Chapter 6: Functions and Modular Programming

Dona Hertel

CS161 – Winter 2013

Portland State University

What is a Function

- A block of code set aside (either in the same file or another file) which can be run from somewhere else in the code.

```
def print_message(message, num):
    for i in range(num):
        print(message)
    print(“The end”)
.....
.....
print_message(“Hello”,5)
```

What is a Function

- There are two types of statements needed by your code in order to use a function.
  - **function definition statement** followed by the body of the function:

```
def print_message(message, num):
    for i in range(num):
        print(message)
    print("The end")
```

  - **Function call statement:**

```
print_message("Hello World",5)
```

Why Do You Want to Create Your Own Functions?

- You can code without using functions.
- However, the code will get very complex and repetitive. Breaking up the code into functions:
  - Makes the code more understandable.
  - Eliminates repetition.
  - Makes code re-usable.

Parameters (input)

- A **parameter** is a variable used inside the function to pass in values needed during the running of the function. The parameters are indicated by a comma separated list inside parenthesis () after the function name.

```
def print_message(message, num):
    for i in range(num):
        print(message)
    print(“The end”)
```

NOTE: If the function has no parameters, a pair of empty parenthesis () are still needed.

Arguments (input)

- To set the parameter's values during a function call, a corresponding list of **arguments** are provided.

  ```
  def print_message(message, num):
  ....
  ....
  contmessage = “Hello”
  max_times = 5
  print_message(contmessage, max_times)
  ```

- NOTE: The parameters and arguments don't need to have the same name.

Parameters (Positional)

- For parameters with just a name or value, what argument supplies a value to what parameter is based on its ***position*** in the lists.

print_message("Hello",5)

  - message will equal "Hello"
  - num will equal 5

m = "Goodbye"

i = 8

print_message(m, i)

  - message will equal "Goodbye"
  - num will equal 8

Positional Parameter Example

- (See moodle site for a zip file of code examples)
  - positional_params.py

Return Statement (output)

- In order to get back any values from a function, a **return statement** needs to be used. A return statement can return one value or a list of values (rare in other languages).

```
def  get_input(message, valid_responses):
       ans = “”
       i=0
       while ans not in valid_responses:
            ans = input(message)
            i = i + 1
        return ans, i
```

return statement (cont)

- To receive the values when the function is called, just use an assignment with the function call on the right and the return values on the left (in the same order as in the return statement).

```
answer, count = get_input(message, valid_responses)
```

Code Examples

- (See moodle site for a zip file of code examples):
    - no_function.py
    - one_function.py
    - two_functions.py
    - full_functions.py

Structure of a python file

- Comment section:
  - year, name, short description
- import statements
- Global variables
- function definitions
- main part of code

Keyword Arguments

- Sometimes you would like to tell the function which parameter gets what value.
- You can by specifying the name of the parameter and the value using **keyword arguments**.

```
print_function(message=”HELLO”, num=5)
```

Keyword Arguments Example

- (See moodle site for a zip file of code examples):
  - keyword_args.py

Default Parameters

- You can also give a parameter a 'default' value. If the function call then doesn't have to provide a value for this parameter if it only wants to use the default value.
  - **def print_function(message, num=5):**

    .

    .

    .
  - **print_function("Hello")**

    will assign "Hello" to message and 5 to num.

Default Parameter Example

- (See moodle site for a zip file of code examples):
  - default_params.py

Using function in other files

- An import statement allows the use of functions from other files. Import statements are usually put at the top of the file just under the top comments.

```
# copyright 2013 by Dona Hertel
import sys
from myModule import doSomething
....
....
if (something_went_bad):
    sys.exit(0)
....
....
doSomething(here)
```