3.2.11 Printing Formats

While performing the election analysis, you and Tom will have to test your code with Python print statements. Learning how to use a variety of printing formats and adding print statements in your Python scripts is a common practice among programmers. Creating print statements will you help determine if your code is correct when you run it.

As you have already seen in your work so far, there are several ways to print information to the output screen. Knowing how to print in a variety of formats and how to format floating-point decimal numbers is useful for programmers and is often used when writing any Python script.

The different print methods we will cover are the `print()` function, printing single and multiline f-strings, and formatting numbers in print statements.

The print() Function

So far, we have used the Python `print()` function to print the following:

1. A string, or a sentence displayed between quotes. For example: `print("Hello World")` and `print("Arapahoe and Denver are not in the list of counties.")`.
2. A string with integer values converted to a string using concatenation with the "+" sign. For example: `print("Your interest for the year is $" + str(interest))` .

Both of these are sufficient when printing simple statements, but using concatenation can become cumbersome when we need to print items from a list of values from a dictionary. This is where f-strings come in.

F-strings: Formatted String Literals

With Python 3.7, printing has become much easier with the use of **f-string literals,** which can be used in place of concatenation. The general format for f-strings is as follows:

1. The f-string begins with an `f` followed by a string contained within quotes. (The term f-string comes from the leading "f" character preceding the string literal.)
2. In the f-string, curly braces are used to add variables or expressions to the f-string.

To see an example, let's edit the code we wrote to calculate the percentage of votes using f-string literals.

Here's the original code:

```
my_votes = int(input("How many votes did you get in the election? "))
total_votes = int(input("What is the total votes in the election? "))
percentage_votes = (my_votes / total_votes) * 100
print("I received " + str(percentage_votes)+"% of the total votes.")
```

And here's how you would edit the code to use f-strings.

```
my_votes = int(input("How many votes did you get in the election? "))
total_votes = int(input("What is the total votes in the election? "))
print(f"I received {my_votes / total_votes * 100}% of the total votes.")
```

Look at the last line in particular. Inside the curly braces, the f-string performs the calculation `my_votes / total_votes * 100` and formats the value as a string. There is no need to convert `percentage_votes` in the original code to the string format. This makes the code more concise and easier to read!

Using F-Strings with Dictionaries

F-strings can be used to print out the keys or values of a dictionary. This will make our code easier to write and read.

Let's edit the code you may have written for the Skill Drill where you needed to print each county and registered voter from the counties dictionary.

Here's the counties dictionary and the solution for that Skill Drill if we use concatenation.

```
counties_dict = {"Arapahoe": 369237, "Denver":413229, "Jefferson": 390222}
for county, voters in counties_dict.items():
    print(county + " county has " + str(voters) + " registered voters.")
```

The output should when this code is executed should look like this:

```
Arapahoe county has 369237 registered voters.
Denver county has 413229 registered voters.
Jefferson county has 390222 registered voters.
```

If we use f-stings, we can rewrite the print statement to be more intuitive and clear.

```
for county, voters in counties_dict.items():
    print(f"{county} county has {voters} registered voters.")
```

Multiline F-Strings

Another use for f-strings is to print multiple strings or lines to the screen. Let's say you need to tell a candidate how many votes they won, the total number of votes, and the percentage of votes they received. You can use the code you wrote to calculate the percentage of votes and create a message to be printed to a screen, like this:

```
candidate_votes = int(input("How many votes did the candidate get in the ele
total_votes = int(input("What is the total number of votes in the election?
message_to_candidate = (
    f"You received {candidate_votes} number of votes. "
    f"The total number of votes in the election was {total_votes}. "
    f"You received {candidate_votes / total_votes * 100}% of the total votes

print(message_to_candidate)
```

When we run this file in VS Code, the output will look like this when you input 3,345 for the candidate's votes and 23,123 for the total votes.

```
You received 3345 number of votes. The total number of votes in the election
```

Format Floating-Point Decimals

Notice that in the output from the algorithm above, the percentage of votes is, 14.466115988409808! In Python, we can format numbers with a thousands separator and specify a decimal precision.

The general format for a number to format it in an f-string is as follows:

```
f'{value:{width}.{precision}}'
```

In the format, the `width` specifies the number of characters used to display the value. However, if a value needs more space than the width specifies, the additional space is used.

The `precision` indicates the number of decimal places to format the value. For example, to format the interest to two decimal places, we would use `.2f`, where `f` means "floating-point decimal format".

When formatting a number, we can also add a thousands separator with a comma, using the following format. The comma is placed after the `{width}`.

```
f'{value:{width},.{precision}}'
```

Let's add a thousands separator to the output for the candidate votes and total votes and then format the percentage of votes to two decimal places. The code should look like this:

```
message_to_candidate = (
    f"You received {candidate_votes:,} number of votes. "
    f"The total number of votes in the election was {total_votes:,}. "
    f"You received {candidate_votes / total_votes * 100:.2f}% of the total v
```

When this file is run in VS Code, the output will look like this when you input 3,345 for the candidate's votes and 23,123 for the total votes.

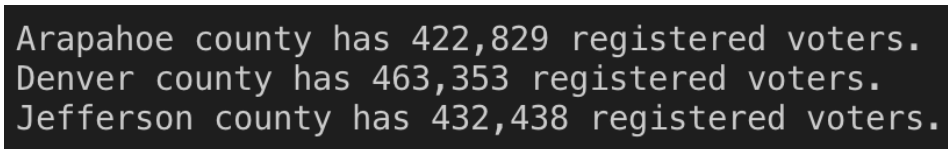
```
You received 3,345 number of votes. The total number of votes in the electio
```

SKILL DRILL

Refer to the following dictionary to complete the activity.

`counties_dict = {"Arapahoe": 422829, "Denver": 463353, "Jefferson": 432438}`

Print each county and registered voter from the dictionary. The output should look like the following:



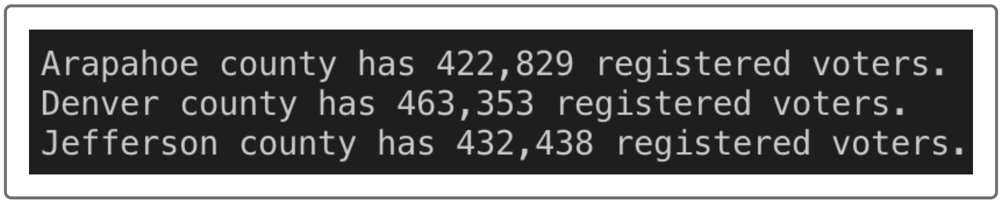
```
Arapahoe county has 422,829 registered voters.
Denver county has 463,353 registered voters.
Jefferson county has 432,438 registered voters.
```

SKILL DRILL

Refer to the following dictionary to complete the activity.

`voting_data = [``{"county":"Arapahoe", "registered_voters": 422829}, {"county":"Denver", "registered_voters": 463353}, {"county":"Jefferson", "registered_voters": 432438}]`

Print each county and registered voter from the dictionary. The output should look like the following:



```
Arapahoe county has 422,829 registered voters.
Denver county has 463,353 registered voters.
Jefferson county has 432,438 registered voters.
```

SHOW HINT

NOTE

> For more information, see the **documentation on formatting f-string literals** (https://www.python.org/dev/peps/pep-0498/).

Congratulations, you made it through this programming-packed day! That was a lot of information thrown at you in a short time. We recommend more practice so that you can keep these concepts fresh in your memory.