USU CI-WATER Code Camp

Coding Introduction and Warm Up

Jeffery S. Horsburgh

Utah State University

Support:
EPS 1135482

Solving a Problem Using Code

How long it takes to write a program

Human time

How long it takes that program to run

Computer time

Every programming language makes a tradeoff between these

Python
MATLAB

Java
C#

Fortran
C

Why Python?

- Python is generally:
  - Comparatively easy to learn
  - Freely available
  - Cross-platform (Windows, Mac, Linux)
  - Widely used
  - Well documented
  - Well supported
  - Simple, interpreted language – no compilation step

http://www.python.org

python™

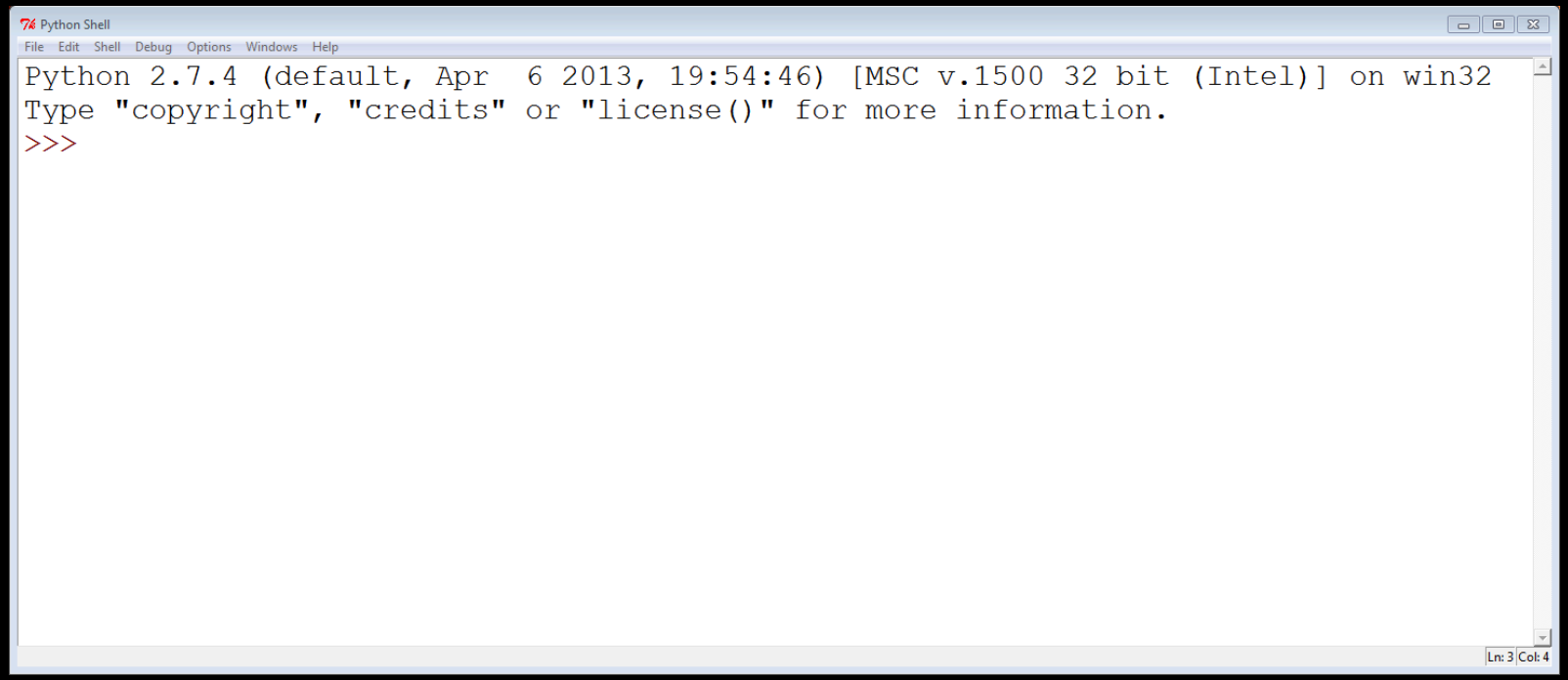
Python Basics

- To run Python interactively, we will use Idle (the Python GUI)

- Windows: Start → All Programs → Python 2.7 → Idle (Python GUI)

- Mac: Applications → Python 2.7 → Idle

Idle


Python Shell
File Edit Shell Debug Options Windows Help
Python 2.7.4 (default, Apr 6 2013, 19:54:46) [MSC v.1500 32 bit (Intel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>>
Ln: 3 Col: 4

Simple Arithmetic

```
>>>print 1 + 2
3
>>>
```

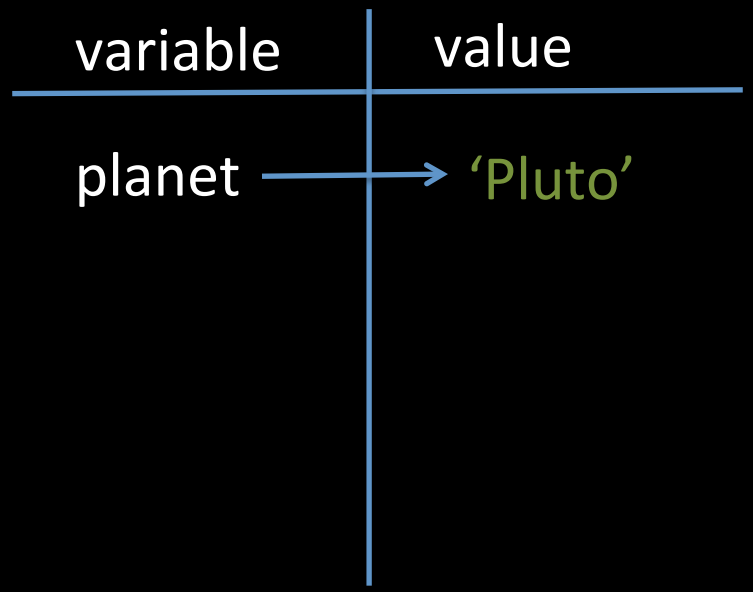
Manipulate Strings

```
>>>print ‘charles’ + ‘darwin’
charlesdarwin
>>>
```

Variables

- Variables are names for values
- Created by use – no declaration necessary

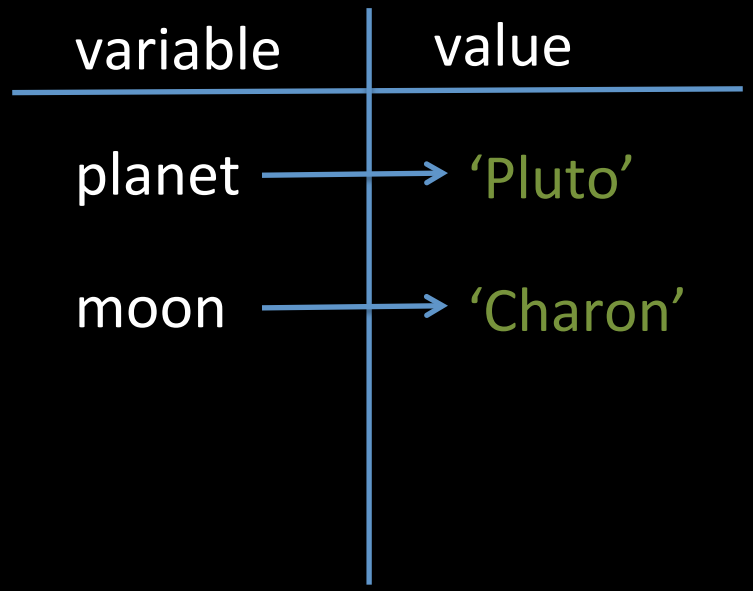
```
>>>planet = 'Pluto'
>>>print planet
Pluto
>>>
```



Variables

- Variables are names for values
- Created by use – no declaration necessary

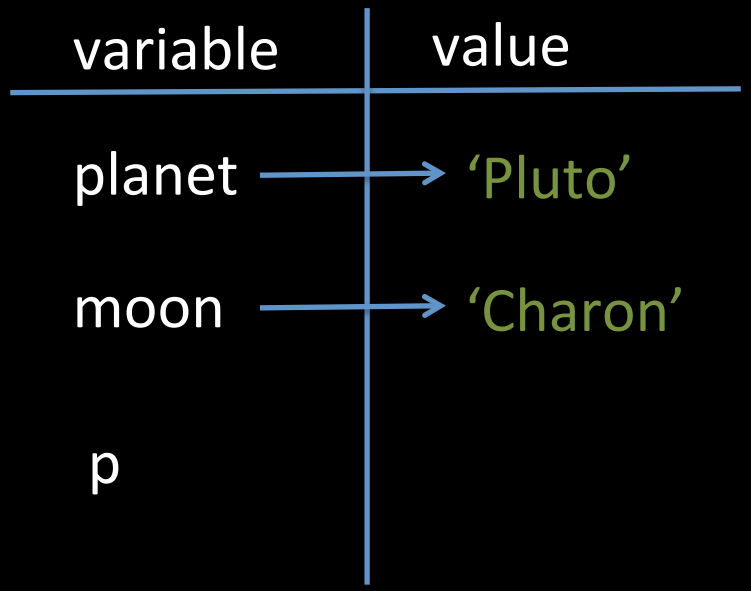
```
>>>planet = 'Pluto'
>>>print planet
Pluto
>>>moon = 'Charon'
>>>
```



Variables

- Variables can be assigned to other variables

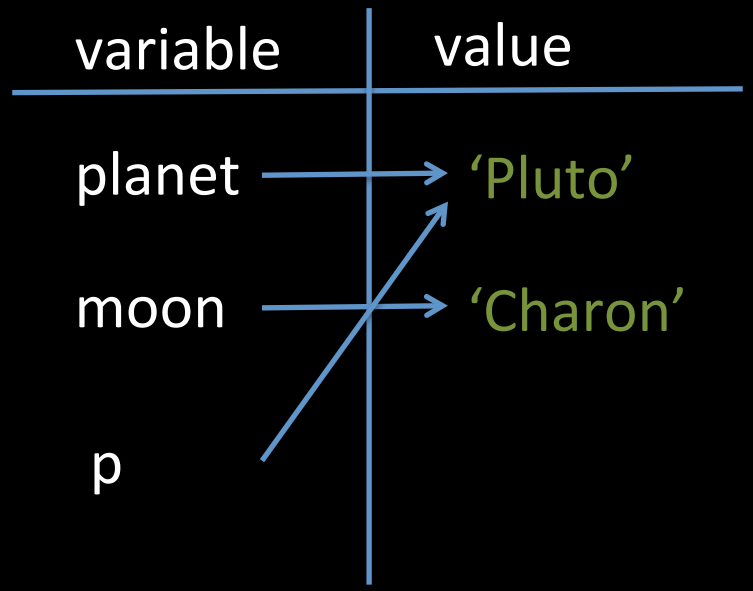
```
>>>p = planet
>>>
```



Variables

- Variables can be assigned to other variables

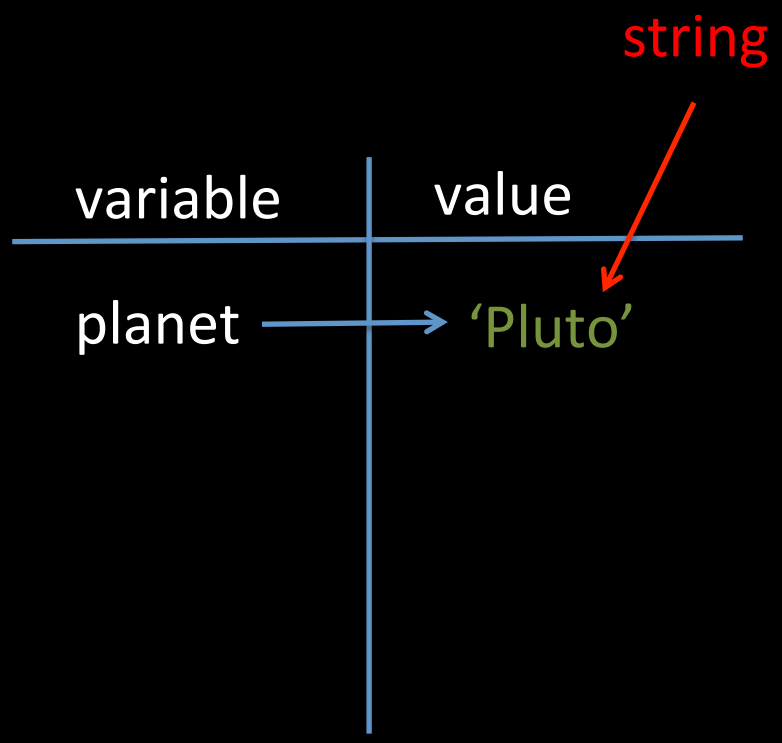
```
>>>p = planet
>>>print p
Pluto
>>>
```



Variables

- In Python, variables are just names
- Variables do not have data types

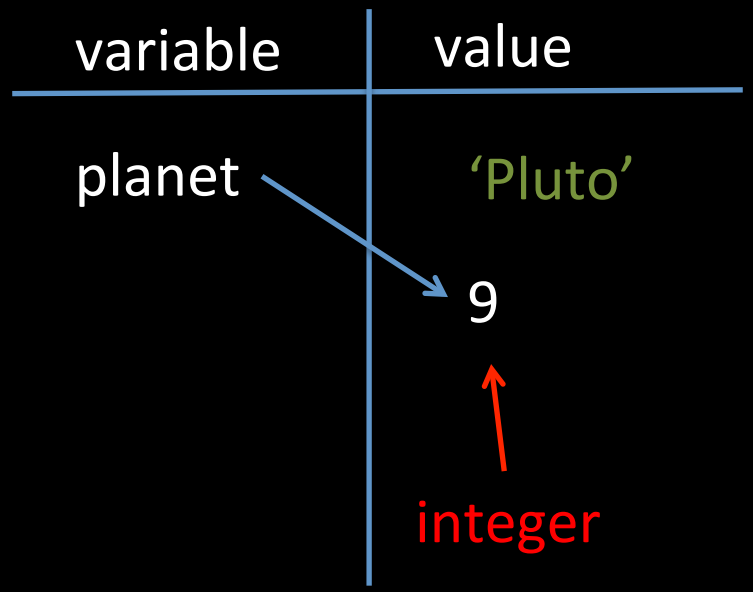
```
>>>planet = 'Pluto'
>>>
```



Variables

- In Python, variables are just names
- Variables do not have data types

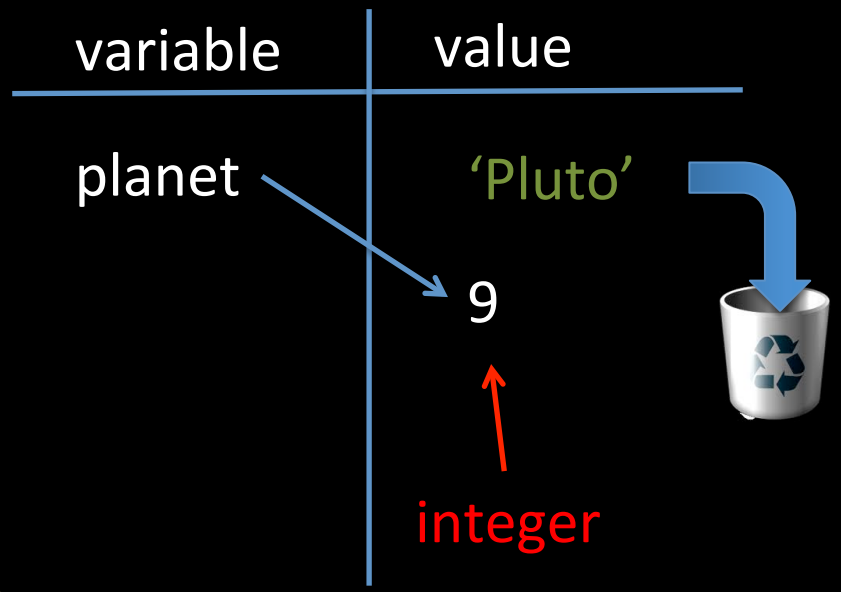
```
>>>planet = 'Pluto'
>>>planet = 9
>>>
```



Variables

- In Python, variables are just names
- Variables do not have data types

```
>>>planet = 'Pluto'
>>>planet = 9
>>>
```



Python collects the garbage and recycles the memory (e.g., 'Pluto')

Variables

- You must assign a value to a variable before using it

```
>>>planet = ‘Sedna’
>>>
```

Variables

- You must assign a value to a variable before using it

```
>>>planet = 'Sedna'
>>>print plant     #Note the deliberate misspelling
```

Variables

- You must assign a value to a variable before using it

```
>>>planet = 'Sedna'
>>>print plant     #Note the deliberate misspelling
Traceback (most recent call last):
  File "<pyshell#11>", line 1, in <module>
    print plant
NameError: name 'plant' is not defined
```

Unlike some languages – Python does not initialize variables with a default value

Values Do Have Types

```
>>>string = 'two'
>>>number = 3
>>>print string * number
```

Values Do Have Types

```
>>>string = 'two'
>>>number = 3
>>>print string * number  #Repeated concatenation
twotwotwo
>>>
```

Values Do Have Types

```
>>>string = 'two'
>>>number = 3
>>>print string * number  #Repeated concatenation
twotwotwo
>>>print string + number
```

'two3' ????

If so, then what is the result of '2' + '3'

- Should it be the string '23'
- Should it be the number 5
- Should it be the string '5'

Values Do Have Types

```
>>>string = ‘two’
>>>number = 3
>>>print string * number  #Repeated concatenation
twotwotwo
>>>print  string + number
Traceback (most recent call last):
  File "<pyshell#15>", line 1, in <module>
    print string + number
TypeError: cannot concatenate 'str' and 'int' objects
```

Use Functions to Convert Between Types

```
>>>print int('2') + 3
5
>>>
```

Arithmetic in Python

Addition	+	35 + 22	57
		‘Py’ + ‘thon’	‘Python’
Subtraction	-	35 - 22	13
Multiplication	*	3 * 2	6
		‘Py’ * 2	‘PyPy’
Division	/	3.0 / 2	1.5
		3 / 2	1
Exponentiation	**	2 ** 0.5	1.41421356…

Comparisons

```
>>>3 < 5
True
>>>
```

Comparisons turn numbers or strings into True or False

Comparisons

3 < 5	True	Less than
3 != 5	True	Not equal to
3 == 5	False	Equal to (Notice double ==)
3 >= 5	False	Greater than or equal to
1 < 3 < 5	True	Multiple comparisons

Single '=' is assignment
Double '==' is comparison

Behold the Power of Programming!

- The real power of programming comes from the ability to perform:
  - **Selection** – the ability to do one thing rather than another
  - **Repetition** – the ability to automatically do something many times

Selection – if, elif, and else

```
moons = 3
if moons < 0:
    print 'less'
elif moons == 0:
    print 'equal'
else:
    print 'greater'
```

Always starts with if and a condition

There can be 0 or more elif clauses

The else clause has no condition and is executed if nothing else is done

Tests are always tried in order

Since moons is not less than 0 or equal to zero, neither of the first two blocks is executed

Selection – if, elif, and else

```
>>>moons = 3
>>>if moons < 0:
    print 'less'
elif moons == 0:
    print 'equal'
else:
    print 'greater'

greater
>>>
```

Indentation

- Python uses indentation to show which statements are in an if, elif, else statement

- Any amount of indentation will work, but the standard is 4 spaces (and you must be consistent)

Repetition - Loops

- Simplest form of repetition is the while loop

```
numMoons = 3
while numMoons > 0:
    print numMoons
    numMoons -= 1
```

Repetition - Loops

- Simplest form of repetition is the while loop

```
numMoons = 3
while numMoons > 0:
    print numMoons
    numMoons -= 1
```

While this is true

Do this

Repetition - Loops

```
>>>numMoons = 3
>>>while numMoons > 0:
    print numMoons
    numMoons -= 1
3
2
1
>>>
```

Combine Looping and Selection

```
numMoons = 0
while numMoons < 10:
    if numMoons > 0:
        print numMoons
    numMoons += 1
```

Combine Looping and Selection

```
>>>numMoons = 0
>>>while numMoons < 5:
    if numMoons > 0:
        print numMoons
    numMoons += 1
1
2
3
4
>>>
```

Saving and Executing Code

- Writing and executing complex code is too difficult to do line by line

- Write code using a text editor
- Save the code as a text file with a “.py” extension
- Execute code in the file from the command line

Credits

- Some instructional materials adapted from the Software Carpentry website http://software-carpentry.org

Copyright © Software Carpentry

Support:
EPS 1135482