Discussion 11: Intro to Python

Warm-Up

1. What is the difference between `print` and `return` in Python?

Python's `print` is like Snap!'s `say` (display/side effect); `return` is like `report` (gives a value).

2. Once you write Python code, how do you run it?

First, navigate to the folder in which the file is stored in your computer's terminal.

Then, use one of the following: `python3 <filename>` or `python3 -i <filename>`

3. What is the difference between `python3`, `python3 <filename>`, and `python3 -i <filename>`? What do each of them do?

`python3`: starts up a blank Python session

`python3 <filename>`: runs the code in the provided file; immediately returns to the terminal

`python3 -i <filename>`: runs your python script, opening an interactive session

4. How are `while` loops in Python similar to `repeat until` loops in Snap? How do they differ?

Both complete the relevant tasks in the loop based on a given condition. Python's `while` loops run while the condition is true; Snap!'s `repeat until` loops run until the condition is false.

Learning a Not-So-Foreign Language

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


```
foo == 5
```


```
foo = 5
```


```
foo += 5
```


```
foo = 'foo'
```


```
len('word')
```


```
'word'[2]
```


```
'hello ' + 'world'
```

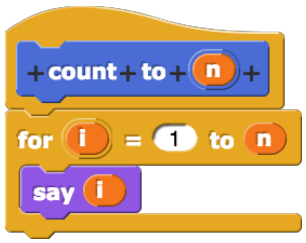

```
'word'[1:]
```


```
[1, 2, 3]
```


```
for item in  [1, 2, 3]:
    print(item)
```


```
while len(lst) != 0:
    lst.remove(lst[-1])
```


```
[1, 2, 3][0]
[1, 2, 3][-1]
len([1, 2, 3])
[1, 2, 3].append(4)
```



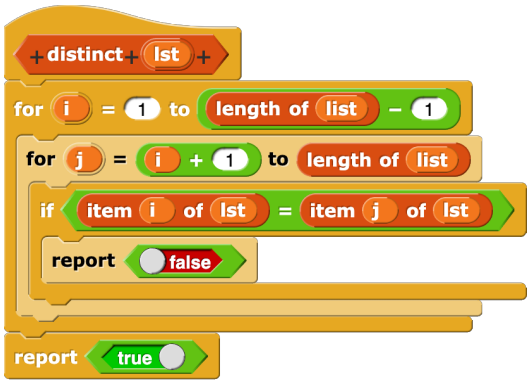

```
def count_to(n):
    for i in range (1, n+1):
        print(i)
```

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

a.


```
def is_long(word):
    if len(word) > 5:
        return True
    else:
        return False
```

b.




```
def distinct(string):
    for i in range(0, len(string) - 1):
        for j in range(i+1, len(string)):
            if string[i] == string[j]:
                return False
    return True
```

Let's Write Some Python

1. Write a function that counts the number of times a given letter appears in a given string. Try writing this both iteratively and recursively!

```
def find_num_letters(letter, str):
    num_letters = 0
    for ltr in str:
        if ltr == letter:
            num_letters += 1
    return num_letters
```

```
def find_num_letters(letter, str):
    if len(str) == 0:
        return 0
    if str[0] == letter:
        return 1 + find_num_letters(letter, str[1:])
    else:
        return find_num_letters(letter, str[1:])
```

2. Define the function Fizzbuzz so that it does the following:

- Iterates through the numbers 1 – 100, and for each number:
  - Prints “fizz” if it is divisible by 3.
  - Prints “buzz” if it is divisible by 5.
  - Prints “fizzbuzz” (and *not* “fizz” or “buzz”) if it is divisible by 15.
  - Prints the number otherwise.

```
def Fizzbuzz():

    for i in range(1, 101):
        if i % 15 == 0:
            print('fizzbuzz')
        elif i % 3 == 0:
            print('fizz')
        elif i % 5 == 0:
            print('buzz')
        else:
            print(i)
```

Errors Galore

We wrote the function `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, it has a lot of syntax errors and doesn’t run. Identify and fix the syntax errors in the code below:

```
def floor_divide(big_num, small_num):
    if small_num = 0: #should be ==
    return You cannot divide by zero! #should be in quotes
    current_num = small_num
    num times = 0
    while current_num <= big_num
        current_num += small_num
        num times += 1
    report num times  #should be return, not report
```