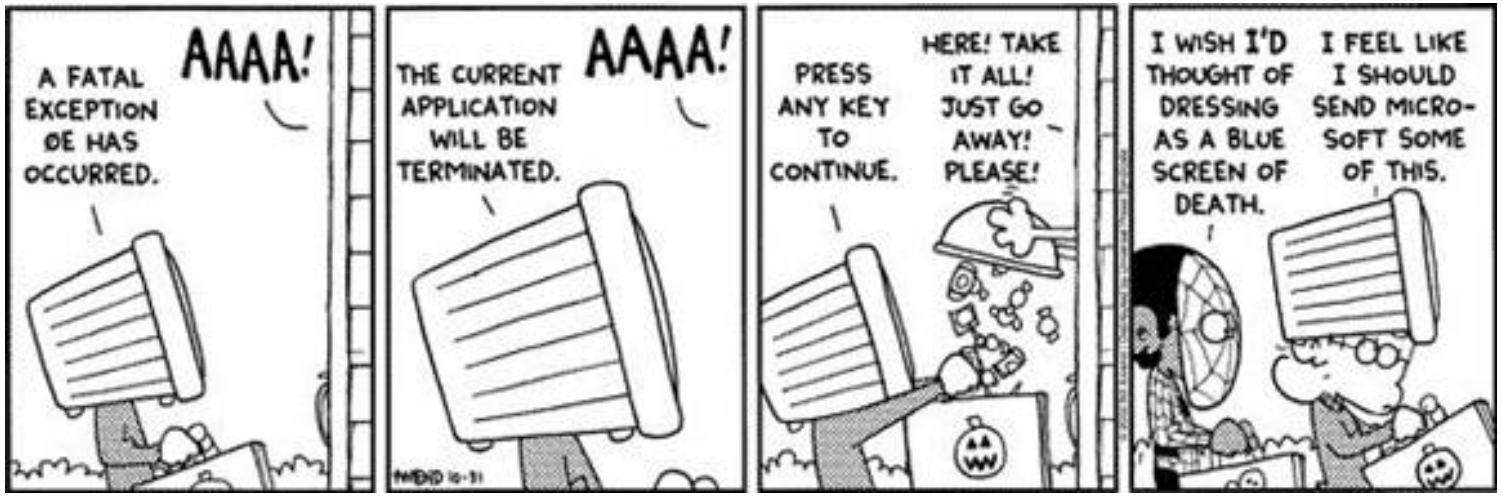
CSc 110, Spring 2017

Lecture 17: Line-Based File Input

Adapted from slides by Marty Stepp and Stuart Reges



Gas prices question

- Write a program that reads a file `gasprices.txt`
  - Format: *Belgium $/gal*
    US $/gal
    date

```
8.20
3.81
3/21/11
8.08
3.84
3/28/11
...
```

- The program should print the average gas price over all data in the file for both countries:

```
Belgium average: 8.3 $/gal
USA average: 3.9 $/gal
```

Gas prices solution

['8.20\n', '3.81\n', '3/21/11\n', '8.08\n', '3.84\n', '3/28/11\n', ….]

```
def main():
    file = open("gasprices.txt")
    belgium = 0
    usa = 0
    lines = file.readlines()

    for i in range(0, len(lines), 3):
      belgium = belgium + float(lines[i])
      usa = usa + float(lines[i + 1])
      count = count + 1

    print("Belgium average: " + str(belgium / count) + " $/gal")
    print("USA average: " + str(usa / count) + " $/gal")
```

Recall the hours.txt file

- File `hours.txt` has the following contents:

```
123 Brett 12.5 8.1 7.6 3.2
456 Sarina 4.0 11.6 6.5 2.7 12
789 Nick 8.0 8.0 8.0 8.0 7.5
```

How would we process this file if we wanted to extract just the names?

Processing files

- After using readlines(), each element of the list is a string
- Each string is a group of characters separated by spaces
- We may need a new method….

```
>>> f = open("hours.txt")
>>> f.readlines()
['123 Brett 12.5 8.1 7.6 3.2\n',
'456 Sarina 4.0 11.6 6.5 2.7 12\n',
'789 Nick 8.0 8.0 8.0 8.0 7.5\n']
>>>
```

Method for splitting strings

`split()`- a method that splits a string into a list of substrings

by default, uses spaces to split the string

```
s = "This is a line of words separated by spaces"
s = s.split()
for i in range(0, len(s)):
    print(s[i])
```

Line-based file processing

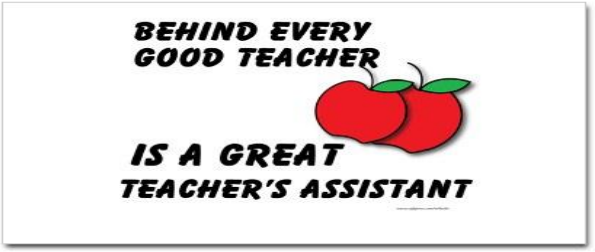
- Use `readlines()` to read the file
- Then use `split()` on each line

```
file = open("<filename>")
lines = file.readlines()
for single_line in lines:
    parts = single_line.split()
    <process the parts of the line>
```

Hours question

- Given a file `hours.txt` with the following contents:

```
123 Brett 12.5 8.1 7.6 3.2
456 Sarina 4.0 11.6 6.5 2.7 12
789 Nick 8.0 8.0 8.0 8.0 7.5
```



- Consider the task of computing hours worked by each person:

```
Brett  (ID#123) worked 31.4 hours (7.85 hours/day)
Sarina (ID#456) worked 36.8 hours (7.36 hours/day)
Nick   (ID#789) worked 39.5 hours (7.90 hours/day)
```

Hours answer

```
# Processes an employee input file and outputs each employee's hours.
def main():
    file = open("hours.txt")
    lines = file.readlines()
    for single_line in lines:
        process_employee(single_line)

def process_employee(line):
    parts = line.split()
    id = parts[0]           # e.g. "123"
    name = parts[1]         # e.g. "Brett"
    sum = 0
    count = 0
    for i in range(2, len(parts)):
        sum = sum + float(parts[i])
        count = count + 1
    average = sum / count
    print(name + " (ID#" + id + ") worked " +
        str(sum) + " hours (" + str(average) + " hours/day)")
```

```
123 Brett 12.5 8.1 7.6 3.2
456 Sarina 4.0 11.6 6.5 2.7 12
789 Nick 8.0 8.0 8.0 8.0 7.5
```

IMDb movies problem

- Consider the following Internet Movie Database (IMDb) data:

```
1 9.1 196376 The Shawshank Redemption (1994)
2 9.0 139085 The Godfather: Part II (1974)
3 8.8 81507 Casablanca (1942)
```

- Write a program that displays any movies who titles contain some specified text.
- It will also display the number of matches found.

```
Search for: part

Rank    Votes   Rating  Title
2       139085  9.0     The Godfather: Part II (1974)
40      129172  8.5     The Departed (2006)
95      20401   8.2     The Apartment (1960)
192     30587   8.0     Spartacus (1960)
. . .
4 matches
```

Pseudocode

```
1 9.1 196376 The Shawshank Redemption (1994)
2 9.0 139085 The Godfather: Part II (1974)
3 8.8 81507 Casablanca (1942)
```

ask the user for the search word
open the IMDb data file
create a list of the files contents
print the header of the output
for each line in the list of contents
    if the search word is in the line
        increment a matches counter
        print the line in the proper format
print the number of matches

```
Rank     Votes    Rating  Title
 2       139085   9.0     The Godfather: …
```

Problem: What if there are no matches?

Solution: Create a function to search for the word in a file

What else should be in a function?

Pseudocode

```
1 9.1 196376 The Shawshank Redemption (1994)
2 9.0 139085 The Godfather: Part II (1974)
3 8.8 81507 Casablanca (1942)
```

```
ask the user for a search word
open the IMDb data file
create a list of the files contents
if search word is in the list of files contents
    print the header for the output
    set matches counter
   for each line in the list
        if the search word is in the line
            increment the matches counter
            print the line in the proper format
   print the number of matches
```

```
Rank     Votes    Rating  Title
  2      139085   9.0     The Godfather: …
```

Better IMDb code

```
# Displays IMDB's Top 250 movies that match a search string.

def main():
    search_word = get_phrase()
    file = open("imdb.txt")
    line_list = file.readlines()
    line = search_list(line_list, search_word)

    if (len(line) > 0):
        print("Rank\tVotes\tRating\tTitle")
        matches = 0
        for a_line in line_list:
            ans = search_line(a_line, search_work)
            if (len(ans) > 0):
                matches = matches + 1
                display(a_line)
        print(str(matches) + " matches.")

# Asks the user for their search word and returns it.
def get_phrase():
    search_word = input("Search word: ")
    search_word = search_word.lower()
    print()
    return search_word
...
```

Better IMDb functions

```
# Breaks apart each line, looking for lines that match the search word.
def search_list(line_list, search_word):
    for line in line_list:
        line_lower = line.lower()     # case-insensitive match
        if (search_word in line_lower):
            return line
    return ""   # not found

# Looks for the search word in a single line.
def search_line(line, search_word):
    line_lower = line.lower()     # case-insensitive match
    if (search_word in line_lower):
        return line
    return ""   # not found


# displays the line in the proper format on the screen.
def display(line):
    parts = line.split()
    rank = parts[0]
    rating = parts[1]
    votes = parts[2]
    title = ""
    for i in range(3, len(parts)):
        title += parts[i] + " "    # the rest of the line
    print(rank + "\t" + votes + "\t" + rating + "\t" + title)
```