Finishing Strings and introducing While loops

Calling methods vs calling functions

- You need to know whether you are calling a method or a function.
- Example: len is a function, so:

```
len(s)        # Fine
s.len()       # Error
```

- Example: lower is a method, so:

```
lower(s)     # Error
s.lower()    # Fine
```

Some string methods

- S.replace(old, new): return a string, same as S but with all occurrences of old replaced by new. Does not change S.
- S.count(substring): return the number of times substring occurs in S.
- S.find(substring): return the index of the first occurrence of substring in S, starting from the left.
- S.startswith(substring): return True iff S begins with the substring.
- And a very useful *function*: len(string)

Why are some things methods and other things functions?

- Python could have defined len (and find etc.) as a method or a function.
- Programmers can define their own new kinds of objects too. (We'll learn how later.)
- The same decision has to be made for every operation you want to define for your new type of object object: method or function? There is always a choice.

So how does one decide?

- One guideline: if the operation is only relevant to one type of object, make it a method defined for that type of object.
    - Eg: converting to lowercase.
- But if the operation is relevant to other types of object, make it a function, so it can be called with all of those types of object.
    - Eg: finding the length of something.
- You won't be asked to make these decisions in csc108. But the issue may have been bugging you.

Special Characters

- example in Wing

Formatted Printing

```
"%d pounds of %s will cost %.2f dollars"
                    %(amt, item, total_cost)
```

While Loops

How they work

- General format

  while *expression*:
  statements

- Sounds like: while the expression is true, execute the statements.

- But really means: if the expression is true, execute the statements one [more] time. Then go back and consider doing it all again.

- The difference: if the expression becomes false during the statements, it still completes all of them.
  It doesn't stop the loop immediately.