

TP 1 - éléments de correction

```
1 import math
2
3 # Exercice 1 -----
4 print("EXERCICE 1\n")
5 def n_premiers_termes_u(n):
6     u_k = 1 # initialisation
7     for k in range(n):
8         print(f"u_{k} = {u_k}")
9         u_k = math.sqrt(1+u_k) # maj de uk
10
11 n_premiers_termes_u(5)
12
13 print(f"Le nombre d'or est : {(1 + math.sqrt(5))/2}")
14
15 # Exercice 2 -----
16 print("\nEXERCICE 2")
17 print("\na")
18 def n_premiers_termes__F(n):
19     F_km2, F_km1 = 1, 1
20     print(f"F_0 = {F_km2}")
21     if n >= 1 :
22         print(f"F_1 = {F_km1}")
23     for k in range(2,n):
24         F_k = F_km1 + F_km2
25         print(f"F_{k} = {F_k}")
26         F_km2 = F_km1 # maj de F_km2 et F_km1
27         F_km1 = F_k
28
29 n_premiers_termes__F(5)
30
31 print("\nb")
32 def n_premiers_termes__adaptee_F(n):
33     F_km2, F_km1 = 1, 1
34     print(f"F_1 / F_0 = {F_km1 / F_km2}")
35     for k in range(2,n):
36         F_k = F_km1 + F_km2
37         print(f"F_{k}/F_{k-1} = {F_k/F_km1}")
38         F_km2 = F_km1 # maj de F_km2 et F_km1
39         F_km1 = F_k
40
41 n_premiers_termes__adaptee_F(5)
42
43 print(f"Le nombre d'or est : {(1 + math.sqrt(5))/2}")
44
45 # Exercice 3 -----
46 print("\nEXERCICE 3\n")
47 print("a")
48
49 def syracuse_avant_1(S_0):
50     S_k = S_0
51     print(f"S_0 = {S_k}")
52     k = 1
53     while S_k != 1 :
54         if S_k % 2 == 0 :
55             S_k = S_k / 2
56         else :
57             S_k = 3 * S_k + 1
58         print(f"S_{k} = {S_k}")
59         k = k + 1
60
61 syracuse_avant_1(3)
62
63 print("\nb")
64
65 def nb_etapes_avant_syracuse_1(N):
66     for S_0 in range(1,N+1):
67         S_k = S_0
68         print(f"S_0 = {S_k}")
69         k = 1
70         while S_k != 1 :
```

```

71         if S_k % 2 == 0 :
72             S_k = S_k / 2
73         else :
74             S_k = 3 * S_k + 1
75         k = k + 1
76         print(f"Nombre d'étapes = {k-1}\n")
77
78 nb_etapes_avant_syracuse_1(5)
79
80 # Exercice 4 -----
81 print("\nEXERCICE 4\nVoir l'autre document pour la correction ")
82
83 ## METHODE DICHOTOMIE
84 print("\nMETHODE DICHOTOMIE\n")
85
86 def dichotomie(f,a,b,tol,prec,n_max):
87     a_n = a
88     b_n = b
89     n = 0
90     while n < n_max and abs(b_n - a_n) >= prec and abs(f((a_n+b_n)/2)) >= tol :
91         c_n = (a_n + b_n) / 2
92         if f(c_n) * f(a_n) > 0 :
93             a_n = c_n
94         if f(c_n) * f(a_n) < 0 :
95             b_n = c_n
96         if f(c_n) * f(a_n) == 0 :
97             print(f"Une solution est {c_n}")
98             n = n + 1
99         if n == n_max :
100             print("ATTENTION ! L'algorithmme s'est arrêté car le n_max a été dépassé.\n")
101             return [n,a_n,b_n]
102
103 def f1(x):
104     return x - math.exp(-x)
105
106 def f2(x):
107     return x**2-2
108
109 tol = 10**(-16)
110 prec = 10**(-10)
111 nmax = 100
112
113 # pour f1
114 f = f1
115 a = 0
116 b = 1
117 [n,a_n,b_n] = dichotomie(f,a,b,tol,prec,nmax)
118 print(f"pour f1:\nmb_itérations = {n}\n[a_n,b_n]=[{a_n},{b_n}]\n")
119
120 # pour f2
121 f = f2
122 a = 1
123 b = 2
124 [n,a_n,b_n] = dichotomie(f,a,b,tol,prec,nmax)
125 print(f"pour f2:\nmb_itérations = {n}\n[a_n,b_n]=[{a_n},{b_n}]\n")
126 print(f"Vérification:\nracine(2)={math.sqrt(2)}")
127
128 ## METHODE DE NEWTON -----
129 print("\nMETHODE NEWTON\n")
130
131 def newton(f,df,x0,tol,prec,nmax):
132     x_n = x0
133     x_np1 = x_n - f(x_n)/df(x_n)
134     n = 0
135     while n < n_max and abs(x_np1 - x_n) > prec and abs(f(x_n))>tol:
136         n = n + 1
137         x_n = x_np1
138         x_np1 = x_np1 - f(x_np1)/df(x_np1)
139     if n == n_max :
140         print("ATTENTION ! L'algorithmme s'est arrêté car le n_max a été dépassé.\n")
141     return [n,x_n]
142
143 def df1(x):

```

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