Chapter 3
Simple Functions

STARTING OUT WITH

Python

First Edition

by
Tony Gaddis

3.1 Introduction to Functions

Concept:

A function is a group of statements that exists within a program for the purpose of performing a specific task.

3.1 Introduction to Functions

Divide and Conquer

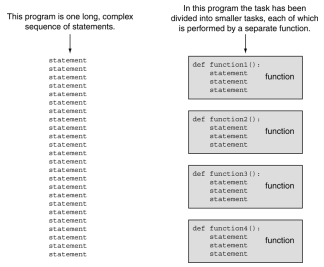


Figure 3-1 Using functions to divide and conquer a large task

3.1 Introduction to Functions

Benefits of Using Functions

- Simple Code
- Code Reuse
- Better Testing
- Faster Development
- Easier Facilitation of Teamwork

3.2 Defining and Calling a Function

Concept:

The code for a function is known as a function definition. To execute the function, you write a statement that calls it.

3.2 Defining and Calling a Function

Function Names

- Cannot use Python key words
- Cannot contain spaces
- First character must be one of the letters *a* through *z*, *A* through *Z*, or an underscore character(_)
- After the first character use letters *a* through *z*, *A* through *Z*, digits *0* through *9*, or underscore
- Uppercase and lowercase characters are distinct

3.2 Defining and Calling a Function

Defining and Calling a Function

Function header

Block

```
def function_name():
    statement
    statement
    etc.
```


3.2 Defining and Calling a Function

Defining and Calling a Function

Figure 3-2 The function definition and the function call

These statements cause the `message` function to be created.

```
# This program demonstrates a function.
# First, we define a function named message.
def message():
    print 'I am Arthur,'
    print 'King of the Britons.'

# Call the message function.
message()
```

This statement calls the `message` function, causing it to execute.

3.2 Defining and Calling a Function

Indentation in Python

Figure 3-7 All of the statements in a block are indented

The last indented line is the last line in the block.

```
def greeting():
    print 'Good morning!'
    print 'Today we will learn about functions.'
```

These statements are not in the block.

```
print 'I will call the greeting function.'
greeting()
```


3.3 Designing a Program to Use Functions

Concept:

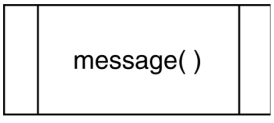
Programmers commonly use a technique known as top-down design to break down an algorithm into functions.

3.3 Designing a Program to Use Functions

Flowcharting a Program with Functions

Function Call

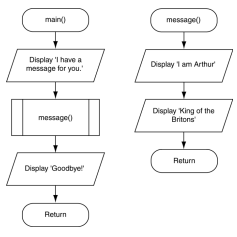
Figure 3-8 Function call symbol



3.3 Designing a Program to Use Functions

Flowcharting a Program with Functions

Figure 3-9 Flowchart for Program 3-2



3.3 Designing a Program to Use Functions

Top-Down Design

- The overall task of the program is broken down into a series of subtasks
- Each of the subtasks is examined to determine whether it can be further broken down into more subtasks.
- Once the subtasks are identified, they are written in code.

3.3 Designing a Program to Use Functions

Pausing Execution Until the User Presses Enter

```
raw_input('Press Enter to see Step 1.')
```


3.4 Local Variables

Concept:

A local variable is created inside a function and cannot be accessed by statements that are outside the function. Different functions can have local variables with the same names because the functions cannot see each others' local variables.

3.4 Local Variables

Scope and Local Variables

A local variable's scope is the function in which the variable is created

3.4 Local Variables

Program 3-5 (birds.py)

```
1  # This program demonstrates two functions that
2  # have local variables with the same name.
3
4  def main():
5      # Call the texas function.
6      texas()
7      # Call the california function.
8      california()
9
10 # Definition of the texas function. It creates
11 # a local variable named birds.
12 def texas():
13     birds = 5000
14     print 'texas has', birds, 'birds.'
15
16 # Definition of the california function. It also
17 # creates a local variable named birds.
18 def california():
19     birds = 8000
20     print 'california has', birds, 'birds.'
21
22 # Call the main function.
23 main()
```


3.5 Passing Arguments to Functions

Concept:

An argument is any piece of data that is passed into a function when the function is called. A parameter is a variable that receives an argument that is passed into a function.

3.5 Passing Arguments to Functions

Program 3-6 (pass_arg.py)

```
1  # This program demonstrates an argument being
2  # passed to a function.
3
4  def main():
5      value = 5
6      show_double(value)
7
8  # The show_double function accepts an argument
9  # and displays double its value.
10 def show_double(number):
11     result = number * 2
12     print result
13
14 # Call the main function.
15 main()
```


3.5 Passing Arguments to Functions

Figure 3-13 The value variable is passed as an argument

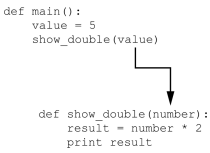
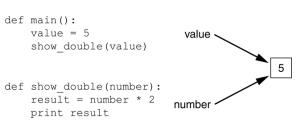


Figure 3-14 The value variable and the number parameter reference the same value



3.5 Passing Arguments to Functions

Parameter Variable Scope

- A parameter variable's scope is the function in which the parameter is used.
- Inside the function, the parameter variable can be accessed, but no statement outside the function can access it.

3.5 Passing Arguments to Functions

Passing Multiple Arguments

Program 3-8 (multiple_args.py)

```
1  # This program demonstrates a function that accepts
2  # two arguments.
3
4  def main():
5      print 'The sum of 12 and 45 is'
6      show_sum(12, 45)
7
8  # The show_sum function accepts two arguments
9  # and displays their sum.
10 def show_sum(num1, num2):
11     result = num1 + num2
12     print result
13
14 # Call the main function.
15 main()
```


3.5 Passing Arguments to Functions

Making Changes to Parameters

Program 3-10 (change_me.py)

```
1  # This program demonstrates what happens when you
2  # change the value of a parameter.
3
4  def main():
5      value = 99
6      print 'The value is', value
7      change_me(value)
8      print 'Back in main the value is', value
9
10 def change_me(arg):
11     print 'I am changing the value.'
12     arg = 0
13     print 'Now the value is', arg
14
15 # Call the main function.
16 main()
```


3.5 Passing Arguments to Functions

Keyword Arguments

Specifies which parameter the argument should be passed into. Its position in the function call does not matter.

Program 3-11 (keyword_args.py)

```
1  # This program demonstrates keyword arguments.
2
3  def main():
4      # Show the amount of simple interest, using 0.01 as
5      # interest rate per period, 10 as the number of periods,
6      # and $10,000 as the principal.
7      show_interest(rate=0.01, periods=10, principal=10000.0)
8
9  # The show_interest function displays the amount of
10 # simple interest for a given principal, interest rate
11 # per period, and number of periods.
12
13 def show_interest(principal, rate, periods):
14     interest = principal * rate * periods
15     print 'The simple interest will be $%.2f.' % interest
16
17 # Call the main function.
18 main()
```

3.6 Global Variables and Global Constants

Concept:

A global variable is accessible to all the functions in a program file.

3.6 Global Variables and Global Constants

Program 3-13
(global1.py)

```
1  # Create a global variable.
2  my_value = 10
3
4  # The show_value function prints
5  # the value of the global variable.
6  def show_value():
7      print my_value
8
9  # Call the show_value function.
10 show_value()
```


3.6 Global Variables and Global Constants

Restrict the use of global variables for the following reasons:

- Global variables make debugging difficult
- Functions that use global variables are usually dependent on those variables.
- Global variables make a program hard to understand.

3.6 Global Variables and Global Constants

Global Constants

It is a global name that references a value that cannot be changed during the program's execution.

Chapter 3
Simple Functions

QUESTIONS

?