Totems

CompSci 101

January 22, 2013
1 Goal

The goal of this first assignment is to introduce students to the notion of encapsulating different functionalities in separate functions. We do so by writing a program that in the end will print a totem pole in the output.

First of all let’s define what a totem pole is. A totem pole consists of several ‘heads’ each of which on its own has different parts. Moreover, each totem ‘head’ consists of:

- The top of the head - e.g. its hair.
- The middle of the head - e.g. its eyes/ears/nose.
- The bottom of the head - e.g. its nose/mouth/chin.

Example  Below you can see two examples of a totem head written with ASCII characters (that is characters found on a keyboard!). Again, do not confuse a totem head with a totem pole! A totem pole is 3 totem heads one on top of the other.
2 Builder Functions

In this part you will write python functions that implement different parts of a head. *Each function that you write must return a string that is 15 characters wide and as many tall as you want it to be.*

2.1 Basic Parts

Initially you are needed to write one python function for each of the following parts:

- Hair
- Side
- Eyes
- Mouth
- Chin
- Nose

The goal here is to have functions that return strings that - given some imagination - look like the correlated part of the head.

An example for the hair function would be:

```python
def hair_build():
    hair = "|||||||||||||||" + "\n"
    return hair
```

2.2 More Diversity

Choose at least 3 parts for which you already wrote a function in section 2.1 and for each of these parts write an additional builder function which this time implements the same part of the totem pole but in a different way.

An example of the second function that implements the hair part would be:

```python
def hair_fancy_build():
    hair = r"\\||||\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\n" + "\n"
    return hair
```

As you can see in the above code, we used the “r” predicate before the string representation. We do saw to declare that whatever follows is raw characters. *The results otherwise would be highly different...*
3 Caller Functions

Now that we have the basic ingredients, we need to somehow combine them into building a complete totem pole. In this section you are needed to build two different functions, both of which call the builder functions that you wrote in section 2.

**totempole():** This function must print a totem pole that consists of three different heads. Each head should be different than the other two heads - you should use the ‘diverse’ parts you implemented on section 2.2.

In order to implement this function, you should make the right calls to the functions you wrote in sections 2.1 and 2.2 and print their returned values.

**randompole():** This function should print a random totem pole. Again the totem pole consists of 3 heads.

Use the random library and its functions provided by Python. Based on random values that you get from these functions, choose different head parts that you wrote in parts 2.1 and 2.2 in order to build each head. This time, it is possible (but highly unlikely) for a totem pole to have 2 heads exactly the same.

An example of how randompole() uses probabilistic decision:

```python
import random

def randompole():
    # randint returns a random integer in the range of [0,1]
    choice = random.randint(0, 1)
    if choice == 0:
        print(hair_build())
    else:
        print(hair_fancy_build())
```

...
4 Deliverables

Turn in a python module that contains all the functions you wrote in sections 2 and 3, as well as a README file - i.e. a plain text file - that contains valuable information like comments, and the resources that you used to finish the assignment. Working together is permitted, what is not permitted is working together without stating so in the README file.

Plagiarism will not be tolerated and we expect everybody to comply with the Duke Community Standard.

Important Dates

• Turn in by Feb. 4: Full points
• Turn in by Feb. 7: 10% down
• Turn in by Feb. 14: 50% down

All deadlines are at 11:55pm on that day. Anything turned in after 11:55 on Feb. 14 will not be graded.

How to submit: Either via Eclipse Ambient or the web submit system. Links to these two methods can be found on the Large Assignment tab on the course website.
5 Appendix

An example of a Python function returning a string of 5 characters wide and 3 characters tall:

```python
1  def 15charWide ():
2    string = "12345" + "\n" + r"\\\n" + "\n" + "54321"
3    return string
```

The “\n” character is the “newline” character. The “\” (backslash) is also known as the ‘escape’ character, which means that whatever follows this character is automatically escaped (i.e. not printed) and instead a special character is printed. In the case of “n” when it is escaped we get a newline instead! The “r” predicate before a string start denotes that whatever follows is raw characters, this means that escape characters are ignored and the string consists exactly of what was typed.

When the multiply operator is applied on a string then it repeats the string as many times as the number with which we multiply. Open your Eclipse and try this:

```python
1  print "This is mildly interesting" + 15*"!
```

Random numbers are a powerful tool in Computer Science. It allows us to build systems that are through randomness are versatile and useful for many inherently different real world scenarios. For more on random functions in Python see here: [Python Random](http://docs.python.org/2/library/random.html)

Given that you have 2 different functions for hair, 2 different functions for eyes, and 2 different functions for mouth, you can build in total 8 different heads of a totem pole. Since a totem pole has 3 heads, that means that we can have 512 different totem poles in total!