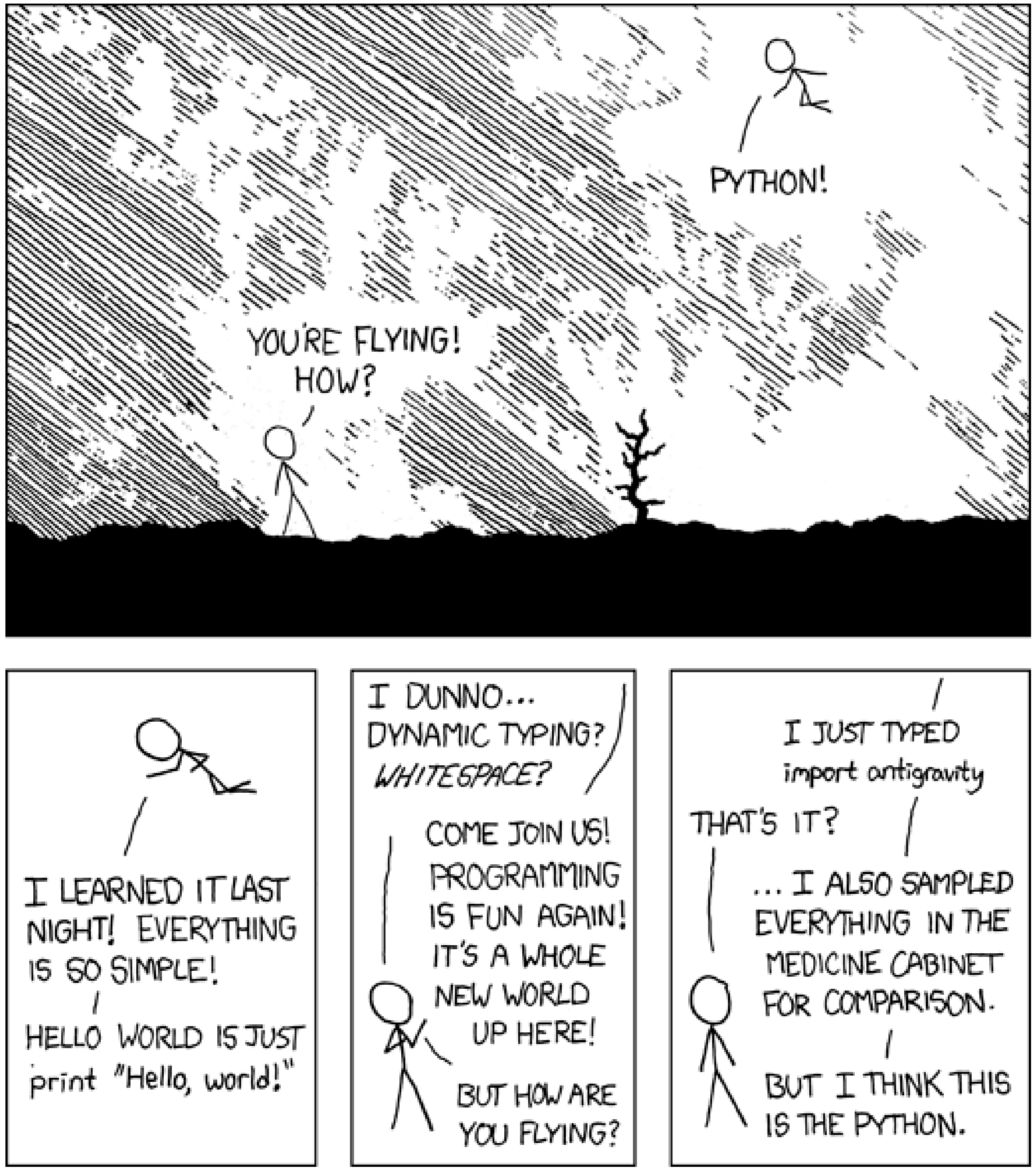
Lect 2 – Python Data, Loops, Modules

Rob Capra

INLS 490-172


PYTHON!
YOU'RE FLYING! HOW?
I LEARNED IT LAST NIGHT! EVERYTHING IS SO SIMPLE!
HELLO WORLD IS JUST print "Hello, world!"
I DUNNO... DYNAMIC TYPING? WHITESPACE?
COME JOIN US! PROGRAMMING IS FUN AGAIN! IT'S A WHOLE NEW WORLD UP HERE!
BUT HOW ARE YOU FLYING?
I JUST TYPED import antigravity
THAT'S IT?
... I ALSO SAMPLED EVERYTHING IN THE MEDICINE CABINET FOR COMPARISON.
BUT I THINK THIS IS THE PYTHON.

http://xkcd.com/353/

Data Types

- Integer: 4, 123
- Float: 3.14, 89.75
- String: “Hello”, ‘Hello’, ‘3.75’

- **`type()`** will return the type of an object

Triple-Quoted Strings

- `"Bruce's beard“`

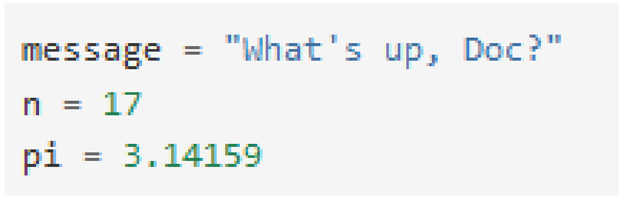
- `'Knights who say "Ni!"'`

- `'''"Oh no", she exclaimed.`
  `"Ben's bike is broken!"'''`

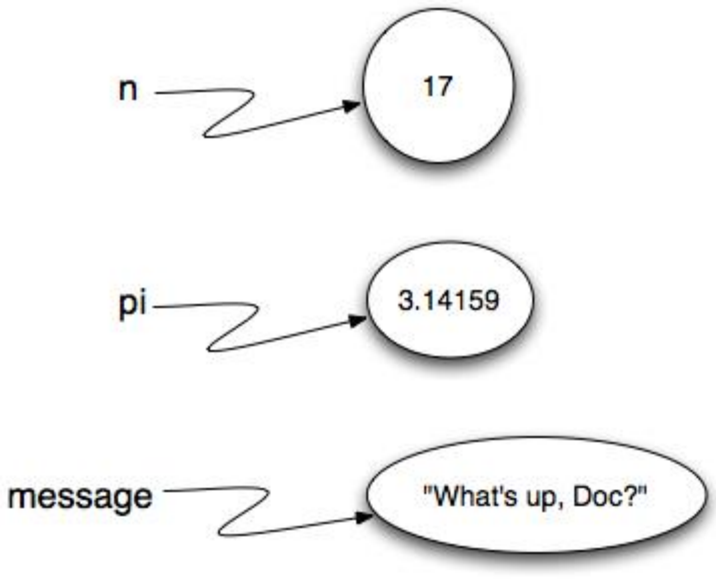
Type Conversion

- Examples in INTPY (activecode 10, ch02_20)

Variables

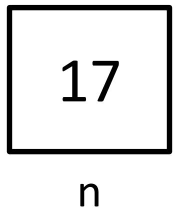


```
message = "What's up, Doc?"
n = 17
pi = 3.14159
```



The type of a variable is the type of the object it currently refers to. (dynamic typing)

Contrast with statically-typed languages.



Variables

- In Python, you don't have to declare variables to be a particular type, but the objects referenced do have a type.
  - Ex:

```
x = 10
type(x)
x = 3.14
type(x)
```

Variable Initialization

- And you still need to initialize variables before you use them.
  - Java:
    ```
    int x = 10;
    x = x + 1;
    ```
  - Python:
    ```
    x = 10
    x = x + 1
    ```

Variable Names

- Can be as long as you like
- Can contain letters and digits and underscores
- Must start with a letter or underscore
- Cannot be a Python keyword
- Typically are all lower-case
- Case matters
  - `Spam` and `spam` are different variables

Statements vs Expressions

- A statement is an instruction to be executed
- An expression combines values, variables, operators and functions and are evaluated to produce a value

Expressions

- Operators and Operands
- Floating point vs. Integer division (Py2 vs Py3)
- Modulus, remainder, integer remainder
  - Examples from INTPY (activecode 19, ch02_17)

Input

- `input()` – get input from the user
- Can include a prompt string
- Examples: activecode 21 (inputfun)

Boolean Expressions

- Evaluate to **`True`** or **`False`** `(bool)`
- Relational operators:

  == != > < >= <=

  - Remember == instead of =
  - There is no =< or =>
- Logical operators:
  - and, or, not

Conditional Execution

- If statement

```
    if x>0:
        print 'x is positive'
```

- No curly braces {}
- Colon : at the end of the if
- Indented statement is executed if true (whoa!)
  - Spaces vs tabs debate
  - Next slide: aside about PEP 8

Python Enhancement Proposals (PEPs)

- http://www.python.org/dev/peps/
- PEP 20 – The Zen of Python
- PEP 8 – Style Guide for Python Code
    - Your code should follow this guide

Coding Style Guide (PEP8)

- Use 4 spaces per indentation level.
- Limit all lines to a maximum of 79 characters.
  - Docstrings/comments limit to 72 characters

Alternative Execution

- If..else

```
if x%2 == 0:
    print ' x is even'
else:
    print 'x is odd'
```

- No curly braces {}
- Colon : at the end of the if

Chained conditionals

```
if x < y:
    print 'x is less than y'
elif x > y:
    print 'x greater than y'
else:
    print 'x and y are equal'
```

Nested conditionals

```
if 0 < x:
    if x < 10:
        print 'x is pos single digit'
```

```
if 0 < x and x < 10:
    print 'x is pos single digit'
```

For loop – Iteration

- For loop processes each item in a list
- In turn, each item is assigned to the loop var
- Then the body of the loop is executed

```
for name in ["Amy", "Brad", "Cathy"]:
   print "Hi,", name, "!!!"
```

For loop – range()

- range(n) – returns a list [0 .. n-1]
- range(n,m) – returns a list [n .. m-1]

```
for i in range (3):
    print i, "squared =", i*i
```

Modules

- Modules contain Python code designed for use in other programs
- Many modules in the Standard Library
- Use a module by importing it
- Example “import turtle”: activecode 1 (chmod_01)

Documentation and Standard Library

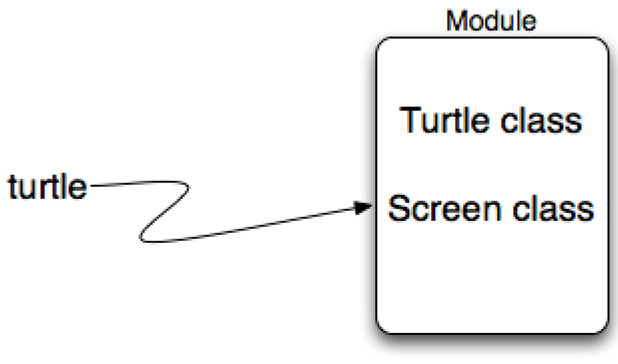
- Python Documentation
  - http://docs.python.org/2.7/
- Language Reference
  - http://docs.python.org/2.7/reference/index.html
- Module Index (standard library)
  - http://docs.python.org/2.7/py-modindex.html

Modules

- Modules are objects

```
import turtle
wn = turtle.Screen()
alex = turtle.Turtle()

alex.forward(150)
alex.left(90)
alex.forward(75)
wn.exitonclick()
```

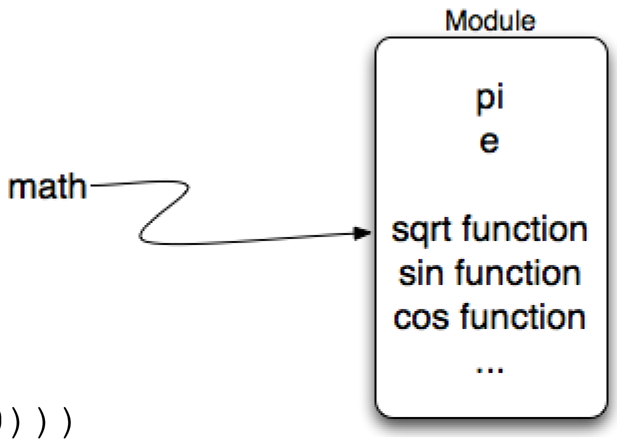


```
import math

print(math.pi)
print(math.e)

print(math.sqrt(2.0))

print(math.sin(math.radians(90)))
```



Debugging

- Avoid (major) debugging by:
  - Start small
  - Keep it working / small victories
  - Example from INTPY
- Hints
  - Test boundary conditions
  - Know your error messages
    - 90% = ParseError, TypeError, NameError, ValueError
    - Examples from INTPY

Debugging Error Types

- ParseError – syntax error
  - Ex: missing parens, quotes, commas
  - Try: comment out line, see what errors change
  - Try: narrow the source of the error
- TypeError – incompatible objects
  - Ex: try to add an int and str
  - Often math/expression statements
  - Try: print values

Debugging Error Types

- NameError – use a var before it has a value
  - Often caused by typos, speeling mistaches, mis-remembering var/function name
  - Try: use search feature of editor
- ValueError – pass wrong type parameter to a function