Computing Recursive Functions

cs4: Computer Science Bootcamp

Çetin Kaya Koç
http://koclab.cs.ucsb.edu/teaching/cs4
cetinkoc@ucsb.edu

Recursively Defined Sequences

- Fibonacci numbers are defined as the sequence of integers:

$$1,\ 1,\ 2,\ 3,\ 5,\ 8,\ 13,\ 21,\ 34,\ 55,\ 89, \ldots$$

- The generation rule: Every number is the sum of two numbers immediately preceding it:

$$\begin{aligned}
1+1 &\rightarrow 2\\
1+2 &\rightarrow 3\\
2+3 &\rightarrow 5\\
3+5 &\rightarrow 8\\
5+8 &\rightarrow 13\\
&\ldots
\end{aligned}$$

- The starting point: The first two numbers, which are 1, 1

Fibonacci Numbers

- Question: What is F_{100}?
- In general, how do we compute F_n for a given n?
- We know the starting points: $F_0 = 1$ and $F_1 = 1$
- We know the generation rule: the nth Fibonacci number is the sum of the previous two:

$$F_n \;=\; F_{n-1} + F_{n-2}$$

- We can use the starting points and the generation rule to write an iterative function for computing F_n for a given n

Iterative Computation of Fibonacci Numbers

```
def ifib(n):
    f0 = 1
    f1 = 1
    for i in range(2,n+1):
        f2 = f0 + f1
        f0 = f1
        f1 = f2
    return(f1)
```

F_0	F_1	F_2	F_3	F_4	F_5
1	1				
		2			
	F_0	F_1	F_2		
	1	2			
			3		
		F_0	F_1	F_2	
		2	3		
				5	
			F_0	F_1	F_2

Recursive Computation of Fibonacci Numbers

- However, it seems more “natural” to compute them recursively
- Imagine we have a function defined as `fib(n)`
- When we invoke the function with $n = 0$ or $n = 1$, that is we make calls `fib(0)` or `fib(1)`, the function will simply return 1
- When we invoke the function with any value of n larger than 1, the function will execute 2 invocations to itself as: `fib(n-1)` and `fib(n-2)`, and when these values are computed, it will sum them and return the sum

Recursive Python Function for Fibonacci Numbers

```
def fib(n):
    if n==0 or n==1:
        return(1)
    else:
        fn1 = fib(n-1)
        fn2 = fib(n-2)
        return(fn1+fn2)
```

- When we attempt to compute F_0 or F_1, the function returns 1
- When we attempt to compute F_n for $n > 1$, what happens?
- Illustration of F_5

Properties of Fibonacci Numbers

- Fibonacci numbers appear in many areas of arts and sciences
- Several interesting properties have been discovered
- Some of these properties have computational implications

- The ratio of consecutive Fibonacci numbers converge to $\frac{1+\sqrt{5}}{2} \approx 1.618033988749895\cdots$

$$\tfrac{1}{1} = 1.0,\ \tfrac{2}{1} = 2.0,\ \tfrac{3}{2} = 1.5,\ \tfrac{5}{3} = 1.6666666\cdots,\ \tfrac{8}{5} = 1.6,$$

$$\tfrac{13}{8} = 1.625,\ \tfrac{21}{13} = 1.615384\cdots,\ \tfrac{34}{21} = 1.619047\cdots$$

- This constant is termed as “Golden Ratio” and denoted by the Greek letter Φ

Golden Ratio

- In mathematics, two quantities are “in the golden ratio” if their ratio is the same as the ratio of their sum to the larger of the two quantities, that is

$$\frac{a+b}{b} = \frac{a}{b} \quad \rightarrow \quad 1 + \frac{b}{a} = \frac{a}{b}$$

- By assigning $r = \frac{a}{b}$, we obtain

$$1 + \frac{1}{r} = r \quad \rightarrow \quad r^2 - r - 1 = 0$$

- The roots of this quadratic equation are

$$\Phi = \frac{1+\sqrt{5}}{2}, \quad \Psi = \frac{1-\sqrt{5}}{2}$$

Golden Ratio

- The golden ratio also is called the golden mean or golden section (Latin: sectio aurea)
- Other names include extreme and mean ratio, medial section, divine proportion, divine section (Latin: sectio divina), golden proportion, golden cut, and golden number
- It has a fascinating history from the time of Greeks to the 20th century
- It has applications in aesthetics, architecture, painting, music, design, nature, and optimization

Properties of Fibonacci Numbers

- Φ and Ψ were the two roots of $r^2 - r - 1 = 0$

$$\Phi = \frac{1+\sqrt{5}}{2}, \quad \Psi = \frac{1-\sqrt{5}}{2}$$

- Fibonacci numbers have a compact expansion in terms of the Φ and Ψ

$$F_n = \frac{\Phi^{n+1} - \Psi^{n+1}}{\Phi - \Psi} = \frac{\left(\frac{1+\sqrt{5}}{2}\right)^{n+1} - \left(\frac{1-\sqrt{5}}{2}\right)^{n+1}}{\sqrt{5}}$$

- This formula is named Binet's formula or de Moivre's formula
- Computationally, this is an interesting situation since both Φ and Ψ are irrational numbers, while F_n is an integer!
- This implies that if we use this formula for computing F_n for very large values of n, we will have to use infinite precision real arithmetic

Computing F_n using de Moivre's Formula

```
def deMoivre(n):
    sq5 = math.sqrt(5)
    phi = (1+sq5)/2
    psi = (1-sq5)/2
    fn = (phi**(n+1)-psi**(n+1))/sq5
    return(fn)
```

```
n Fn
0 1.0
1 1.0
2 2.0
3.0000000000000004
4 5.000000000000001
5 8.000000000000002
6 13.000000000000002
7 21.000000000000004
```

```
n  Fn
68 117669030460994.27   (94)
69 190392490709135.44   (35)
70 308061521170129.7    (29)
71 498454011879265.2    (64)
72 806515533049395.0    (93)
73 1304969544928660.0   (57)
74 2111485077978055.2   (50)
75 3416454622906715.5   (07)
```

Fractals

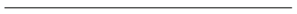
- A fractal is geometric object that exhibits a repeating pattern that displays at every scale
- If the replication is exactly the same at every scale, it is called a self-similar
- Mathematical history of fractals goes back to Gottfried Leibniz who noticed the recursive self-similarity of certain sets (objects)
- After Leibniz Karl Weierstrass presented the first definition of a function with a graph that would be considered fractal
- Georg Cantor published examples of subsets of the real line known as Cantor sets, which are now recognized as fractals

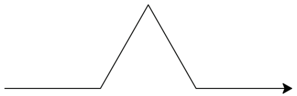
Koch Curves

- One of the important milestones came in 1904, when Helge von Koch gave a more geometric definition including hand drawn images of a similar function, which is now called the Koch curve
- Niels Fabian Helge von Koch (1870-1924) was a Swedish mathematician
- Koch curve has an order n (which is an integer) and a length d (which is a real number)

Koch Curves for $n = 0$ and $n = 1$

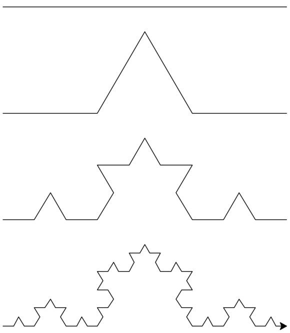
- Koch curve for $n = 0$ and $d = 100$ is just a straight line of length 100
- When the order $n = 1$, the straight line is divided into 3 equal segments and over these segments 4 new Koch curves are drawn such that each line is a Koch curve of order $n = 0$ and length $d/3$
- The angles of the 2nd and 3rd segments are 60 degrees from the 1st and 4th





Koch Curves for any n

- A Koch curve of order n and length d creates 4 new Koch curves, each of which is of order $n - 1$ and length $d/3$, such that:
  - The 1st Koch curve continues in the same direction
  - The 2nd Koch curve turns left 60 degrees and continues
  - The 3rd Koch curve turns right 120 degrees and continues
  - The 4th Koch curve turns left 60 degrees and continues



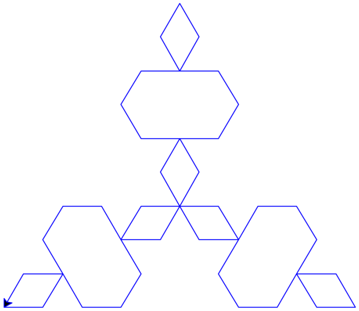
Recursive Python Function for Koch Curve Order *n*

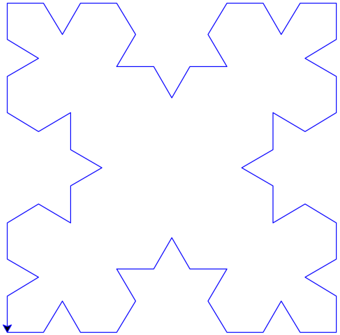
```
def koch(turtle,n,d):
    if n == 0:
        turtle.forward(d)
    else:
        koch(turtle,n-1,d/3)
        turtle.left(60)
        koch(turtle,n-1,d/3)
        turtle.right(120)
        koch(turtle,n-1,d/3)
        turtle.left(60)
        koch(turtle,n-1,d/d)
```

Koch Triangles and Squares

- We can also create Koch triangles, by starting with a equilateral triangle of side d (and thus, $n = 0$), and applying the Koch fractal rule to each side of the triangle
- Or, similarly, we can create Koch squares, by starting with a square of side d (and thus, $n = 0$), and applying the Koch fractal rule to each side of the square
- However, when the Koch rule is applied, the top corner of Koch line fractal can be pointing either to the inside or to the outside of the triangle or square, creating different Koch "snowflakes"

Koch Triangles and Squares for $n = 3$





Koch Triangles and Squares for $n = 3$