```
# C:
\Users\edoua\Documents\math\2016-2017\a1a_2023\info\Exercices_correct
ion.py
001| # Exercice 1 :
002|
003| def TermeSuite(n):
004|     u = 2
005|     for i in range(n):
006|         u = u**2 - 5
007|     return u
008|
009| def TermeSuiteRec(n):
010|     if n==0:
011|         return 2
012|     else:
013|         return TermeSuiteRec(n-1)**2 - 5
014|
015| # Exercice 2 :
016|
017| def Monnaie(S,x,y,z):
018|     L = []
019|     for a in range(x+1):
020|         for b in range(y+1):
021|             c = S - 5*a - 2*b
022|             if 0 <= c <= z:
023|                 L.append( (a,b,c) )
024|     return L
025|
026| # Exercice 3 :
027|
028| def Puissance7(n):
029|     k = 0
030|     while 7**k < n:
031|         k = k+1
032|     # Lorsqu'on sort de la boucle, on a 7^k qui est supérieur ou
égal à n
033|     return (7**k == n)
034|
035| # Exercice 4 :
036|
037| def NbVoyelles(s):
038|     Nb = 0
039|     for x in s:
040|         if x=='a' or x=='e' or x=='i' or x=='o' or x=='u' or
x=='y':
041|             Nb += 1
042|     return Nb
043|
044| # Exercice 5 :
045|
046| def Rajoutav(s):
047|     S = ''
048|     n = len(s)
049|     for i in range(n-1):
050|         S = S + s[i]
```

```
051|         if s[i] in 'bcdfghjklmnpqrstvwxz' and s[i+1] in
'aeiouy':
052|             S = S + 'av'
053|     S = S + s[n-1]
054|     return S
055|
056| # Exercice 6 :
057|
058| def MinStrictPos(t):
059|     m = 0
060|     for x in t:
061|         if x>0 and (x<m or m==0):
062|             m = x
063|     if m > 0 :
064|         return m
065|     else :
066|         return None
067|
068| # Exercice 7 :
069|
070| def Distincts(t):
071|     n = len(t)
072|     for i in range(n):
073|         for j in range(i+1,n):
074|             if t[i]==t[j]:
075|                 return False
076|     return True
077|
078| # La complexité est en O(n^2) dans le pire des cas
079|
080| ##
081| # Exercice 8 :
082|
083| def decompose(n):
084|     a = n//1000
085|     n = n-a*1000
086|     b = n//100
087|     n = n-b*100
088|     c = n//10
089|     d = n%10
090|     return (a,b,c,d)
091|
092| def recompose(a,b,c,d):
093|     return a*1000 + b*100 + c*10 + d
094|
095| def tri(a,b,c,d):
096|     # on organise un tournoi pour classer les 4 valeurs
097|     # Demi finales :
098|     if b<a: a,b = b,a
099|     if d<c: c,d = d,c
100|     # Finales (des vainqueurs et des perdants)
101|     if c<a: a,c = c,a
102|     if d<b: b,d = d,b
103|     # Match de la deuxième et troisième place
104|     if c<b: b,c = c,b
```

```
105|     return (a,b,c,d)
106|
107| def nPlus(n):
108|     (a,b,c,d) = decompose(n)
109|     (a,b,c,d) = tri(a,b,c,d)
110|     return recompose(d,c,b,a)
111|
112| def nMoins(n):
113|     (a,b,c,d) = decompose(n)
114|     (a,b,c,d) = tri(a,b,c,d)
115|     return recompose(a,b,c,d)
116|
117| def suivant(n):
118|     return nPlus(n)-nMoins(n)
119|
120| def iterPlusMoins(n):
121|     print(n)
122|     while suivant(n)!=n:
123|         n = suivant(n)
124|         print(n)
125|
126| # En expérimentant la fonction iterPlusMoins(n) pour plusieurs
valeurs de n,
127| # on a l'impression que le résultat finit presque toujours sur
le nombre 6174
128| # (sauf lorsque n a 4 chiffres identiques, et dans ce cas on
tombe sur la valeurs 0 en une itération)
129|
130| # Preuve de la conjecture
131|
132| def DernierResultat_iterPlusMoins(n):
133|     while suivant(n)!=n:
134|         n = suivant(n)
135|     return n
136|
137| RES = []
138|
139| for i in range(1000):
140|     r = DernierResultat_iterPlusMoins(i)
141|     if r not in RES:
142|         RES.append(r)
143|
144| # on obtient : RES = [0,6174]
145|
146| RES_Nul = [i for i in range(10000) if
DernierResultat_iterPlusMoins(i)==0]
147|
148| # RES_Nul contient seulement les 10 entiers où les 4 chiffres
sont identiques
149|
150| ##
151| # Exercice 9 :
152|
153| import matplotlib.pyplot as plt
154| import matplotlib as mpl
```

```
155|
156| def exo1(n):
157|     G = [[0 for i in range(n)] for j in range(n)]
158|     for i in range(n):
159|         G[i][n-1-i] = 1
160|     CM = mpl.colors.ListedColormap(['white','black'])
161|     plt.imshow(G, cmap = CM)
162|     plt.show()
163|
164| def exo2(n):
165|     G = [[0 for i in range(n)] for j in range(n)]
166|     for i in range(n):
167|         for j in range(n):
168|             if (i+j)%2==0 : G[i][j] = 1
169|     CM = mpl.colors.ListedColormap(['white','black'])
170|     plt.imshow(G, cmap = CM)
171|     plt.show()
172|
173| def exo3(n):
174|     G = [[0 for i in range(n)] for j in range(n)]
175|     for i in range(n):
176|         if i % 2 == 0:
177|             for j in range(n) : G[i][j] = 1
178|         if i % 4 == 1:
179|             G[i][n-1] = 1
180|         if i % 4 == 3:
181|             G[i][0] = 1
182|     CM = mpl.colors.ListedColormap(['white','black'])
183|     plt.imshow(G, cmap = CM)
184|     plt.show()
185|
186|
187| def exo4(n):
188|     G = [[0 for i in range(n)] for j in range(n)]
189|     for i in range(n):
190|         for j in range(n):
191|             if (i % 3 == 0) or (j % 3 == 0):
192|                 G[i][j] = 1
193|     CM = mpl.colors.ListedColormap(['white','black'])
194|     plt.imshow(G, cmap = CM)
195|     plt.show()
196|
197| def exo5(n):
198|     G = [[0 for i in range(n)] for j in range(n)]
199|     for i in range(n):
200|         for j in range(n):
201|             if (i % 3 == 0) or (j % 3 == 0):
202|                 G[i][j] = 1
203|             if (i % 3 == 0) and (j % 3 == 0):
204|                 G[i][j] = 2
205|     CM = mpl.colors.ListedColormap(['white','black', 'red'])
206|     plt.imshow(G, cmap = CM)
207|     plt.show()
208|
209| def exo6(n):
```

```
210|     if n%3 != 0:
211|         return 'n doit être un multiple de 3'
212|     G = [[0 for i in range(n)] for j in range(n)]
213|     for i in range(n//3):
214|         for j in range(3*i, 3*i + 3):
215|             for k in range(3*i, 3*i+3):
216|                 G[n-1-j][k] = 1
217|         G[n-2-3*i][3*i+1] = 2
218|
219|     CM = mpl.colors.ListedColormap(['white','black', 'yellow'])
220|     plt.imshow(G, cmap = CM)
221|     plt.show()
222|
223| ##
224| # Exercice 10 :
225|
226| def transition(t):
227|     n = len(t)
228|     T = [0] * n
229|     for i in range(1,n-1):
230|         if t[i-1]==t[i]==t[i+1]:
231|             T[i] = 0
232|         else :
233|             T[i] = 1
234|     return T
235|
236| def Sierpinski():
237|     t = [0]*500 + [1] + [0]*500
238|     G = [t]
239|     for k in range(500):
240|         t = transition(t)
241|         G.append( t )
242|
243|     CM = mpl.colors.ListedColormap(['white', 'black'])
244|     plt.imshow(G, cmap = CM)
245|     plt.show()
246|
247| def AutreConfiguration():
248|     t = [ (k % 2) for k in range(101)]
249|     G = [t]
250|     for k in range(100):
251|         t = transition(t)
252|         G.append( t )
253|     CM = mpl.colors.ListedColormap(['white', 'black'])
254|     plt.imshow(G, cmap = CM)
255|     plt.show()
```