Institut **P**réparatoire aux **E**tudes d'**I**ngénieurs de **N**abeul
Informatique : TP Python

TP 1. Simulation Numérique : Corrigé
Modules de calcul scientifique : `Numpy` et `Matplotlib`

Exercice 1 :

Construire un objet de type `numpy.ndarray`

a) dont les termes sont les multiples de 3 compris entre 0 et 27 ;

```
import numpy as np
a = np.arange(0,28,3);
print(a)
```

b) dont les termes sont les puissances de 2, de $2^0 = 1$ à $2^{12} = 4096$;

```
import numpy as np
b = np.arange(0,13);
print(2**b)
```

Exercice 2 :

Pour cet exercice, on prend n = 1000000. On pourra augmenter ou diminuer cette valeur en fonction de la machine utilisée.

1. Calculer $\sum_{i=0}^{n} i$ sans utiliser `numpy`.
2. Chronométrer le temps nécessaire pour le calcul précédent, par exemple en utilisant `time.clock()`.

```
import numpy as np
from time import clock
# 1ère méthode : avec une boucle for ordinaire
t0 = clock()
a = range(1000001)
t1 = clock()
S = 0
for i in a:
    S += i
t2 = clock()
print('1ère méthode : {:6.6f} + {:6.6f}  = {:6.6f} secondes'.
                format(t1-t0,t2-t1,(t1-t0)+(t2-t1)))

# 2ème méthode : avec une liste en compréhension
t0 = clock()
a = range(1000001)
t1 = clock()
S = sum(i for i in a)
t2 = clock()
print('2ème méthode : {:6.6f} + {:6.6f}  = {:6.6f} secondes'.
                format(t1-t0,t2-t1,(t1-t0)+(t2-t1)))
```

3. Utiliser un tableau numpy et la méthode sum pour calculer à nouveau la somme proposée.
4. Comparer le temps de calcul avec la méthode précédente.

```
import numpy as np
from time import clock
t0 = clock()
v = np.array(range(1000001))
t1 = clock()
S = v.sum() # ou np.sum(v)
t2 = clock()
print('3ème méthode : {:6.6f} + {:6.6f}  = {:6.6f} secondes'.
                format(t1-t0,t2-t1,(t1-t0)+(t2-t1)))
```

Exercice 3 :

On peut justifier que :

$$\pi = 2\prod_{n=1}^{+\infty}\frac{4n^2}{4n^2-1}$$

appelé le produit de Wallis.

1. Écrire une fonction itérative, d'argument n, calculant :

$$2\prod_{i=1}^{n}\frac{4i^2}{4i^2-1}$$

```
import numpy as np
from time import clock
def wallis (n):
    p = 2
    for i in range(1, n+1):
        k = 4 * i * i
        p *= k / (k - 1)
    return p
t0 = clock()
print(wallis(1000000))
t1 = clock()
print('{:6.6f} secondes'.format(t1-t0))
```

2. Écrire une fonction utilisant un tableau numpy, effectuant le même calcul.
3. Comparer le temps de calcul de ces deux fonctions.

```
import numpy as np
from time import clock
def wallis_np (n):
    x = np.arange(1,n+1, dtype=np.float)
    x **= 2
    x *= 4
    y = x - 1
    x /= y
    return 2*np.prod(x)
t0 = clock()
print(wallis_np(1000000))
t1 = clock()
print('{:6.6f} secondes'.format(t1-t0))
```

Exercice 4 :

Soit la fonction $h(x) = \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}x^2}$. Créer des tableaux xt et ht qui contiennent respectivement 41 valeurs reparties uniformement pour $x \in [-4,4]$ et l'évaluation de la fonction h sur ces valeurs.

```
from math import sqrt,pi,exp
import numpy as np

def h(x):
    return (1/sqrt(2*pi))*exp(-1/2*x**2)

xt = np.linspace(-4,4,41)
print(xt)
h = np.vectorize(h)
ht=h(xt)
print(ht)
```

Exercice 5 :

Ecrire un script qui demande à l'utilisateur des nombres $a_1, a_2, b_1, b_2, c_1, c_2$ et trace le triangle ABC.

```
"""
On rappelle que : les points du triangle ABC sont  :
A(a1;a2), B(b1;B2) et C(c1;c2)
et que : plt.plot([a1, b1, c1, a1],[a2, b2, c2, a2])
relie les points  (a1;a2), (b1;B2), (c1;c2) et (a1;a2)
on fait donc:
"""

import matplotlib.pyplot as plt

a1= float(input('a1 = '))
a2= float(input('a2 = '))

b1= float(input('b1 = '))
b2= float(input('b2 = '))

c1= float(input('c1 = '))
c2= float(input('c2 = '))

plt.plot([a1, b1, c1, a1],[a2, b2, c2, a2])
plt.show() # Afficher la fenetre graphique
```

Exercice 6 :

Réaliser le graphe de la fonction $y(t) = v_0 t - \frac{1}{2} g t^2$ pour $v_0 = 10$, $g = 9.81$, et $t \in [0, 2v_0/g]$. Le label sur l'axe des x devra être "temps (s)" et le label sur l'axe des y "hauteur (m)".

```
from math import *
import numpy as np
import matplotlib.pyplot as plt
v0 = 10
```

```python
g = 9.81
a=0
b= 2 *v0 / g
t = np.linspace(a,b,100)

def y(t):
        return v0 * t - 1/2 * g * t**2

plt.plot(t, y(t))
plt.xlabel("Temps (s)")
plt.ylabel("Hauteur (m)")
plt.show()
```

Exercice 7 :

La fonction $f(x,t) = e^{-(x-3t)^2} sin(3\pi(x-t))$ décrit pour une valeur fixe de t une onde localisée en espace. Faire un programme qui visualise cette fonction comme une fonction de x dans l'intervalle $x \in [-4,4]$ pour $t = 0$.

```python
from math import exp, pi
import numpy as np
import matplotlib.pyplot as plt

def f(x,t):
    return exp(-(x-3*t)**2) * sin(3*pi*(x-t))

f = np.vectorize(f)

x = np.linspace(-4,4)
print("x=",x)

y=f(x,0)
print("y =",y)

plt.plot(x,y)

plt.xlabel("x")
plt.ylabel("f(x,0)")

plt.show()
```