Discussion 11: Python I

Learning a Not-So-Foreign Language

1. Translate the following expressions from Snap! to Python:

<table>
<thead>
<tr>
<th>Snap! Expression</th>
<th>Python Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>foo = 5</code></td>
<td><code>foo = 5</code></td>
</tr>
<tr>
<td><code>set foo to 5</code></td>
<td><code>foo = 5</code></td>
</tr>
<tr>
<td><code>change foo by 5</code></td>
<td><code>foo += 5</code></td>
</tr>
<tr>
<td><code>set foo to foo</code></td>
<td><code>foo = foo</code></td>
</tr>
<tr>
<td><code>length of word</code></td>
<td><code>len(word)</code></td>
</tr>
<tr>
<td><code>letter 3 of word</code></td>
<td><code>word[2]</code></td>
</tr>
<tr>
<td><code>join hello world</code></td>
<td><code>join('hello world')</code></td>
</tr>
</tbody>
</table>

2. Translate the following blocks of code from Snap! to Python line by line:

   a. 
   ```
   + is+ word + long?+
   if length of word > 5
   report true
   else
   report false
   ```

   b. 
   ```
   + is+ string +
   for i = 1 to length of string - 1
   for j = i + 1 to length of string
   if letter i of string = letter j of string
   report false
   ```

3. What is the difference between `print` and `return`?
More Practice

1. Define the function Fizzbuzz so that it does the following:
   - Print out the numbers 1 through 100
   - If the number is divisible by 3, print “fizz”.
   - If it is divisible by 5, print “buzz”.
   - If it is divisible by 15, print “fizzbuzz”.

2. Write a function that counts the number of times a given letter appears in a given string. Try writing this iteratively and recursively! Finish one way? Try the other way!

```python
def find_num_letters(letter, str):
```

Errors Galore

We wrote a function called `floor_divide`, which divides a number, `big_num`, by another number, `small_num`, and then reports the answer rounded down to the nearest whole number. Unfortunately, there are a lot of syntax errors! Identify and fix all of the bugs in the code below.

```python
def floor_divide(big_num, small_num):
    if small_num == 0:
        return You cannot divide by zero!
    current_num = small_num
    num_times = 0
    while current_num <= big_num:
        current_num = current_num + small_num
        num_times = num_times + 1
    report num_times
```