Chapter 5
Functions—QuickStart
What is a Function?
Functions

• From mathematics we know that functions perform some operation and return one value.
• They “encapsulate” the performance of some particular operation, so it can be used by others (for example, the sqrt() function).
Why Have Them?

- Support divide-and-conquer strategy
- Abstraction of an operation
- Reuse: once written, use again
- Sharing: if tested, others can use
- Security: if well tested, then secure for reuse
- Simplify code: more readable
Mathematical Notation

• Consider a function which converts temperatures in Celsius to temperatures in Fahrenheit:
  – Formula: \( f = c \times 1.8 + 32.0 \)
  – Functional notation: \( f = \text{celsius2Fahrenheit}(c) \)
    where
    \( \text{celsius2Fahrenheit}(f) = c \times 1.8 + 32.0 \)
Function Invocation

- Math: \( f = \text{celsius2Fahrenheit}(c) \)
- Python, the invocation is much the same
  \[ f = \text{celsius2Fahrenheit}(c) \]

Terminology: argument “c”
Function Definition

• Math: celsius2Fahrenheit(c) = c*1.8 + 32.0

• Python
def celsius2Fahrenheit(c):
    return c*1.8 + 32.0

• Terminology: parameter “c”
Return Statement

• The return statement indicates the value that is returned by the function.
• The statement is optional (the function can return nothing). If no return, the function is often called a procedure.
FIGURE 5.1 Function parts.
Code Listing 5.1
Temp Convert
# Temperature conversion

def celsius2fahrenheit(celsius):
    """ Convert Celsius to Fahrenheit."""
    return celsius*1.8 + 32
Triple Quoted String in Function

- A triple quoted string just after the def is called a docstring
- docstring is documentation of the function’s purpose, to be used by other tools to tell the user what the function is used for.
Operation

1. Call copies argument c to parameter temp

2. Control transfers to function “celsius2Fahrenheit”

```python
f = celsius2Fahrenheit(c)

def celsius2Fahrenheit(temp):
    return temp * 1.8 + 32.0
```
3. Expression in `celsius2Fahrenheit` is evaluated

```python
def celsius2Fahrenheit(temp):
    return temp * 1.8 + 32.0
```

4. Value of expression is returned to the invoker

```python
f = celsius2Fahrenheit(c)
```
FIGURE 5.2 Function flow of control.
Implement \texttt{len()} 

- How might we count the number of characters in a string without using \texttt{len()}?
def length(s):
    """Return the length of s."""
    count = 0
    for c in s:
        count += 1
    return count
Count lowercase characters

• How might we count the number of lowercase characters in a string?
Count lowercase characters

- import string
- use string.lowercase, string of lowercase
  - 'abcdefghijklmnopqrstuvwxyz'
- check if each letter is a member (using the in operator) of string.lowercase
import string

def lowercaseCount(s):
    """Return the lowercase count in s."""
    count = 0
    for c in s:
        if c in string.lowercase:
            count += 1
    return count
Example: Word Puzzle

• Find an English language word that has the vowels ‘a’, ‘e’, ‘i’, ‘o’, and ‘u’ in sequence
Example: Word Puzzle

• Clean the text (i.e., covert to lowercase and remove whitespace and punctuation characters.

• Create a string containing the sequence of vowels in the word

• Check to see if that string contains ‘aeiou’
def cleanWord(word):
    """Return word in lower case stripped of whitespace and punctuation characters"""
    word = word.strip().lower()
    badChars = string.whitespace + string.punctuation
    for char in badChars:
        word = word.replace(char, '')
    return word
def getVowelsInWord(word):
    """ Return vowels in string, include repeats"""
    vowelStr = 'aeiou'
    vowelsInWord = ''
    for char in word:
        if char in vowelStr:
            vowelsInWord += char
    return vowelsInWord
Yet another function

• Let’s add a function which determines if a word contains the vowels ‘aeiou’ in order:
def hasVowelsInOrder(word):
    """ Return true if the word contains vowels in order, false otherwise"""
    vowels = getVowelsInWord(cleanWord(word))
    index = vowels.find('aeiou')
    return index != -1
Now automate the process

• Can read a file using the open() function:
  – data = open("filename.txt")

• Then we can do something like:
  – Then we can search the words in the file using:
    for line in data:
      print line

• Let’s find and download a dictionary file.
data = open("dictionary.txt")
for line in data:
    if hasVowelsInOrder(line):
        print(line)
Example: Palindromes

• Remember palindromes?
import string

inputString = raw_input("Enter input: ")
lowerString = inputString.lower()
removeCharacters = string.whitespace + string.punctuation
for char in removeCharacters:
    lowerString = lowerString.replace(char, "")
if lowerString == lowerString[::-1]:
    print "PALINDROME! "
else:
    print "NOT A PALINDROME!"
print lowerString + " " + lowerString[::-1]
Example: Palindromes

- How might we simplify this code by defining functions?
Example: Palindromes

- Define two helper functions:
  - clearText(text)
    - Returns a lowercase version of the text stripped of whitespace and punctuation characters.
  - reverseText(text)
    - Returns a reverse version of the text.
- Makes defining isPalindrome(text) easy!
def cleanText(text):
    """Return text in lower case stripped of whitespace and punctuation characters"""
    text = text.strip().lower()
    badChars = string.whitespace + string.punctuation
    for char in badChars:
        text = text.replace(char, '')
    return text
def reverseText(text):
    """Return text in reverse order"""
    return text[::-1]
def isPalindrome(text):
    """Return True if the text is a palindrome, False otherwise"""
    text = cleanText(text)
    return text == reverseText(text)
How to Write a Function

- **Does one thing.** If it does too many things, it should be broken down into multiple functions (refactored).
- **Readable.** How often should we say this? If you write it, it should be readable.
- **Reusable.** If it does one thing well, then when a similar situation (in another program) occurs, use it there as well.
More on Functions

• **Complete.** A function should check for all the cases where it might be invoked. Check for potential errors.

• **Not too long.** Kind of synonymous with “does one thing”. Use it as a measure of doing too much.
Procedures

• Functions that have no return statements are often called procedures.
• Procedures are used to perform some duty (print output, store a file, etc.)
• Remember, return is not required.
Multiple Returns in a Function

- A function can have multiple return statements.
- Remember, the first return statement executed ends the function.
- Multiple returns can be confusing to the reader and should be used judiciously.
Example: Classify a Number

• Write a function which returns “positive” if the number is positive, “negative” if the number is negative, or “zero” if the number is zero.
def classifyNumber(number):
    """Return “positive” if the number is positive, “negative” if the number is negative, “zero” if the number is zero”""
    if number > 0:
        return "positive"
    elif number < 0:
        return "negative"
    else:
        return "zero"
Example: Palindromes (cont)

• If text has less than 2 characters, it must be a palindrome.
  – Modify isPalindrome() accordingly.
def isPalindrome(text):
    """Return True if the text is a palindrome, False otherwise"""
    if len(text) < 2:
        return True
    text = cleanText(text)
    return text == reverseText(text)