TP 1 - éléments de correction

```python
import math

# Exercice 1 ------------------------------------------------
print("EXERCICE 1\n")
def n_premiers_termes_u(n):
    u_k = 1 # initialisation
    for k in range(n):
        print(f"u_{k} = {u_k}")
        u_k = math.sqrt(1+u_k) # maj de uk

n_premiers_termes_u(5)

print(f"Le nombre d'or est : {(1 + math.sqrt(5))/2}")

# Exercice 2 ------------------------------------------------
print("\nEXERCICE 2")
print("\na)")
def n_premiers_termes__F(n):
    F_km2, F_km1 = 1, 1
    print(f"F_0 = {F_km2}")
    if n >= 1 :
        print(f"F_1 = {F_km1}")
    for k in range(2,n):
        F_k = F_km1 + F_km2
        print(f"F_{k} = {F_k}")
        F_km2 = F_km1 # maj de F_km2 et F_km1
        F_km1 = F_k

n_premiers_termes__F(5)

print("\nb)")
def n_premiers_termes__adaptee_F(n):
    F_km2, F_km1 = 1, 1
    print(f"F_1 / F_0  = {F_km1 / F_km2}")
    for k in range(2,n):
        F_k = F_km1 + F_km2
        print(f"F_{k}/F_{k-1} = {F_k/F_km1}")
        F_km2 = F_km1 # maj de F_km2 et F_km1
        F_km1 = F_k

n_premiers_termes__adaptee_F(5)

print(f"Le nombre d'or est : {(1 + math.sqrt(5))/2}")

# Exercice 3 ------------------------------------------------
print("\nEXERCICE 3\n")
print("a)")

def syracuse_avant_1(S_0):
    S_k = S_0
    print(f"S_0 = {S_k}")
    k = 1
    while S_k != 1 :
        if S_k % 2 == 0 :
            S_k = S_k / 2
        else :
            S_k = 3 * S_k + 1
        print(f"S_{k} = {S_k}")
        k = k + 1

syracuse_avant_1(3)

print("\nb)")

def nb_etapes_avant_syracuse_1(N):
    for S_0 in range(1,N+1):
        S_k = S_0
        print(f"S_0 = {S_k}")
        k = 1
        while S_k != 1 :
```

```
            if S_k % 2 == 0 :
                S_k = S_k / 2
            else :
                S_k = 3 * S_k + 1
            k = k + 1
        print(f"Nombre d'étapes = {k-1}\n")

nb_etapes_avant_syracuse_1(5)

# Exercice 4 ------------------------------------------------
print("\nEXERCICE 4\nVoir l'autre document pour la correction ")

## METHODE DICHOTOMIE
print("\nMETHODE DICHOTOMIE\n")

def dicho(f,a,b,tol,prec,n_max):
    a_n = a
    b_n = b
    n= 0
    while n < n_max and abs(b_n - a_n) >= prec and abs(f((a_n+b_n)/2)) >= tol :
        c_n = (a_n + b_n)/ 2
        if f(c_n) * f(a_n) > 0 :
            a_n = c_n
        if f(c_n) * f(a_n) < 0 :
            b_n = c_n
        if f(c_n) * f(a_n) == 0 :
            print(f"Une solution est {c_n}")
        n = n + 1
    if n == n_max :
        print("ATTENTION ! L'algorithme s'est arrêté car le n_max a été dépassé§.\n")
    return [n,a_n,b_n]

def f1(x):
    return x - math.exp(-x)

def f2(x):
    return x**2-2

tol =  10**(-16)
prec =  10**(-10)
nmax = 100

# pour f1
f = f1
a = 0
b = 1
[n,a_n,b_n] = dicho(f,a,b,tol,prec,nmax)
print(f"pour f1:\nnb_itérations = {n}\n[a_n,b_n]=[{a_n},{b_n}]\n")

# pour f2
f = f2
a = 1
b = 2
[n,a_n,b_n] = dicho(f,a,b,tol,prec,nmax)
print(f"pour f2:\nnb_itérations = {n}\n[a_n,b_n]=[{a_n},{b_n}]")
print(f"Vérification:\nracine(2)={math.sqrt(2)}")

## METHODE DE NEWTON -------------------
print("\nMETHODE NEWTON\n")

def newton(f,df,x0,tol,prec,nmax):
    x_n = x0
    x_np1 = x_n - f(x_n)/df(x_n)
    n = 0
    while n < n_max and abs(x_np1 - x_n) > prec and abs(f(x_n))>tol:
        n = n + 1
        x_n = x_np1
        x_np1 =  x_np1 - f(x_np1)/df(x_np1)
    if n == n_max :
        print("ATTENTION ! L'algorithme s'est arrêté car le n_max a été dépassé§.\n")
    return [n,x_n]

def df1(x):
```

```
144     return 1 + math.exp(-x)
145
146 def df2(x):
147     return 2 * x
```