**TP 4 - éléments de correction**

```python
import numpy as np
import matplotlib.pyplot as plt
import scipy.interpolate as inter

## Exemple 1
x = np.array([0,1,2])
w = np.array([4,-2,4.])
p = inter.lagrange(x,w)
print(p)
print(p(0), p(1), p(2))
xpt=np.linspace(-1,3,40)
plt.plot(x,w,'+',label = "points d'interpolation")
plt.plot(xpt,p(xpt),'k',label ="polynôme d'interpolation (lagrange)")
plt.plot(xpt,inter.barycentric_interpolate(x,w,xpt),'r-.', label = "polynôme d'interpolation (barycentric_interpolate)")
plt.legend()
plt.show()

## Exemple 2
x=np.array([-4,0,1,4])
w=np.cos(x)
p=inter.lagrange(x,w)
print(p)
print(p(-4), p(0), p(1), p(4))
xpt=np.linspace(-4,4,40)
plt.clf()
plt.plot(x,w,'+',label="points d'interpolation")
plt.plot(xpt,np.cos(xpt),'k',label="fonction à interpoler (cosinus)")
plt.plot(xpt,inter.barycentric_interpolate(x,w,xpt),'r-.',label="polynôme d'interpolation (barycentric_interpolate)")
plt.legend()
plt.show()

## 1.1. Premier test
x = np.array([0,1,2,3,4])
w = np.sin(x)
p = inter.lagrange(x,w)
print(f"valeurs aux points d'interpolation avec routine lagrange :\n{p(x)}")
plt.clf()
plt.plot(x,w,'bo',label="points d'interpolation")
xpt = np.linspace(-2,4,40)
plt.plot(xpt,np.sin(xpt),'go',label = "fonction à interpoler (sinus)")
plt.plot(xpt,p(xpt),'k',label="polynôme d'interpolation (lagrange)")
print(f"valeurs aux points d'interpolation avec barycentric_interpolate :\n {inter.barycentric_interpolate(x,w,x)}")
plt.plot(xpt,inter.barycentric_interpolate(x,w,xpt),'r-.',label="polynôme d'interpolation (barycentric_interpolate)")
plt.legend()
plt.show()

## 1.2. Exercice
def f(x):
    return x**(1/3)

# a)
x = np.array([0,1,8,27,64])
w = f(x)
p = inter.lagrange(x,w)
print(f"a) Le polynôme d'interpolation de Lagrange est, selon la routine lagrange : \nP(x) = {p}")

# b)
print(f"b) P(20) = {p(20)}\n20**(1/3) = {20**(1/3)}")

# c)
xpt = np.linspace(0,64,200)
plt.clf()
plt.plot(x,w,'o',label="points d'interpolation")
plt.plot(xpt,f(xpt),'g',label="fonction à interpoler (racine cubique)")
plt.plot(xpt,p(xpt),'r-.',label = "polynôme d'interpolation (lagrange)")

# d)
x_sans0 = np.array([1,8,27,64])
w_sans0 = f(x_sans0)
p_sans0 = inter.lagrange(x_sans0,w_sans0)
```

```
print(w_sans0)
print(p_sans0(x_sans0))
plt.plot(xpt,p_sans0(xpt),'k',label = "polynôme d'interpolation sans (0,0) (lagrange)")
plt.legend()
plt.show()

## 2.1. Points équidistants
def einterp(n,f,a,b):
    plt.clf()
    xpt = np.linspace(a,b,1000)
    plt.plot(xpt,f(xpt),'g',label="f")
    x = np.linspace(a,b,n+1)
    plt.plot(xpt,inter.barycentric_interpolate(x,f(x),xpt),'k-.',label = "f interpolée")
    plt.plot(x,f(x),'x',label="points d'interpolation")
    plt.legend()
    plt.show()

a , b = -1 , 1
def f1(x):
    return 1/(1+x**2)
n = 57
einterp(n,f1,a,b)

def f2(x):
    return 1/(4/25 +x**2)
n = 20
einterp(n,f2,a,b)

## 2.2. Points de Tchebychev
def tinterp(n,f,a,b):
    plt.clf()
    xpt = np.linspace(a,b,1000)
    plt.plot(xpt,f(xpt),'g',label="f")
    x = np.zeros(n+1)
    for k in range(0,n+1):
        x[k] = (a+b)/2 + (b-a)/2 * np.cos( np.pi * (2 * k + 1)/(2*(n+1)))
    plt.plot(xpt,inter.barycentric_interpolate(x,f(x),xpt),'k-.',label = "f interpolée")
    plt.plot(x,f(x),'x',label="points d'interpolation")
    plt.legend()
    plt.show()

a , b = -1 , 1
def f1(x):
    return 1/(1+x**2)
n = 60
tinterp(n,f1,a,b)

def f2(x):
    return 1/(4/25 +x**2)
n = 20
tinterp(n,f2,a,b)
```