Devoir d'informatique 2 – Corrigé

D.Malka – MPSI 2017-2018 – Lycée Saint-Exupéry

8.12.2017

Exercice 1 – Evaluation du nombre e

1. Fonction `fact` renvoyant la factorielle de l'entier naturel passé en argument.

```
def fact(n):
    facto=1
    i=0
    while i<n:
        i=i+1
        facto=facto*i
    return facto
```

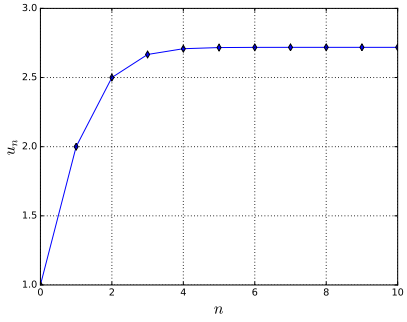
2. Fonction `u` renvoyant la valeur du terme de rang n de u_n.

```
def u(n):
    s=1
    i=1
    while i<=n:
        s=s+1/fact(i)
        i=i+1
    return s
```

.

3. La suite u_n semble converger. On note e sa limite. On peut estimer la valeur de e par le terme de plus haut rang soit u_{10} : $e \approx u_{10} = 2.7182818011463845$. Faut-il garder toutes les décimales ? Le tableau montre qu'à partir du rang 3, à chaque itération, la première décimale non modifiée recule d'un cran vers la droite. A la $10^{\text{ème}}$ itération, les six premières décimales sont conservées : on peut donc raisonnablement supposer qu'elles sont correctes. On retient donc :

$$\boxed{e \approx 2.718281}$$



(a) Graphe

n	$u(n)$
0	1
1	2.0
2	2.5
3	2.6666666666666665
4	2.708333333333333
5	2.7166666666666663
6	2.7180555555555554
7	2.7182539682539684
8	2.71827876984127
9	2.7182815255731922
10	2.7182818011463845

(b) Valeurs

Figure 1 – Suite u_n

Exercice 2 – Polynômes d'interpolation de Lagrange

1. Représentation mémoire du polynôme $5x^4 - 2x^2 + 1$: $[1, 0, -2, 0, 5]$.
2. Polynôme de Lagrange P_0 relatif aux points $(-9, 5)$; $(-4, 2)$; $(-1, -2)$; $(7, 9)$.

$$P_0 = \frac{x+4}{-9+4} \times \frac{x+1}{-9+1} \times \frac{x-7}{-9-7}$$

$$\boxed{P_0 = -\frac{1}{640}(x^3 - 2x^2 - 30x - 28)}$$

3. Fonction `mult_pol` :

```
1 def mult_pol(P,Q):
2     p=len(P)
3     q=len(Q)
4     r=(p+q)-2#degre max de R
5     R=init_pol(r)
6     for i in range(p):
7         for j in range(q):
8             R[i+j]=R[i+j]+P[i]*Q[j]
9     return R
```

FIGURE 2 – Fonction `mult_pol`

La fonction `mult_pol` renvoie le polynôme `R=P x Q`. Le fondement de l'implémentation est le suivant :

$$P(x)Q(x) = R(x)$$

$$\Leftrightarrow \sum_{i=0}^{p} a_i x^i \sum_{j=0}^{q} b_j x^j = \sum_{k=0}^{p+q} c_k x^k$$

$$\Leftrightarrow \sum_{i=0}^{p} \sum_{j=0}^{q} a_i b_j x^{i+j} = \sum_{k=0}^{p+q} c_k x^k$$

$$\Leftrightarrow \sum_{k=0}^{p+q} \sum_{\substack{i,j \\ i+j=k}}^{p,q} a_i b_j x^k = \sum_{k=0}^{p+q} c_k x^k$$

L'ensemble des monômes x^k formant une base des polynômes, on en déduit que $c_k = \sum_{\substack{i,j \\ i+j=k}}^{p,q} a_i b_j$. La fonction `mult_pol` calcule les coefficients c_k suivant cette dernière formule.

4. Fonction `base_pol` :

```
1 def base_pol(x,y,j):
2     n=len(x)
3     P=[1]
4     for i in range(n):
5         if i!=j:
6             Q=[-x[i]/(x[j]-x[i]),1/(x[j]-x[i])]
7             P=mult_pol(P,Q)
8     return P
```

FIGURE 3 – Fonction `base_pol`

5. Fonction `add_pol` :

```
def add_pol(P,Q):
    p=len(P)
    q=len(Q)
    n=min(p,q)
    if p>q :
        R=P
    else:
        R=Q
    for i in range(n):
        R[i]=P[i]+Q[i]
    return R
```

FIGURE 4 – Fonction `add_pol`

6. Fonction `eval_pol` :

```
def eval_pol(P,x):
    val=0
    for i in range(0,len(P)):
        val=val+P[i]*x**i
    return val
```

FIGURE 5 – Fonction `eval_pol`

7. Instructions permettant de tracer le polynôme L interpolant les points $(-9,5)$; $(-4,2)$; $(-1,-2)$; $(7,9)$: fig.6. Il faut calculer le polynôme interpolateur L puis évaluer ses images $L(x)$ sur l'intervalle $[-8,10]$. Enfin, on trace $L(x)$ en fonction de x grâce à la fonction `plot`.

```
y=[5,2,-2,9]
s=['r-','b--','m-.','k:']
plot(x,y,'go',label="Points d'interpolation")
l=linspace(-10,8,100)
L=inter_pol(x,y)
y_tot=eval_pol(L,l)
plot(l,y_tot,'g',lw=1,label=r"$L$")
```

FIGURE 6 – Tracé du polynôme L

Exercice 3 – Simulation d'une file de voitures

1. **Préliminaires**

1.1 On implémente la file de voitures par une liste `A` de booléens, l'élément i de la liste représentant la case i de la file. Si la case i est occupée par une voiture alors `L[i]` vaut `True`, si la case i est libre alors `L[i]` vaut `False`. Exemple fig.7.

`A=[True, False, True, True, False, False, False, False, False, False, True]`

FIGURE 7 – Représentation d'une file de longueur onze comprenant quatre voitures, situées respectivement sur les cases d'indices 0, 2, 3 et 10 ; et son implémentation par une liste de booléens.

1.2 La fonction `occupe(L,i)` n'a qu'à renvoyer la valeur `L[i]` ! L'intérêt d'une telle fonction réside seulement dans la lisibilité du code à suivre. Voir fig.8.

```
1 def occupe(L,i):
2     return L[i]
```

FIGURE 8 – Fonction `occupe`

1.3 La file comprenant n cases et chaque case ne pouvant prendre que deux états possibles (`True` ou `False`), il existe 2^n files de longueur n.

2. **Déplacement des voitures dans une file**

2.1 L'instruction `avancer(avancer(A,False),True)` appelle deux fois la fonction `avancer` : une première fois sans introduire de nouvelle voiture à gauche (`avancer(A,False)`), puis une deuxième fois sur la liste renvoyée par `avancer(A,False)` en introduisant une nouvelle voiture à gauche (`avancer(avancer(A,False),True)`). Résultat de l'appel dans le tableau fig.**??**.

Instruction	Liste
`A=[True, False.....True]`	`[True, False, True, True, False, False, False, False, False, False, True]`
`avancer(A,False)`	`[False, True, False, True, True, False, False, False, False, False, False]`
`avancer(avancer(A,False),True)`	`[True, False, True, False, True, True, False, False, False, False, False]`

FIGURE 9 – Liste renvoyée par l'instruction `avancer(avancer(A,False),True)`

2.2 Implémentation en `Python` la fonction `avancer` : fig.10.

```
1  def avancer(L,voit_depart):
2      n=len(L)
3      Q=L[:]#copie locale pour ne pas modifier la liste pointee par L
4      #Dans tous les cas, la case la plus a droite prend la valeur False initialement
5      #puisqu'elle est soit vide, soit la voiture qui l'occupe sort de la file
6      Q[n-1]=False
7      #On deplace les voitures de la droite vers la gauche donc il faut parcourir
8      #la liste a l'envers
9      i=n-2
10     while i>=0:
11         if occupe(Q,i) and not occupe(Q,i+1):
12             Q[i+1]=True
13             Q[i]=False
14         i=i-1
15     #Voiture la plus a gauche
16     Q[0]=voit_depart
17     return Q
```

FIGURE 10 – Fonction `avancer`