

Captures d'écran des propositions de programme à exécuter sur Python (parties B, C, D)

Partie B : Autres transformations et représentation de la fractale « dragon de Heighway »

(« dragon heighway.py »)

```
1 import matplotlib.pyplot as plt
2 from random import*
3
4 nbpoints=10000
5 # point initial
6 p = (0, 0)
7
8 def transformation_1(p):
9     x = p[0]
10    y = p[1]
11    x1=0.5*x-0.5*y
12    y1=0.5*x+0.5*y
13    return x1,y1
14
15 def transformation_2(p):
16    x = p[0]
17    y = p[1]
18    x1=-0.5*x-0.5*y+1
19    y1=0.5*x-0.5*y
20    return x1,y1
21
22 def transforme(p):
23     # Choix aléatoire (avec équiprobabilité) entre les 2 transformations de fonctions
24     tirage=random()
25     if tirage <1/2 :
26         x, y = transformation_1(p)
27     else :
28         x, y = transformation_2(p)
29     return x, y
30
31 def construction(p, nbpoints):
32     x = [p[0]]
33     y = [p[1]]
34     for i in range(nbpoints):
35         p = transforme(p)
36         x.append(p[0])
37         y.append(p[1])
38     # Graphique
39     plt.plot(x, y, 'o')
40     plt.title('Dragon de Heighway')
41     plt.show()
42
43 construction(p, nbpoints)
44
```

Partie C : La fougère de Barnsley (1998) («fougere barnsley.py »)

```
1 import matplotlib.pyplot as plt
2 from random import*
3
4 nbpoints=10000
5 # point initial
6 p = (0, 0)
7
8 def transformation_1(p):
9     x = p[0]
10    y = p[1]
11    x1=0*x-0*y+0
12    y1=0*x+0.16*y+0
13    return x1,y1
14
15 def transformation_2(p):
16    x = p[0]
17    y = p[1]
18    x1=0.85*x-0.04*y+0
19    y1=-0.04*x+0.85*y+1.6
20    return x1,y1
21
22 def transformation_3(p):
23    x = p[0]
24    y = p[1]
25    x1=0.2*x-0.26*y+0
26    y1=0.23*x+0.22*y+1.6
27    return x1,y1
28
29 def transformation_4(p):
30    x = p[0]
31    y = p[1]
32    x1=-0.15*x+0.28*y+0
33    y1=-0.26*x+0.24*y+0.44
34    return x1,y1
35
36 def transforme(p):
37     # Choix aléatoire entre les 4 transformations de fonctions, probabilités donnés dans l'énoncé : 0.01 ; 0.85 ; 0.07 ; 0.07
38     tirage=random()
39     if tirage <0.01 :
40         x, y = transformation_1(p)
41     elif tirage <0.86:
42         x, y = transformation_2(p)
43     elif tirage <0.93 :
44         x, y = transformation_3(p)
45     else :
46         x, y = transformation_4(p)
47     return x, y
48
49 def construction(p, nbpoints):
50     x = [p[0]]
51     y = [p[1]]
52     for i in range(nbpoints):
53         p = transforme(p)
54         x.append(p[0])
55         y.append(p[1])
56     plt.plot(x, y,'o')
57     plt.title('Fougère de Barnsley')
58     plt.show()
59
60 construction(p, nbpoints)
61
```

Partie C : Une fractale en forme d'arbre (« arbre fratal.py »)

```
1 import matplotlib.pyplot as plt
2 from pylab import *
3 from random import *
4 from math import *
5
6 nbpoints=10000
7 # point initial
8 p = (0, 0)
9 c=0.255
10 r=0.75
11 q=0.625
12 A1=-pi/8
13 A2=pi/5
14
15 def transformation_1(p):
16     x = p[0]
17     y = p[1]
18     x1=0.5
19     y1=c*y
20     return x1,y1
21
22 def transformation_2(p):
23     x = p[0]
24     y = p[1]
25     x1=r*cos(A1)*x-r*sin(A1)*y+0.5-0.5*r*cos(A1)
26     y1=r*sin(A1)*x+r*cos(A1)*y+c-0.5*r*sin(A1)
27     return x1,y1
28
29 def transformation_3(p):
30     x = p[0]
31     y = p[1]
32     x1=q*cos(A2)*x-r*sin(A2)*y+0.5-0.5*q*cos(A2)
33     y1=q*sin(A2)*x+r*cos(A2)*y+0.6*c-0.5*q*sin(A2)
34     return x1,y1
35
36
37 def transforme(p):
38     # Choix aléatoire (avec équiprobabilité) entre les tris transformations de fonctions
39     tirage=random()
40     if tirage < 1/3 :
41         x, y = transformation_1(p)
42     elif tirage < 2/3 :
43         x, y = transformation_2(p)
44     else :
45         x, y = transformation_3(p)
46     return x, y
47
48 def construction(p, nbpoints):
49     x = [p[0]]
50     y = [p[1]]
51     for i in range(nbpoints):
52         p = transforme(p)
53         x.append(p[0])
54         y.append(p[1])
55     # Graphique
56     plt.plot(x, y,'o')
57     plt.title('Un modèle de fractal type arbre')
58     plt.show()
59
60 construction(p, nbpoints)
61
```