CSC108H Lecture 3

Dan Zingaro

September 14, 2012

Functions

- In math, we could define a function such as $f(x) = x^2$
  - What is the value of $f(3)$? $f(5)$?
- In Python, we can achieve a similar effect

```
def f(x):
  return x ** 2
```

Now, what does the call `f (3)` do?

Function Definitions

```
def f(x):
  return x ** 2
```

- `def` is a keyword; it has a special meaning to Python and cannot be used as a variable or function name
- The `return` statement terminates the function, and determines the value returned to the caller
- If there is no `return`, the function terminates when it reaches the bottom (and returns `None`)
- Notice how the body of the function is indented! This is required by Python!
- `x` is the **parameter** of function `f`
- `x` is a variable accessible only from within the function
- We could use any name we like for the parameter and/or function name

ConcepTest

Which of the following contains a function call?

(1)

```
type(4.5)
```

(2)

```
def add_one(x):
  return x + 1
```

(3)

```
area(2, 9)
```

(4)

```
print("Hello")
```

- A. (3) only
- B. (2) and (3)
- C. (1), (3), and (4)
- D. All of (1), (2), (3), and (4) include a function call

ConcepTest

What is the output of this code?

```
def calculate(w, x, y):
  a = x
  b = w + 1
  return a + b + 3

print(calculate(3, 2, 0))
```

- A. 5
- B. 9
- C. 0
- D. 3

ConcepTest

What is the output of this code?

```
def calculate(w, x, y):
  a = x
  b = w + 1
  return a + b + w

print(calculate(1, 2, 0))
```

- A. 3
- B. 4
- C. 5
- D. 6

ConcepTest

What are the bugs in the following code?

```
def add_one(x):
return x + 1

x = 2
x = x + add_one(x)
```

- A. No bugs. The code is fine
- B. The function body is not indented
- C. We are using `x` as a parameter and a variable, but we are not allowed to do that
- D. Both B and C are bugs

Function Design Recipe

We are going to use a recipe for writing functions.
Step 1: write some examples of the function call.

```
>>> glue_strings('John', 'Smith')
'John Smith'
```

Step 2: write the type contract (what the function accepts and what it returns)

```
(str, str) -> str
```

Function Design Recipe...

Step 3: write the header.

```
def glue_strings(s1, s2):
```

Step 4: add a description of the function, referring to each parameter by name

```
  Return the string composed of s1, a space, and s2.
```

Step 5: put it all together and write the function. Let's do that now!

Python Modules

- Modules are used to group functions
- Use `import` to make a module's functions available
- Any file ending in `.py` can act as a module
- `import` runs all the code in a `.py` file, but only the first time (i.e. if you make changes to a file and try to import it again, nothing will happen)
- If we `import module_name`, we access its functions through `module_name.function_name`
- To access `function_name` without having to type the `module_name` prefix, we can use
  `from module_name import function_name`