Devoir d'informatique 2

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

8.12.2017

Durée de l'épreuve : 2h00

L'usage de la calculatrice est autorisé.

L'énoncé de ce devoir comporte 4 pages.

— *Si, au cours de l'épreuve, vous repérez ce qui vous semble être une erreur d'énoncé, signalez le sur votre copie et poursuivez votre composition en expliquant les raisons des initiatives que vous êtes amené à prendre.*
— *Il ne faudra pas hésiter à formuler des commentaires. Le barème tiendra compte de ces initiatives ainsi que des qualités de rédaction de la copie.*
— *La numérotation des exercices doit être respectée. Les résultats doivent être systématiquement encadrés. Les pages doivent être numérotées de la façon suivante : n°page courante/nombre total de page.*

Données pour tous les exercices.

— La méthode `append()` ajoute l'argument en queue de liste. Appel : `l.append(elt)`.
— La fonction `len()` renvoie un entier égal à la longueur de la liste ou de la chaîne de caractères passée en argument. Appel : `len(L)`.
— La fonction `float()` convertit l'argument en flottant. Appel `float(n)`.
— La fonction `int()` convertit l'argument en un entier. Si l'argument est un flottant, la fonction `int()` renvoie la partie entière de ce nombre. Appel `int(x)`.
— La fonction `range(a,b,p)` génère la liste suivante : $[a, a+p, a+2p, \dots, a+jp, \dots, b-p]$. L'appel `range(a,b)` est équivalent à `range(a,b,1)` et l'appel `range(b)` est équivalent à `range(0,b,1)`.
— La fonction `linspace(a,b,N)` génère la liste suivante : $[a, a+p, a+2p, \dots, a+jp, \dots, a+(N-1)p]$ avec $p = \frac{b-a}{N}$.
— Slicing : l'expression `l[i:j]` renvoie la sous-liste comprenant les éléments d'indice i inclus à j exclu de la liste `l`.

Exercice 1 – Evaluation du nombre e

On considère la suite réelle définie par :

$$u_n = 1 + \frac{1}{1!} + \frac{1}{2!} + \cdots + \frac{1}{n!} = \sum_{i=0}^{n} \frac{1}{i!}$$

où $i!$ est la factorielle de i soit

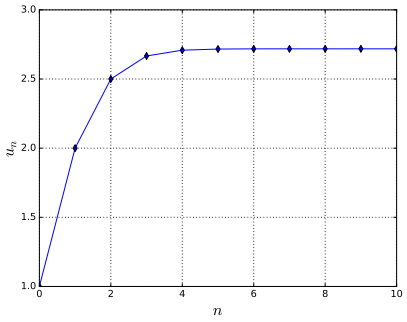
$$i! = \prod_{k=1}^{i} k$$

.

Par convention $0! = 1$.

On montre que la suite $(u_n)_{n\in\mathbb{N}}$ converge vers le nombre irrationnel e. On cherche à en estimer la valeur.

1. Ecrire une fonction `fact` renvoyant la factorielle de l'entier naturel passé en argument.
2. Ecrire une fonction `u` renvoyant la valeur du terme de rang n de $(u_n)_{n\in\mathbb{N}}$.
3. A l'aide de la fonction `u`, on obtient les résultats fig.1. Commenter la courbe puis proposer une valeur approchée de e. On justifiera le nombre de chiffres significatifs retenus.



(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

Une façon de représenter en mémoire le polynôme $P : x \mapsto P(x) = \sum_{i=0}^{n} a_i x^i$ consiste à stocker ses coefficients a_i dans une liste P. $P[i]$ contient alors la valeur de a_i tandis que l'indice i est le degré du monôme de coefficient a_i. Ainsi, par exemple, on représente le polynôme $P(x) = 2x^3 + 4x + 1$ par la liste $[1, 4, 0, 2]$.

Les polynômes interpolateurs de Lagrange P_j permettent d'interpoler une série de $n+1$ points $\{(x_i,y_i)\}$ (i allant de 0 à n) par un polynôme L de degré au plus égal à n passant exactement par chacun des points de la série (fig.2). Le polynôme $L(x)$ s'écrit :

$$L(x) = \sum_{j=0}^{n} y_j P_j(x) \quad \text{avec} \quad P_j(x) = \prod_{i=0, i\neq j}^{n} \frac{x - x_i}{x_j - x_i}$$

Les polynômes P_j constituent une base de l'espace vectoriel des polynômes de degré inférieur ou égal à n.

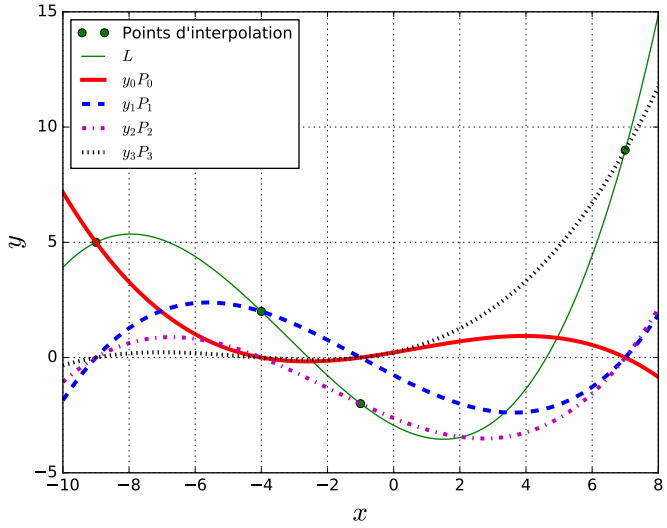


FIGURE 2 – Interpolation polynomiale. Le polynôme L passe par chacun des points $(-9, 5)$; $(-4, 2)$; $(-1, -2)$; $(7, 9)$. Il est une combinaison linéaire des polynômes interpolateurs de Lagrange P_0, P_1, P_2 et P_3.

Pour calculer le polynôme L on dispose des fonctions fig.3 dont certaines restent à écrire. **Pour écrire les fonctions qui sont demandés par la suite, on pourra utiliser toute fonction implémentée fig.3.**

1. Donner la représentation mémoire du polynôme $5x^4 - 2x^2 + 1$.
2. Donner l'expression mathématique du polynôme P_0 relatifs aux points $(-9, 5)$; $(-4, 2)$; $(-1, -2)$; $(7, 9)$.
3. Expliquer brièvement ce que fait la fonction `mult_pol` et justifier son implémentation.

```python
def init_pol(n):
    N=[]
    for i in range(n+1):
        N.append(0)
    return N

def add_pol(P,Q):
    #A IMPLEMENTER

def mult_pol(P,Q):
    p=len(P)
    q=len(Q)
    r=(p+q)-2#degre max de R
    R=init_pol(r)
    for i in range(p):
        for j in range(q):
            R[i+j]=R[i+j]+P[i]*Q[j]
    return R

def base_pol(x,y,j):
    """
    x : liste des abscisses des points a interpoler
    y : liste des ordonnees des points a interpoler
    j : int
    """
    #A IMPLEMENTER

def inter_pol(x,y):
    n=len(x)#degre de L
    L=[0]
    for j in range(n):
        P=base_pol(x,y,j)
        Q=mult_pol([y[j]],P)
        L=add_pol(L,Q)
    return L

def eval_pol(P,x):
    #A IMPLEMENTER
```

FIGURE 3 – Ensembles des fonctions permettant l'interpolation polynomiale d'une série de points

4. Implémenter la fonction `base_pol` qui prend en argument la liste `x` des abscisses $\{x_i\}$ et la liste `y` des ordonnées $\{y_i\}$ des points à interpoler ainsi que l'entier `j` et renvoie le polynôme de Lagrange P_j.
5. Implémenter la fonction `add_pol` qui renvoie le polynôme `R` égal à la somme des deux polynômes `P` et `Q` passés en argument.
6. Implémenter la fonction `eval_pol` qui renvoie la valeur $P(x)$ du polynôme `P` pour la valeur `x` passée en argument.
7. Compléter le code fig.4 par les trois instructions permettant de tracer le polynôme L interpolant les points $(-9, 5)$; $(-4, 2)$; $(-1, -2)$; $(7, 9)$.

```python
#Points a interpoler
x=[-9,-4,-1,7]
y=[5,2,-2,9]
l=linspace(-10,8,100)#liste de cents points equirepartis entre -10 et 8
# instruction 1
# instruction 2
# instruction 3
```

FIGURE 4 – Tracé du polynôme L

Exercice 3 – Simulation d'une file de voitures

On modélise le déplacement d'un ensemble de voitures sur des files à sens unique (voir fig.5). C'est un schéma simple qui peut permettre de comprendre l'apparition d'embouteillages et de concevoir des solutions pour fluidifier le trafic.

FIGURE 5 – Représentation d'une file de longueur onze comprenant quatre voitures, situées respectivement sur les cases d'indices 0, 2, 3 et 10

On considère le cas d'une seule file, illustré par la figure 5. Une file de longueur n est représentée par n cases. Une case peut contenir au plus une voiture. Les voitures présentes dans une file circulent toutes dans la même direction (sens des indices croissants, désigné par les flèches sur la figure 5) et sont indifférenciées.

1. **Préliminaires**

   1.1 Proposer une implémentation de la file de voitures à l'aide d'une liste de booléens. Ecrire alors l'instruction `Python` permettant de définir une liste `A` représentant la file de voiture illustrée par la figure 5.

   1.2 Soit `L` une liste représentant une file de longueur n et i un entier tel que $0 \leq i < n$. Définir en Python la fonction `occupe(L, i)` qui renvoie `True` lorsque la case d'indice i de la file est occupée par une voiture et `False` sinon.

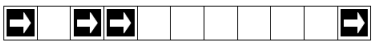
   1.3 Combien existe-t-il de files différentes de longueur n ? Justifier votre réponse

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

   On identifie désormais une file de voitures à une liste. On considère les schémas de la figure 6 représentant des exemples de files. Une étape de simulation pour une file consiste à déplacer les voitures de la file, à tour de rôle, en commençant par la voiture la plus à droite, d'après les règles suivantes :
   — une voiture se trouvant sur la case la plus à droite de la file sort de la file ;
   — une voiture peut avancer d'une case vers la droite si elle arrive sur une case inoccupée ;
   — une case libérée par une voiture devient inoccupée ;
   — la case la plus à gauche peut devenir occupée ou non, selon le cas considéré.

   On suppose avoir écrit en `Python` la fonction `avancer` prenant en paramètres une liste de départ, un booléen indiquant si la case la plus à gauche doit devenir occupée lors de l'étape de simulation, et renvoyant la liste obtenue par une étape de simulation.

   Par exemple, l'application de cette fonction à la liste illustrée par la figure 6(a) permet d'obtenir soit la liste illustrée par la figure 6(b) lorsque l'on considère qu'aucune voiture nouvelle n'est introduite, soit la liste illustrée par la figure 6(c) lorsque l'on considère qu'une voiture nouvelle est introduite.



(a) Liste initiale A

(b) $B =$ `avancer`$(A,$ `False`$)$

(c) $C =$ `avancer`$(A,$ `True`$)$

FIGURE 6 – Etape de simulation

2.1 Étant donnée `A` la liste définie à la question 1.1, que renvoie l'instruction `avancer(avancer(A,False),True)` ?

2.2 Implémenter en `Python` la fonction `avancer`.

Par la suite, on peut simuler un bouchon en bloquant une voiture, simuler un carrefour...