Introduction to Functions

Intro to Computer Science

CS1510

Dr. Sarah Diesburg

Review

- So far, most of our programs have retrieved data from the keyboard and written data to the screen
  - Data must be entered on every program run
  - Programs have no way to write permanent output
- Text files provide convenient input/output storage
  - e.g. programs can read configuration data or input files to process, and can write output to files

Question #1

- A program is designed to retrieve some data from a file, process it, and output the revised data to another file. Which of the following functions/methods will not be called in the program?
  - A. open
  - B. A loop or method for reading (e.g. read)
  - C. write
  - D. close
  - E. All should be called

Review

- Several methods for reading text from files:
  - readline: reads and returns next line; returns empty string at end-of-file
  - read: reads the entire file into one string
  - readlines: reads the entire file into a list of strings
- All of these leave a trailing '\n' character at the end of each line.

Review

- A file is a sequence of lines. Can be read with a for-loop

```
f = open('data.txt',"r")
for line in f:
    print(line.strip())
```

- …or using a while-loop:

```
f = open('data.txt',"r")
line = f.readline()
while line:
    print(line.strip())
    line = f.readline()
```

Question #2 – What is the last thing printed?

data.txt

```
Reading Assignments
#Each line lists the reading
#assignment for that date
Sep, 17, Section 1.1-1.3
Sep, 19, Section 1.4-1.8
Sep, 21, Section 2.1-2.4
```

program

```
line = f.readline()
line = f.readline()
while line.startswith('#'):
    line = f.readline()
print(f.readline())
```

Question #3 – What is the last thing printed?

data.txt

```
Reading Assignments
#Each line lists the reading
#assignment for that date
Sep, 17, Section 1.1-1.3
Sep, 19, Section 1.4-1.8
Sep, 21, Section 2.1-2.4
```

program

```
line = f.readline()
line = f.readline()
while line.startswith('#'):
    line = f.readline()
print( line )
```

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 len() function).

Why Have Them?

- 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
- Support divide-and-conquer strategy

Mathematical Notation

- Consider a function which converts temperatures in Celsius to temperatures in Fahrenheit:
  - Formula: F = C * 1.8 + 32.0
  - Functional notation: F = celsisus2Fahrenheit(C) where

    celsius2Fahrenheit(C) = C*1.8 + 32.0

Python Invocation

- Math: F= celsius2Fahrenheit(C)
- Python, the invocation is much the same

  F = celsius2Fahrenheit(C)

  Terminology: argument “C”

Function Definition

- Math: g(C) = C*1.8 + 32.0
- Python

```
def celsius2Fahrenheit (C):
 return C*1.8 + 32.0
```

- Terminology: parameter “C”

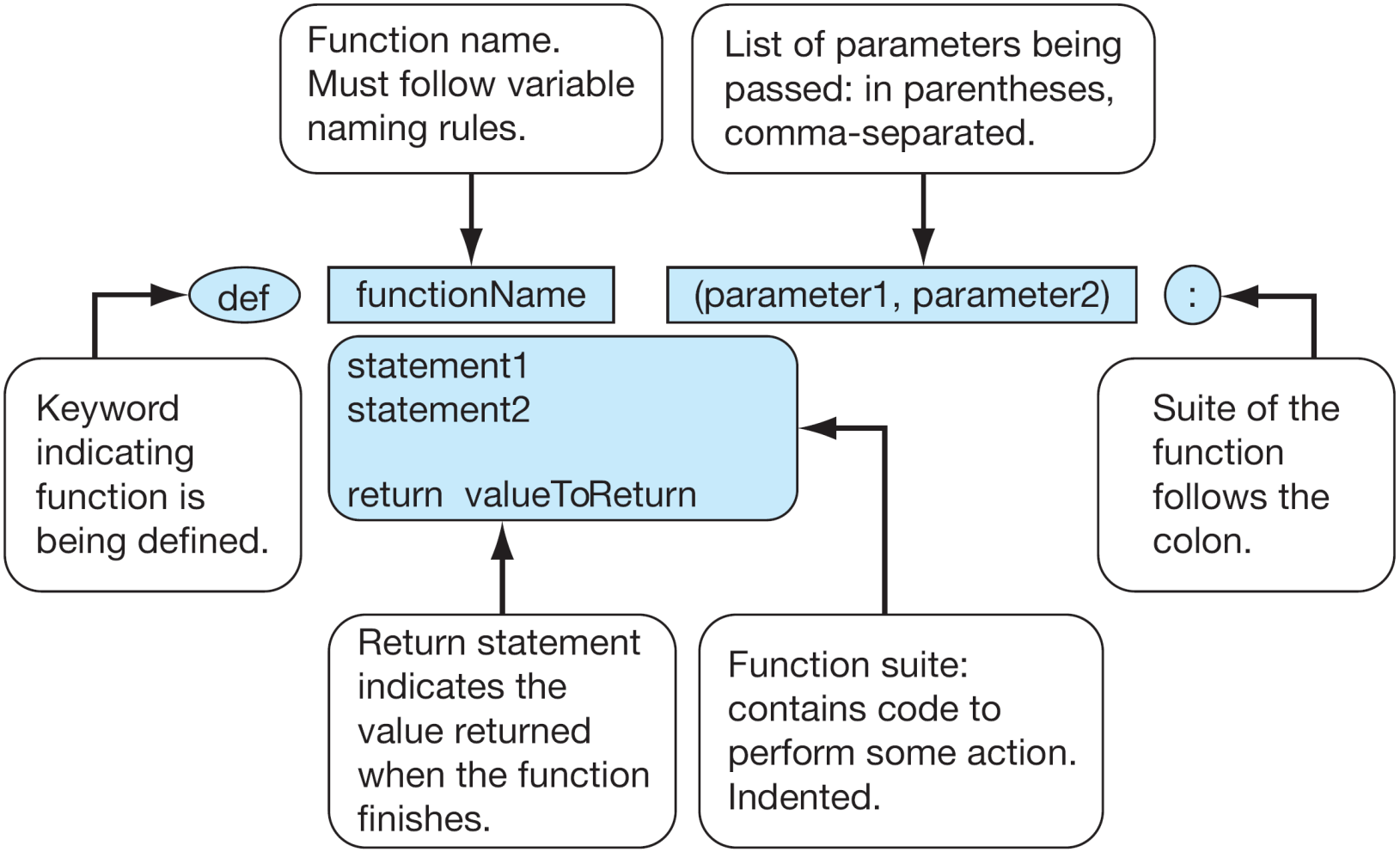


FIGURE 5.1 Function parts.

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.

Code Listing 6.1

- Temp Convert

Code Listing 6.1

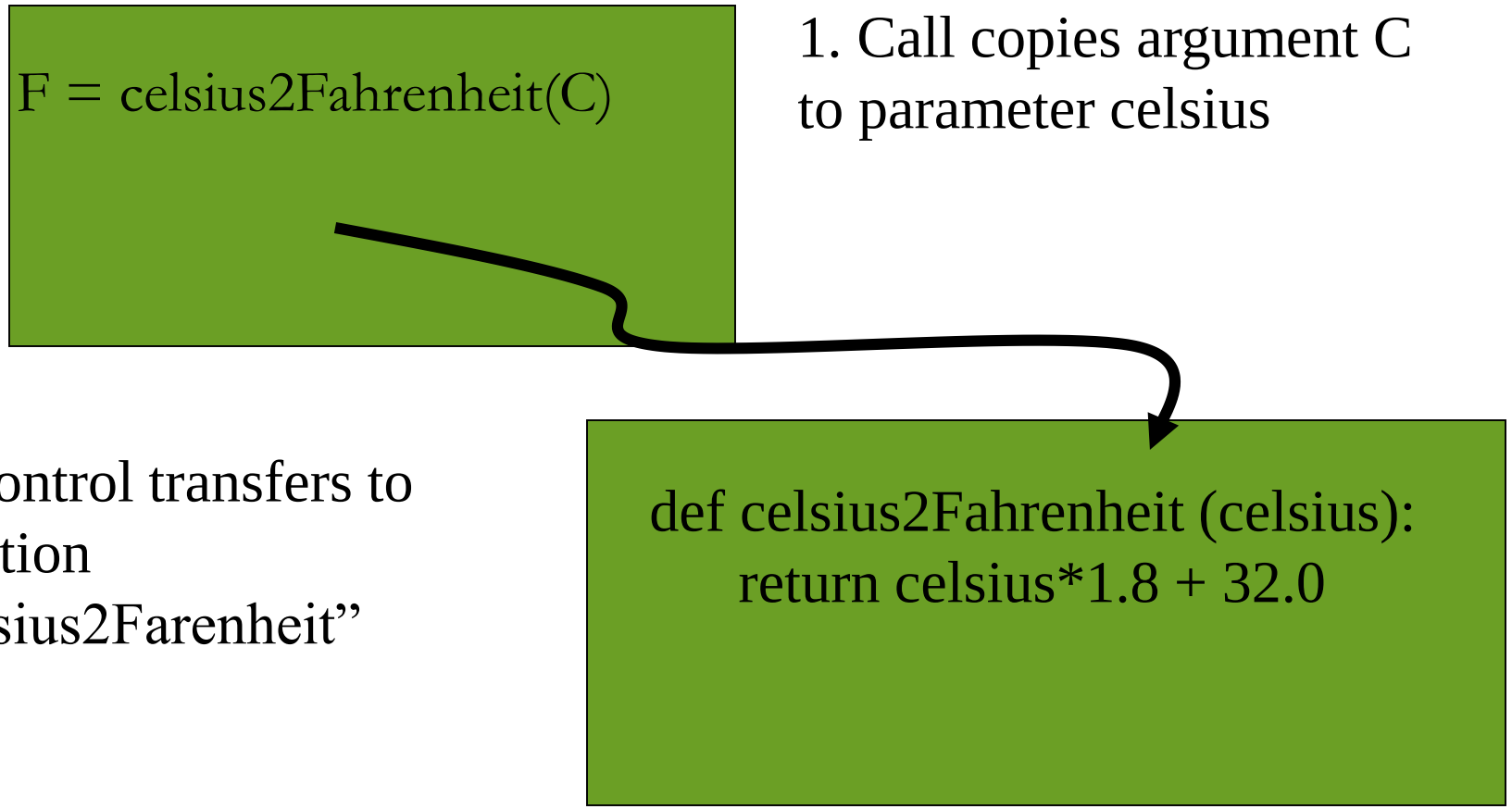
```
# Temperature conversion

def celsius2fahrenheit(celsius):
    """ Convert Celsius to Fahrenheit."""
    return celsius*1.8 + 32
```

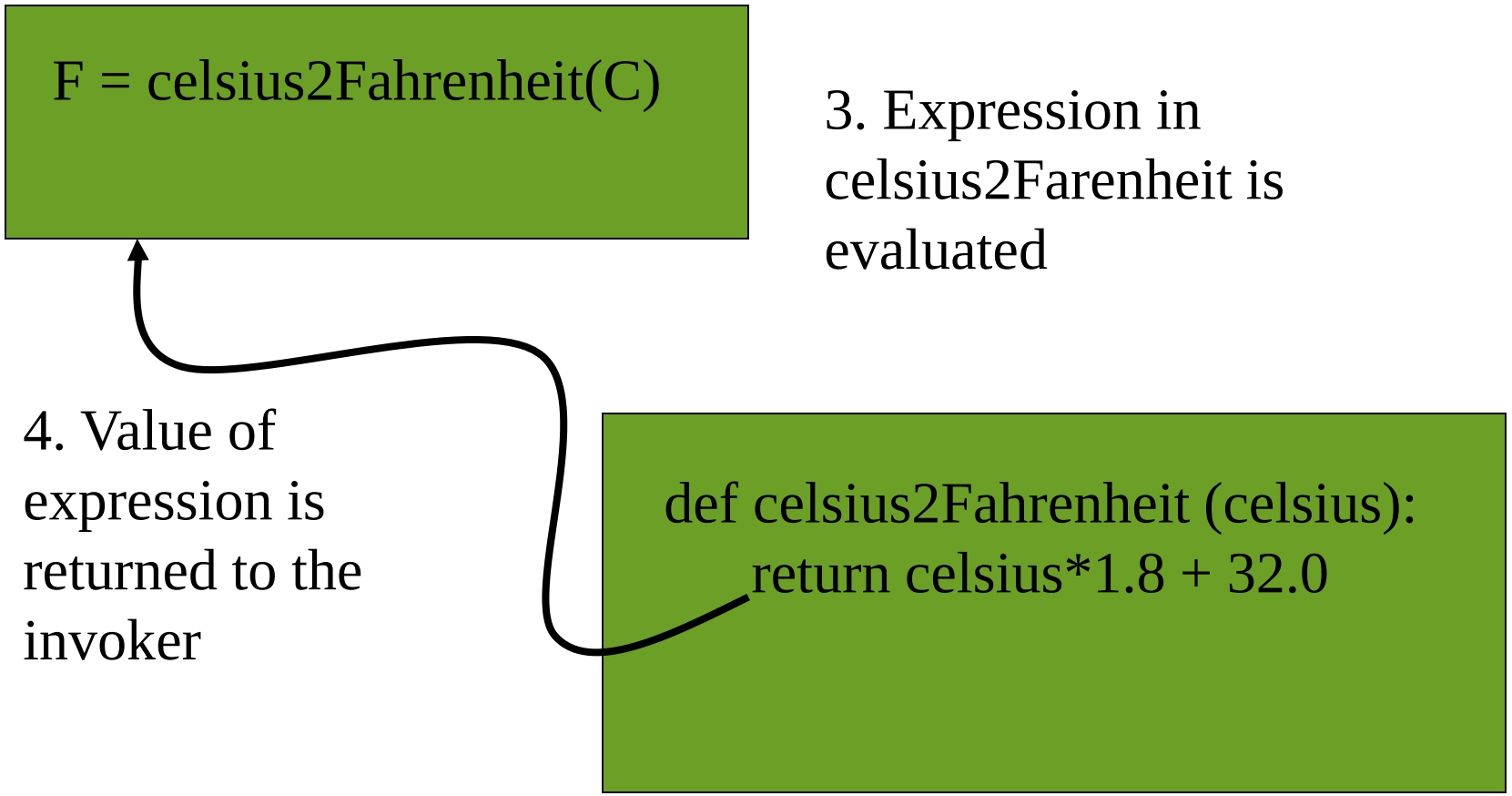
Triple Quoted String in Function

- 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


F = celsius2Fahrenheit(C)
1. Call copies argument C to parameter celsius
2. Control transfers to function "celsius2Farenheit"
def celsius2Fahrenheit (celsius):
return celsius*1.8 + 32.0

Operation (con't)



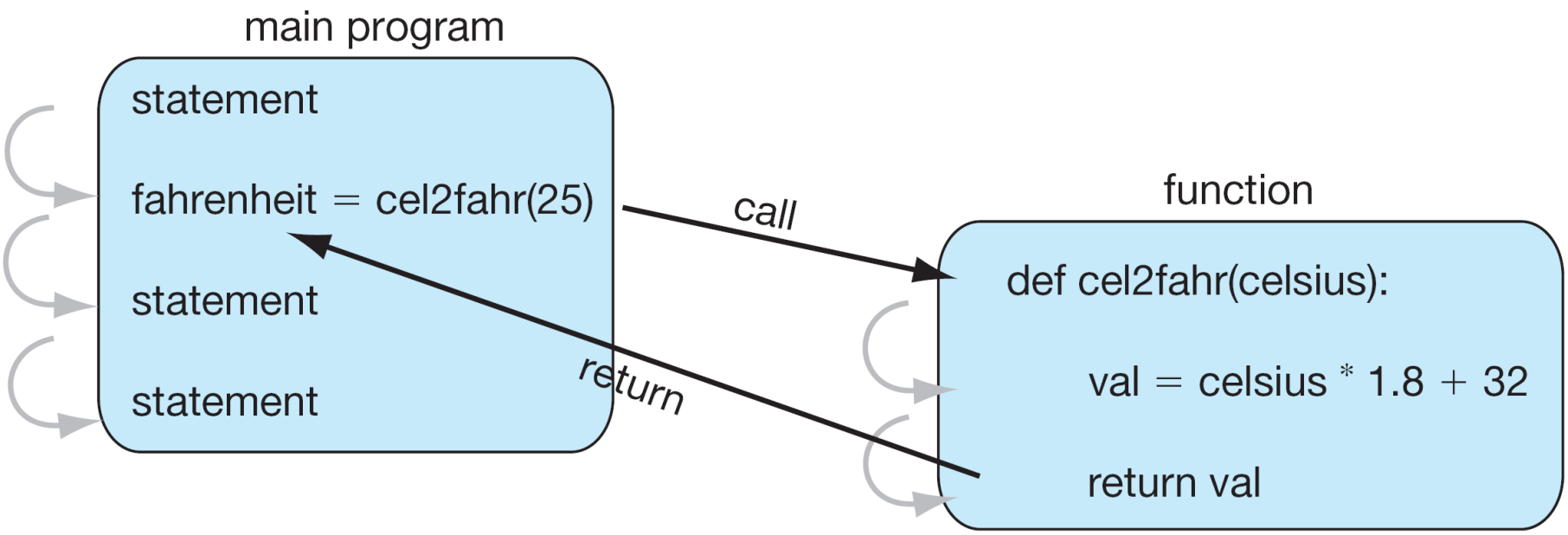


FIGURE 5.2 Function flow of control.

Code Listing 6.3

- Implement len

Code Listing 6.3

```
def length(S):
    """Return the length of S."""
    count = 0
    for s in S:
        count += 1
    return count
```

Code Listing 6.4

- Count Letters in String

Check Membership in lowercase

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

Code Listing 6.4

```
import string

def letterCount(S):
    """Return the count of letters in S."""
    count = 0
    for s in S:
        if s.lower() in string.ascii_lowercase:
            count += 1
    return count
```

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.