

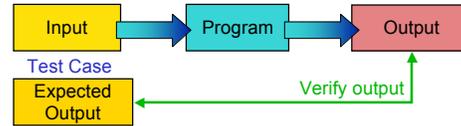
## Objectives

- Wrap up arithmetic
- A few programming tricks
- String operations
- Formatting output

Jan 16, 2008

Sprenkle - CS111

## Review: Testing Process



- Test case: **input** used to test the program, **expected output** given that input
- Verify if **output** is what you expected
- Need good test cases
  - For now, good that you know the “problematic” test cases, even if we don’t know how to address them yet

Jan 16, 2008

Sprenkle - CS111

## Review: Arithmetic Operations

Symbol	Meaning	Associativity
+	Addition	Left
-	Subtraction	Left
*	Multiplication	Left
/	Division	Left
%	Remainder (“mod”)	Left
**	Exponentiation (power)	Right

Precedence rules: P E - DM% AS  
 ↑  
 negation

*Associativity matters when you have the same operation multiple times*

Jan 16, 2008

Sprenkle - CS111

## Review: Two Types of Division

- Float Division: Result is a **float**
    - 3.0/6.0 --> 0.5
    - 6.0/3.0 --> 2.0
    - **At least** one of numerator and denominator must have a decimal, i.e., have type **float**
  - Integer Division: Result is an **int**
    - 3/6 --> 0
    - 6/3 --> 2
    - x/y, if both x and y are **ints**
    - If both numerator and denominator are **ints**, result is **int**
- Not always obvious*

Jan 16, 2008

Sprenkle - CS111

4

## Modulo Operator: %

- Modular Arithmetic: Remainder from division
  - $x \% y$  is the remainder of  $x/y$
  - Read as “x mod y”
- Works only with integers
  - Typically just positive numbers
- Example:  $6 \% 4$ 
  - Read as “six mod four”
  - 6/4 is 1 with a remainder of 2, so  $6\%4$  evaluates to 2

Jan 16, 2008

Sprenkle - CS111

5

## Brainstorm

- What useful thing does  $\% 10$  do?
  - $3 \% 10 =$
  - $51 \% 10 =$
  - $40 \% 10 =$
  - $678 \% 10 =$
  - $12543 \% 10 =$
- What useful thing does  $/10$  do (integer division)?
  - $3/10 =$
  - $51/10 =$
  - $40/10 =$
  - $678/10 =$
  - $12543 / 10 =$
- What useful thing does  $\% 2$  do?

Jan 16, 2008

Sprenkle - CS111

6

## Trick #1: Type Conversion

- You can convert a variable's type
  - Use the type's **constructor**

Conversion Function/Constructor	Example	Value Returned
int(<number or string>)	int(3.77) int("33")	3 33
long(<number or string>)	long(12)	12L
float(<number or string>)	float(22)	22.0
str(<any value>)	str(99)	"99"

Jan 16, 2008

Sprenkle - CS111

7

## Trick #2: Arithmetic Shorthands

- Called **extended assignment operators**
- Increment Operator
  - $x = x + 1$  can be written as  $x += 1$
- Decrement Operator
  - $x = x - 1$  can be written as  $x -= 1$
- Shorthands are similar for  $-$ ,  $*$ ,  $/$  :
  - $x /= 2$
  - amount \*= 1.05

Jan 16, 2008

Sprenkle - CS111

8

## Programming, Testing Practice

- Average three numbers
  - What are good test cases?

average3.py

Jan 16, 2008

Sprenkle - CS111

9

## Programming, Testing Practice

- Average three numbers
  - What are good test cases?
  - How can we make sure we get a floating point number?
    - Two alternatives

average3.py

Jan 16, 2008

Sprenkle - CS111

10

## String Operations

Operand	Syntax	Meaning
+	str1 + str2	Concatenate two strings into one string
*	str * num	Concatenate string <code>num</code> times

- Examples:
  - $str = "I feel " + "sleepy"$
  - $str = "Oops! " * 3$

Jan 16, 2008

Sprenkle - CS111

## Practice

- Given the following code

```
SCALE_MIN = "1"  
SCALE_MAX = "10"
```

```
scale = input( prompt )
```

- Create the string variable `prompt` for the `input` statement so that it prompts the user:

On a scale of 1 to 10, how much do you like Matt Damon?

Jan 16, 2008

Sprenkle - CS111

scale.py

## Strings: `str`

- Used for text
- Indicated by double quotes "" or single quotes ''
  - In general, I'll use double quotes
  - Empty string: "" or ''
- Use triple quotes """ for strings that go across multiple lines

```
"""This string  
is long.  
Like, really long """
```

Jan 16, 2008

Sprengle - CS111

13

## Printing Output

- **print**
  - Simple statement
  - Displays the result of expression(s) to the terminal
- `print "Hello, class"`
  - ← `print` automatically adds a '\n' (carriage return) after it's printed Unless ends with ,
  - string literal
- `print "Your answer is", answer`
  - Use commas to print multiple "things" in one line

Jan 16, 2008

Sprengle - CS111

demo\_str.py

14

## Escape Sequences

- Escape Sequences
  - newline character (carriage return) -> \n
  - tab -> \t
  - quote -> \"
  - backslash -> \\
- Example:
  - `print "To print a \\, you must use \\"`
    - What does this display?

Shell demonstration

Jan 16, 2008

Sprengle - CS111

15

## Practice

- Display To print a tab, you must use '\t'.
- Display I said, "How are you?"

escape\_practice.py

Jan 16, 2008

Sprengle - CS111

16

## Problem with `print`

- By default, `print` puts spaces around numbers when they get printed out
  - Example:

```
x = 13.54  
print "You owe $", x, "."
```

Displays:  
You owe \$ 13.54 .  
Extra spaces

Jan 16, 2008

Sprengle - CS111

## Solution: using `str()`

- Recall: `str()` is constructor/converter function to convert other data types to strings
  - Example: `str(33)` → '33'
- Use constructor with the `+` (i.e., concatenation) operator when printing output
  - `print "You owe $" + str(x) + "."`

Jan 16, 2008

Sprengle - CS111

## Another problem with print

```
SALES_TAX=.05 # the sales tax in VA
value = input("How much does your item cost? ")
tax = value * (1+SALES_TAX)
print "Your item that cost ($", value, ")",
print "costs $", tax, "with tax"
```

Jan 16, 2008

Sprenkle - CS111

sales\_tax.py

## Solution: Format Operator & Specifiers

- Format operator: %
- Format specifiers give control over how output is displayed to user
  - Right, left justification
  - Number of decimals to display

Jan 16, 2008

Sprenkle - CS111

## Formatting Strings

- Syntax is
    - <templatestring> % (<value1>, <value2>, ..., <valueN>)
- Formatting operator
- Replacement values
- Semantics: creates a formatted string
    - Means "format the templatestring, using the format(s) specified by **format specifiers** on the corresponding replacement values"
  - Typically used with print statements

Jan 16, 2008

Sprenkle - CS111

## Formatting Strings

- **templatestring** is a template for the resulting string with format specifiers instead of the values
    - For each format specifier in templatestring, should have a **replacement value**
    - Throws **TypeError** if not enough replacements for specifiers in templatestring
    - If only one replacement value, don't need ()
  - Example:
    - "%0.2f" % 3.14159 result is "3.14"
- One format specifier
- Corresponding replacement value

Jan 16, 2008

Sprenkle - CS111

## Format Specifiers

- General format: %**[flags][width][.precision]**code
- The [] mean "optional"
- flags:
    - 0: zero fills
    - +: adds a + sign before positive values
    - -: left-justification (default is right-justification)
  - width:
    - *Minimum* number of character spaces reserved to display the entire value
    - Includes decimal point, digits before and after the decimal point and the sign

Jan 16, 2008

Sprenkle - CS111

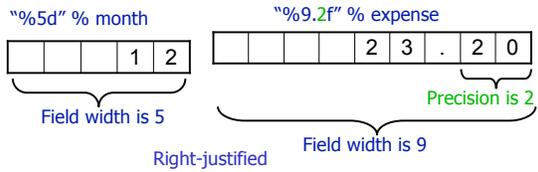
## Format Specifiers

- General format: %**[flags][width][.precision]**code
- precision:
    - Number of digits after the decimal point for **real** values
  - code:
    - Indicates the value's **type**/way to format
      - s - string
      - d (or i) - integer
      - f - floating point
      - e - floating point with exponent

Jan 16, 2008

Sprenkle - CS111

## Example Format Specifiers

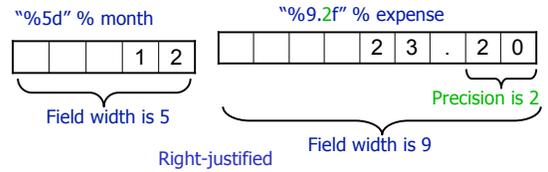


- What if precision is bigger than the decimal places?
- What if field width is smaller than the length of the value?

Jan 16, 2008

Sprengle - CS111

## Example Format Specifiers

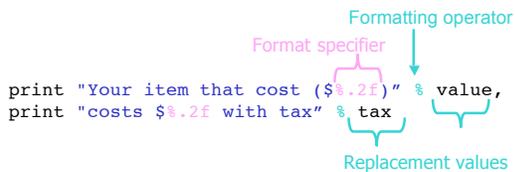


- What if precision is bigger than the decimal places?
  - > Fills decimal with 0s
- What if field width is smaller than the length of the value?
  - > String contains entire value

Jan 16, 2008

Sprengle - CS111

## Example using Format Operator



Jan 16, 2008

Sprengle - CS111

## Formatting Practice

- x=10
- y = 3.5
- z = "apple"
- "%6.2d" % x
- "%6.2f" % x
- "%06.2f" % y
- "%+6.2f" % y
- "%-10s" % z
- "%5d %-7.3f" % (x,y)

Jan 16, 2008

Sprengle - CS111