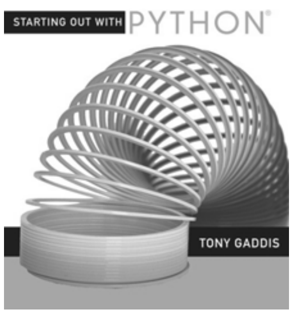
Topic 6: Functions

What's a function?

How can we use functions to write better software?

1

Recommended Readings



- Chapter 3
- Chapter 6

2

What is a Function?

- What is a function?
  – A named set of statements
  – Perform some task
- Functions:
  – May take parameters
  – May return values
- What functions have you already used?

3

Motivation

- Ideally, a function should
  – perform a clearly defined specific purpose
  – hide details from the caller
  – be sufficiently small to be easily understood
  – be well documented

4

Defining a Function

- Creates a function for later use
  - The function does not execute until it is called
  - Function may be called many times (from different places) after it has been defined
- General form:
  - `def functionName(parameters):`
    `statement(s)`

5

Example

- Create a function that draws a music note
  - Head will be a solid oval, 20 pixels wide and 10 pixels high
  - Stem will be 50 pixels tall on the right side

6

Calling Our Function

- A function does not execute when it is defined
  - It must be called
- Execution for the entire program begins at the first statement outside of a function

7

Example

- What's the problem with our function?
- How do we fix it?

8

Parameters

- Allow us to provide data to a function
  - Data is placed in brackets after the function name when the function is called
  - Parameter variables appear in brackets after the function name in the function definition
  - Values appear in parameter variables when the function executes
  - Parameters are positional

9

Terminology

- Actual Parameter
  - The value placed in brackets after the function name when the function is called
- Formal Parameter
  - The name of the parameter variable in the function that is called

10

Example

- Extend our note drawing function so that it takes two parameters that control the position of the note

11

Named Parameters

- Positional parameters assign values to parameter variables in the order that they occur
- Named parameters allow us to assign values in any order
  - Allow for optional parameters / default values for some parameters
  - May still be used in a positional manner

12

Example

- Extend the note drawing function so that it takes additional parameters that specify the color of the note

13

Default Parameter Values

- Python permits default values for parameters
  - If the function call does not supply a value then the default is used
  - If the call includes a value for that parameter then the default value is overridden

14

Functions can Call Functions

- Create a second function for drawing a note
  - Head will be a solid oval, 20 pixels wide and 10 pixels high
  - Stem will be 50 pixels tall on the left side
  - Flag will be a cubic curve

15

Functions can Call Functions

16

Variables & Functions

- Variables can be defined inside functions
  - A variable defined inside of a function can only be used inside that function
  - Behaves just like the variables we have used previously

17

Variables & Functions

- Variables can be defined outside of functions
  - Referred to as global variables
  - Can be read anywhere in the program after it is assigned a value
  - All of the constants we have created are global variables that we choose not to change
  - Use of global variables (other than as constants) is strongly discouraged

18

Variables & Functions

- Changing global variables
  - By default, an assignment statement inside of a function creates a new variable within that function
    - Even if a global variable with that name already exists
  - Want to change a global variable?
    - Include a global statement at the beginning of the function

19

Example

20

Scope

- Scope determines the portion of a program where a name can be used
  - Impacts functions, variables, …
- Functions
  - Functions can't be called before they have been defined
  - Functions in other modules cannot be used until after the import statement for that module

21

Scope

- Variables
  - Cannot be read before they are given a value
  - Can be used from the point where they are first assigned a value until the end of the function
  - Variables created inside a function are destroyed when the function exits

22

Parameter Variables

- Parameter variables hold values passed to a function from the calling scope
  - Parameter variables are normally read
  - It is also possible to store a new value into a parameter variable (don't usually do this!)
    - Value of the variable will change in the called function
    - For the types we have used so far, the value will **not** change in the main program

23

Another Example

- Create a function called readInteger
  - requires two parameters
    - The lowest permitted value
    - The highest permitted value
  - returns one result
    - The value entered by the user between lowest (inclusive) and highest (inclusive)
  - readInteger will ensure that the value returned is within the specified range
  - Use this function to improve the number game

24

Another Example

25

Why Functions are Useful

- Facilitate Code Reuse
  - Write once, use many times
- Reduce Complexity
  - Low level details are hidden
  - Programmer can concentrate on higher level problems
- Ease Maintenance
  - Bugs only need to be corrected once
  - Functions can be tested separately

26

Comments

- Every function should begin with a comment
  - Describe the action taken by the function
  - Describe the parameters that need to be provided
  - Describe the value returned by the function

27

Preconditions / Postconditions

- Function comments may also describe
  - Preconditions:
    - Conditions that must be true before the function executes
    - If any precondition is not met, the function may not behave correctly
  - Postconditions:
    - Conditions that are guaranteed to be true after the function executes
    - If the function doesn't make a post-condition true then the function contains a bug that must be fixed

28

Returning Multiple Values

- What if we need to return more than one value from a function?
  - Comma separated list of values in return statement
  - Comma separated list of variables to the left of the equals sign

Example

Flash card add and multiply practice:

- 10 random questions, add or multiply 2 integers between 1 and 10

Testing

- Test each function you write individually
  - Errors are easier to find
    - Generally only need to look inside the function being tested
  - Only use the function in the rest of your program once you have tested it thoroughly

Design

- How do functions relate to top down design?
  - Use top down design to break the problem into smaller pieces
  - Each smaller piece is a good candidate for a function
  - Look at each function
    - Is it too big?
    - Does it contain repeated code?
    - Should it call other functions?

Wrapping Up

- Functions
  - A named group of statements that perform a task
  - Allow us to break our program into separate units that each have a specific purpose
  - Ease program creation and debugging

33

Where Are We Going?

- Now that we can write larger programs we want to be able to manage more data
  - How do we read and write values in files?
  - How can we work with many values at the same time in a reasonable way?

34