Chapter 4:
For Loops, Strings and Tuples
Objectives

In this chapter, you will learn how to:

- Construct *for* loops to move through a sequence
- Use the *range()* function to create a sequence of numbers
- Treat strings as sequences
- Use tuples to harness the power of sequences
- Use sequence functions and operators
- Index and slice sequences
Using **for** Loops

**For loop**

- Repeats part of a program based on a *sequence*
  - Repeats the loop body for each element of the sequence
  - Loop ends when it reaches end of the sequence
  - Not based on a condition
- *Sequence*

**An ordered list of things**

- *E.g.*, {1,2,3,4,5,6,7,8,9,10}, “Computers”, “hello”
for Loop Structure

for <var> in <seq>:
    Statement #1
    Statement #2

Assign the first element (character) in <seq> to <var> then execute statements 1 & 2
Assign the next character to in <seq> to <var> then execute statements 1 & 2
Repeat the above steps until you exhaust all characters in <seq>

- <var> ~ loop variable for each element in <seq>
- Statement #1, Statement #2 ~ for loop body
  - Observe the colon (:)  
  - Observe the mandatory indentation for the for loop block
Understanding for Loops

Python Simulation

Loopy strings Python Program

- Review program on the CD
Counting with a *for* Loop

**`range()`**

- **Function call:** `range( n )`
  - Generates list of numbers “n” numbers starting from 0
    - `range(6)`
      - `[0, 1, 2, 3, 4, 5]`
    - `range(9)`
      - `[0, 1, 2, 3, 4, 5, 6, 7, 8]`

- **Counter Program Simulation**
  - Counting forward
Counting with a *for* Loop

Step Forward

- Function call: `range( istart, iend, istep )`
  - Generates list of numbers:
    - Starting from *istart*
    - Ending at *iend -1*
    - In steps of *istep*

- `range(0, 60, 10)`
  - [0, 10, 20, 30, 40, 50]
- `range (10, 30, 5)`
  - [10, 15, 20, 25]

- Review Counter Program Simulation on CD
Counting with a *for* Loop

Step Backwards

- Function call: `range( istart, iend, -istep )`
  - Generates list of numbers:
    - Starting from `istart`
    - Ending at `iend -1; where istart > iend`
    - In steps of `-istep`
  - `range(10, 0, -1)`
    - `[10, 9, 8, 7, 6, 5, 4, 3, 2, 1]`
  - `range (30, 10, -5)`
    - `[30,25, 20, 15]`
Indexing Strings

How do you access the individual characters in \textit{Word}?

- **Sequential Access**
  - Construct a \textit{for} loop and go thru’ One character at a time

- **Random Access**
  - Construct a \textit{for} loop and use the function \textit{randrange()} to generate valid random indices for the string
Indexing Strings

- How do you access the letters in variable: `word`?

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>i</td>
<td>n</td>
<td>d</td>
<td>e</td>
<td>x</td>
</tr>
</tbody>
</table>


Demo CD Programs

Python Simulation

- String Immutability
- Creating new strings from old string
- Slice (range)
- Creating Tuples
String Indexing

```python
word = "game"
print word[0]
print word[1]
print word[-4]
raw_input("\nPress enter to exit")
```

Program Output:
```
g
a
g```

Indexing: Restricted to one element at a time
String Immutability

- Cannot Change element(s) of a string
  - `word = “game”`
  - `word[0] = “L”`  
  
  This is an illegal assignment!

```
word = "game"
word[0] = "L"
```
Creating New Strings from an Existing String

- Create a new string without vowels
  - Prompt user to enter a new string (message)
  - Check for any vowels in the sequence
  - Create new string:
    - Concatenate non-vowel characters in the sequence

- Python CD Program Simulation:
  - no_vowel.py
Slicing Strings

- **Slice**
  - A set of contiguous elements in a string
  - `word = “pizza”`
    - Slice ~ `word[Start Position: End Position]`
    - `word[0:5]`, `word[1:3]`, `word[-4:-2]`, `word[-4:3]`
Creating Slice

- Slice Program Simulation
  - Prompt user to enter begin and end positions of slice
  - Convert the user input, string, to integer value
  - Reference the slice and print the contiguous elements

- More on Slice:
  - word[0:], word[:5], word[:]

Tuple

Objectives

- Creating a Tuple
- Printing Tuple Elements
- Using the `len()` function with Tuples
- Indexing Tuples
- Slicing Tuples
- Tuple Immutability
- Concatenating Tuples
**Tuples**

- **Tuple:**
  - Sequence of different types of elements
- Let’s create a tuple: `inventory`:
  
  - `inventory = ("sword", 4, 2.5)`

- **Empty tuple:**
  - `inventory = ( )`
Tuple
Printing Tuple elements

- Print elements in tuple

- inventory = ("sword",
               4,
               2.5)

- for item in inventory:
  - print item

- raw_input ("\n\nPress the enter key to exit.")
Tuple

Use the \texttt{len()} Function with Tuple

- Get length of a Tuple:
- \texttt{len (inventory)}
  - returns number of elements in the tuple variable, \textit{inventory}

- print “You have”, len(inventory), “ in your possession”
Tuple

Indexing Tuples

print inventory[0]
print inventory[1]
print inventory[-1]

raw_input("\nPress enter to exit")

Program Output:
sword
4
2.5

Indexing:
Restricted to
one element at a time
**Tuple Slicing**

Slice inventory:

inventory\[Start Position: End Position]\n
inventory[0:3] → sword4
inventory[1:3] → 42.5
inventory[-3:-1] → sword4
*Cannot* Change element(s) of a Tuple

inventory[0] = “helmet”

- This is an illegal assignment!
inventory = ("sword", 4, 2.5)
jewelry = ("gold", "gem")
print "inventory is: " inventory
print "jewelry is " jewelry
#My new collection
inventory = inventory + jewelry
print inventory
raw_input ("\n\nPress enter key to exit")

Tuple
Concatenate

Concatenates 2 Tuples