GLOBAL EDITION

The Practice of Computing Using Python

THIRD EDITION

Punch • Enbody

Pearson

Digital Resources for Students

Your new textbook provides 12-month access to digital resources that may include VideoNotes (step-by-step video tutorials on programming concepts), source code, web chapters, quizzes, and more. Refer to the preface in the textbook for a detailed list of resources.

Follow the instructions below to register for the Companion Website for William Punch and Richard Enbody's *The Practice of Computing Using Python*, Third Edition, Global Edition.

1. Go to www.pearsonglobaleditions.com/punch
2. Enter the title of your textbook
3. Click Companion Website
4. Click Register and follow the on-screen instructions to create a login name and password.

Use a coin to scratch off the coating and reveal your access code.
Do not use a sharp knife or other sharp object as it may damage the code.

Use the login name and password you created during registration to start using the digital resources that accompany your textbook.

IMPORTANT:

This prepaid subscription does not include access to MyProgammingLab, which is available at **www.myprogramminglab.com** for purchase.

This access code can only be used once. This subscription is valid for 12 months upon activation and is not transferrable. If the access code has already been revealed it may no longer be valid.

For technical support go to https://support.pearson.com/getsupport

38. Similar to `.lower`: write a program that prompts for a string and prints the string in lowercase without using the string.lower method.

39. In the palindrome example we used `replace` to remove bad characters. Refactor that program to **keep** the good characters rather than remove the bad characters.

40. Given the following code:

```
x=input("Enter a string: ")
y=0
for i in x:
    print(y,i)
    y+=1
```

(a) What will be printed, if `"hello"` is entered?
(b) Refactor the code using enumerate.

41. Although Python's formatted printing can be cumbersome, it can often drastically improve the readability of output. Try creating a table out of the following values:

Melting and Boiling Points of Alkanes

Name	Melting Point (deg C)	Boiling Point (deg C)
Methane	−162	−183
Ethane	−89	−172
Propane	−42	−188
Butane	−0.5	−135

42. Write a program to prompt the user to input a string, and then count the number of digits, letters, and special characters present in the given string. Format the output into a table.

43. Write a program that plays the game of hangman. Use characters to print the hangman's status. Triple-quoted strings will be useful.
Hint: Draw the entire hangman status as a string picture with a full picture for each partial hangman status.

44. Write a program that prompts for a sentence and calculates the number of uppercase letters, lowercase letters, digits, and punctuation. Output the results neatly formatted and labeled in columns.

45. Write a program that returns True if the sentence string `sentence_str` contains every letter of the alphabet. If the sentence does not contain each letter, print which ones are missing. The results should not depend on capitalization.

46. Jacob has been passing notes in class without a problem until his teacher intercepted one today. In order to prevent future embarrassment, he wants a way to encode his notes so his teacher cannot read them. Help him by writing a program that encrypts a string by switching every letter with alphabetical position i with the letter with alphabetical position 26-i. It should be: letter `12 = 'm'` becomes the letter `26-12 = 14 = 'o'`.

47. You are creating a new account and need to provide a password. The password has the following requirements:

    (a) The password must be at least 6 characters and at most 20 characters.
    (b) It must contain at least one lowercase letter, one uppercase letter, and one number.

    Write a program that prompts the user to input a password and checks if the password is valid. If the password is valid, print a confirmation statement. If it is not, print a statement that the password is not valid.

48. Write a program that prints a multiplication table for the numbers from 0 to 12. Do not do any of your calculations by hand. Format the output into a table.

49. Using the incredible power of the human mind, according to research at Cambridge University, it doesn't matter in what order the letters in a word are, the only important thing is that the first and last letter be in the right place. The rest can be total mess and you can still read it without a problem. This is because the human mind does not read every letter by itself, but the word as a whole. Amazing, huh? Yeah and I always thought spelling was important! Write a program that prompts for a word and prints the scrambled version (with first and last fixed).

    Hint: The random module helps scramble words.

50. Develop a Python program that will: (1) prompt for input, (2) extract all the vowels in the input string and make them lowercase, (3) add all the nonduplicate vowels you found into a temporary string, and (4) stop the code when you have collected all five vowels (aeiou) and print the number of trials that you did to collect all five vowel characters.

51. Write a program that prompts for the user to input a sentence. Then check this sentence to make sure the first word of the sentence is capitalized and the sentence ends with a punctuation mark. If it is not properly written, fix the sentence, print the type of error, and print the fixed sentence.

52. Write a program that prompts the user to enter a three-digit number such that the digits are in order, for example, 123789. The program loops until a correct value is entered.

Programming Projects

1. **Mastermind**
   Mastermind is a code-breaking game for two players; play can be simulated in text on a computer. Online versions exist and are useful for understanding how the game is played, but if you can get a hold of the actual board game, that is even better. The game is played using the following:

   - A decoding board, with a shield at one end covering a row of four large holes, and twelve additional rows containing four large holes next to a set of four small holes;

- Code pegs of six different colors (we'll use "colors" ABCDEF), with round heads, which will be placed in the large holes on the board; and
- Scoring pegs, some black, some white, that are flat-headed and smaller than the code pegs; they will be placed in the small holes on the board. Only the quantity of black and white scoring pegs in each row matter in the game.

One player, the codemaker, selects four colors that are shielded from the other player, the codebreaker. In our version, colors cannot repeat, that is, AABB is illegal. The goal of the game is for the codebreaker to correctly determine both the four colors selected by the codemaker and their position in the code.

The codebreaker tries to guess the pattern, in both order and color, within 12 turns. Each guess is made by placing a row of code pegs on the decoding board. Once placed, the codemaker provides feedback by placing from zero to four scoring pegs in the small holes of the row with the guess. A black scoring peg is placed for each code peg from the guess that is correct in both color and position. A white peg indicates the existence of a correct color peg placed in the wrong position.

Once feedback is provided, another guess is made; guesses and feedback continue to alternate until either the codebreaker guesses correctly, or 12 incorrect guesses are made.

Write a program that simulates the game by providing the feedback. The codebreaker will input each guess by entering a string of 'colors'. Your simulation will ensure that guessing rules are followed: the guess consists of exacly four colors from 'ABCDEF'. Feedback will be a count of black pegs and a count of white pegs. Your program will determine the feedback and print it. The program will declare a win if the guess is correct or a loss after 12 incorrect guesses. In addition, the program should print the complete board state so the codebreaker can more easily view the history of guesses made.

Hints:

- Play the game using paper and pencil to understand how the game is played before designing your game playing algorithm.
- Use strings for the code and guesses.
- Use a string `'ABCDEF'` for the set of allowable colors so you can check membership using "in".
- The `isalpha` string method is useful for checking input.
- The history can be built as a long string using concatenation. The end-of-line character `'\n'` will be useful for readable output.

(a) The first version of the program should prompt for the codemaker's code. Such a game program isn't much fun to play, but it is easier to test.

(b) The final version should use the random module to create the codemaker's code so it can be kept shielded from the codebreaker.

   i. Use the `index = random.randint(start,end)` function (from Section 2.2.10) to generate random indices, *start* ≤ *index* ≤ *end*, to select code characters from `'ABCDEF'`.

ii. Or, use `random.sample(population,k)` that returns a sample of length k from a specified population. The `join` expression (that we will learn the meaning of later in Chapter 7) converts to a string what is returned by the `sample` function:

```
code = ''.join(random.sample('ABCDEF',4))
```

2. Mad Libs

Mad Libs (madlibs.com) is an old word game for children. (If you have never played Mad Libs, try it at eduplace.com/tales/) You are prompted for categories of words (color, girl's name, place, etc.) and then those words are inserted into a predefined story. The predefined story has place holders for those words which get replaced by the values prompted for. For example, Suppose you are prompted for a verb and a noun and respond with "giggle" and "spark". If the predefined string was the first line of Hamlet in this form:

To VERB or not to VERB: that is the NOUN:

The revised version will be:

To giggle or not to giggle: that is the spark:

Create your own predefined story with parts of speech replaced with their description in all capitals: VERB, NOUN, ADJECTIVE, and so on. Your story will be more interesting if you augment your words to be replaced with others appropriate for your story: BOYS_NAME, COLOR, ACTIVE_VERB, and so on. Be creative. If you are at a loss for ideas, begin with a fairy tale. For example, here is the beginning of "Little Red Riding Hood" (Note the backslash continuation character \):

```
story = "Once upon a time in the middle of a ADJECTIVE_ONE NOUN_ONE stood a \
         ADJECTIVE_TWO NOUN_TWO, the home of a ADJECTIVE_ONE ADJECTIVE_THREE \
         NOUN_THREE known to everyone as GIRLS_NAME."
```

Prompt the user for strings to replace the various parts of speech that you have specified. Print out the revised story.

Hint: The string method `replace(old,new)` will be helpful.

3. Pig Latin

Pig Latin is a game of alterations played on words. To make the Pig Latin form of an English word the initial consonant sound is transposed to the end of the word and an "ay" is affixed. Specifically there are two rules:

(a) If a word begins with a vowel, append "yay" to the end of the word.
(b) If a word begins with a consonant, remove all the consonants from the beginning up to the first vowel and append them to the end of the word. Finally, append "ay" to the end of the word.

For example:

- dog ⇒ ogday
- scratch ⇒ atchscray
- is ⇒ isyay
- apple ⇒ appleyay

Write a program that repeatedly prompts for an English word to translate into Pig Latin and prints the translated word. If the user enters a period, halt the program.

Hints:

- Slicing is your friend: it can pick off the first character for checking, and you can slice off pieces and concatenate to yield the new word.
- Making a string of vowels allows use of the "in" operator: `vowels = 'aeiou'`.

CHAPTER 5

Functions—QuickStart

Function, the exercise, or executing of some office or charge.

T. Blount, Glossographia, 1656 Earliest definition of function in the *Oxford English Dictionary*

YOU HAVE HAVE SEEN EXAMPLES OF USING PYTHON FUNCTIONS. IN THIS CHAPTER, you'll learn how to create your own functions.

The concept of a function should be familiar from its use in mathematics. Functions in programming languages share many of the characteristics of mathematical functions, but add some unique features as well that make them more useful for programming.

One of the main advantages for using functions is that they support divide-and-conquer problem solving. This technique encourages you to break a problem down into simpler subproblems, solve those subproblems, and then assemble the smaller solutions into the overall solutions. Functions are a way to directly encode the "smaller subproblem" solution. You'll see more about this as we work through this chapter.

5.1 WHAT IS A FUNCTION?

In mathematics, a function defines the relationship between values. Consider the function $f(x) \Rightarrow \sqrt{x}$. If you provide a particular value of x, for example, $x = 4$, the function will perform a calculation (here the square root operation) and return the associated value, for example, 2. Mathematicians term the variable x the *argument* to the function and say that the function *returns* the value 2.

It is possible for a function to have multiple arguments, for example a function that calculates multiplication requires two arguments: $f(x, y) \Rightarrow x * y$. However, a mathematical function returns only one object. Note that the returned object can be a *compound* object; an object with multiple elements. For example, when working with graph paper, each point is