Objectives

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

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
• Precedence rules: P E - DM% AS
• Example: 6 % 4
  ➢ Read as “six mod four”
  ➢ 6/4 is 1 with a remainder of 2, so 6%4 evaluates to 2

Modulo Practice

• 7 % 2 =
• 3 % 6 =
• 6 % 2 =
• 7 % 14 =
• 14 % 7 =
• 6 % 0 =

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?

Trick #1: Type Conversion

• You can convert a variable’s type
  ➢ Use the type’s constructor

<table>
<thead>
<tr>
<th>Conversion Function/ Constructor</th>
<th>Example</th>
<th>Value Returned</th>
</tr>
</thead>
<tbody>
<tr>
<td>int(&lt;number or string&gt;)</td>
<td>int(3.77)</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>int(&quot;33&quot;)</td>
<td>33</td>
</tr>
<tr>
<td>long(&lt;number or string&gt;)</td>
<td>long(12)</td>
<td>12L</td>
</tr>
<tr>
<td>float(&lt;number or string&gt;)</td>
<td>float(22)</td>
<td>22.0</td>
</tr>
<tr>
<td>str(&lt;any value&gt;)</td>
<td>str(99)</td>
<td>&quot;99&quot;</td>
</tr>
</tbody>
</table>

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
Programming, Testing Practice

- Average three numbers
  - What are good test cases?

  `average3.py`

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

  `average3.py`

String Operations

<table>
<thead>
<tr>
<th>Operand</th>
<th>Syntax</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>+</td>
<td>str1 + str2</td>
<td>Concatenate two strings into one string</td>
</tr>
<tr>
<td>*</td>
<td>str * num</td>
<td>Concatenate string num times</td>
</tr>
</tbody>
</table>

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

Practice

- Given the following code:
  ```python
  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?
  ```

Use of Constants

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

scale = input( prompt )

- How difficult is it to modify the code to change the scale?
- What if I the min and max were “hard-coded” into the prompt?

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

```python
scale.py
```
Escape Sequences

• Escape Sequences
  ➢ newline character (carriage return) -> 
  ➢ tab -> \t
  ➢ quote -> \\
  ➢ backslash -> \\

• Example:
  ➢ print "To print a \, you must use "\\\\""
  ➢ What does this display?

Practice

• Display To print a tab, you must use \t.
• Display I said, "How are you?"

For This Week

• Lab 1: Due Friday by classtime
• Broader Issues: Four Puzzles from Cyberspace
  ➢ Through “Jake’s Communities”
  ➢ Posted on Sakai by 10 a.m. on Friday
• Office Hours
  ➢ Thursday: 2:30-3
• Next Monday’s Class
  ➢ 2:05-2:50 p.m.