esprit — Se former autrement — HONORIS UNITED UNIVERSITIES	**EXAMEN** Semestre : 1 ☐ 2 ■ Session : Principale ■ Rattrapage ☐
ETUDIANT(e) **Nom et Prénom :** .... **Classe :**	**Code :**

Module : Calcul Scientifique (CS)
Enseignant(s) : Equipe CS de l'UP Math
Classes : 3A1→3A28
Documents autorisés : OUI ■ NON ☐ Nombre de pages : 7 pages
Calculatrice autorisée : OUI ■ NON ☐ Internet autorisée : OUI ☐ NON ■
Date : 01/06/2023 Heure : 11h Durée : 1h30min

****NB :**

1. Toute réponse non justifiée (sans code) ne sera pas considérée.
2. Vous êtes appelés à utiliser :
   (a) **une flèche $\longrightarrow$ pour indiquer une indentation,**
   (b) **l'importation des modules** "`numpy`","`matplotlib.pyplot`" **et** "`sympy`" **en utilisant les alias présentés ci-dessous :**

[ ]:
```
import numpy as np
import matplotlib.pyplot as plt
import sympy as sp
```

Exercice 1 : (10 points) Soient trois points de coordonnées $(x_i, y_i)_{0\leq i\leq 2}$ tels que $x_i \neq x_j$, pour $0 \leq i,j \leq 2$ et $i \neq j$. On rappelle qu'il existe un unique polynôme d'interpolation $P \in \mathbb{R}_2[X]$ vérifiant $P(x_i) = y_i$, $\forall i \in \{0,1,2\}$ et s'exprimant comme suit :

$$\begin{aligned} P(t) &= \sum_{i=0}^{2} \beta_i \omega_i(t), \quad t \in \mathbb{R} \\ &= \beta_0.\underbrace{1}_{\omega_0(t)} + \beta_1\underbrace{(t-x_0)}_{\omega_1(t)} + \beta_2\underbrace{(t-x_0)(t-x_1)}_{\omega_2(t)}, \end{aligned}$$

avec

$$\omega_i(t) = \begin{cases} 1 & \text{si } i = 0, \\ \displaystyle\prod_{j=0}^{i-1}(t-x_j) & \text{si } i \in \{1,2\}. \end{cases} \tag{1}$$

Les réels β_i, $i \in \{0,1,2\}$ correspondent aux coefficients de Newton. Pour les déterminer, nous utilisons la méthode des différences divisées.

1. (a) Compléter la fonction `BaseNewton(t,i,x)` prenant en entrée une liste t, l'indice i des coefficients de Newton et une liste x contenant les points d'interpolation $(x_i)_{0\leq i\leq 2}$.

   Cette fonction évalue en t, le polynôme ω_i, $i \in \{0,1,2\}$ associé aux points d'interpolation $(x_i)_{0\leq i\leq 2}$. **(1,5 points)**

[ ]:
```
def BaseNewton(t,i,x):
    W=np.ones((len(t)))
    if i==0:
        return  W
    else:
        for j in np.arange(0,i):
                                                        #utilisez  (1)
                W*=(t-x[j])
    return W
```

(b) En appliquant la fonction `BaseNewton` et en considérant les points d'interpolation $\{-2, 0, 2\}$, évaluer les polynômes ω_0, ω_1 et ω_2 en 5 valeurs de t reparties uniformément sur l'intervalle $[-2, 2]$. **(1.5 points)**

[ ]:
```
# Code

x=np.arange(-2,3,2)
#x=[-2,0,2]
t=np.linspace(-2,2,5)
print(BaseNewton(t,0,x))
print(BaseNewton(t,1,x))
print(BaseNewton(t,2,x))
```

Les valeurs de ω_0 en t sont [1 1 1 1 1].

Les valeurs de ω_1 en t sont [0 1 2 3 4].

Les valeurs de ω_2 en t sont [0 -1 0 3 8].

On cherche maintenant à déterminer les coefficients de Newton.

On considère trois points $(x_i, y_i)_{0\leq i\leq 2}$ d'abscisses deux à deux distincts et On rappelle que

• La différence divisée d'ordre 0 de x_0 est:

$$\beta_0 = [y_0] = y_0$$

• La différence divisée d'ordre 1 de x_0 et x_1 est :

$$\beta_1 = [y_0, y_1] = \frac{y_1 - y_0}{x_1 - x_0}$$

• La différence divisée d'ordre 2 des x_0, x_1 et x_2 est:

$$\beta_2 = [y_0, y_1, y_2] = \frac{[y_1, y_2] - [y_0, y_1]}{x_2 - x_0}$$

.

2. (a) Ecrire une fonction `diffdiv(x,y)` prenant en entrée deux listes x et y composées respectivement par $(x_i)_{0\leq i\leq 2}$ et $(y_i)_{0\leq i\leq 2}$, et qui retourne une liste contenant les coefficients de Newton β_i, $i \in \{0, 1, 2\}$. **(1.5 points)**

[ ]:
```
def diffdiv(x,y):

    beta0=y[0]
    beta1=(y[1]-y[0])/(x[1]-x[0])
    beta2=(((y[2]-y[1])/(x[2]-x[1]))-beta1)/(x[2]-x[0])

    return [beta0,beta1,beta2]
```

(b) En prenant les points d'interpolation $(-2,4)$, $(0,3)$ et $(2,6)$, afficher les coefficients de Newton β_0, β_1 et β_2. **(1 point)**

[ ]:
```
# Code
x=[-2,0,2]
y=[4,3,6]

diffdiv(x,y)
```

$$\beta_0 = 4$$
$$\beta_1 = -0.5$$
$$\beta_2 = 0.5$$

3. (a) Compléter la fonction `InterpolationNewton(t,x,y)` qui évalue en t, le polynôme d'interpolation de Newton associé aux points d'interpolation $(x_i, y_i)_{0 \leq i \leq 2}$, avec x est une liste composée par les $(x_i)_{0 \leq i \leq 2}$ et y est une liste composée par les $(y_i)_{0 \leq i \leq 2}$. **(2 points)**

[ ]:
```
def InterpolationNewton(t,x,y):
    P=np.zeros((len(t)))            # Initialisation

    beta=diffdiv(x,y)    # beta : les coefficients de Newton

    for i in np.arange(0,3):
        P+=beta[i]*BaseNewton(t,i,x)

    return P
```

(b) Evaluer le polynôme qui interpole les points $(-2,4)$, $(0,3)$ et $(2,6)$, noté P, en 5 valeurs de t reparties uniformément sur l'intervalle $[-2,2]$. **(1 point)**

[ ]:
```
# Code
# x, y et t sont déjà déclarées.

P=InterpolationNewton(t,x,y)
P
```

Les valeurs de P en t sont : array([4., 3., 3., 4., 6.])

(c) Compléter le code suivant qui permet de tracer les courbes représentatives de P et du polynôme Q défini sur $[-2,2]$ par $Q(t) = \frac{1}{2}t^2 + \frac{1}{2}t + 3$. Puis, interpréter les résultats obtenus. **(1.5 points)**

[ ]:
```
# Code

t=np.linspace(-2,2,100)
P=InterpolationNewton(t,x,y)
Q=lambda t:0.5*(t**2)+0.5*t+3
plt.plot(t,P,'mo',t,Q(t),'b')
plt.grid(True)
plt.legend(('Polynome dínterpolation P','polynome Q'))
```

Interprétation graphique :

$P = Q$

Exercice 2 : (10 points)

L'objectif de cet exercice est d'approcher la valeur de $I = \int_a^b f(t)\,dt$, avec f une fonction continue sur un intervalle $[a,b]$ en utilisant deux méthodes d'intégration numérique.

En subdivisant l'intervalle $[a,b]$ en N sous-intervalles de même longueur, les points d'intégration seront notés x_k, où $0 \leq k \leq N$.

NB: Le nombre de sous-intervalles N égal au nombre de points d'intégration -1.

La formule de la première méthode d'intégration est donnée par

$$I_1 = \sum_{k=0}^{N-1} f(x_k)\left(x_{k+1} - x_k\right). \quad \textbf{Formule 1}$$

ETUDIANT(e)	
Nom et Prénom : ………………………………………………..	Code :
Classe: ……………………………………………………..................	

1. Ecrire une fonction appelée `methode1(f,x)` prenant en entrée une fonction continue f sur l'intervalle $[a,b]$ et $x = [x_0, x_1, ..., x_N]$ une liste (ou un tableau 1D) contenant les points d'intégration et retournant I_1 la valeur approchée de I donnée par **Formule 1**. **(1.5 points)**

[ ]:
```
def methode1(f,x):
    I=0
    for k in np.arange(0,len(x)-1):
        I+=(x[k+1]-x[k])*f(x[k])
    return I
```

La formule de la deuxième méthode d'intégration est donnée par

$$I_2 = \sum_{k=0}^{N-1} \frac{f(x_k) + f(x_{k+1})}{2} (x_{k+1} - x_k). \quad \textbf{Formule 2}$$

2. Ecrire une fonction appelée `methode2(f,x)` prenant en entrée une fonction continue f sur l'intervalle $[a,b]$ et $x = [x_0, x_1, ..., x_N]$ une liste (ou un tableau 1D) contenant les points d'intégration et retournant I_2 la valeur approchée de I donnée par **Formule 2**. **(1,5 points)**

[ ]:
```
def methode2(f,x):
    I=0
    for k in np.arange(0,len(x)-1):
        I+=(x[k+1]-x[k])*(f(x[k])+f(x[k+1]))/2
    return I
```

3. Soit $I = \int_0^5 \frac{1}{1+(t-\pi)^2}\,dt$.

   (a) Pour $N = 10$, donner les deux valeurs approchées de I obtenues en appliquant les fonctions **methode1** et **methode2**. **(1.5 points)**

[ ]:
```
# Code

t=np.linspace(0,5,11)
f=lambda t: 1/(1+(t-np.pi)**2)
I1=methode1(f,t)
I2=methode2(f,t)
print(I1,I2)
```

   - La valeur approchée de I en appliquant **methode1** est 2.301644622937016
   - La valeur approchée de I en appliquant **methode2** est 2.3347780876447075

   (b) En utilisant une fonction pré-définie dans la bibliothèque **sympy**, donner la valeur de I.**(1.5 points)**

[ ]:
```
# Code

x=sp.symbols('x')
I=sp.integrate(1/(1+(x-np.pi)**2),(x,0,5)).evalf()
#I=sp.integrate(1/(1+(x-sp.pi)**2),(x,0,5)).evalf()
```

   La valeur de I vaut 2.33976628366847

   (c) Donner les erreurs d'intégration de I par les deux méthodes. **(1.5 points)**

[ ]:
```
# Code

E1=np.abs(I-I1)
E2=np.abs(I-I2)
print(E1,E2)
```

   - L'erreur d'intégration de I en appliquant **methode1** est 0.0381216607314538
   - L'erreur d'intégration de I en appliquant **methode2** est 0.00498819602376255

   (d) Ecrire un code qui permet d'afficher deux listes contenant les erreurs d'intégration obtenues par les deux méthodes **methode 1** et **methode 2** pour les différentes valeurs de N suivantes : 30,40,50 et 60. **(1.5 points)**

[ ]:
```python
# Code
# Indications : vous pouvez utiliser cette démarche
#                   1- Créer pour chaque N une liste
#                    contenant les points d'intégration
#                    2-Créer deux listes vides pour les remplir avec
#                    les erreurs d'intégration demandées

L1=np.linspace(0,5,31)
L2=np.linspace(0,5,41)
L3=np.linspace(0,5,51)
L4=np.linspace(0,5,61)
L=[L1,L2,L3,L4]
ErrM1=[ ]
ErrM2=[ ]
#f=lambda t: 1/(1+(t-np.pi)**2)
for k in L:
    ErrM1.append(np.abs(I-methode1(f,k)))
    ErrM2.append(np.abs(I-methode2(f,k)))
print(ErrM1)
print(ErrM2)
```

(e) Comparer les deux méthodes en terme de précision pour les quatre valeurs considérées de N. **(1 point)**

Pour les quatre valeurs de N données, la deuxième méthode approche le mieux la valeur de I.