Objectives

• Wrap up lists, dictionaries
• Default parameters
• Writing documentation
• Broader Issue: Digital Humanities

Creating Dictionaries in Python

Syntax:
{<key>:<value>, ..., <key>:<value>}

empty = {}

Accessing Values Using Keys

• Typically, you will check if dictionary has a key before trying to access the key
  if 'z' in ascii:
    value = ascii['z']

• Or handle if get default back
  val = ascii.get('z')
  if val is None:
    # do something ...

Defaults for Parameters

• Saw with the xrange function
  – Didn’t have to specify start or increment when calling the function
  – Default start = 0
  – Default increment = 1
• Can assign a default value to a parameter
  – In general, in function header, default parameter(s) should come after all the parameters that need to be defined

Using Default Parameters

• By default, the rollDie function could assume that a die has 6 sides
  def rollDie(sides=6):
    return random.randint(1,sides)

Examples of calling the function:
rollDie(6)
rollDie()  # Show help
rollDie(12)
Problem: Student Majors
• We want to keep track of the number of majors of each type
  ➢ Twist: Not every student has a major (don’t declare until sophomore year)

Problem: Student Majors, revised
• Students can have more than one major
  ➢ Should count these separately
• How can we modify the previous program to do that?

Lists vs. Dictionaries
<table>
<thead>
<tr>
<th>Lists</th>
<th>Dictionaries</th>
</tr>
</thead>
<tbody>
<tr>
<td>integer positions (0,...) to any type of value</td>
<td>Map immutable keys (int, float, string) to any type of value</td>
</tr>
<tr>
<td>Ordered</td>
<td>Unordered</td>
</tr>
<tr>
<td>Slower to find a value (in)</td>
<td>Fast to find a value (use key)</td>
</tr>
<tr>
<td>Fast to print in order</td>
<td>Slower to print in order (by key)</td>
</tr>
<tr>
<td>Only as big as you make it</td>
<td>Takes up a lot of space (so can add elements in the middle)</td>
</tr>
</tbody>
</table>

Getting Documentation
• dir: function that returns a list of methods and attributes in an object
  ➢ dir(<type>)
• help: get documentation
  ➢ In the Python shell
    ➢ help(<type>)
    ➢ import <modulename>
    ➢ help(<modulename>)

Where is Documentation Coming From?
• Comes from the code itself in “doc strings”
  ➢ i.e., “documentation strings”
• Doc strings are simply strings after the function header
  ➢ Typically use triple-quoted strings because documentation goes across several lines

```python
def verse(animal, sound):
    """ prints a verse of Old MacDonald, filling in the strings for animal and sound """
```

Digital Humanities
• Process large amounts of information to learn new information
  ➢ Not necessarily numeric information
    ➢ Text-based, images, multi-media, etc.
• Area growing in importance
  | Aaron | Charles | Greg | Russ |
  | Ben   | Chen    | Kevin| Sara |
  | Camille | David  | Mallory | Taylor |
  | Carrie | Dylan  | Mike | Thomas |
Learning Algorithms

• Train on data
  ➢ This is an example of \( x \)
  ➢ This is not an example of \( x \)

• Critical question: how to model/characterize info

Analyzing Shakespeare

• hyphenated compound words
• Relative clauses per thousand
  ➢ Shakespeare used less often
• Grade-level of writing, as measured by word- and sentence-length
• Percentage of open- and feminine-ended lines
  ➢ Open: no punctuation
  ➢ Feminine: unstressed syllable
  ➢ Shakespeare used more often

Discussion

• How did the researchers characterize a metaphor?
  ➢ What are some other ways to characterize it?
  ➢ How accurate can we get?

• How did computing change the “game” of the metaphoric field?
  ➢ With this new information, what follow up questions can be answered?

• What other gains do you see possible by applying computing to the humanities?
• Will digital humanities take the “art” out of the humanities?

Relation to Class

• Analyzing texts
  ➢ Strings
• Files containing the data

• Natural Language Processing
  ➢ Growing field of computer science
    ➢ E.g., Speech recognition
  ➢ Process sentences
    ➢ Determine subject/verb/direct object, etc.