Chapter 6:
Functions
Objectives

In this chapter, you will learn how to:

- Create your own functions
- Accept values into your functions through parameters
- Return information from your functions through return values
- Work with Global Variables and Constants
def classic_burger():

    Type Program
    Steps to be executed
    Remember to indent steps

    return <result>

    This is the Value returned by your Program: “instruction”
    Omit entire line if no value is returned
Creating Functions

Documenting a Function ~ instruction

- `def tv_remote()`:

  """Remote Control Instructions""

  Program Steps to be executed

  Documentation String (Optional)

  return <result>
Calling your Function

- classic_burger()
Functions
Creating Parameters but no Return Values

Case I: Input Parameters but no Return Value

```python
def display(message):
    print message
```

- Function name
- One parameter
- Input parameter
Functions
Creating Parameters & Return Values

- **Case II:** Returns a Value, but does not specify input parameter

```python
def give_me_five()
    five = 5
    return five
```

- Function name
- Return value
- No parameter
Functions
Creating Parameters & Return Values

Case III: Specifies input parameter and Returns a Value,

```python
def ask_yes_no(question):
    """Ask a yes or no question""
    response = None
    while response not in ("y", "n"):
        response = raw_input(question).lower()
    return response
```

Return value
# My Main Program

```python
display ("Here is a message for you")
number = give_me_five()
print "Here is what I got from give_me_five:", number
answer = ask_yes_no("\nPlease enter ‘y’ or ‘n’: ")
print "Thanks for entering:", answer
raw_input ("\n\nPress the enter key to exit."")
```

Calling (Referencing): display, give_me_five, ask_yes_no

Catching return values
Functions

Receiving Information Thru’ Parameters

```python
def display(message):
    message = "here is a message for you"

def ask_yes_no(question):
    question = "Please enter ‘y’ or ‘n’: 
```

Function
Returning Information to Calling program

```python
def give_me_five( ):
    return five  # returns the value of variable five
```

```python
def ask_yes_no(question):
    return response  # returns the value of variable response
```
Suppose Function Definition

```python
def birthday1(name, age):
```

Calling Program:

```python
birthday1(   "Jackson", 1)
```

1. 1st Parameter is assigned the value of the first argument
2. 2nd Parameter is assigned the value of the second argument

- Parameter *name* is assigned the value "Jackson"
- Parameter *age* is assigned the value 1
Function

Using Positional Parameters and Positional Arguments

- Suppose Function Definition
  - `def birthday1(name, age):
  - Calling Program:
    - `birthday1( 1, “Jackson”)

1. 1st Parameter is assigned the value of the first argument
2. 2nd Parameter is assigned the value of the second argument

- Parameter `name` is assigned the value `1`
- Parameter `age` is assigned the value “Jackson”
Function

Using Keyword Arguments

- Suppose Function Definition
  ```python
def birthday1(name, age):
```
- Calling Program:
  ```python
birthday1(name = "Jackson", age = 1) or
```
  ```python
birthday1(age = 1, name = "Jackson")
```

• Parameter `name` is assigned the value "Jackson"
• Parameter `age` is assigned the value 1

Keyword arguments lets you pass values in any order
How do you define a Function with Default Values?

```python
def birthday2(name = "Jackson", age = 1):
```

**NOTE:** You MUST SPECIFY ALL OR NO DEFAULT VALUES
Why specify Default Values in a Function Header?

- The default value is used if calling program omits the value of one or more argument
Function

Default Values

def birthday2(name = “Jackson”, age = 1):
    print “Happy birthday,”, name, “!”, “I hear you’re”, age, “today.

Calling Program:
birthday2( ) → Happy birthday Jackson! I hear you’re 1 today
birthday2(name = “Katherine”) → Happy birthday Katherine! I hear you’re 1 today
birthday2(age = 12) → Happy birthday Jackson! I hear you’re 12 today
birthday2(name = “Katherine”, age = 12) → Happy birthday Katherine! I hear you’re 12 today
birthday2( “Katherine”, 12) → Happy birthday Katherine! I hear you’re 12 today
Functions

Global Variables

Main Program

```python
def func1()
    variable1 = 1
```

```python
def func2()
    variable2 = 2
```

variable0 = 0

Local Variables

Global Variable
Functions

Global & Local Variables

- Local Variables
  - Exits only within the Function
  - Cannot be accessed by other Function or the Main Program

- Global Variables
  - Can be **read** by every Function linked to the main program
  - Special Declaration required to change the value of a global variable within a function
How do you change the value of a global variable within a function?

You must first declare the global variable within the function:

```python
def func1():
    global variable0
    variable0 = -100
```

---

**Functions**

**Global Variables**

- How do you change the value of a global variable within a function?
- You must first declare the global variable within the function:

```python
def func1():
    global variable0
    variable0 = -100
```
def read_global():
    print "From inside the local scope of read_global(), value is:\", value

def shadow_global():
    value = -10
    print "From inside the local scope of shadow_global(), value is:\", value

def change_global():
    global value
    value = -10
    print "From inside the local scope of change_global(), value is:\", value

# main
value = 10
print "In the global scope, value has been set to:\", value, "\n"

read_global()
print "Back in the global scope, value is still:\", value, "\n"

shadow_global()
print "Back in the global scope, value is still:\", value, "\n"

change_global()
print "Back in the global scope, value has now changed to:\", value

raw_input("\n\nPress the enter key to exit.")
def read_global():
    print "From inside the local scope of read_global(), value is:", value

def shadow_global():
    value = -10
    print "From inside the local scope of shadow_global(), value is:", value

def change_global():
    global value
    value = -10
    print "From inside the local scope of change_global(), value is:", value

def main():
    print "In the global scope, value has been set to:", value, "\n"
    read_global()
    print "Back in the global scope, value is still:\", value, "\n"
    shadow_global()
    print "Back in the global scope, value is still:\", value, "\n"
    change_global()
    print "Back in the global scope, value has now changed to:\", value
    return

#Start execution of your program here
value = 10
main()
raw_input("\n\nPress the enter key to exit.\n")