PSI - Correction des exercices-types arts et métiers

Exercice 0

1. On trouve `q=123` et `r=4`.
   On réitère : `q=12` et `r=3`, `q=1` et `r=2`, `q=0` et `r=1`.
   On remarque que les restes successifs donnent l'écriture décimale de `n`, lue de droite à gauche.
2. Par exemple :

```
q=1234
r=0
s=0
for k in range(4):
    r=q%10
    q=q/10
    s+=r**3
print(s)
```

   On obtient `100`.
3. Par exemple :

```
def somcube(n):
    r=0
    s=0
    while n!=0:
        r=n%10
        n=n/10
        s+=r**3
    return s
```

4. Par exemple :

```
liste=[]
for k in range(1001):
    if k==somcube(k):
        liste=liste+[k]
print(liste)
```

   On obtient `[0,1,153,370,371,407]`.
5. Par exemple :

```
def somcube2(n):
    n=str(n)
    s=0
    for k in range(len(n)):
        s+=int(n[k])**3
    return s
```

Exercice 1

1. Par exemple :

```
f=open('ex_001.csv','r')
ligne=f.readline()
LX=[]
LY=[]
while ligne:
    lignesplit=ligne.split(';')
    LX=LX+[float(lignesplit[0])]
    LY=LY+[float(lignesplit[1])]
    ligne=f.readline()
f.close()
```

2. Par exemple :

```
import matplotlib.pyplot as plt

plt.plot(LX,LY)
plt.show()  #facultatif
```

3. Par exemple :

```
def trapeze(x,y):
    assert len(x)==len(y)
    n=len(x)
    s=0
    for k in range(n-1):
        s+=(x[k+1]-x[k])*(y[k+1]+y[k])/2.
    return(s)
```

4. Par exemple :

```
from scipy.integrate import trapz #accès à l'aide en ligne le jour de l'oral?

print(trapeze(LX,LY))
print(trapz(LY,LX))
# la méthode des trapèzes n'est pas au programme mais aucune connaissance n'est requise
```

Exercice 2

1. La matrice $M = \begin{pmatrix} 0 & 9 & 3 & -1 & 7 \\ 9 & 0 & 1 & 8 & -1 \\ 3 & 1 & 0 & 4 & 2 \\ -1 & 8 & 4 & 0 & -1 \\ 7 & -1 & 2 & -1 & 0 \end{pmatrix}$.

   Remarquer qu'elle est symétrique, par définition.

   ```
   from numpy import *

   M=matrix([[0,9,3,-1,7],[9,0,1,8,-1],[3,1,0,4,2],[-1,8,4,0,-1],[7,-1,2,-1,0]],dtype=int)
   ```

   On aurait aussi bien pu se dispenser de `numpy` en utilisant une liste de listes :

   `M=[[0,9,3,-1,7],[9,0,1,8,-1],[3,1,0,4,2],[-1,8,4,0,-1],[7,-1,2,-1,0]]`

   en prenant garde de remplacer les appels `M[i,j]` par `M[i][j]`.

2. Par exemple :

3. Par exemple :

   ```
   voisin4=[]
   for k in range(5): # 5 est le nombre de lignes de M
       if M[k,4]<=0: #teste si k voisin de 4
           voisin4.append(k) #si oui ajout dans la liste
   ```

4. Par exemple :

   ```
   def voisins(i):
       sortie=[]
       for k in range(5): # 5 est le nombre de lignes de M = nombre de sommets du graphe
           if M[k,i]!=-1 and M[k,i]!=0: #teste si k voisin de i
               sortie.append(k) #si oui ajout dans la liste
       return sortie
   ```

5. Par exemple :

   ```
   def degre(i):
       return len(voisins(i))
   ```

6. Par exemple :

   ```
   def longueur(L):
       n=len(L)
       if n==0: # pas de sommets dans le trajet
           trajet=-1
       elif n==1: # un seul sommet dans le trajet
           trajet=0
       else:  # au moins 2 sommets parcourus
           trajet=0
           i=0
           while trajet!=-1 and i<n-1:
               if M[L[i],L[i+1]]>0:
                   trajet+=M[L[i],L[i+1]]
               else:
                   trajet=-1
               i+=1
       return trajet
   ```

Exercice 3

1. Par exemple :

```
def nombreZeros(t,i):
    sortie=0    # cas où t[i]=1
    n=len(t)
    if t[i]!=1:
        nb=0
        k=i
        while k<n and t[k]==0: #parcours de la liste tant que la case contient zéro
            nb+=1 #on compte un zéro de plus
            k+=1
        sortie=nb
    return sortie
```

2. Il suffit de déterminer l'élément maximal de la liste `nombreZeros(t,i)`.

```
def nombreZerosMax(t):
    liste=[]
    n=len(t)
    for i in range(n): # création de la liste suggérée dans l'énoncé
        liste.append(nombreZeros(t,i))
    max=liste[0]
    for i in range(1,n): # algorithme de recherche du maximum
                         # dans une liste de nombres
        if max<liste[i]:
            max=liste[i]
    return max
```

3. La complexité dans le meilleur des cas (liste constituée exclusivement de 1) est en $O(n)$, et dans le pire des cas (liste constituée exclusivement de 0) elle est en $O(n^2)$.

4. Par exemple :

```
def nombreZerosMax2(t):
    n=len(t)
    i=0
    max=0
    while i<n: #algorithme de recherche du maximum
        nbzero=nombreZeros(t,i)
        if max<nbzero: # teste si on trouve un nb de zéros contigus supérieur
                       # à ce qu'a donné le début de la liste
            max=nbzero
        i+=nbzero+1 # si on a nbzero zéros contigus, la recherche
                    # reprend après cette "série"
    return max
```

Cette fois la complexité dans le meilleur et dans le pire des cas est en $O(n)$.

Exercice 4

1. Par exemple :

```
from math import factorial,exp

def Px(k,n,p):
    return exp(-n*p)*(n*p)**k/factorial(k)
```

On obtient la liste demandée avec le script :

```
valeursX=[]
for k in range(31):
    valeursX.append(Px(k,30,0.1))
```

2. Par exemple :

```
from scipy.misc import comb

def Py(k,n,p):
    return comb(n,k)*p**k*(1-p)**(n-k)
```

On obtient la liste demandée avec le script :

```
valeursY=[]
for k in range(31):
    valeursY.append(Py(k,30,0.1))
print(valeursY)
```

3. Il s'agit de l'algorithme de recherche du maximum.

```
def Ecart(n,p):
    max=0
    for k in range(n+1):
        ecart=abs(Px(k,n,p)-Py(k,n,p))
        if ecart>max:
            max=ecart
    return max
```

4. Par exemple :

```
def N(e,p):
    n=1 # n non nul d'après l'énoncé; important car Ecart(0,p)=0
    while Ecart(n,p)>e:
        n+=1
    return n
```

5. Les instructions `N(0.008,0.075)`, `N(0.005,0.075)`, et `N(0.008,0.1)`, donnent respectivement `1`, `124` et `71`.
Par contre l'instruction `N(0.005,0.1)` donnent une « OverflowError ». Si on modifie la fonction `Px` ainsi :

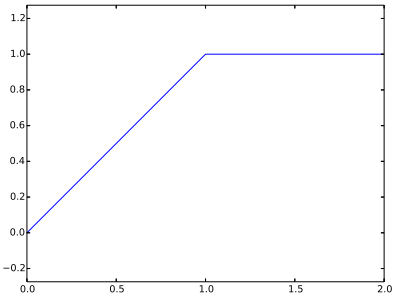
```
def Px(k,n,p): #classique: version par récurrence qui évite le calcul de k!
    temp=exp(-n*p)
    for i in range(k):
        temp*=n*p/(i+1)
    return temp
```

alors l'instruction `N(0.005,0.1)` donne `182`.
Il semblerait que pour `p=0.1` la convergence est plus lente que `p=0.075`.

Exercice 5 1. Par exemple :

```
import matplotlib.pyplot as plt

def g(x):
    assert 0<=x and x<2
    if 0<=x and x<1:
        return x
    if 1<=x<2:
        return 1

x=[k*0.01 for k in range(200)]
y=[g(k) for k in x]
plt.plot(x,y)
plt.axis("equal")
```

On obtient la figure :



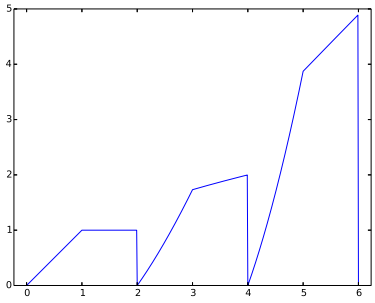
2. Par exemple :

```
from math import sqrt

def f(x):
    if 0<=x and x<2:
        return g(x)
    else:
        return sqrt(x)*f(x-2)
```

3. On utilise les instructions :

```
x=[k*0.01 for k in range(601)]
y=[f(k) for k in x]
plt.plot(x,y)
plt.axis("equal")
```

On obtient la figure :



4. Par exemple :

```
a=0.01
while f(a)<=4:
    a+=0.01
```

On obtient `a=5.13`.

Exercice 6

1. `d(4)` et `d(10)` renvoient respectivement `[1,2,4]` et `[1,2,5,10]`.
   La fonction `d` renvoie la liste des diviseurs positifs de l'entier naturel `n` donné en entrée.
2. Par exemple :

```
def DNT(n):
    L=[]
    for nombre in range(2,n):
        if n%nombre==0:
            L.append(nombre)
    return L
```

3. Par exemple :

```
def sommeCarresDNT(n):
    s=0
    L=DNT(n)
    for k in L:
        s+=k**2
    return s
```

4. On utilise les instructions :

```
liste=[]
for n in range(1000):
    if n==sommeCarresDNT(n):
        liste.append(n)
```

   On obtient : `[0, 4, 9, 25, 49, 121, 169, 289, 361, 529, 841, 961]`.
   On peut conjecturer que les entiers vérifiant la propriété sont des carrés (mais la condition n'est pas suffisante).

Exercice 7

1. `alphabet='abcdefghijklmnopqrstuvwxyz'`
2. Par exemple :

```
def decalage(n):
    chaine=''
    for k in range(26):
        chaine+=alphabet[(k+n)%26]
               # on peut aussi distinguer deux cas suivant que k+n>26 ou non
    return chaine
```

3. Par exemple :

```
def indices(x,phrase):
    liste=[]
    p=len(phrase)
    for k in range(p):
        if phrase[k]==x:
            liste+=[k]
    return liste
```

4. Par exemple :

```
def codage(n,phrase):
    phrase2=list(phrase) # on transforme la chaine en liste
                         # pour pouvoir modifier ses éléments
    alphabet2=decalage(n)
    for l in range(26):
        liste=indices(alphabet[l],phrase) # chaque lettre de l'alphabet
                                          # est recherchée puis décalée
        for k in liste:
            phrase2[k]=alphabet2[l]
    phrase=''
    for x in phrase2:
        phrase+=x # on retransforme la liste en chaîne
    return phrase
```

5. Pour décoder on décale de `26-n`.

Exercice 8

1. Par exemple :

```
M=20
m=10

def f(c):
    u=0
    k=0
    while abs(u)<=M and k<=m+1:
        u=u**2+c
        k+=1
    return k
```

2. On utilise le script :

```
import matplotlib.pyplot as plt

LX=[-2+k*4./400. for k in range(401)]
LY=[f(x) for x in LX]
plt.plot(LX,LY)
plt.show()
```

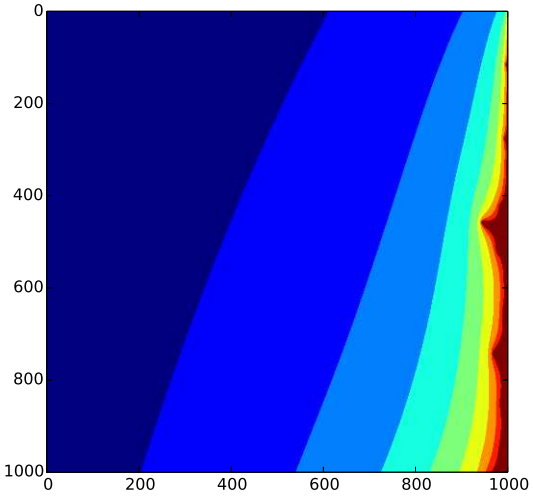
3. Par exemple :

```
import numpy as np

LX=[-2+k*0.5/1000. for k in range(1001)]
LY=[-1.1+k*1.1/1000. for k in range(1001)]
n=len(LX)
p=len(LY)
tableau=np.zeros((n,p))
for i in range(n):
    for l in range(p):
        tableau[i,l]=f(LX[i]+LY[l]*1j)
```

4. On utilise le script :

```
plt.imshow(tableau) #aide en ligne?
plt.show()
```

Pour améliorer la précision, on augmente le pas des subdivisions `LX` et `LY`.
Voici l'image obtenue avec 1000 points :



Exercice 9

1. Par exemple le script :

```
R=array([[1,2,3],[4,5,6]])
S=array([[1,2,3],[4,5,6],[7,8,9]])
print(R,S)
```

2. Par exemple :

```
def test(M):
    n,p=M.shape # ne pas utiliser shape?
    sortie=0
    if n==p:
        sortie=n
    return sortie
```

`test(R)` et `test(S)` donnent respectivement `0` et `3`.

3. On utilise le script :

```
f=open('ex_006.txt','r')
ligne=f.readline()
M1=zeros((5,5))
i=0
while ligne:
    lignesplit=ligne.split(' ')  #valeurs séparées par des espaces
    for j in range(5):
        M1[i,j]=float(lignesplit[j])
    ligne=f.readline()
    i+=1
f.close()
test(M1)
```

4. Par exemple le script :

```
propre=eig(M1)
print(propre[0])
```

5. Par exemple le script :

```
def dansIntervalle(L,a,b):
    sortie=True
    for x in L:
        if x<a or x>b:
            sortie=False
    return sortie

print(dansIntervalle(propre[0],0,1))
```

Exercice 10

1. Par exemple :

```
def comptage(L,N):
    P=[]
    for k in range(N):
        compteur=0
        for x in L:
            if x==k:
                compteur+=1
        P+=[compteur]
    return P
```

2. Par exemple :

```
def tri(L,N):
    Ltri=[]
    P=comptage(L,N)
    for k in range(N):
        for i in range(P[k]):
            Ltri+=[k]
    return Ltri
```

3. Par exemple le script :

```
L=[]
for k in range(20):
    L+=[randint(0,5)]
L
tri(L,6)
```

4. La complexité dans le pire des cas est en $O(N^2)$ (deux boucles `for` imbriquées).
   Elle est identique à celle du *tri par insertion*, et moins bien que celle du *tri fusion*.

Exercice 11

1. Par exemple :

```
def MaxiListe(L):
    max=L[0]
    indice=0
    n=len(L)
    for k in range(1,n):
        if max<L[k]:
            max=L[k]
            indice=k
    return max,indice
```

2. Par exemple :

```
def sup(f,N):
    L=[f(float(k)/N) for k in range(N+1)]
    return MaxiListe(L)
```

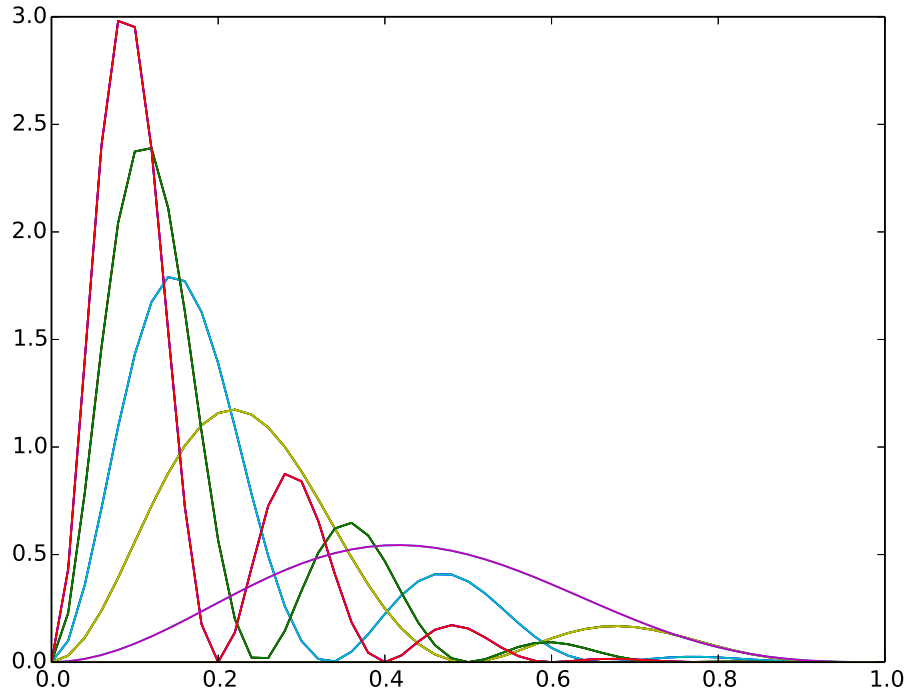
3. Par exemple :

```
from math import pi,sin

def F(n,x):
    return n*sin(n*pi*x)**2*(1-x)**n
```

4. Par exemple :

```
import matplotlib.pyplot as plt

def Graph(L):
    Lx=[k/100. for k in range(101)]
    for n in L:
        y=[F(n,x) for x in Lx]
        plt.plot(Lx,y)
```

`Graph(range(7))` donne :



5. Par exemple :

```
def PlusGrand(M,N):
    n=0
    L=[F(n,float(k)/N) for k in range(N+1)]
    max,abscisse=MaxiListe(L)
    while max<M:
        n+=1
        L=[F(n,float(k)/N) for k in range(N+1)]
        max,abscisse=MaxiListe(L)
    return n,abscisse,max
```

`PlusGrand(2,101)` et `PlusGrand(10,101)` donnent respectivement `(4, 11, 2.420301126604817)` et `(17, 3, 10.179438038033796)`.

6. Par exemple :

```
n,abscisse,max=PlusGrand(20,N)
test=False
Lx=[k/float(N-1) for k in range(N)]
while not(test):
    test=True
    for x in Lx:
        if x>=0.1 and F(n,x)>10**(-8):
            test=False
    n+=1
```

Avec `N=101`, `N=1001` et `N=5001` on trouve respectivement 101, 220 et 227.