15-110: Principles of Computing, Spring 2018

Lab 6 – Thursday, February 22

Goals

- Identify the base case and recursive case of a given recursive function
- Implement simple recursive functions over integers, one-dimensional lists, and nested lists

Part 1 : TA Demonstrations

1.1 Recursive Function

Remember that a recursive function has a base case followed by one or more recursive cases. Example:

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

1.2 List

- How do you create a list with all but the first element of a given list?

- Let `x = [1, [4, 5, [6, 7, 8], 10], 12, [14], 15]`

  Why is `len(x)` equal to 5? What is `len(x[1])` ?
  What is `x[1:]` ?

  How can we access the element 7?

  How can we test if an element of `x` is an integer or a list itself?

```
>>> isinstance(x, list)
True
>>> isinstance(x[0], int)
True
```

Part 2 : Student Activities

1. Create a file `multiply.py` using gedit. In `multiply.py`, define a RECURSIVE Python function `multiply(a, b)` that computes the value of `a * b` based on the following RECURSIVE formula. Assume that the function is always called with a non-negative integer for `b`.

$$a \times b = \begin{cases} a & \text{if } b = 1 \\ a + a \times (b-1) & \text{if } b > 1 \end{cases}$$

NOTE: Do not use the multiplication operation (*) in your solution. Instead, think recursively:

```
def multiply(a, b):
    if ___________________:                # base case
        return __________
    else
        return a + multiply(__________________)
```

(The last blank should be a recursive call to the `multiply` function.)

Example Usage:

```
-bash-4.2$ python3 –i multiply.py
>>> multiply(5, 1)
5
>>> multiply(5, 10)
50
```

2. Create a file `stars.py` using gedit. In `stars.py`, define a function `print_stars(n)` that prints n stars (asterisks) in a row recursively, assuming n is a positive integer (n ≥ 1). What is the base case? In the recursive case, print one star (without moving to the next line) and then use the function recursively to print the rest.

After you do this, then write a function `num_to_bar(numlist)` that prints a bar graph of the numbers given in `numlist` using stars. Your solution must use the `print_stars` function you wrote above. You can assume that `numlist` contains positive numbers only.

Example Usage:

```
-bash-4.2$ python3 –i stars.py
>>> num_to_bar([5, 2, 6, 4])
*****
**
******
****
>>> num_to_bar([5, 7, 1])
*****
*******
*
```

3. Recursion is especially useful when searching nested lists (lists of lists). Create a file `sumlist.py` using gedit. In `sumlist.py`, define a RECURSIVE Python function `sumlist(datalist)` that has a parameter representing a list containing integers and returns the sum of all of the data in the list. The twist here is that the data list can have nested lists inside.

Here is the overview of the algorithm to use:

> 1. If the list is empty, return 0 as its sum.
> 2. If the first element of the list is an integer, then return that value plus (recursively) the sum of the rest of the list.
> 3. If the first element of the list is a list, then return (recursively) the sum of that list plus (recursively) the sum of the rest of the list.

Example Usage:

```
-bash-4.2$ python3 –i sumlist.py
>>> sumlist([2, [4, [6, 7, 8], 10]])
37
>>> sumlist([1, [4, 5, [6, 7, 8], 10], 12, [14], 15])
82
>>> sumlist([5])
5
```

Submission

When you finish the lab, you should be inside the `lab6` folder, which is inside the `private/15110` directory. When you type ‘`ls`’ and press the `Enter` key, you should see the following files: **`multiply.py`**, **`stars.py`**, and **`sumlist.py`**. Once you see all files, please type ‘`cd ..`’ and press the `Enter` key. Then, zip your `lab6` folder by typing ‘`zip –r lab6.zip lab6`’. Please submit the zipped file `lab6.zip` on Autolab under lab 6.