

```

1  """Hilbert Curve"""
2  # N.B. This is a bit of a crude version as I couldn't find my old implementation
3  from calendar import leapdays
4  import turtle
5  count = 0
6  wn = turtle.Screen()
7  turtle.screensize(700,700)
8  turtle.hideturtle()
9  turtle.penup()
10 turtle.backward(300)
11 turtle.left(90)
12 turtle.backward(300)
13 turtle.pendown()
14
15 def segments(count):
16     if(count > 1):
17         return 2*segments(count-1) +1
18     else:
19         return 3
20
21 def seedL(length):
22     turtle.forward(length)
23     turtle.left(90)
24     turtle.forward(length)
25     turtle.left(90)
26     turtle.forward(length)
27
28 def seedR(length):
29     turtle.forward(length)
30     turtle.right(90)
31     turtle.forward(length)
32     turtle.right(90)
33     turtle.forward(length)
34
35 def recursiveR(count, length):
36     count -= 1
37     rec = length/segments(count)
38     rec_ = segments(count-1)*rec
39     len = length/3
40     if count>1:
41         turtle.right(90)
42         recursiveL(count,rec_)
43         turtle.right(90)
44         turtle.forward(rec)
45         recursiveR(count,rec_)
46         turtle.left(90)
47         turtle.forward(rec)
48         turtle.left(90)
49         recursiveR(count,rec_)
50         turtle.forward(rec)
51         turtle.right(90)
52         recursiveL(count,rec_)
53         turtle.right(90)
54     else:
55         turtle.right(90)
56         seedL(len)
57         turtle.right(90)
58         turtle.forward(len)
59         seedR(len)

```

```

60     turtle.left(90)
61     turtle.forward(len)
62     turtle.left(90)
63     seedR(len)
64     turtle.forward(len)
65     turtle.right(90)
66     seedL(len)
67     turtle.right(90)
68
69 def recursiveL(count, length):
70     count -= 1
71     rec = length/segments(count)
72     rec_ = segments(count-1)*rec
73     len = length/3
74     if count>1:
75         turtle.left(90)
76         recursiveR(count, rec_)
77         turtle.left(90)
78         turtle.forward(rec)
79         recursiveL(count, rec_)
80         turtle.right(90)
81         turtle.forward(rec)
82         turtle.right(90)
83         recursiveL(count, rec_)
84         turtle.forward(rec)
85         turtle.left(90)
86         recursiveR(count, rec_)
87         turtle.left(90)
88     else:
89         turtle.left(90)
90         seedR(len)
91         turtle.left(90)
92         turtle.forward(len)
93         seedL(len)
94         turtle.right(90)
95         turtle.forward(len)
96         turtle.right(90)
97         seedL(len)
98         turtle.forward(len)
99         turtle.left(90)
100        seedR(len)
101        turtle.left(90)
102
103 try:
104     count = int(input("Psudeo Order: "))
105     recursiveR(count, 600)
106     print("Finished")
107 except ValueError:
108     raise ValueError("Order must be positive interger")
109

```

```
1 """Koch Snoflake"""
2 import turtle
3 count = 0
4 wn = turtle.Screen()
5 turtle.hideturtle()
6 turtle.penup()
7 turtle.backward(400)
8 turtle.pendown()
9
10 def drawseed(length):
11     len = length/3
12     turtle.forward(len)
13     turtle.left(60)
14     turtle.forward(len)
15     turtle.right(120)
16     turtle.forward(len)
17     turtle.left(60)
18     turtle.forward(len)
19
20 def recursive(count, length):
21     count -= 1
22     len = length/3
23     if count>1:
24         recursive(count, len)
25         turtle.left(60)
26         recursive(count, len)
27         turtle.right(120)
28         recursive(count, len)
29         turtle.left(60)
30         recursive(count, len)
31     else:
32         drawseed(len)
33
34 try:
35     count = int(input("Psudeo Order: "))
36     recursive(count, 2400)
37     print("Finished")
38 except ValueError:
39     raise ValueError("Order must be positive interger")
40
```

```

1  """Dragon Curve"""
2  import sys, turtle
3  Gcount = 0
4  wn = turtle.Screen()
5  turtle.hideturtle()
6  turtle.penup()
7  turtle.backward(250)
8  turtle.left(90)
9  turtle.forward(200)
10 turtle.right(135)
11 turtle.pendown()
12
13 def rRecursive(count, length):
14     count -= 1
15     newLength = ((length**2)/2)**0.5
16     if count > 0:
17         turtle.left(45)
18         lRecursive(count, newLength)
19         turtle.right(90)
20         rRecursive(count, newLength)
21         turtle.left(45)
22     else:
23         turtle.left(45)
24         turtle.forward(length)
25         turtle.right(90)
26         turtle.forward(length)
27         turtle.left(45)
28
29 def lRecursive(count, length):
30     count -= 1
31     newLength = ((length**2)/2)**0.5
32     if count > 0:
33         turtle.right(45)
34         lRecursive(count, newLength)
35         turtle.left(90)
36         rRecursive(count, newLength)
37         turtle.right(45)
38     else:
39         turtle.right(45)
40         turtle.forward(length)
41         turtle.left(90)
42         turtle.forward(length)
43         turtle.right(45)
44
45 try:
46     Gcount = int(input("Pseudo Order: "))
47     filename = "Order{0}PseudoDragonCurve.ps".format(Gcount)
48     print(filename)
49     rRecursive(Gcount, 350)
50     print("Finished")
51     turtle.getscreen().getcanvas().postscript(file=filename)
52 except ValueError:
53     raise ValueError("Order must be positive interger")
54
55 sys.exit()

```

```

1  """Sierpinski Triangle"""
2  import sys, turtle
3  Gcount = 0
4  wn = turtle.Screen()
5  turtle.hideturtle()
6  turtle.penup()
7  turtle.backward(350)
8  turtle.left(90)
9  turtle.backward(300)
10 turtle.right(30)
11 turtle.pendown()
12
13 def seed(sideLength):
14     for i in range(3):
15         turtle.forward(sideLength)
16         turtle.right(120)
17
18 def recursive(count, sideLength):
19     count -= 1
20     if count > 0:
21         recursive(count, sideLength/2)
22         turtle.forward(sideLength/2)
23         recursive(count, sideLength/2)
24         turtle.right(120)
25         turtle.forward(sideLength/2)
26         turtle.left(120)
27         recursive(count, sideLength/2)
28         turtle.left(120)
29         turtle.forward(sideLength/2)
30         turtle.right(120)
31     else:
32         seed(sideLength)
33
34 try:
35     Gcount = int(input("Pseudo Order: "))
36     filename = "Order{0}PseudoSierpinskiTriangle.ps".format(Gcount)
37     print(filename)
38     recursive(Gcount, 700)
39     print("Finished")
40     turtle.getscreen().getcanvas().postscript(file=filename)
41 except ValueError:
42     raise ValueError("Order must be positive integer")
43
44 sys.exit()
45

```