Intro to Computer Science

Previous

- What is computer science
- Programming languages
- Syntax v. Semantics
- Python!

Readings	
Gaddis	• Chapter 1

Next

- Types
  - Introspection
  - Casting
- Variables
- Operators
- Input
- String methods

Readings	
Gaddis	• Chapter 2 • Chapter 8.3*
Python Std. Lib.	• Section 4.7.1

* "Searching, Manipulating"

From before

- Programming languages have difference aspects
  - High-level/low-level
  - Compiled/interpreted
- Syntax are rules about structure
- Semantics provide meaning
- *Python should be up and running!*

Values and Types

- *Everything* on a computer reduces to numbers
  - Deep down, these are just groups of binary numbers
  - Letters represented with codes (ASCII, Unicode)
  - Pixels are either red (some number), blue (some number) or green (some other number)
- How do programs distinguish these numbers?
- Types
  - An interpretation of the numbers to give them meaning

Value	Type
10	integer
12.32	float
‘Hello’	string

Types and computation

In life

- When programmers talk about their programs as a whole or even aspects of their programs, they talk in terms of types
  - "This variable is a string"
  - "This operation is over integers"
  - "This method returns a float"
- Talking in terms of types will make people understand you

In this class

- Understanding types is imperative to programming
- When deciding how to solve a problem, you should be thinking in terms of types
- I will often pose problems, and frame discussions, in terms of types

Operators

- Types can be combined using *operators*
  - Probably familiar with arithmetic operators
  - Combination of operators and values is known as an *expression*

Value	Type	Expression
10	integer	10 + 34
12.32	float	12.32 * 10.23
‘Hello’	string	‘Hello’ + ‘World!’

- Operators are specific to a given type
  - Addition means something different depending on the type

Numeric Types

int

- Set of integers
  …, -2, -1, 0, 1, 2, …

float

- Fractional values
  Ex: 1.0, 0.5, 0.25, 0.125, …
- Actually approximations of real numbers
- Decimal (.) is important
  - 2.0 is a float
  - 2 is an int (by default)

Type: string (str)

Values

- Any sequence of characters
- String literals come with quotes
  - “Hello World!”
  - ‘This is awesome’
  - “That’s with apostrophes”

Operations

Concatenation	+
Repetition	*

- Concatenation operates on two strings
- Repetition operates on a string and an integer

Operators Have Order

Precedence

Group	Operators			
Exponentiation	**			
Unary	+	-		
Binary arithmetic	*	/	%	
Binary arithmetic	+	-		

- Parenthesis make order explicit
- Order extends to string operators as well!

Let’s discuss…

1. Print the chorus of Queen’s “We Will Rock You” using only two strings
   – Hint: The chorus is
     we will we will
     rock you rock you

Manipulating Types

Introspection

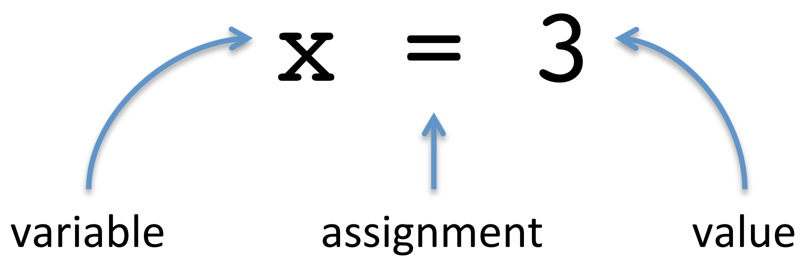
- Examine a type at run-time
  - `type(2)`
  - `type(2.0)`
- Sometimes useful for preventative maintenance

Casting

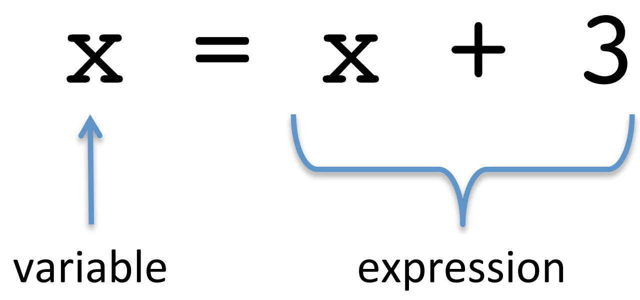
- Convert one type to another
  - `float(2)`
  - `int(2.0)`
  - `str(10)`
- Not all casts are legal
  - `int('Hello')`
- *Narrow* cast loses information
  - `float(int(2.6))`
  - Never done automatically
- *Wide* cast preserves information
  - 1 / 2.0
  - Usually done automatically

Variables

- Variables allow us to save values for later use
  - Technically: the association of a name to a memory location
- Variables are created through assignment
  - Variables must be assigned before they can be used
- Assignment statements can
  - contain expressions
  - reassign variable values



```
x = 3
```



```
x = x + 3
```

Variable Gotchas

- There are restrictions on legal names
  - Python reserves some names for internal use -- program variable names *should* be different
  - Variables names
    - must start with a letter or underscore
    - cannot contain spaces
    - are case-sensitive
- Must be declared before they can be used
  - Assignment is a declaration in Python
  - Be careful!
- Variables implicitly take on the type of their assignment
  - Assigning 2 to x means x is now an int
  - Up to the programmer to keep track!

Let’s discuss…

Does not need to work for negative numbers ☺

1. We saw that

   1 / 2 = 0.5, while
   1 // 2 = 0

   How can we change an expression that uses / so that it produces the same value as // ?
2. Can you think of a program that rounds floating point numbers to the nearest integer? This code should be two lines:
   1. Assign a floating point number to a variable
   2. Print the rounded integer version

Input

- Programs generally require input to do anything useful
- Input also makes programs generic
- Input comes from
  - The command line
  - Devices: network, mouse, keyboard, screen, speaker
  - *Explicit requests during execution*
    - Your program tells the computer to obtain it!

The Python “input” function

- A three-step process:
  1. Prints a string
  2. Waits for input
  3. Returns whatever’s typed
- The returned value is a string!
  – Don’t forget about our discussion on casting

String Methods: The Homestretch

- String manipulation is a large portion of what many researchers do
  1. Remove whitespace
  2. Convert to lower case
  3. Replace one character with another
- Python has a `ton**2` of functions that manipulate strings
- These functions are easy to write, but handy to have already written
  – You'll rewrite a few of these in the future ☺

Calling (all) String Methods

- *String methods* are functions available to values of data-type string
- They are called using *dot notation*:

```
x = “my new string”
print(x.upper())
y = x.upper()
```

Assign the new string

‘upper’ is an expression, so we can print it…

… or save it for later

‘upper’ is the string method

- A new string is returned
  - Methods do not alter the original string!