Computer Science 1 — CSci 1100
Lecture 12 — Files

Overview

- All good things must eventually be stored in a file!
- File read and write are basic form of input and output of programs.
- Files can store data that is too large to read into a program all at once.
- To read a text file, you must first open it for reading and when you are done you must close it.
- To write a text file, you must first open it for writing and when you are done you must close it.
- There are many different data formats for files. Often, one needs specific modules to read these files (e.g. images).
- We will see some special text formats in the next lecture.

Reading files

- To read a file, you must open it.

  ```
  >>> f = open("census_data.txt")

  >>> import urllib
  >>> f = urllib.urlopen("http://www.cs.rpi.edu/academics/undergrad/12.html")
  ```

  After opening, f points to the first line in the file.

- Read what you want.
  - Read a single line, return the line (including the newline at the end). At the end, `f` points to the next line.

    ```
    >>> line = f.readline()
    ```

  - Read the whole file, starting with where `f` is pointing to. Includes all new lines as well.

    ```
    >>> line = f.read()
    ```

- And, eventually close it. `f` does not point anywhere anymore. Closing a file is much more important for writing.

  ```
  >>> f.close()
  ```

Exercise

- Given the file `census_data.txt`:

  ```
  Location 2000 2011
  New York State 18,976,811 19,378,102
  New York City 8,008,686 8,175,133
  ```

- What is the value of variables `line1, line2, line3` after the following code executes?

```
f = open("census_data.txt")
line1 = f.readline()


line2 = f.read()


line3 = f.readline()
```

Reading a file completely

- Often we need to read a whole file line by line. All of the methods below return the same result.

  ```
  #########################################
  f = open('abc.txt')
  for line in f:
      print line
  f.close()


  #########################################
  for line in open('abc.txt'):
      print line


  #########################################
  f = open('abc.txt')
  line = f.readline()
  while line:
      print line
      line = f.readline()
  f.close()
  ```

- Let's examine what happens in each case!

Writing files

- Writing files is pretty similar to reading files. But, we must open the file to write or to append.

  ```
  f = open("outfile.txt","w")
  ```

  Write mode (`"w"`) means the old contents of the file is deleted completely.

  ```
  f = open("outfile.txt","a")
  ```

  Append mode (`"a"`) means the old contents of the file is kept, and the new content is added to the end of the file.

- To write to a file, we use the following command (you must put your own new lines):

  ```
  f.write("Hello world!")
  ```

- You must close the files you write! Otherwise, the changes you made will not be recorded.

  ```
  f.close()
  ```

- NOTE: we always mean the write mode when we say "write a file". Append mode is used very rarely and mentioned explicitly if it is meant to be used.

Exercise

- Given the file `census_data.txt`, write a new file called `census_summary` that contains a line with the total number of rows in `census_data.txt`, followed with a complete copy of `census_data.txt`.

Example: Search in a file

- We are given a file containing movies order by year, from later to earlier. Each line contains:

  `Actor Name | Movie Name | Year`

  Write code that returns all the movies by an input actor between two input years.

  ```
  maxyear = int(raw_input("Enter the latest year in the range ==>"))
  minyear = int(raw_input("Enter the earlier year in the range ==>"))
  ```

Sets vs. Lists

- Sets are like lists, but they do not contain duplicates. You can convert lists to sets and sets to lists.

```
>>> x = [1,2,3,3,4,5]
>>> y = set(x)
>>> y
set([1, 2, 3, 4, 5])
>>> z = {1,2,2,2,2}
>>> z
set([1, 2])
>>> x = list(z)
>>> x
[1, 2]
>>> z.add(4)
>>> z
set([1, 2, 4])
```

- You cannot index sets, but you can check if an item is in a set in the same way as a list.

```
>>> z[1]
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: 'set' object does not support indexing
>>> 1 in [1,2,3,4]
True
>>> 2 in {1,3}
False
>>> 'cad' in 'abracabadra'
False
>>> 'cad' in 'abracadabra'
True
```

- Checking for item in set is faster than checking for item in list. But adding a new item is faster for lists than sets.
- We will see more set operations in Lecture 14.

Example: Degrees of Kevin Bacon

- We are going to write a program to play degrees of Kevin Bacon (KB), invented by Kevin Bacon's publicist to promote him. It was a success!
- The Bacon number ranks people based on how close they are to KB. The claim is that almost everyone in movie business is at most 6 steps away from KB, but even finding people of Bacon number 4 or higher is very difficult. For example, Obama and Shiloh Jolie-Pitt both have Bacon number of 2. Hugo Chavez has Bacon number 3.
- `Kevin Bacon`'s Bacon number is 0. He is himself.
- If an actor starred in a movie with KB, then his/her Bacon number is 1. For example, `Tom Hanks` starred with KB in `Apollo 13`. So his Bacon number is 1.
- If an actor starred in a movie with an actor of Bacon number i, and if the actor has no Bacon number less than or equal to i, then that actor has Bacon number i+1 .

- Example:

  Justin Bieber's Bacon number is 3

  (Bacon #3) Justin Bieber and Miley Cyrus appeared in Justin Bieber: Never Say Never.

  (Bacon #2) Miley Cyrus and Sarah Jessica Parker appeared in Sex and the City 2.

  (Bacon #1) Sarah Jessica Parker and Kevin Bacon appeared in Footloose.

- Example:

  Musidora's Bacon number is 5

  (Bacon #5) Musidora and Stacia Napierkowska appeared in Les Vampires.

  (Bacon #4) Stacia Napierkowska and Jean Dax appeared in The Hunchback of Notre Dame.

  (Bacon #3) Jean Dax and Charles Boyer appeared in Mayerling.

  (Bacon #2) Charles Boyer and Eli Wallach appeared in How to Steal a Million.

  (Bacon #1) Eli Wallach and Kevin Bacon appeared in Mystic River.

- Now, let's first find all actors of Bacon number 1 using our movie file. Should KB's Bacon number be 1?

Example: Degrees of Kevin Bacon, continued

- Now, that we have computed all actors with Bacon number 1, can we compute higher numbers?
- If an actor starred in a movie with a Bacon number 1 actor, than they are a Bacon number 2 actor.
- By the way, did you know that you can Google someone's Bacon number? We can check if we are doing this correctly.
- Here is a puzzle for you. Our data set is a subset of the IMDB data for space considerations (though it is still very large). It does not have all the movies and it does not have all the actors in a movie.

  So, if we compute Bacon number of an actor to be 3, is it possible for the real Bacon number to be higher? How about lower?

Summary

- To read/write a file, you must first open it.
- To read all the lines in a file, you generally need a form of loop.
- To write a file, it is crucial to close it at the end to save it properly.
- Many other file operations use special data types with associated modules that help interpret the special formats in these files.