Reminder: Please keep the middle part of the classroom "device free"

If you wish to use a laptop, move to the sides or back.

Coming Up

- Friday: Built-in Functions
  Intro to Functions
- Sunday @ 10: Fri Review, Weekly Exercises, Mon Prep due
- Monday: Booleans
- Tuesday @ 10: Mon Review and Wed Prep due
- ... expect A1 to be posted soon!

Getting Help

- Office hour after every lecture
- Closed Labs (led by your TA)
- Online discussion boards
  - Find/build study groups
- Email
  - Please read announcements on the discussion board first
  - Use a good subject, such as "108: A1 question about pixels"
- Don't spin your wheels -- ask for help!

Discussion Board

- Please make sure you have discussion board access.
  - Visit the discussion board (linked via the course webpage)
  - Use your UTORID and password to log in

- If you are unable to log in, the mcs system administrator will help.
  - Send me an email, and I will forward it.

Mea Culpa

- Apologies to those of you who attempted to access the pi.py file last night

- I posted it as a ".py" file, but our webserver tried to execute it, so you saw a "HTTP 500" error
- The system administrator is working on the issue. In the meantime, you can view the files now. Copy and past them into a python file on your machine.

Assignment vs Equality

- Python variables look like math variables.
- This could be Python or math:
  p = 5
  q = p * 7
- But "=" in math means equality
  (stating a fact)
  whereas "=" in Python means assignment
  (asking Python to *do* something)
- This makes a big difference!

Assignment Statement

Form:

```
variable = expression
```

How it's executed:

1. Evaluate the expression on the RHS. (The value of the expression is a memory address.)
2. Store that memory address in the variable on the LHS.

Assignment Statement

Form:

Remember this!

```
variable = expression
```

How it's executed:

1. Evaluate the expression on the RHS. (The value of the expression is a memory address.)
2. Store that memory address in the variable on the LHS.

1. Mutability

- In math, this is inconsistent:

  p = 5
  q = p * 7
  p = q + 10

- p can't be both 5 and 45!
- But in Python, it makes perfect sense. p starts out referring to 5, but then changes to refer to 45.
- You can change a variable's value as many times as you want. You can even change its type.

2. No linked variables

- # What does this do?
  x = 37
  y = x + 2
  # **x refers to 37, y refers to 39**
  x = 20
  # Does y refer to 22? **No**
- You can't use assignment to link the values of two variables together permanently.
- Note: This is different from having two variables referring to the same value. More on this later ...

3. Assignment is not symmetric

	In math	In Python
sum = a + b	they mean the same thing	fine
a + b = sum		illegal

Rules for the format of names

- There are a few rules about names of variables (and other things we'll see later):
  - Must start with a letter (or underscore).
  - Can include letters, digits, and underscores, but nothing else.
  - And case matters, by the way.

```
age = 11
aGe                 # Error!  This is not defined.
```

- Valid: `_moo_cow`, `cep3`, `I_LIKE_TRASH`
- Invalid: `49ers`, `@home`

Conventions for the format of names

- thEre'S a GoOD rEasON wHy WorDs haVE A StaNDaRd caPITaLizAtIon sCHemE
- Python convention: `pothole_case`
- `CamelCase` is sometimes seen, but *not for functions and variables*
- Rarely, single-letter names are capitalized: `L`, `X`, `Y`
- When in doubt, use `lowercase_pothole`

Producing Output

- In Python, you normally make full statements, eg:
  - assignment statements
  - def statements
  - if statements
- But the shell lets you provide just an expression, and it then shows you the value of the expression.
- So to show output in the shell, you can just give an expression.

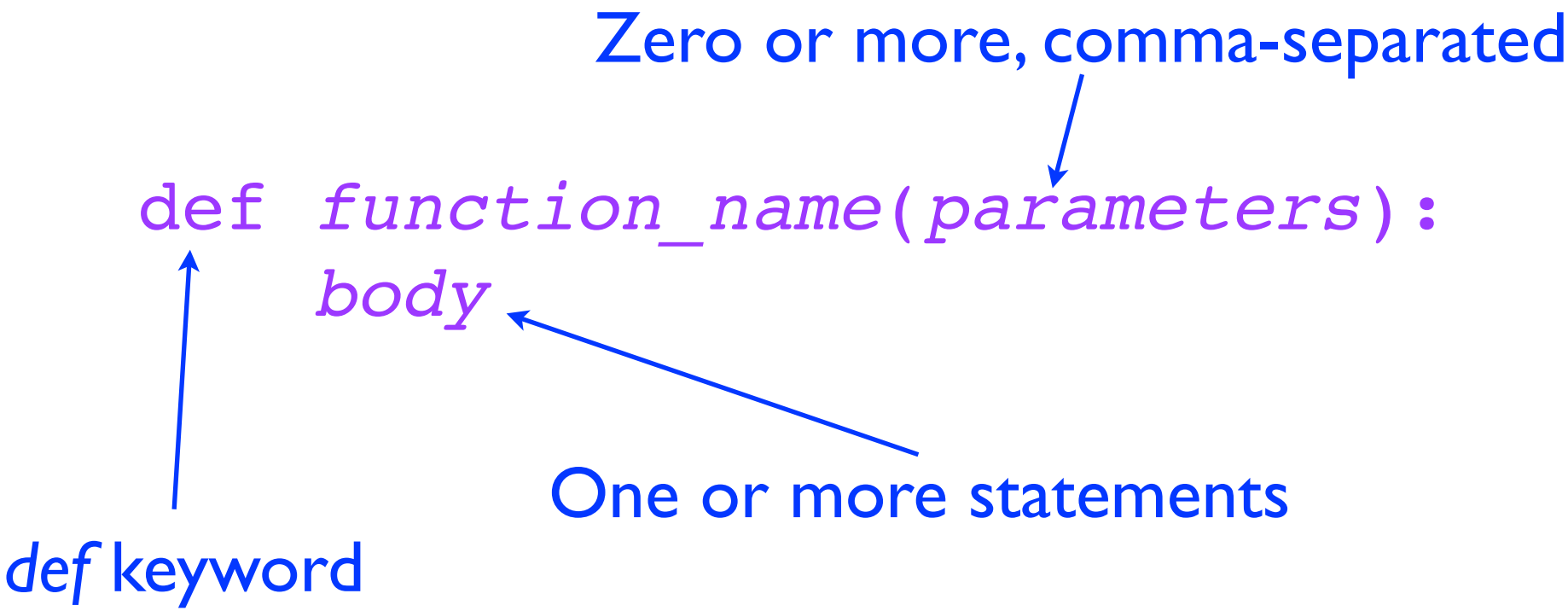
- To show output in the editor, use print. Example:

```
print “Hello!”
mark1 = raw_input(“First mark: “)
mark2 = raw_input(“Second mark: ”)
print(“The average is”, average(mark1, mark2))
```

- Comma is for printing multiple items separated by blanks.

Any burning questions
before we start?

Function Definition



Function Call

Form:

zero or more, comma-separated

```
function_name(arguments)
```

How it's executed:

1. Each argument is an expression. Evaluate these expressions, in order. (The value of each expression is a memory address.)
2. Store those memory addresses in the corresponding parameters.
3. Execute the body of the function.

Return Statement

Form:

```
return expression
```

How it's executed:

1. Evaluate the expression. (The value of the expression is a memory address.)
2. Exit the function, using that memory address as the value of the function call.

NB: The function ends *immediately*. Any remaining statements in it are not executed.

Reminder: NO TUTORIAL TODAY

Remember: We're in DV2072 on Monday!