A. String indexing

We begin by developing a program that asks a user to input a seven-digit phone number using the standard representation (e.g., `'977-6667'`). However, we wish to store this number as a string that does not include the hyphen (e.g., `'9776667'`). For the sake of demonstration, we wish to have a program behave as follows.

Sample Execution
`Enter phone number: 977-6667` `Dialing 9776667`

We consider a first implementation for such a task as follows.

Implementation #1	
1.	`raw = input('Enter phone number: ')`
2.	`stored = raw[0]+raw[1]+raw[2]+raw[4]+raw[5]+raw[6]+raw[7]`
3.	`print('Dialing', stored)`

1. For the sample execution shown above, what is the value of variable `raw`?

2. For the sample execution shown above, what is the value of variable `stored`?

3. What is the index of the hyphen within the string known as `raw`?

4. How was the hyphen excluded from the stored result in this code?

5. Python supports a convenient slicing notation, such as `raw[start:stop]` for integers `start` and `stop`. For string `raw`, this syntax produces a new string that includes all characters starting at index `start` and going up to but not including index `stop`. Rewrite line 2 of the code above, so that it concatenates two slices of the original string, namely all digits preceding the hyphen and all digits following the hyphen.

6. Because we often want slices from the beginning or end of a string, the slicing notation has another convenience, The slicing notation has another convenience, which is that if you omit index start (but still include the colon), it presumes you want to start at the beginning of the string. If you omit the stop index, it presumes you want the slice to go all the way to the end. Rewrite your answer to Question #5 to take advantage of this convenience (if you did not already do so).

7. There is a single space after the colon in the first line of the execution because we included that space in the prompt string, not because the user typed a space before the digit `'9'`. However, consider how this program would behave had the user also typed a single space character prior to the `'9'` as shown in the following execution.

Sample Execution
`Enter phone number:  977-6667` `Dialing ???????`

What would be displayed as the stored value in this case? Why?

Our original implementation took advantage of the fact that we knew the standard convention placed the hyphen after the first three digits of the phone number. This second implementation assumes that there will be a single hyphen, but it avoids the assumption that there are three digits before the hyphen.

Implementation #2	
1.	`raw = input('Enter phone number: ')`
2.	`hyphenLoc = raw.index('-')`
3.	`stored = raw[ :hyphenLoc] + raw[1+hyphenLoc: ]`
4.	`print('Dialing', stored)`

8. In our original execution, in which raw = `'977-6667'`, what is the value of hyphenLoc? What is the resulting value of `stored`?

9. In our second execution, in which raw = `' 977-6667'`, what is the value of hyphenLoc? What is the resulting value of `stored`?

10. In India, phone numbers do not always have the same number of digits preceding a hyphen. For example, the local phone number of the US consulate in Mumbai is 4625-822. Run the second implementation using such input. What is reported as the stored value?

11. What happens if the user enters input without a hyphen, thus with raw = `'9776667'`?

[PAUSE FOR A CLASS DISCUSSION]

B. String methods

There are a variety of useful methods of the string class in Python. For example the method `s.strip()` produces a version of string `s` with all leading or trailing whitespace removed. The method `s.replace(from, to)` produces a new string that is a copy of `s`, but with all occurrences of pattern `from` replaced by string `to`.

12. Returning to Implementation #1, modify the code to use the strip method to remove extraneous whitespace from the user input. Then test that implementation in the scenario in which the user entered extraneous whitespace, such as when raw = `' 977-6667'`.

13. Give a new implementation that does not use string indexing or slicing at all, and instead which uses the replace method to replace any hyphen with an empty string. Test your implementation on input `'977-6667'`. Test it on input `' 977-6667'`.

14. If including area code, or even country codes, a user may enter a phone number such as `'1-314-977-6667'`. Test your latest implementation on that number.

15. What happens if the user enters input without a hyphen, thus with raw = `'9776667'`?

C. Splitting and joining strings

We can accomplish our task another way by taking advantage of the split and join methods of the string class. Consider the following implementation.

Implementation #3

```
1. raw = input('Enter phone number: ')
2. pieces = raw.split('-')
3. stored = ''.join(pieces)    # join on empty string
4. print('Dialing', stored)
```

16. Assuming the original user interaction with `raw = '977-6667'`, what is the precise value of variable `pieces`?

17. Assuming a user interaction with `raw = '1-314-977-6667'`, what is the precise value of variable `pieces`?

18. What happens if the user enters input without a hyphen, thus with raw = `'9776667'`?

19. Our programs thus far have output the dialing sequence purely as numeric digits, such as

Sample Execution
`Enter phone number: 1-314-977-6667` `Dialing 13149776667`

Modify the Implementation #3 so that it displays the dialing sequence with a space between each pair of numbers, thus as

Sample Execution
`Enter phone number: 1-314-977-6667` `Dialing 1 3 1 4 9 7 7 6 6 6 7`

20. Modify the program to instead produce the following output, assuming that the user input will always use a form that starts with a country code, a hyphen, an area code, another hyphen, and then the remainder of the local phone number.

Sample Execution

`Enter phone number: 1-314-977-6667`
`Country code: 1`
`Area code: 314`
`Local number: 977-6667`

[PAUSE FOR A CLASS DISCUSSION]

D. Lists and tuples

Implementation #4

```
1. depts = [ ('CS','977-6667'), ('Math','977-2444') ]
2. print(depts[0])
3. print(depts[1])
4. print(depts[0][1])
5. print(depts[1][0])
6. depts.append( ('Biology','977-3910') )
7. depts[1][0] = 'Math/Stat'
```

Examine this code but do not execute it yet.

21. What output do you think will be produced by line 2?

22. What output do you think will be produced by line 3?

23. What output do you think will be produced by line 4?

24. What output do you think will be produced by line 5?

25. In your own words, what will be the effect of executing line 6?

26. In your own words, what will be the effect of executing line 7?

27. Now, go ahead and execute the code. How were your predictions?

E. [Bonus] Compare and Contrast the Three Sequence Types

Let's compare and contrast the three sequence classes we studied. Fill out the table below, putting an "✓" in the cell where the functionality listed in the row is supported by the class.

Supports	**str**	**tuple**	**list**
indexing			
slicing			
mixed object types			
methods that change the object			
methods that return a modified version of the object (and leave the original object unchanged)			

Based on your comparison, draw a conclusion about similarities and differences of str, tuple, and list classes.