1. Try using the Python interpreter as a calculator, and typing expressions like 12 / (4 + 1).
2. Given an alphabet of 26 letters, there are 26 to the power 10, or 26 ** 10, 10-letter strings we can form. That works out to 141167095653376L (the L at the end just indicates that this is Python's long-number format). How many hundred-letter strings are possible?
3. The Python multiplication operation can be applied to lists. What happens when you type ['Monty', 'Python'] * 20, or 3 * sent1?
4. Consider the following Python expression: len(set(text4)). State the purpose of this expression. Describe the two steps involved in performing this computation.
5. String Processing
    a. Define a string and assign it to a variable, e.g., my_string = 'My String' (but put something more interesting in the string). Print the contents of this variable in two ways, first by simply typing the variable name and pressing Enter, then by using the print statement.
    b. Try adding the string to itself using my_string + my_string, or multiplying it by a number, e.g., my_string * 3. Notice that the strings are joined together without any spaces. How could you fix this?
6. Define a variable my_sent to be a list of words, using the syntax my_sent = ["My", "sent"] (but with your own words, or a favorite saying).
    a. Use ' '.join(my_sent) to convert this into a string.
    b. Use split() to split the string back into the list form you had to start with.
7. Define several variables containing lists of words, e.g., phrase1, phrase2, and so on. Join them together in various combinations (using the plus operator) to form 1.8 Exercises | 35 whole sentences. What is the relationship between len(phrase1 + phrase2) and len(phrase1) + len(phrase2)?
8. What is the difference between the following two lines? Which one will give a larger value? Will this be the case for other texts?
    a. >>> sorted(set([w.lower() for w in text1]))
    b. >>> sorted([w.lower() for w in set(text1)])
9. What is the difference between the following two tests: w.isupper() and not w.islower()?
10. Write the slice expression that extracts the last two words of text2.
11. Write expressions for finding all words in text6 that meet the following conditions. The result should be in the form of a list of words: ['word1', 'word2', ...].
    a. Ending in ize
    b. Containing the letter z
    c. Containing the sequence of letters pt
    d. All lowercase letters except for an initial capital (i.e., titlecase)

12. Define sent to be the list of words ['she', 'sells', 'sea', 'shells', 'by', 'the', 'sea', 'shore']. Now write code to perform the following tasks:
    a. Print all words beginning with sh.
    b. Print all words longer than four characters
13. What does the following Python code do? sum([len(w) for w in text1]) Can you use it to work out the average word length of a text?
14. Define a function called vocab_size(text) that has a single parameter for the text, and which returns the vocabulary size of the text.
15. Define a function percent(word, text) that calculates how often a given word occurs in a text and expresses the result as a percentage.
16. We have been using sets to store vocabularies. Try the following Python expression: set(sent3) < set(text1). Experiment with this using different arguments to set(). What does it do? Can you think of a practical application for this?
17. Define a string s = 'colorless'. Write a Python statement that changes this to “colourless” using only the slice and concatenation operations.
18. We can use the slice notation to remove morphological endings on words. For example, 'dogs'[:-1] removes the last character of dogs, leaving dog. Use slice notation to remove the affixes from these words (we’ve inserted a hyphen to indicate the affix boundary, but omit this from your strings): dish-es, run-ning, nationality, un-do, pre-heat.
19. Write regular expressions to match the following classes of strings:
    a. A single determiner (assume that a, an, and the are the only determiners)
    b. An arithmetic expression using integers, addition, and multiplication, such as 2*3+8
20. Write a utility function that takes a URL as its argument, and returns the contents of the URL, with all HTML markup removed. Use urllib.urlopen to access the contents of the URL
21. Create a variable words containing a list of words. Experiment with words.sort() and sorted(words). What is the difference?
22. Earlier, we asked you to use a text editor to create a file called test.py, containing the single line monty = 'Monty Python'. If you haven’t already done this (or can’t find the file), go ahead and do it now. Next, start up a new session with the Python interpreter, and enter the expression monty at the prompt. You will get an error from the interpreter. Now, try the following (note that you have to leave off the .py part of the filename):
    a. >>> from test import msg
    b. >>> msg

c. This time, Python should return with a value. You can also try import test, in which case Python should be able to evaluate the expression test.monty at the prompt.

23. Write code to access a favorite web page and extract some text from it. For example, access a weather site and extract the forecast top temperature for your town or city today
24. Try to write code to convert text into hAck3r, using regular expressions and substitution, where e → 3, i → 1, o → 0, l → |, s → 5, . → 5w33t!, ate → 8. Normalize the text to lowercase before converting it. Add more substitutions of your own. Now try to map s to two different values: $ for word-initial s, and 5 for word-internal s
25. Define the variable saying to contain the list ['After', 'all', 'is', 'said', 'and', 'done', ',', 'more', 'is', 'said', 'than', 'done', '.']. Process the list using a for loop, and store the result in a new list lengths. Hint: begin by assigning the empty list to lengths, using lengths = []. Then each time through the loop, use append() to add another length value to the list
26. Define a variable silly to contain the string: 'newly formed bland ideas are inexpressible in an infuriating way'. (This happens to be the legitimate interpretation that bilingual English-Spanish speakers can assign to Chomsky's famous nonsense phrase colorless green ideas sleep furiously, according to Wikipedia). Now write code to perform the following tasks:
    a. Split silly into a list of strings, one per word, using Python's split() operation, and save this to a variable called bland.
    b. Extract the second letter of each word in silly and join them into a string, to get 'eoldrnnnna'.
    c. Combine the words in bland back into a single string, using join(). Make sure the words in the resulting string are separated with whitespace.
    d. Print the words of silly in alphabetical order, one per line.
27. Write code to convert nationality adjectives such as Canadian and Australian to their corresponding nouns Canada and Australia (see http://en.wikipedia.org/wiki/List_of_adjectival_forms_of_place_names).
28. An interesting challenge for tokenization is words that have been split across a linebreak. E.g., if long-term is split, then we have the string long-\nterm.
    a. Write a regular expression that identifies words that are hyphenated at a linebreak. The expression will need to include the \n character.
    b. Use re.sub() to remove the \n character from these words.
    c. How might you identify words that should not remain hyphenated once the newline is removed, e.g., 'encyclo-\npedia'?
29.