## Introduction to Python

### Learning a Not-So-Foreign Language

Translate the following expressions.

<table>
<thead>
<tr>
<th>Expression</th>
<th>Translation</th>
</tr>
</thead>
<tbody>
<tr>
<td>foo = 5</td>
<td></td>
</tr>
<tr>
<td>set foo to 5</td>
<td></td>
</tr>
<tr>
<td>change foo by 5</td>
<td></td>
</tr>
<tr>
<td>length of word</td>
<td></td>
</tr>
<tr>
<td>letter 3 of word</td>
<td></td>
</tr>
<tr>
<td>join h l</td>
<td></td>
</tr>
<tr>
<td>for i = 1 to 10</td>
<td>say i</td>
</tr>
</tbody>
</table>

### Monty Python’s Practice Circus

1. Fill in the `fizzbuzz` function 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”.

```python
def distinct(string):
    for i in range(1, len(string) + 1):
        for j in range(i + 1, len(string) + 1):
            if letter i of string = letter j of string:
                report = False
                return
return True
```

### What is the difference between `print` and `return`?

______________________________________________________________________

```python
def fizzbuzz():
    
```
2. Write a function that will count the number of times a letter appears in a string. For example, if the string was “tinny”, and we were going to find the number of times the letter “n” appears in the string, our function will return 2. If we tried to find the number of times “d” appeared in the string, our function would return 0.

Try writing this iteratively and recursively. Finish one way? Try it the other way!

```python
def find_num_of_letters(str, letter):
    # Code for finding number of times letter appears in str
```

**Bugs? What Bugs?**

We decide to write a function called `floor_divide` which will report the number of times a smaller number can fit into a bigger number. We know our algorithm is right but we notice there are a lot of Python syntax bugs in our code. Identify and fix them!

```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
    return num_times
```

**Extra for Experts: Falling Factorial**

Write a function `falling`, which is a “falling” factorial that takes two arguments, `n` and `k`, and returns the product of `k` consecutive numbers, starting from `n` and working downwards. For example, `falling(10, 3)` will return `720` (`10 * 9 * 8`).

```python
def falling(n, k):
    # Code for calculating falling factorial
```