Case Study:

Data Structure Selection

Chapter 13

Data Structure

- **Data structure:** is about how data is organized
- **Algorithm:** step by step operations to reach a goal

  Data structures + Algorithms = Problem Solution

- Good choice produces good results...

`string` Module

- `string` module has some useful stuff:

```
>>> import string

>>> string.punctuation
'!"#$%&\'()*+,-./:;<=>?@[\\]^_`{|}~'

>>> string.whitespace
' \t\n\r\x0b\x0c'
```

- What about numbers?

Random Numbers

- `random` module provides functions that generate **pseudorandom** numbers.

```
import random

for i in range(10):
    x = random.random()
    # returns random float between 0.0 and 1.0 (including 0.0 but not 1.0)
    print(x)
```

Random Numbers (cont.)

- The function `randint` takes parameters `low` and `high` and returns an integer between `low` and `high` (including both):

```
>>> random.randint(5, 10)
5

>>> random.randint(5, 10)
9
```

Random Numbers (cont.2)

- To choose an element from a sequence at random, you can use `choice`:

```
>>> t = [1, 2, 3]
>>> random.choice(t)
2
>>> random.choice(t)
3
```

Keyword Arguments

```
>>> t = ('a' , 'b' , 'c' , 'd' , 'e')
>>> t.sort()

>>> t.sort(reverse=True)

sorted?

for freq, word in t[:10]:
    print(word, freq, sep='\t')

t.sort(key=?)
```

Optional Parameters

```
>>> s='banana'

>>> s.find('a')

>>> s.find('a',2)

>>> s.find('a',2,3)
```

Optional Parameters (cont.)

```
def print_most _common(hist, num=10):
    t = most_common(hist)
    print(' The most common words are: ' )
    for freq, word in t[:num]:
        print(word, freq, sep='\t' )

print_most_common(hist)
print_most_common(hist, 20)
```

- If a function has both required and optional parameters, all the required parameters have to come first, followed by the optional ones

Debugging

- Reading
- Running
- Ruminating
- Rubberducking
- Retreating