Recursion

Part 2

November 9, 2022

Administrative Notes

Project 2 is out

- Due Monday, November 21
- We’ll talk more about it

Midterm 2 is NEXT WEDNESDAY, November 16

- It will cover all material through tonight’s lecture
- There will be a sample exam on GitHub in the next few days
- Next Monday will be a review; we’ll go over the sample exam then
- The structure and rules will be the same as Exam 1

More on Recursion

The rules of recursion:

1. Any problem that can be solved by recursion can also be solved by iteration. The reverse is not true.
2. Recursion is always more expensive than iteration. But you use recursion because it’s easier for the programmer to solve the problem correctly
3. There must be at least one base case, which can be solved directly. There can be more than one base case, but there has to be at least one
4. There must be at least one recursive case. The recursive case must make the problem simpler; that is, it must be closer to a base case

Frames and the Python Stack

A ***frame*** is the set of all symbols (variables, constants, function names) currently in scope - currently known to the Python interpreter

When the program starts, it pushes the main program’s frame onto the stack, and the main program executes.

When the main program calls a function, Python creates a new frame for that function and pushes that frame onto the stack

- Since the function’s frame is on top of the stack, that function is now executing

Now some more recursive examples

- Summing a list of numbers
- Summing the digits of a number
- Calculating x**y

Fibonacci sequence

1, 1, 2, 3, 5, 8, 13,...

After the first two numbers, each number is the sum of the previous two numbers

That is, f(n) = f(n-1) + f(n-2)

Iterative

```
def fib(n):
  If n<= 3:
      return n
  Else:
      fib = [1,1]
      for i in range(3,n+1)
         fib.append(fib[i-1] + fib[i-2])
      return (fib[n])
```

Recursive

```
def fib(n):
  if n < 2
    return 1
  else:
    return (fib(n-1) + fib(n-2))
```