return False
```

Code Python
```python
def appartient(L, e) :
    for k in range(len(L)) :
        if L[k] == e :
            return True
    return False
```

- Renvoie l'occurence d'un élément :

Code Python
```python
# Version 1
def occurence(L, e) :
    occ = 0
    for k in L :
        if k == e :
            occ += 1
    return occ
```

Code Python
```python
# Version 2
def occurence(L, e) :
    occ = 0
    for k in range(len(L)) :
        if L[k] == e :
            occ += 1
    return occ
```

- Recherche du second maximum dans une liste :

Code Python
```python
def secondMax(L) :
    M  = maximum(L)  #maximum est la fonction définie précédemment
    debut = 0
    while L[debut] == M :
        debut += 1
    M2 = L[debut]
    for k in range(debut, len(L)) :
        if L[k] > M2 and L[k] != M :
            M2 = L[k]
    return M2
```

- Recherche des deux valeurs les plus proches dans une liste :

Code Python
```python
def deuxPlusProchesValeurs(L) :
    d = abs(L[0]-L[1])
    ind1, ind2 = 0, 1
    for i in range(len(L)) :
        for j in range(i) :
            e = abs(L[i]-L[j])
            if e < d :
                d = e
                ind1, ind2 = i, j
    return (L[ind1], L[ind2])
```

- Moyenne des éléments d'une liste à deux dimensions :

Code Python
```python
def moyenne_2D(L) :
    S = 0
    n = 0
    for l in L :
        n += len(l)
        for x in l :
            S += x
    return S/n
```

Code Python
```python
def moyenne_2D(L) :
    S = 0
    n = 0
    for i in range(len(L)) :
        n += len(L[i])
        for j in range(len(L[i])) :
            S += L[i][j]
    return S/n
```

3 Chaînes de caractères

- Indique si `mot` est un palindrome :

Code Python
```python
# Version 1
def estPalindrome(mot) :
    n = len(mot)
    for l in range(n//2) :
        if mot[i] != mot[n-1-i] :
            return False
    return True
```

Code Python
```python
# Version 2
def estPalindrome(mot) :
    return mot == mot[::-1]
```

- Recherche d'un mot dans un texte :

Code Python
```python
def rechercheMot(texte, mot) :
    #indique la présence de mot dans texte
    lmot = len(mot)
    ltexte = len(texte)
    for j in range(ltexte-lmot+1) :
        i = 0
        while i < lmot and texte[j+i] == mot[i] :
            i += 1
        if i == lmot :
            return True
    return False
```

4 Tri à bulles

Code Python

```python
# Version 1
def tri_bulles(L) :
    j = len(L)-1
    while j > 0 :
        for i in range(j) :
            if L[i] > L[i+1] :
                # on échange les termes
                L[i], L[i+1] = L[i+1], L[i]
        j = j-1
```

Code Python

```python
# Version 2
def tri_bulles(L) :
    for j in range(1, len(L)) :
        for i in range(len(L)-j) :
            if L[i] > L[i+1] :
                L[i], L[i+1] = L[i+1], L[i]
```

5 Tracé de graphiques

- **Tracé d'un nuage de points.**

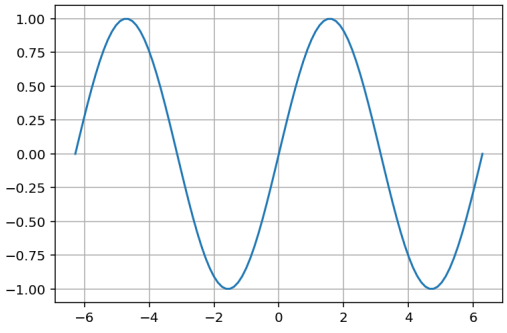
On veut tracer le nuage constitués des points de coordonnées (-2, 3), (-1, 0), (3, 1) et (5, 2).
On définit la liste des abscisses et celles des ordonnées puis on effectue la commande tracé avec `plt.plot(X, Y, 'x')`.
Le "x" est une option faisant apparaître les points avec un symbole de croix (on peut le changer en "o", ".", "v" ...).

Code Python

```python
# import des bibliothèques utiles
import matplotlib.pyplot as plt
# Définition des listes des abscisses et ordonnées
X = [-2, -1, 3, 5]
Y = [3, 0, 1, -2]
plt.plot(X, Y, "x") # Tracé du nuage de points
plt.grid()  #Tracé du quadrillage
plt.show() # Affichage du tracé
```

- **Tracé du graphe d'une fonction.**

Pour tracer le graphe d'une fonction f sur l'intervalle $[a, b]$, on définit la liste X des abscisses en discrétisant l'intervalle $[a, b]$ et Y celles des images par f.
On obtient le tracé suivant avec les instructions ci-contre :



Code Python

```python
# import des bibliothèques utiles
import matplotlib.pyplot as plt
import numpy as np
# Définition de la fonction
def f(x) :
    return np.sin(x)
# Définition de l'intervalle
a = -2*np.pi
b = 2*np.pi
# Nombre de points
n = 100
# Définition de la liste des abscisses et des ordonnées
X = [a+i*(b-a)/n for i in range(n+1)]
Y = [f(x) for x in X]
# Tracé de la courbe
plt.plot(X, Y)
plt.grid()  #Tracé du quadrillage
plt.show() # Affichage du tracé
```

- **Tracé d'un histogramme :** La liste des commandes ci-dessous donne l'histogramme ci-contre :

Code Python

```python
import matplotlib.pyplot as plt
X = [2, 2, 3, 4, 4, 4, 4, 5, 5, 5, 6, 6, 6, 6, 6, 6, 7, 7, 7, 8, 8, 9]
plt.hist(X, range=(0, 10), bins=10, color="yellow", edgecolor='red')
plt.xlabel('valeurs')
plt.ylabel('effectif')
plt.title("Exemple d'histogramme simple")
plt.show()
```