Intro to Computing
ES112

Lecture 9

Chap 14,15 from book

Simple dictionary problem

- Write a function that given a list, returns a dictionary, that has keys the elements and values the number of times an element appears
- save it in a file hist.py

Creating modules

- As you have used the math module, you can create your own module
- You can make a module out of hist.py

- Just do import hist
  - You might have issues if there are statements outside functions

- If you want hist.py to be executable by itself, then add these lines in hist.py:

```
if __name__ == "__main__":
    # write down whatever you want to execute
```

Filenames and paths

- The os module provides access to functions to work with files and directories

```
>>> import os
>>> cwd = os.getcwd()
>>> print cwd
/home/anirban
>>> os.path.abspath('memo.txt')
'/home/dinsdale/memo.txt'
```

`os.getwd()`	Current working directory
`os.path.exists(..)`	Tells you whether a file exists
`os.path.abspath(..)`	Converts a relative path to absolute
`os.path.isfile(...)`	Whether a name is a file
`os.path.isdir(...)`	Whether directory
`os.listdir(..)`	Lists all files and directories inside the directory
`os.path.join(dir, file)`	Takes a directory name and a file and joins them to create a complete path

Sample program to traverse dirs

- Write a program that given a directory lists the names of all files in it and in all its subdirectories

- Hint: try out the functions in the interactive mode of python before writing

Sample program to traverse dirs

- Write a program that given a directory lists the names of all files in it and in all its subdirectories

```
def walk(dirname):
    for name in os.listdir(dirname):
        path = os.path.join(dirname, name)

        if os.path.isfile(path):
            print path
        else:
            walk(path)
```

Exceptions

- Sometimes, when you commit an error in code, this is the kind of message that you see

```
>>> fin = open('bad_file')
IOError: [Errno 2] No such file or directory: 'bad_file'

>>> d = {}
>>> print d[1]
Line 1: KeyError: 1
```

- In such situations, it is possible to pre-empt the error inside code and do something to rectify it

```
d = {}
try:
  # this code is tried first
  print d[1]
  # this line is not evaluated if above line gives error
  X = d[1] + "something"
except:
  # first add a key 1
  d[1] = "some value"

print d[1]
```

Exceptions

- It is also possible to be specific about which exceptions you are going to catch
  - ValueError, KeyError, IOError,...

```
d = {}
try:
  # this code is tried first
  print d[1]
  # this line is not evaluated if above line gives error
  X = d[1] + "something"
except KeyError:
  # first add a key 1
  d[1] = "some value"

print d[1]
```

If we specified some other exception it would not catch it

Common example: adding to dictionary

- Suppose we have a list of words, and we want to create its histogram using a dictionary. Write three variants of this function
  - Using if-else
  - Using get()
  - Using try-exception

- Which do you think is preferable, and why?

Creating data structures: example 1

- We will write a program that tries to tries to see the difference in writing between two authors. Our dataset will be two works of detective fiction: You can download the books from there:
  - http://www.gutenberg.org/cache/epub/583/pg583.txt
  - http://www.gutenberg.org/cache/epub/2852/pg2852.txt
- After you download the books, write a program that:
  - Reads the files
  - Replaces all punctuation by space
  - Converts all words into lowercase
- Print also
  - the number of distinct words used in the book.
  - Which author uses the most extensive vocabulary?

Dictionary operations

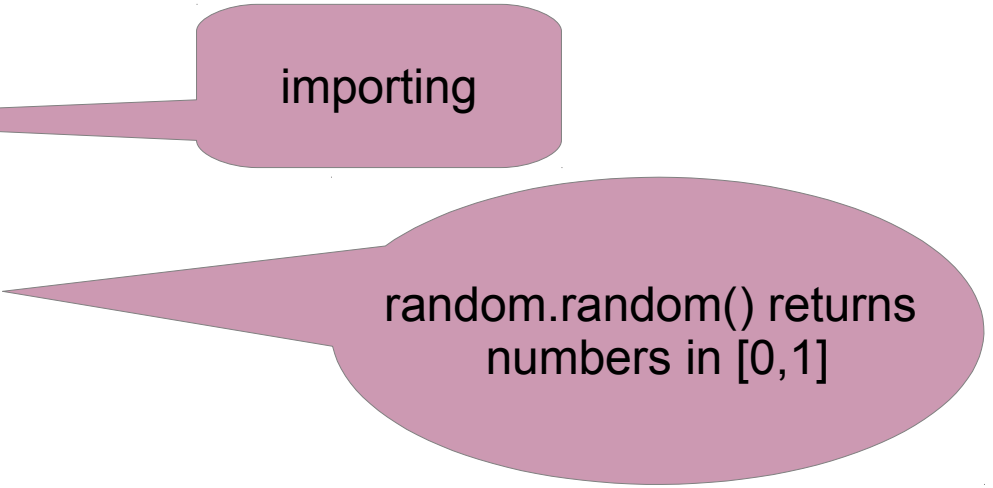
- Builds a dictionary of words for each book, keys are words and values are the counts
- Find 20 words in the first book which are not present in the second book and vice versa?
- Finds out the top 20 words in each book--- how?

Random numbers

- Sometimes, it is useful to generate random numbers
- The module random will be useful
- To generate few random numbers

```
import random

for i in range(10):
    x = random.random()
    print x
```



Other random methods

- `random.choice(list)` : Given a list it chooses a random element of this list
- `random.randint(a, b)` : Returns a random integer in the range a to b – 1

- Could you have done these using only `random.random()`?

Analysis

- From each dictionary, pick words proportional to their frequency – how?

Analysis

- From each dictionary, pick words proportional to their frequency
  - Create a list that contains all words, each being repeated the number of times it appeared in dictionary
  - Use random.choice()

Further analysis

- Try to solve exercise 8 in chapter 13
  - Can be made into a project (needs more work)

Projects

- List of ideas will be shared
- Overall 5% marks, additional 5% bonus if a very good project
- Can work in a group of upto 3 people
- Needs to be
  - Demo
  - Short 1-2 page description of what you did

Classes

- A class is a user defined type
- Suppose we want to do some geometry in 2D. We need to define points

```
class Point(object):
    """Represents a point in 2-D space."""
```

- You have a data type called Point

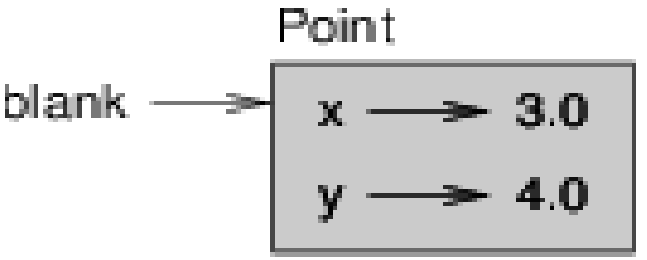
```
>>> blank = Point()
>>> print blank
<__main__.Point instance at 0xb7e9d3ac>
```

- We can now already use this to store more attributes

```
>>> blank.x = 3.0
>>> blank.y = 4.0
```

Attributes

- We first created a class (Point) and then an object from the class (blank)
- Once you have assigned attributes of an object, you can access them and change them
  - Objects are mutable




```
>>> print blank.y
4.0
>>> blank.x = 3.0
```

  - Can pass to functions

```
def print_point(p):
    print '(%f, %f)' % (p.x, p.y)
```

Exercise

Write a function called `distance_between_points` that takes two `Points` as arguments and returns the distance between them.

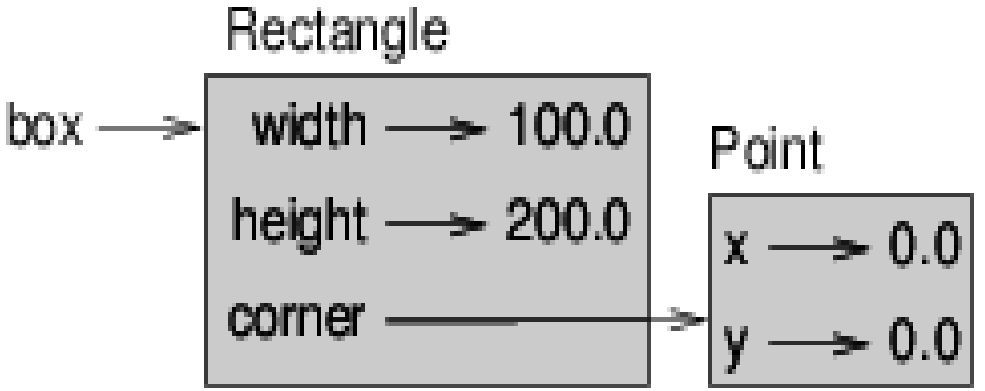
More classes

- Suppose now, we want to create a class for a rectangle
- Should have a
  - Length, width and the left hand corner point
-

```
class Rectangle(object):
    """Represents a rectangle.

    attributes: width, height, corner.
    """
```

```
box = Rectangle()
box.width = 100.0
box.height = 200.0
box.corner = Point()
box.corner.x = 0.0
box.corner.y = 0.0
```



Exercise

- Objects can be return values:

- Write a function find_center that takes a Rectangle as an argument and returns a Point that contains the coordinates of the center of the Rectangle

Mutability

- Objects are mutable

- Write a function named move_rectangle that takes a Rectangle and two numbers named dx and dy. It should change the location of the rectangle by adding dx to the x coordinate of corner and adding dy to the y coordinate of corner.

Comparing and copying

- Issues when copying lists of lists...

Comparing and copying

- Issues when copying lists of lists...

- When comparing objects, the default behavior is the same as checking identity

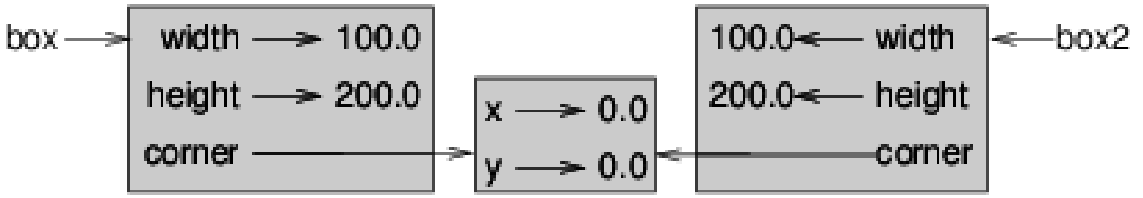
-
```
>>> p1.x = 3
>>> p1.y = 4
>>> p2.x = 3
>>> p2.y = 4
>>> p1 is p2
False
>>> p1 == p2
False
```

This behavior can be changed
--- more on this later

Shallow vs deep copying

- We can use the module copy to do what is known as a “shallow” copy of the object

```
>>> import copy
>>> box2 = copy.copy(box)
>>>box2 is box
False
>>>box2.corner is box.corner
True
```



Shallow vs deep copying

- In order to make the objects completely distinct in memory, need to deep-copy

```
>>> box3 = copy.deepcopy(box)
>>> box3 is box
False
>>> box3.corner is box.corner
False
```

Debugging

- How do we know whether a particular object has a given field?

```
>>> p = Point()
>>> print p.z
AttributeError: Point instance has no attribute 'z'
```

- To get the type of a variable

```
>>> type(p)
<type '__main__.Point'>
```

- To get whether a particular object has a field

```
>>> hasattr(p, 'x')
True
>>> hasattr(p, 'z')
False
```

Exercise

- Write a function that allows us to take two rectangles and find out whether they are intersecting or not