Recursive Functions

Every recursive solution has this structure:

1. Base case, where the problem is simple enough to be solved directly
2. Recursive case, which has three components
   a. Divide problem into one or more simpler or smaller parts of the problem,
   b. Call the function (recursively) on at least one part
   c. Combine the solutions of the parts into a solution for the problem

Writing Recursive Functions

If you are asked to solve a problem using recursion, follow these steps:
1. (Doc) Write the docstring first, it helps!
2. (Base) Figure out the base case: think of inputs where the answer is easy. If the input is a number this is often 0 or 1. If it’s a string or a list, this is often the empty string or list, or sometimes a string/list with just one letter/item.
3. For the recursive case:
   a. (Divide) Break the problem into two pieces: a piece you can “handle” easily and another piece which is a smaller version of the same problem.
   b. (Recurse) Follow the “have faith” principle. Make a recursive call and have faith the function will work correctly. This is where the docstring is helpful. You assume the function you are calling does what the docstring says it does to that smaller piece.
   c. (Combine) Take the result of the recursive call and the solution to the other smaller piece and combine them into a complete solution.

For example:

Write a recursive function count_e that takes a string s and returns the number of times ‘e’ occurs in s.

Step 1: We know that the function is called count_e and takes a string s as an argument. We define the function and write the docstring:

```python
def count_e(s):
    ```
Step 2: Base case: We know the answer is easy if the string is of length zero. If the string is of length 0 we know there are zero ‘e’s.

```python
if len(s) == 0:
    return 0
```

Step 3: Recursive Case:

a) Divide – We know we can handle a string of length 1 easily. If a string of length 1 is our first piece our second piece gets smaller by removing that one piece we can handle. For the string ‘bee knees’ we handle b (is it an ‘e’ or not) and then shrink the string – the second piece is ‘ee knees’.

```python
# first letter in current string is an ‘e’
eelif s[0] == ‘e’:
    count = 1    # the count of ‘e’s in the string
                 # equals 1
    smaller_piece = s[1:0]   # shrink the string
                 # by removing the
                 # piece we’ve
                 # solved

# first letter in current string is not an ‘e’
else:
    count = 0    # the count of ‘e’s in the string
                 # equals 0
    smaller_piece = s[1:0]   # shrink the string
                 # by removing the
                 # piece we’ve
                 # solved
```

b) Recurse – We know we want to count the number of ‘e’s in the smaller piece. Since our function counts the number of ‘e’s in a string we figure out how many ‘e’s in the smaller piece by calling our function with the smaller piece of the string. So, we modify the code in the Divide step above to read:

```python
# first letter in current string is an ‘e’
eelif s[0] == ‘e’:
    count = 1    # the count of ‘e’s in the string
                 # equals 1
    smaller_piece = s[1:0]   # shrink the string
                 # by removing the
                 # piece we’ve
                 # solved

    # call the function count_e on the smaller
# piece, trust it will return the correct
# count for that string
count_in_smaller_piece = count_e(smaller_piece)

# first letter in current string is not an ‘e’
else:
    count = 0  # the count of ‘e’s in the string
    # equals 0
    smaller_piece = s[1:0]  # shrink the string
    # by removing the
    # piece we’ve
    # solved

    # call the function count_e on the smaller
    # piece, trust it will return the correct
    # count for that string
    count_in_smaller_piece = count_e(smaller_piece)

    # sum up and return
    return count + count_in_smaller_piece

# first letter in current string is an ‘e’
elif s[0] == ‘e’:
    count = 1  # the count of ‘e’s in the string
    # equals 1
    smaller_piece = s[1:0]  # shrink the string
    # by removing the
    # piece we’ve
    # solved

    # call the function count_e on the smaller
    # piece, trust it will return the correct
    # count for that string
    count_in_smaller_piece = count_e(smaller_piece)

    # sum up and return
    return count + count_in_smaller_piece

# first letter in current string is not an ‘e’
else:
    count = 0  # the count of ‘e’s in the string
    # equals 0
    smaller_piece = s[1:0]  # shrink the string
    # by removing the
    # piece we’ve
    # solved
# call the function count_e on the smaller
# piece, trust it will return the correct
# count for that string
count_in_smaller_piece = count_e(smaller_piece)
# sum up and return
return count + count_in_smaller_piece

We are done. The full function now looks like:

```python
def count_e(s):
    """(str) -> int
    Returns the number of times ‘e’ occurs in s.
    Examples
    >>> count_e(‘abc’)
    0
    >>> count_e(‘bees knees’)
    4
    >>>
    if len(s) == 0:
        return 0

    # first letter in current string is an ‘e’
    elif s[0] == ‘e’:
        count = 1  # the count of ‘e’s in the string
                    # equals 1
        smaller_piece = s[1:0]  # shrink the string
                                # by removing the
                                # piece we’ve
                                # solved

        # call the function count_e on the smaller
        # piece, trust it will return the correct
        # count for that string
        count_in_smaller_piece = count_e(smaller_piece)

        # sum up and return
        return count + count_in_smaller_piece

    # first letter in current string is not an ‘e’
    else:
        count = 0  # the count of ‘e’s in the string
                    # equals 0
        smaller_piece = s[1:0]  # shrink the string
                                # by removing the
                                # piece we’ve
                                # solved
```
# call the function count_e on the smaller piece, trust it will return the correct count for that string
count_in_smaller_piece = count_e(smaller_piece)

# sum up and return
return count + count_in_smaller_piece