CSC 280 Introduction to Computer Science: Programming with Python

Lecture 2 : Values, Types and Expressions

September 2, 2015

Prof. Bei Xiao

Fall, 2015

American University

Outline

- Compiled versus Interpreted language
- Values and Expressions
- Types
- Variables
- First Python Program

Compiled vs. Interpreted

- Interpreted (easier to learn):

  source code (you wrote) -> checker->interpreter->output

  Examples: Python, Perl, Lisp, PhP.

  Strength: easier to debug, Weakness: slower

- Compiled (more efficient):

  source code (you wrote) -> checker/compiler->object code (language close to computer hardware knows) -> interpreter by hardware->output

  Error message will be in object code

Python is

- A scripting language (strong in text-processing)
- An interpreted language, like Perl.
- A very high-level language (close to human semantics).
- Procedural (like C, Pascal, but much more)
- But also object-oriented (like C++ Java).
- And even functional

On to Python

“hello world”

- C

```
#include <stdio.h>
int main(int argc, char ** argv)
{
printf(“Hello, World!\n”);
}
```

- Java

```
public class Hello
{
public static void main(String argv[])
{
! ! System.out.println(“Hello, World!”);
}
}
```

- Python

```
print “Hello, World!”
```

From last lecture: What is a program

- Program is a set of instructions that specifies how to perform a computation
- Input
- Output
- Math
- Conditional execution
- Repetition

Questions

What is the advantage of stored-program computer?

What is syntax, static semantics, semantics?

Let’s get started

IDE (Integrated Development Environment)
IDEL (Python) is also told by inspired by the movie Monty Python

IDEL

- Integrated Development Environment
- Shell, where we can type things.
- Syntax Highlighting
- Integrated Debugger (but it is not super useful).
- Use printing statements for debugging.
- Color means something.

Demo: Printing

- Before we do anything, let's try to print something. Open an editor, type the following and save it as a file:

```
print "Hello World!"
print "Hello Again"
print  "This is fun."
print 'Yay! Printing.'
print "I 'd much rather you 'not' ."
print "I like typing this.
```

Run the file by either select " Run Module" or hit F5

- What is your output? What type of error is it?

“Double Quotes”

- >>> "doesn't"
  "doesn't"
- >>> 'doesn\'t'
  "doesn't"
- >>> '"yes"he said.'
- '"yes"he said.'
  >>> "\"Yes,\"he said."
  '"Yes,"he said.

What is the output of the following statement?

>>> '"Isn\'t," she said.'

Objects

- Everything in Python is an object
- Each object has a type
- Two types: Scalar and non-Scalar

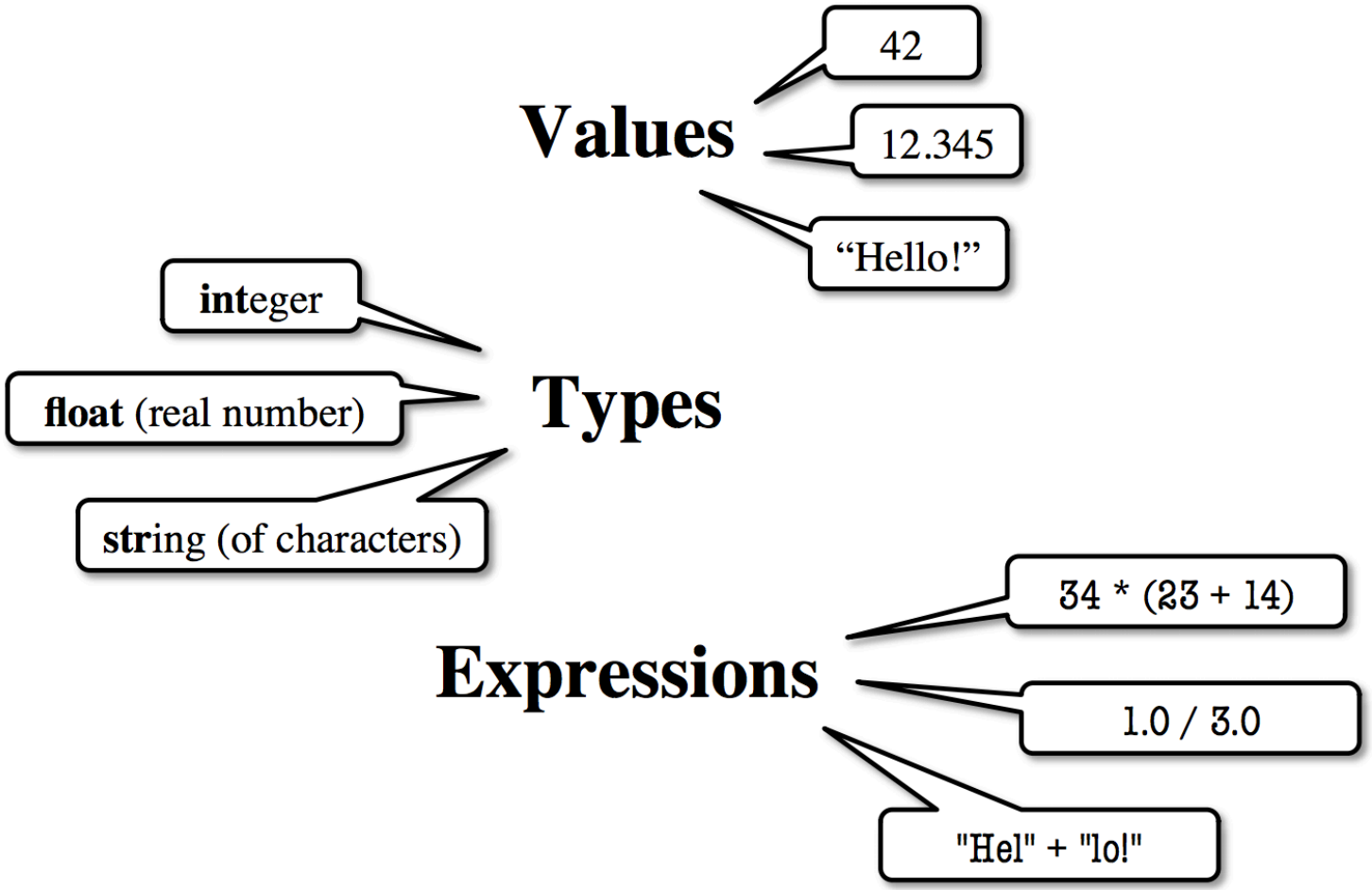
Three ways to run a Python program

- Interactive

```
>>> for i in range(5):
    print i,
    0 1 2 3 4
```

- Save to a file, say foo.py
  in command line: foo.py
- Add a special line pointing to the default interpreter:
  add #!/usr/bin/env python to the top of foo.py
  make foo.py executable (chmod +x foo.py)
  in the command-line: ./foo.py

The Basics


Values
42
12.345
“Hello!”
Types
integer
float (real number)
string (of characters)
Expressions
34 * (23 + 14)
1.0 / 3.0
"Hel" + "lo!"

Representing Values

- Everything on a computer reduces to numbers
  - Letters represented by numbers (ASCII codes)
  - Pixel colors are three numbers (red, blue, green)
  - So how can Python tell all these numbers apart?
- Type:
- A set of values and the operations on them.
  - Examples of operations: +,-,/,*
  - The meaning of these depends on the type

Expressions vs. Statements

Expression

- Represents something
  - Python evaluates it
  - End results is a value
- Examples:
  - 2.3
  - (3*7+2)*0.1

Statements

- Does something
  - Python executes it
  - Need no results in a value
- Examples:
  - print “hello world”
  - import math

Values

- Numbers

  3

  3.14

- Strings

  “hello, world”

  “John”

  “ 3.1415”

Operators & Operands

- +, *, -, /,**
- 1+2 addition
- 2*3 multiplication
- 3**5 power
- 3/5 division

```
>>> 3/5
0
```

?????? What happened? To get float number, use float.

```
>>> 3.0/5
0.6
```

Read more about operators in Python:
http://www.tutorialspoint.com/python/python_basic_operators.htm

Operators on String

```
>>> helloworld = "hello" + " " + "world"
>>> print helloworld
>>> lotsofhellos = "hello" * 10
```

Type: int

- Type int (integer):
  - Values: …. -3, -2, -1, 0, 1, 2, 3, 5, …
    - Integer literals look like this: 1, 45, 4302830 (no commons or periods).
  - Operations: +,-,*,/,** (power)
- Principle: operations on int values must yield an int.
  - Example: 1 /2 round the results to 0
    - Companion operation: % (reminder)
    - 7% 3 evaluates to 1, remainder when 7 dividing by 3

Type: float

- Type float (floating point):
  - Values: approximation of real numbers
    - In Python a number with a "." is a float literal (e.g. 2.0).
    - Without a decimal a number is an int literal (e.g. 2)
  - Operators: +,-,*,/,**
  - Meanings for float are different form int, 1.0/2.0 evaluates to 0.5
- Exponent notation is useful for large (or small) values
  - -222.51e6 is -22.51 * 106 or -2251000

A second kind of **float** literal

Type: float binary fractions

- Floating-point numbers are represent in computer hardware as base 2 (binary) fractions. Example: 1.25 is $5 * 2^{-2}$
- Impossible to write most real numbers this way exactly.
- This approximation results in representation error.
  - Similar to problem of writing 1/3 decimals
  - Python choose the closest binary fraction it can
- When combined in expressions, the error can get worse.
  - Example: type 0.1 + 0.2 at the prompt >>>

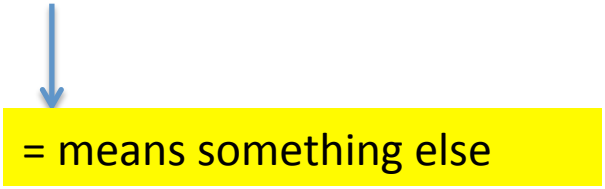
  Read more here:
  https://docs.python.org/2/tutorial/floatingpoint.html

Type: str

- Type String or str:
  - Values: any sequence of characters
  - Operation (s): + (catenation, or concatenation)
- String literal: sequence of characters in quotes
  - Double quotes: "abcex3$<&" or "hello world"
  - Single quotes: 'Hello World!'
- Concatenation can only apply to strings
  - "ab" + "cd" evalutes to "abcd"
  - "ab" + 2 produces an error

Type: bool

- Type boolean or bool:
  - values: True, False
    - Boolean literals are just True or False (have to be capitalized)
  - operations: not, and, or
  - not b: True if b is false and False if b is true.
  - b and c: True if both b and c are true; False otherwise
  - b or c: True if b is true or c is true; False otherwise



Type: bool

- Often come from comparing **int** or **float** numbers
  - Order comparison: i<j, i<=j, i>=j, i>j
  - Equality, inequality: i==j, i!=j,

= means something else

Type: bool

```
>>> bool(1)
True
>>> bool(0)
False
>>> True + 1
2
>>> False + 1
1>>> True and True
True
>>> print 3+2 < 5+7
True
```

Converting values between types

- Basic form: type(value)
  - float(2) converst value 2 to type float (value now 2.0)
  - Int(2.6) converts value 2.6 to type int (value now 2)
- Narrow to wide: bool => int => float
- Widening, Python does this automatically if needed.
  - Example: 1 / 2.0 evaluates to 0.5 (casts 1 to float)
- Narrowing. Python never does this automatically
- Narrowing conversations cause information to be lost
  - Example: float(int(2.6)) evaluates to 2.0

Types

- type(3) int
- type(3.14) float
- type(“hello,world!”)

```
>>> type("hello,world!")
<type 'str'>
>>> type(17)
<type 'int'>
>>> type(100.0)
<type 'float'>
```

Variables

- myString = “bei”
- myString = myString + “xiao”
- print myString

```
>>> myString = 'bei';
>>> myString
'bei'
>>> myString = myString + 'xiao'
>>> mySTring
```

Exercise

- Please yell out the answer of the following expressions WITHOUT using IDLE

- print "Is it greater?", 5 > -2
- print "Is it greater or equal?", 5 >= -2
- print "Is it less or equal?", 5 <= -2
- print 3+2 < 5-7
- print 3 + 2 + 1- 5 -4 % 2-1/4 +6

- Then verify your answers with IDLE

Quiz

- Please write down the answers of the following statements without using Python on a piece of paper and turn your answers in (your full name please):

- 1 == 1 and 1==2
- print 2**4 + 5
- type('100')
- type(4.0/2)
- True and 1== 1
- True or 2 == 1
- print 7 % 2
- print 2 * '**'

- After you turn in your answer, you can verify with IDLE

Exercise

- Generate programs that print the following:

1. What is 3+5? 8
2. What is 5-7? -2

3. Assume that we execute the following assignments:

   w = 17, h = 12, delimiter = '.'

   For each of the following expressions, write the value of the expression and the type:

   1. width/2
   2. width/2.0
   3. height/3
   4. 1 + 2*5
   5. delimiter *5

Next Lecture

- Variables
- More on Boolean operations
- Expressions and statements
- Operator Precedence
- Comments

Take home reading

- Chapter 2
- Number and Math tutorials:
- http://learnpythonthehardway.org/book/ex1.html

- http://learnpythonthehardway.org/book/ex3.html