Inserting values into strings

Option 1 - the string `format` method

You can use the string method `format` method to create new strings with inserted values. This method works for all current releases of Python. Here we insert a string into another string:

```
>>> shepherd = "Mary"
>>> string_in_string = "Shepherd {} is on duty.".format(shepherd)
>>> print(string_in_string)
Shepherd Mary is on duty.
```

The curly braces show where the inserted value should go.

You can insert more than one value. The values do not have to be strings, they can be numbers and other Python objects.

```
>>> shepherd = "Mary"
>>> age = 32
>>> stuff_in_string = "Shepherd {} is {} years old.".format(shepherd, age)
>>> print(stuff_in_string)
Shepherd Mary is 32 years old.
```

```
>>> 'Here is a {} floating point number'.format(3.33333)
'Here is a 3.33333 floating point number'
```

You can do more complex formatting of numbers and strings using formatting options within the curly brackets — see the documentation on curly brace string formatting.

This system allows us to give formatting instructions for things like numbers, by using a `:` inside the curly braces, followed by the formatting instructions. Here we ask to print in integer (`d`) where the number should be prepended with `0` to fill up the field width of `3`:

```
>>> print("Number {:03d} is here.".format(11))
Number 011 is here.
```

This prints a floating point value (`f`) with exactly `4` digits after the decimal point:

```
>>> 'A formatted number - {:.4f}'.format(.2)
'A formatted number - 0.2000'
```

See the Python string formatting documentation for more details and examples.

Option 2 - f-strings in Python >= 3.6

If you can depend on having Python >= version 3.6, then you have another attractive option, which is to use the new formatted string literal (f-string) syntax to insert variable values. An `f` at the beginning of the string tells Python to allow any currently valid variable names as variable names within the string. For example, here is an example like the one above, using the f-string syntax:

```
>>> shepherd = "Martha"
>>> age = 34
>>> # Note f before first quote of string
>>> stuff_in_string = f"Shepherd {shepherd} is {age} years old."
>>> print(stuff_in_string)
Shepherd Martha is 34 years old.
```

Option 3 - old school % formatting

There is an older method of string formatting that uses the `%` operator. It is a bit less flexible than the other two options, but you will still see it in use in older code, and where using `%` formatting is more concise.

For `%` operator formating, you show where the inserted values should go using a `%` character followed by a format specifier, to say how the value should be inserted.

Here is the example above, using `%` formatting. Notice the `%s` marker to insert a string, and the `%d` marker to insert an integer.

```
>>> stuff_in_string = "Shepherd %s is %d years old." % (shepherd, age)
>>> print(stuff_in_string)
Shepherd Martha is 34 years old.
```