Functions

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

Two Parts to a Function

- ***Definition*** – creates the function
- ***Invocation*** – is the application of a function within a program

- A function must be ***defined*** before it is invoked

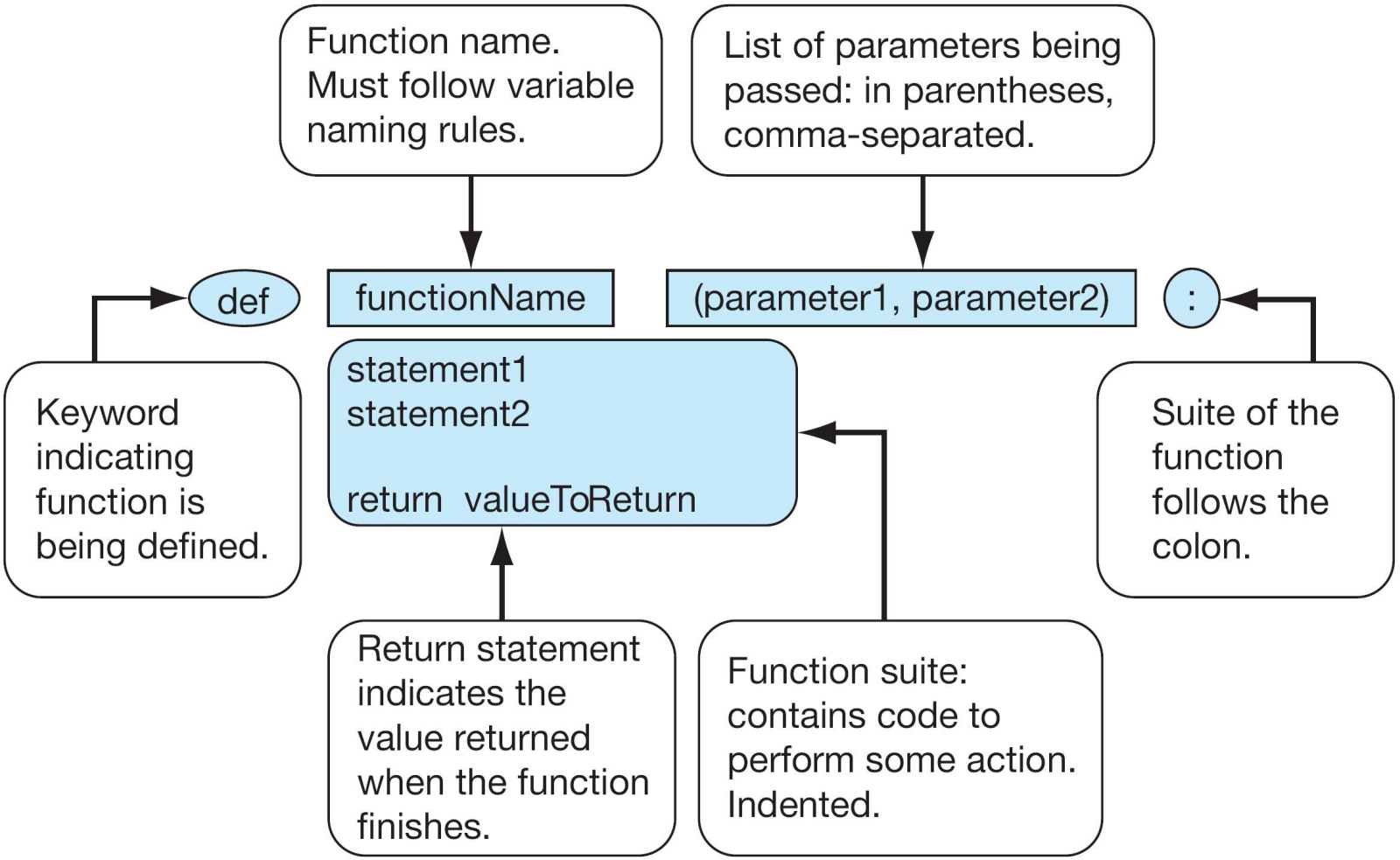
Function Definition

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

- Python:

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

Function Definition



Triple quoted string in function definition

```
def celsius2Fahrenheit (C):
    """ Convert Celsius to Fahrenheit."""
        return C*1.8 + 32.0
```

- 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.

Python Invocation

- Math: F= celsius2Fahrenheit(C)

- Python, the invocation is much the same

  F = celsius2Fahrenheit(C)

Function Invocation Example

- In your program (after the function definition), we can invoke/call the function fahrenheit2Celsius by:

```
originalTemp=90
convertedTemp = fahrenheit2Celsius(originalTemp)
print(originalTemp,”in Celsius is”,convertedTemp)
```

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.

What exactly is return doing?

- When python comes to a function inside your code…

convertedTemp = fahrenheit2Celsius(originalTemp)

- …it runs the function, then substitutes the ***return*** value where the function stood

convertedTemp = 32.22222222222222

Returning None

- None is a special value in Python that represents nothing
  - The first letter of None must be capitalized – it will turn orange
- Use it when you have nothing to return
  - Like if one of the parameters was invalid
- (Take a look at the fahrenheit2Celsius function again…)

Multiple Returns in a Function

- A function can have multiple return statements.
- Remember, the first return statement executed ends the function.

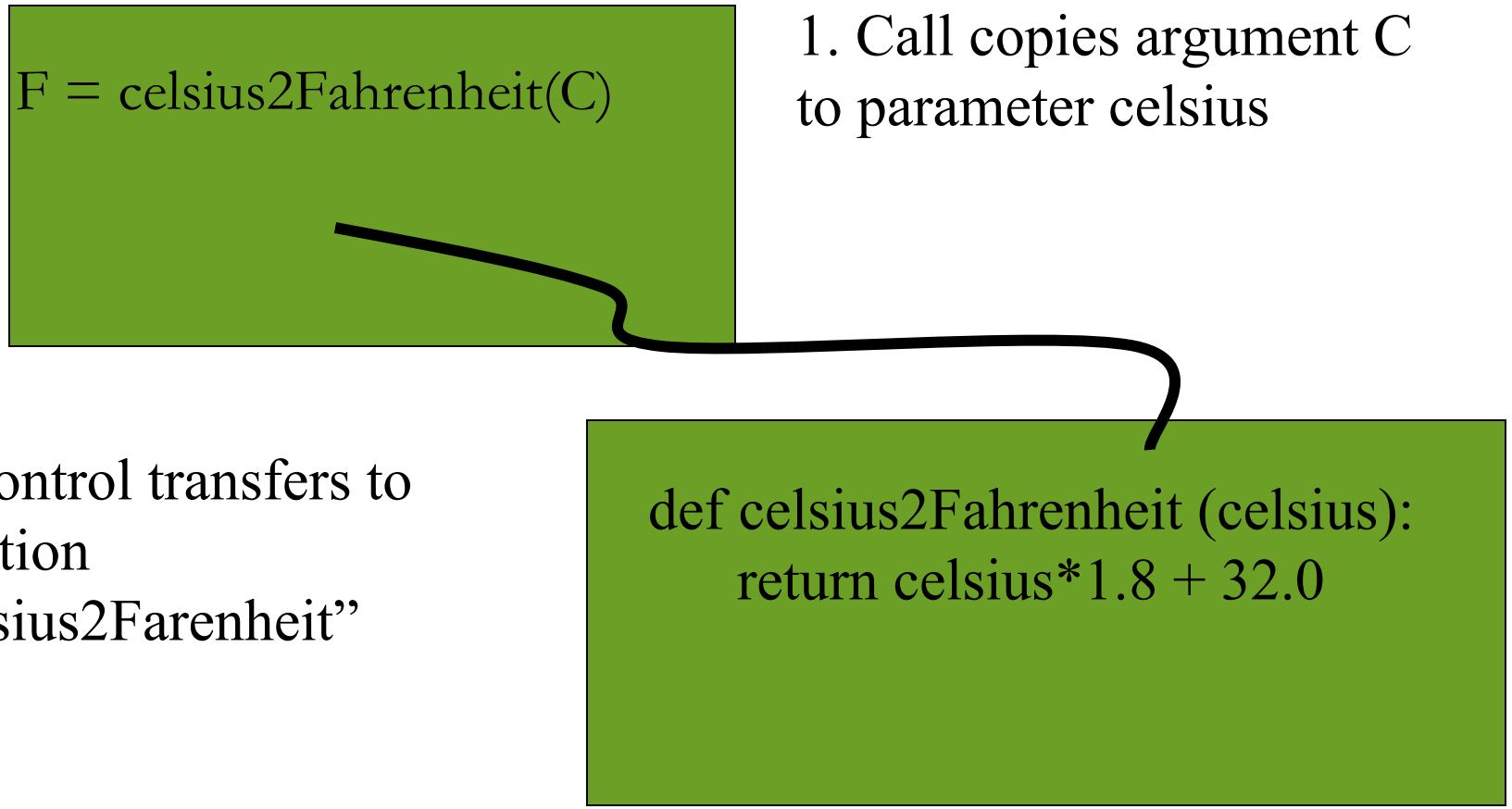
```
#doing function stuff
return result
print("Hello!")     #This line will never happen
```

Multiple Returns in a Function

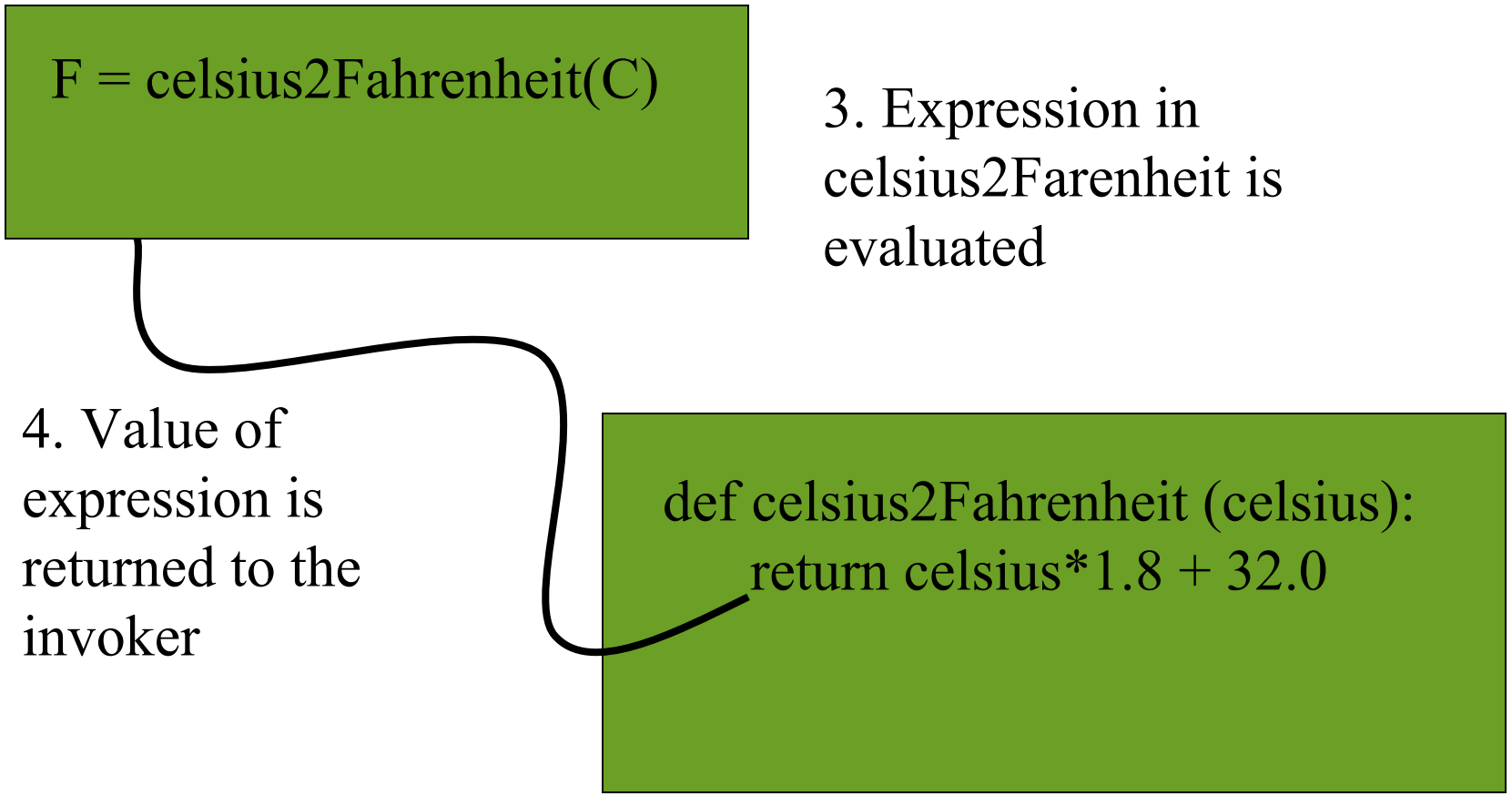
- When you use if/elif/else statements, you could place a return in every branch.

```
if result < 0:
    return None
elif result == 1:
    return 1
else:
    return 42 #the answer to everything else
```

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)



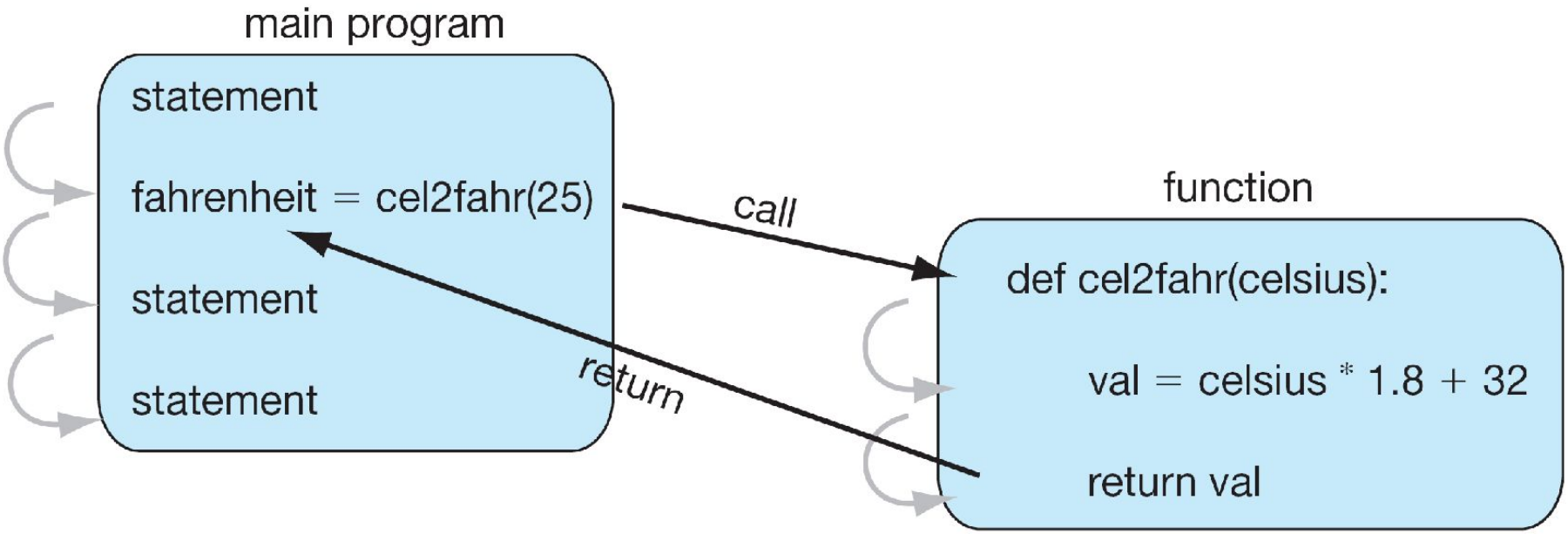


FIGURE 5.2 Function flow of control.

Procedures

- Functions that have no return statements are often called *procedures*.
- Procedures are used to perform a task (print output, store a file, etc.).
- A return statement is not required.

Example

```
def printGameRules():
    print("Please select from one of the following choices:")
    print("     Enter s to show the score")
    print("     Enter p to play")
    print("     Enter e to exit")
    print()

#run program
printGameRules()
```

Example: implement len

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

Example: 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

Example: check membership in lowercase

```
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
- Readable. 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 work for all the cases where it might be invoked.
- Not too long. Kind of synonymous with “does one thing”. Use it as a measure of doing too much.