```
# E:\TP info 24-25\TP3.py
# Importations

import numpy as np
import matplotlib.pyplot as plt

# Exercice 1 Balayage
def balayage(f,a,e):
    k = 0
    while f(a)*f(a+k*e) > 0:
        k += 1
    return (a+(k-1)*e + a+k*e)/2, k

## 1
def f1(x):
    return x**2-2

a,e = 1,10**(-6)
print(balayage(f1,a,e))

## 2
def f2(x):
    return x**3-2*x-5

Lx = np.linspace(0,3)
Ly = f2(Lx)
plt.grid()
plt.plot(Lx,Ly)
plt.show()

a,e = 2,10**(-6)
print(balayage(f2,a,e))

## 3
def f3(x):
    return 1-2**(x**2-6*x+9)
Lx = np.linspace(-5,5)
Ly = f3(Lx)
plt.grid()
plt.plot(Lx,Ly)
plt.show()

a,e = 2,10**(-6)
print(balayage(f3,a,e))

## Exercice 2 : Dichotomie
def dicho(f,a,b,e):
    k = 0
    while abs(a-b) > e:
        c = (a+b)/2
```

```python
        if f(a)*f(c)<0:
            b = c
        else:
            a = c
        k += 1
    return c, k

def f1(x):
    return x**2-2

a,b,e = 1,2,10**(-6)
print(dicho(f1,a,b,e))

## 1
def f1(x):
    return x**5-x**4-5*x**3-x**2+4*x+3

Lx = np.linspace(-2,2)
Ly = f1(Lx)
plt.grid()
plt.plot(Lx,Ly)
plt.show()

e = 10**(-9)
for (a,b) in [(-2,-1),(1,2)]:
    print(dicho(f1,a,b,e))

## 2
def f2(x):
    return (x**2)*(4-x**2)-4/(x**2+1)

# f2 est paire
Lx = np.linspace(0,2)
Ly = f2(Lx)
plt.grid()
plt.plot(Lx,Ly)
plt.show()

e = 10**(-9)
for (a,b) in [(0,1),(1,2)]:
    print(dicho(f2,a,b,e))

## 3
def f3(x):
    return np.exp(-x)-x-2

Lx = np.linspace(-1,1)
Ly = f3(Lx)
plt.grid()
plt.plot(Lx,Ly)
```

```python
plt.show()

a,b,e = -1,0,10**(-9)
print(dicho(f3,a,b,e))

## 4
def f4(x):
    return np.log(4-x**2)-x

Lx = np.linspace(0,1.5)
Ly = f4(Lx)
plt.grid()
plt.plot(Lx,Ly)
plt.show()

a,b,e = 1,2,10**(-9)
print(dicho(f3,a,b,e))

## 5
def f5(x):
    return np.sqrt(x**2-x+1)-2*np.sin(np.pi*x)

Lx = np.linspace(-10,10,1000)
Ly = f5(Lx)
plt.grid()
plt.plot(Lx,Ly)
plt.show()

e = 10**(-9)
for (a,b) in [(0,0.5),(0.5,1)]:
    print(dicho(f5,a,b,e))

# Exercice 3
## 1
def Lu(n):
    L = []
    for k in range(3,n+1):
        def f(x):
            return x**k + x**2 + 2*x - 1
        L.append(dicho(f,0,1,10**(-5))[0])
    return L

import matplotlib.pyplot as plt
n = 15
Ln = range(3,n+1)
Lu = Lu(n)
plt.plot(Ln,Lu,marker = '.')
plt.show()

## 2
```

```
def f(x):
    return x**n*(x-1)-np.exp(-x)

import matplotlib.pyplot as plt
import numpy as np
Lx = np.linspace(1.1,1.2,1000)
for n in range(3,50):
    Ly = f(Lx)
    plt.plot(Lx,Ly)
    plt.grid()
    plt.show()

def Lu(n):
    L = []
    for k in range(3,n+1):
        def f(x):
            return x**k*(x-1)-np.exp(-x)
        L.append(dicho(f,1,2,10**(-5))[0])
    return L

n = 100
Ln = range(3,n+1)
Lu = Lu(n)
plt.plot(Ln,Lu,marker = '.')
plt.show()

## Méthode de Newton
def Newton(f,fprime,a,e):
    x0 = a
    x1 = x0-f(x0)/fprime(x0)
    n = 1
    while abs(x1-x0) > e :
        x0, x1 = x1, x1-f(x1)/fprime(x1)
        n += 1
    return x1,n

# Exercice 5
## 1
def f1(x):
    return x**3-30

def f1prime(x):
    return 3*x**2

a, e = 1, 10**(-6)
print(Newton(f1,f1prime,a,e)[0])

## 2
def f2(x):
    return x**7-1000
```

```

def f2prime(x):
    return 7*x**6

a, e = 1, 10**(-6)
print(Newton(f2,f2prime,a,e)[0])

## Exerce 6
def f3(x):
    return 2*np.sin(x) - x

Lx = np.linspace(0,2)
Ly = f3(Lx)
plt.plot(Lx,Ly)
plt.show()
##
def f3prime(x):
    return 2*np.cos(x) - 1

a, e = 1, 10**(-6)
print(Newton(f3,f3prime,a,e)[0])
```