

Variables and Types

- Python is a **dynamically typed language**
 - Variables can hold values of any type
 - Type of value in variable can change over time
- The following is acceptable in Python:


```
>>> x = 1      ← x contains an int value
>>> x = x / 2.0 ← x contains a float value (why?)
```
- Alternative is a **statically typed language**
 - Each variable restricted to values of just one type
 - Java is an example of such a language

Dynamic Typing

- Often want to track the type in a variable
 - What is the result of evaluating x / y ?
 - Depends on whether x, y are **int** or **float** values
- Use expression `type(<expression>)` to get type
 - `type(2)` evaluates to `<type 'int'>`
 - `type(x)` evaluates to type of contents of x
- Can use in a boolean expression to test type
 - `type("abc") == str` evaluates to **True**

Python Shell vs. Modules

```
Python 2.7.3 (v2.7.3:7827465316d, Apr 9 2012, 18:52:43)
[GCC 4.2.1 (Apple Inc. build 5666) (dot 3)] on darwin
Type "help()", "copyright()", "credits()" or "license()" for more information.
>>> x = 1+2
>>> x = 3*x
>>> x
9
>>> |

module.py
1 # module.py
2 # Walker M. White
3 # June 20, 2012
4
5 """ This is a simple module.
6 It shows how modules work"""
7
8 x = 1+2
9 x = 3*x
10 x
```

- The interactive shell
 - Simple to use
 - Experience with Lab 1
- But very inefficient
 - One command at a time
 - What if need 1000+ lines?
- Alternative: **modules**
 - Files with commands
 - Write in a special editor
- Run module with **import**
 - Loads the file into Python
 - Executes each line

Using a Module

Module Contents

```
# module.py
""" This is a simple module.
It shows how modules work"""
x = 1+2
x = 3*x
x
```

- `# module.py`: Single line comment (not executed)
- `""" This is a simple module. It shows how modules work"""`: Docstring (note the Triple Quotes) Acts as a multiple-line comment Useful for code documentation
- `x = 1+2` and `x = 3*x`: Commands executed on import
- `x`: Not a command Does nothing

Using a Module

Module Contents	Python Shell
<code># module.py</code>	<code>>>> import module</code>
<code>""" This is a simple module. It shows how modules work"""</code>	<code>>>> x</code>
<code>x = 1+2</code>	Traceback (most recent call last): File "<stdin>", line 1, in <module> NameError: name 'x' is not defined
<code>x = 3*x</code>	<code>>>> module.x</code>
<code>x</code>	<code>9</code>
	<code>>>> help(module)</code>

- "Module data" must be prefixed by module name
- Prints docstring and module contents

Function Calls

- Python supports expressions with math-like functions
 - A function in an expression is a **function call**
 - Will explain the meaning of this later
- Function expressions have the form **fun(x,y,...)**
 - function name
 - argument
- Simplest example of functions are in module **math**

```
>>> import math
>>> math.sin(math.pi)
```

 - `import math`: module variable
 - `math.sin(math.pi)`: module function

Using the from Keyword

```
>>> import math
>>> math.pi
3.141592653589793
>>> from math import pi
>>> pi
3.141592653589793
>>> from math import *
>>> cos(pi)
-1.0
```

Must prefix with module name

No prefix needed for variable pi

No prefix needed for anything in math

- Be careful using from!
- Modules are **namespaces**
 - There is only one variable or function for each name
 - Other modules may reuse names for variable/function
 - Prefix keeps them distinct
- **Example:** badpi.py

How Well Are You Following?

Module Contents	Python Shell
# data.py	>>> x = 1
""" Module with two variables """	>>> y = 2
x = 4	>>> from data import x
y = 3	>>> x+y
	???

A: 3
B: 7
C: 6
D: 4
E: I do not know

How Well Are You Following?

Module Contents	Python Shell
# data.py	>>> from data import *
""" Module with two variables """	>>> x = 3
x = 4	>>> from data import x
y = 3	>>> x+y
	???

A: 6
B: 7
C: Error!
D: I do not know

Importing a variable "clobbers" any existing variable of same name

Python Comes with Many Modules

- **io**
 - Read/write from files
- **math**
 - Mathematical functions
- **random**
 - Generate random numbers
 - Can pick any distribution
- **string**
 - Useful string functions
- **sys**
 - Information about your OS
- Complete list:
 - <http://docs.python.org/library>
- **Library:** built-in modules
 - May change each release
 - Why version #s are an issue
- Documentation is the **API**
 - Application
 - Programming
 - Interface
- **Interface:** *specification* of the functions and data in a module

Reading the Python API

Function name: `math.ceil(x)`

Number of arguments: 1

Return the ceiling of x as a float, the smallest integer value greater than or equal to x.

What the function evaluates to: `math.ceil(2.5)` returns `3.0`

The Komodo Editor

Line numbers: 1, 2, 3, 4, 5

Current working directory: /usr/local/bin/python

Current active module: /usr/local/bin/python

Execution output when module is "run": Hello World!

See website for how to add button