TD N°0: Rappel général - Corrigé

Exercice 1 : Crible d'Eratosthène - Corrigé

```python
# Question 1
def init():
    return [1]*100
    # ou bien
    L = []
    for i in range(100):
        L.append(1)
    return L

# Question 2
def multiple(L,i):
    for j in range(i+1,len(L)):
    if j % i == 0:
        L[j]=0

# Question 3
def suivant(L,i):
    j = i + 1
    while L[j] == 0:
        j +=1
    return j

# Question 4
def crible():
    from math import sqrt
    L = init()
    i=2
    while i < int(sqrt(len(L))):
        multiple(L,i)
        i = suivant(L,i)
    p=[]
    for i in range(2,100):
        if L[i]==1:
            p.append(i)
    return p

# Question 5
from math import sqrt
def crible_recursive(L=init(),i=2):
    if i > int(sqrt(len(L))):
        multiple(L,i)
        i = suivant(L,i)
        return crible_recursive(L,i)
    else:
        return [i for i in range(2,len(L)) if L[i]==1]

# Question 7
if __name__ == '__main__':
    from math import sqrt
    print(crible_recursive())
```

Exercice 2 : fonction passée en argument - Corrigé

```python
# Question 1
def carree(n):
    return n*n

# Question 2
def somme_fonction(f,n):
    #calcule et retourne la somme $\sum_{i=0}^{n} f(i)$
    s = 0
    for i in range(n):
        s += f(i)
    return s

#exemple : $n = 10$
somme_fonction(f,10) # 385
```

Exercice 3: Recherche dichotomique - Corrigé

```python
def dicho(x,L):
    if len(L)==1 :
        return 0
    else:
        center = len(L)//2
    if x >= L[center]:
        return center+dicho(x,L[center:])
    else :
        return dicho(x,L[:center])

#test la fonction
def main():
    L = [random.random()*100 for _ in range(10)]
    L.sort()
    # x est un réel vérifiant $L[0] \leq x < L[n-1]$
    x =  random.triangular(L[0],L[-1]);
    # print("x=",x)
    # dicho(x,t) retournant l'unique entier $i$ vérifiant $L[i] \leq x < L[i+1]$
    i = dicho(x,L)
    print("x=",x)
    print("i=",i)
    if not (L[i] <= x < L[i+1]):
        raise Exception

if __name__ == '__main__':
    import random
    n = 30
    # tester la fonction n fois
    for i in range(n):
        main()
    else:
        print("ok")
```

Exercice 4: Récursivité - Corrigé

```python
def est_palindrome(chaine):
    """Vérifie récursivement si une chaîne est un palindrome."""
    # Condition de base : si la chaîne est vide ou de longueur 1
    if len(chaine) <= 1:
        return True
    # Comparaison du premier et du dernier caractère
    if chaine[0] != chaine[-1]:
        return False
    # Appel récursif en enlevant le premier et le dernier caractère
    return est_palindrome(chaine[1:-1])

# Exemple d'utilisation
print(est_palindrome("RADAR"))  # True
print(est_palindrome("HELLO"))  # False
```

Exercice 5: Suite de polynômes - Corrigé

```python
# Question 1
def saisie_deg():
    while 1:
        try:
            n=int(input("deg="))
            if n > 0:
                break #arrêter la boucle while
        except ValueError:
            print("Erreur")
    # arrêter l'exécution de la fonction et retourner n
    return n

# Question 2
def saisie_poly(n):
    p = []
    for i in range(n+1):
        while 1:
            try:
                coef = float(input("coef[{}]= ".format(i))) # (f"P[{i}]=")
                if (i<n) or (i == n and coef != 0):
                    p.append(coef)
                    #arrêt de while et passer au coef P[i] suivant
                    break
            except:
                continue
        return p

# Question 3
def derive(p):
    return [p[i] * i for i in range(1,len(p))]

# Question 4
def opp_poly(p):
    return [-coef for coef in p]

# Question 5
```

```python
def add_poly(p1,p2):
    n1, n2 = len(p1),len(p2)
    p3= [p1[i] + p2[i]  for i in range(min(n1,n2))]
    if n1 > n2 :
        p3 += p1[n2:n1]
    elif n2 > n1:
        p3 += p2[n1:n2]

    while len(p3)>1 and p3[-1]==0:
        p3.pop()

    return p3

# Question 6
def mul_poly(p1,p2):
    deg1 = len(p1)-1
    deg2 = len(p2)-1
    p3 = [0] * (deg1+deg2+1)

    for i in range (len(p1)):
        for j in range(len(p2)):
            p3[i+j] += p1[i]*p2[j]

return p3

# Question 7
def main():
    n = saisie_deg()
    p = saisie_poly(n)

    while 1:
        try:
            i = int(input("i="))
            break
        except :
            continue

    sp0 = p
    sp1 = opp_poly(derive(p))

    if i==0: print(sp0)
    elif i==1: print(sp1)
    else:
        for k in range (2,i+1):
            q =  mul_poly(sp1,sp0)
            spk = add_poly(mul_poly(q,sp1),sp0)
            sp0 = sp1
            sp1 = spk
        print(spk)
```

Exercice 6: Les fichiers - Corrigé

Question 1

```python
def ecrire_fichier(fichier, donnees):
    """Écrit une liste de chaînes de caractères dans un fichier.
    Remplace le contenu du fichier s'il existe déjà."""
    with open(fichier, 'w') as f:
        for enregistrement in donnees:
            f.write(enregistrement + '\n')

# Exemple d'utilisation
# Données à écrire dans le fichier
donnees = ["Ali, 20", "Fahmi, 25", "Jamel, 30"]

# Écrire les données dans le fichier
ecrire_fichier('fichier.txt', donnees)
```

Question 2

```python
def chercher_fichier(fichier, mot_cle):
    """Recherche un enregistrement contenant le mot clé et retourne l'enregistrement."""
    with open(fichier, 'r') as f:
        for ligne in f:
            if mot_cle in ligne:
                return ligne.strip() #  supprimer les espaces au début et à la fin de la ligne
    return None  # Si aucune correspondance n'est trouvée

# Chercher une ligne contenant le mot clé "Fahmi"
resultat = chercher_fichier('fichier.txt', 'Fahmi')
print(resultat)  # Affichera : Fahmi, 25
```

Exercice 7: Dictionnaire, set, tuple, fichier ... - Corrigé

Q1. Distance Euclidienne entre deux Points

```python
import math

def dist(p, q):
    """Retourne la distance euclidienne entre deux points p et q."""
    return math.sqrt((p[0] - q[0])**2 + (p[1] - q[1])**2)

# Exemple d'utilisation
p = (1, 2)
q = (4, 6)
print(dist(p, q))  # 5.0
```

Q2. Recherche des plus proches voisins

```python
def plus_proches_voisins(ensPts):
    """Retourne le couple de points les plus proches dans un ensemble."""
    points = list(ensPts)
    d_min = 0
    proches_voisins = points[0] , points[1]

    for i in range(len(points)-1):
        for j in range(i + 1, len(points)):
            d = dist(points[i], points[j])
            if d < d_min:
                min_dist = d
                proches_voisins = (points[i], points[j])

    return proches_voisins

# Exemple d'utilisation
ensPts = {(1, 2), (4, 6), (5, 7)}
print(plus_proches_voisins(ensPts))  # ((4, 6), (5, 7))
```

Q3. Chargement des villes à partir d'un fichier

```python
def loadCities(filename):
    """Charge les coordonnées des villes à partir d'un fichier texte."""
    Dcities = {}
    with open(filename, 'r') as file:
        for line in file:
            nom_ville, x, y = line.strip().split(':')
            x = float(x)
            y = float(y)
            Dcities[(x, y)] = nom_ville
    return Dcities

# Exemple d'utilisation
Dcities = loadCities('cities.txt')
print(Dcities) # {(10.63699,35.82539):"Sousse", ...}
```

Q4. Recherche des plus proches villes

```python
def plus_proches_villes(Dcities):
    """Retourne le couple de villes les plus proches en utilisant leurs coordonnées."""
    points = list(Dcities.keys())
    plus_proches_villes= plus_proches_voisins(points)
    return (Dcities[plus_proches_villes[0]], Dcities[plus_proches_villes[1]])

# Exemple d'utilisation
print(plus_proches_villes(Dcities))
```

Q5. Écriture des villes dans un fichier

```python
def writeCities(Dcities, nomF):
    """Écrit les coordonnées des villes dans un fichier."""
    with open(nomF, 'w') as file:
        for coords, nom_ville in Dcities.items():
            file.write(f"{nom_ville}:{coords[0]}:{coords[1]}\n")

# Exemple d'utilisation
writeCities(Dcities, 'output.txt')
```