CSc 110, Autumn 2016

Lecture 4: The for Loop

Adapted from slides by Marty Stepp and Stuart Reges
Repetition with \texttt{for} loops

• So far, repeating an action results in redundant code:

```python
makeBatter()
bakeCookies()
bakeCookies()
bakeCookies()
bakeCookies()
bakeCookies()
bakeCookies()
```

• Python's \texttt{for} loop statement performs a task many times.

```python
mixBatter()
for i in range(1, 6):    # repeat 5 times
    bakeCookies()

frostCookies()
```
for loop syntax

for variable in range (start, stop):
    statement
    statement
    ...
    statement

• Set the variable equal to the start value
• Repeat the following:
  • Check if the variable is less than the stop. If not, stop.
  • Execute the statements.
  • Increase the variable's value by 1.
Control structures

- **Control structure**: a programming construct that affects the flow of a program's execution

- Controlled code may include one or more statements

- The for loop is an example of a looping control structure
Repetition over a range

```python
print("1 squared = " + str(1 * 1))
print("2 squared = " + str(2 * 2))
print("3 squared = " + str(3 * 3))
print("4 squared = " + str(4 * 4))
print("5 squared = " + str(5 * 5))
print("6 squared = " + str(6 * 6))
```

- Intuition: "I want to print a line for each number from 1 to 6"

- The `for` loop does exactly that!

```python
for i in range(1, 7):
    print(str(i) + " squared = " + str(i * i));
```

- "For each integer i from 1 through 6, print ..."
Loop walkthrough

for i in range(1, 5):
    print(str(i) + " squared = " + str(i * i))

print("Whoo!")

Output:

1 squared = 1
2 squared = 4
3 squared = 9
4 squared = 16
Whoo!
Multi-line loop body

```python
print("+-----+")
for i in range(1, 4):
    print("\n   /")
    print("/   ")
print("+-----+")
```

• Output:
```
+-----+
 \   /
  \  /
   \ /
    \ 
     
     
     
     
+-----+
```
Expressions for counter

```python
highTemp = 5
for i in range(-3, high_temp // 2 + 1):
    print(i * 1.8 + 32)
```

- Output:
  26.6
  28.4
  30.2
  32.0
  33.8
  35.6
Rocket Exercise

• Write a method that produces the following output:

  T-minus 10, 9, 8, 7, 6, 5, 4, 3, 2, 1,
  blastoff!
  The end.
```
print(' ', end='')

• Adding , end=' ' allows you to print without moving to the next line
  • allows you to print partial messages on the same line

highTemp = 5
for i in range(-3, int(highTemp / 2 + 1)):
    print(i * 1.8 + 32, end=' ')

• Output:
  26.6  28.4  30.2  32.0  33.8  35.6
  • Either concatenate ' ' to separate the numbers or set end=' '
Changing step size

• Add a third number to the end of range, this is the step size
  • A negative number will count down instead of up

```python
print("T-minus ")
for i in range(10, 0, -1):
    print(str(i) + ', ', end=''")
print("blastoff!")
print("The end.")
```

• Output:

  T-minus 10, 9, 8, 7, 6, 5, 4, 3, 2, 1, blastoff!
The end.
Nested loops

• **nested loop**: A loop placed inside another loop.

```python
for i in range(1, 6):
    for j in range(1, 11):
        print("*", end="")
    print()  # to end the line
```

• Output:

```
**********
**********
**********
**********
**********
```

• The outer loop repeats 5 times; the inner one 10 times.
  • "sets and reps" exercise analogy
Nested for loop exercise

• What is the output of the following nested for loops?

```python
for i in range(1, 6):
    for j in range(1, i + 1):
        print("*", end="")
    print()
```

• Output:

```
* 
** 
*** 
**** 
*****
```
Nested for loop exercise

• What is the output of the following nested for loops?

```python
for i in range(1, 6):
    for j in range(1, i + 1):
        print(i, end="")
    print()
```

• Output:

```
1
22
333
4444
55555
```
Complex lines

• What nested for loops produce the following output?

```
....1
...2
..3
 .4
5
```

inner loop (repeated characters on each line)

outer loop (loops 5 times because there are 5 lines)

• We must build multiple complex lines of output using:
  • an outer "vertical" loop for each of the lines
  • inner "horizontal" loop(s) for the patterns within each line
Outer and inner loop

• First write the outer loop, from 1 to the number of lines.
  
  ```python
  for line in range(1, 6):
      ...
  ```

• Now look at the line contents. Each line has a pattern:
  • some dots (0 dots on the last line), then a number

  ....1
  ...2
  ..3
  .4
  5

  • Observation: the number of dots is related to the line number.
Mapping loops to numbers

for count in range(1, 6):
    print(...)

• What statement in the body would cause the loop to print:
  4 7 10 13 16

for count in range(1, 6):
    print(3 * count + 1, end=' ');
Loop tables

for count in range(1, 6):
    print(…)

• What statement in the body would cause the loop to print:
  2 7 12 17 22

• To see patterns, make a table of count and the numbers.
  • Each time count goes up by 1, the number should go up by 5.
  • But count * 5 is too great by 3, so we subtract 3.

<table>
<thead>
<tr>
<th>count</th>
<th>number to print</th>
<th>5 * count</th>
<th>5 * count - 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>5</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>7</td>
<td>10</td>
<td>7</td>
</tr>
<tr>
<td>3</td>
<td>12</td>
<td>15</td>
<td>12</td>
</tr>
<tr>
<td>4</td>
<td>17</td>
<td>20</td>
<td>17</td>
</tr>
<tr>
<td>5</td>
<td>22</td>
<td>25</td>
<td>22</td>
</tr>
</tbody>
</table>
Loop tables question

• What statement in the body would cause the loop to print:
  17 13 9 5 1

• Let's create the loop table together.
  • Each time count goes up 1, the number printed should ...
  • But this multiple is off by a margin of ...

<table>
<thead>
<tr>
<th>count</th>
<th>number to print</th>
<th>-4 * count</th>
<th>-4 * count + 21</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>17</td>
<td>-4</td>
<td>17</td>
</tr>
<tr>
<td>2</td>
<td>13</td>
<td>-8</td>
<td>13</td>
</tr>
<tr>
<td>3</td>
<td>9</td>
<td>-12</td>
<td>9</td>
</tr>
<tr>
<td>4</td>
<td>5</td>
<td>-16</td>
<td>5</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>-20</td>
<td>1</td>
</tr>
</tbody>
</table>
Another view: Slope-intercept

• The next three slides present the mathematical basis for the loop tables. Feel free to skip it.

<table>
<thead>
<tr>
<th>count (x)</th>
<th>number to print (y)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>7</td>
</tr>
<tr>
<td>3</td>
<td>12</td>
</tr>
<tr>
<td>4</td>
<td>17</td>
</tr>
<tr>
<td>5</td>
<td>22</td>
</tr>
</tbody>
</table>
Another view: Slope-intercept

- **Caution:** This is algebra, not assignment!

- Recall: slope-intercept form ($y = mx + b$)

- Slope is defined as “rise over run” (i.e., rise / run). Since the “run” is always 1 (we increment along $x$ by 1), we just need to look at the “rise”. The rise is the difference between the $y$ values. Thus, the slope ($m$) is the difference between $y$ values; in this case, it is +5.

- To compute the $y$-intercept ($b$), plug in the value of $y$ at $x = 1$ and solve for $b$. In this case, $y = 2$.

  \[
  y = m \times x + b \\
  2 = 5 \times 1 + b \\
  \text{Then } b = -3
  \]

- So the equation is

  \[
  y = m \times x + b \\
  y = 5 \times x - 3 \\
  y = 5 \times \text{count} - 3
  \]

<table>
<thead>
<tr>
<th>count (x)</th>
<th>number to print (y)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>7</td>
</tr>
<tr>
<td>3</td>
<td>12</td>
</tr>
<tr>
<td>4</td>
<td>17</td>
</tr>
<tr>
<td>5</td>
<td>22</td>
</tr>
</tbody>
</table>
Another view: Slope-intercept

- Algebraically, if we always take the value of $y$ at $x = 1$, then we can solve for $b$ as follows:
  \[
  y = m \times x + b \\
  y_1 = m \times 1 + b \\
  y_1 = m + b \\
  b = y_1 - m
  \]

- In other words, to get the $y$-intercept, just subtract the slope from the first $y$ value ($b = 2 - 5 = -3$)
  - This gets us the equation
    \[
    y = m \times x + b \\
    y = 5 \times x - 3 \\
    y = 5 \times \text{count} - 3
    \]
    (which is exactly the equation from the previous slides)
Nested for loop exercise

• Make a table to represent any patterns on each line.

<table>
<thead>
<tr>
<th>line</th>
<th># of dots</th>
<th>-1 * line</th>
<th>-1 * line + 5</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>4</td>
<td>-1</td>
<td>4</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>-2</td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>-3</td>
<td>2</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>-4</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>-5</td>
<td>0</td>
</tr>
</tbody>
</table>

• To print a character multiple times, use a for loop.

```python
for j in range(1, 5):
    print("."))        # 4 dots
```
Nested for loop solution

• Answer:
  ```python
  for line in range(1, 6):
    for j in range(1, (-1 * line + 5 + 1)):
      print(".", end='')
    print(line)
  ```

• Output:
  ```
  ....1
  ...2
  ..3
  .4
  5
  ```
Nested for loop exercise

• What is the output of the following nested for loops?
  ```python
  for line in range(1, 6):
      for j in range(1, -1 * line + 6):
          print(".", end='')
     for k in range(1, line):
          print(line, end='')
  print()
  ```

• Answer:
  ```
  ....1
  ...22
  ..333
  .4444
  55555
  ```
Nested for loop exercise

• Modify the previous code to produce this output:
  ....1
  ...2.
  ..3..
  .4...
  5....

• Answer:
  ```python
  for line in range(1,6):
      for j in range(1, -1 * line + 6):
          print(".", end=''
      print(line, end=''
      for j in range(1,line):
          print(".", end=''
      print()
  ```
Modify-and-assign operators

*shortcuts to modify a variable's value*

<table>
<thead>
<tr>
<th>Shorthand</th>
<th>Equivalent longer version</th>
</tr>
</thead>
<tbody>
<tr>
<td>variable += value;</td>
<td>variable = variable + value;</td>
</tr>
<tr>
<td>variable -= value;</td>
<td>variable = variable - value;</td>
</tr>
<tr>
<td>variable *= value;</td>
<td>variable = variable * value;</td>
</tr>
<tr>
<td>variable /= value;</td>
<td>variable = variable / value;</td>
</tr>
<tr>
<td>variable %= value;</td>
<td>variable = variable % value;</td>
</tr>
</tbody>
</table>

x += 3;   # x = x + 3;
gpa -= 0.5;   # gpa = gpa - 0.5;
nnumber *= 2;  # number = number * 2;