COMP10001 Foundations of Computing
Semester 1, 2019

Tutorial Questions: Week 10

— VERSION: 1474, DATE: MAY 8, 2019 —

Discussion

1. What is "recursion"? What makes a function recursive?

   A: *Recursion is where a function calls itself repeatedly to solve a problem. Rather than using a loop to process something, a recursive function usually calls itself with a smaller or broken-down version of the input until it reaches the answer.*

2. What are the two key parts of a recursive function?

   A: *Recursive functions include a "recursive case", where the function calls itself with a reduced or simpler input; and a "base case" where the function has reached the smallest input or simplest version of the problem and stops recursing.*

3. In what cases is recursion useful? Where should it be used with caution?

   A: *Recursion is useful where an iterative solution would require nesting of loops proportionate to the size of the input, such as the powerset problem or the change problem from lectures. Otherwise, there will often be an equally elegant iterative solution, and since function calls are expensive, it's often more efficient to use the iterative approach. Some algorithms you will learn about in future subjects depend on recursion, and it can be a powerful technique when trying to sort data.*

   Now try Exercise 1

4. Today we'll try playing Comp10001-Go. This is a card game which uses a standard deck of playing cards. Discuss some ways of representing a playing card in Python. In Project 3, we'll use a string consisting of a character for the value and one for the suit.

   A: *For card value, we could represent it with an integer value, allowing comparison via relational operators. This would make it possible to find whether one card were bigger than another, or add one to the value to find the next greater card in a sequence. Kings, Queens, Jacks and Aces would need to be stored as integers though, which would make them harder to recognise for a human reading the code. You might like to keep the single-character string representation for value but use a dictionary constant to access an integer value for a card to use when comparing values. Note that with a character, we can't have the number "10" and it has to be the digit "0" instead.*
   For suit, we could potentially represent it with an arbitrary mapping of integers (1: Heart, 2: Diamond) but it is probably clearest to stay with the string representation of "H", "D", "C" and "S". A data type called an "enumeration" allows storing a set of values such as suits, but we won't teach it in this subject.
   Colour is an important attribute of a suit. You may want to map a suit to its colour using a dictionary. Colour could be stored as a character "B" and "R" or perhaps since there are two options, as a boolean (True=Red, False=Black).
   Students often make good use of dictionary constants to access an integer value which allows ordering and colour for a card.

   Now play Comp10001-Go

Exercises

1. Study the following mysterious functions. For each one, answer the following questions:

   - Which part is the base case?
   - Which part is the recursive case?
   - What does the function do?

   (a)
```python
def mystery(x):
    if len(x) == 1:
        return x[0]
    else:
        y = mystery(x[1:])
        if x[0] > y:
            return x[0]
        else:
            return y
```

A: *The `if` block is the base case, and the `else` block is the recursive case. The function returns the largest element in the list/tuple.*

(b)
```
def mistero(x):
    a = len(x)
    if a == 1:
        return x[0]
    else:
        y = mistero(x[a//2:])
        z = mistero(x[:a//2])
        if z > y:
            return z
        else:
            return y
```

A: *The `if` block is the base case, and the `else` block is the recursive case. Like (a), this function returns the largest element in the list/tuple. This function uses two recursive calls, while the first uses one. There's no difference in the calculated output.*