Objectives
• Usability
• Problems with print
• Built-in functions
• Import statements
• Puzzles from Cyberspace

Review from Last Time
• Type conversion
  ➢ Use type’s constructor
• Shorthands, such as x+=1
• String operators
  ➢ +: concatenate strings together
  ➢ *: concatenate the string n number of times
• Escape Sequences
  ➢ Example: ‘\t’ → prints a tab

Usability
• Want users to want to use your software
  ➢ More revenue
  ➢ Develop even better software
• Beyond functionality, what qualities do you like in software?
  ➢ Web included

Usability Goals
• Pretty output, display, layout
  ➢ Easy to read, understand, interpret
• Clear navigation
• Easy to perform frequent tasks
• Undoability
• Difficult to make irrecoverable errors

Problem with print
• By default, print puts spaces around numbers when they get printed out
  ➢ Example:
    \[ x = 13.54 \]
    \[ \text{print "You owe ", x, ", \".\"} \]
    \[ \text{Displays: You owe $ 13.54 .} \]
    \[ \text{Extra spaces} \]
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) + “.”

Another problem with print

sales_tax.py

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

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

Formatting Strings

- Syntax is
  - `<templatestring> % (<value1>, <value2>, ..., <valueN>)`
  - 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

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:
  - “%.2f” % 3.14159 result is “3.14”
- One format specifier
  - Corresponding replacement value
• 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

Field width is 5

Field width is 9

Right-justified

Field width is 9

Right-justified

• General format: `%[flags][width][.precision]code`
  ➢ flags: [] mean "optional"
    • 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
  ➢ 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

Example using Format Operator

```
print "Your item that costs\n\nprint "costs $%.2f with tax" % tax
```

Example using Format Operator

```
print "Your item that cost ($%.2f)" % value,
print "costs $%.2f with tax" % tax
```

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)
Example: Printing Out Tables

• A table of temperature conversions

<table>
<thead>
<tr>
<th>Temp F</th>
<th>Temp C</th>
<th>Temp K</th>
</tr>
</thead>
<tbody>
<tr>
<td>-459.7</td>
<td>-273.1</td>
<td>0.0</td>
</tr>
<tr>
<td>0.0</td>
<td>-17.8</td>
<td>255.4</td>
</tr>
<tr>
<td>32.0</td>
<td>0.0</td>
<td>273.1</td>
</tr>
</tbody>
</table>

• If we want to print data in rows, what is the template for what a row looks like?
  ➢ How do we make the column labels line up?

Broader Issues

• Let’s say I have 19 people in a class
• I want 4 groups of approximately equal size
• How many people will be in each group?

Four Puzzles in Cyberspace

• Context: Book Code v2 by Lawrence Lessig
• You read Chapter 2
  ➢ Presents the problems, not the author’s proposed solutions

Groups:
- Aaron
- Benjamin
- Thomas
- Peter
- Kevin
- Michele
- Craig
- Carrie
- Mallory
- Dylan
- Camille
- Russ
- Taylor
- Sara
- David
- Michael
- Chen
- Greg
- Charles

Four Puzzles from Cyberspace

• How many of you knew about MMOGs before reading this article?
  ➢ How may of you participated in this or something similar?
• Every advancement in technology has positive and negative effects
  ➢ What are the positive and negative effects of email? IM? Facebook?
  ➢ What are the opportunities from Cyberspace in your field?
  ➢ What are the challenges caused by Cyberspace in your field?