Homework #5

Professor Hugh C. Lauer
CS-1004 — Introduction to Programming for Non-Majors

(Slides include materials from *Python Programming: An Introduction to Computer Science*, 2nd edition, by John Zelle and copyright notes by Prof. George Heineman of Worcester Polytechnic Institute)

Assignment — HW5

- **Read one or more files of English text**
- **Create a list of unique words that occurs in those files**
  - With count of number of occurrences of each word
  - Alphabetically
- **Write that list to another file**

Objectives

- Become familiar with working with strings, lists, and files
- Learn how to sort a list
- Learn how read from and write to files
- Learn how to create formatted output

- Your biggest, most advanced *Python* program to date
- Due, Friday, September 30, 6:00 PM

Strongly encouraged to work in 2-person teams

Send e-mail to cs1004-staff@cs.wpi.edu if you would like help in finding a partner

Existing teams from Homework #4 carry over, unless we hear otherwise from you.

Note

- **This is a common assignment in *C* and C++ language courses**

- **Done differently**
  - Usually with a data structure called *binary tree*

Note 2

- §11.6.3 of textbook shows solution using *Python* dictionaries
  - Somewhat simpler
- Use this for inspiration, but …
- … this assignment is more demanding that the implemented by the textbook example.

Structure for HW5

- **Three modules plus wrapper**
- **Primary modules**
  1. Open input file, scan for words, strip punctuation, etc.
  2. Accumulate words from multiple files, eliminate duplicates, count
  3. Write output file in required format
- **Wrapper**
  - Manage other modules
  - Prompt user for file names, etc.
  - (Extra credit) interpret command line arguments

  - Test parts of program

Example — Gettysburg address

Four score and seven years ago our fathers brought forth on this continent, a new nation, conceived in Liberty, and dedicated to the proposition that all men are created equal.

Now we are engaged in a great civil war, testing whether that nation, or any nation so conceived and so dedicated, can long endure. We are met on a great battle-field of that war. We have come to dedicate a portion of that field, as a final resting place for those who here gave their lives that that nation might live. It is altogether fitting and proper that we should do this.

But, in a larger sense, we can not dedicate -- we can not consecrate -- we can not hallow -- this ground. The brave men, living and dead, who struggled here, have consecrated it, far above our ….

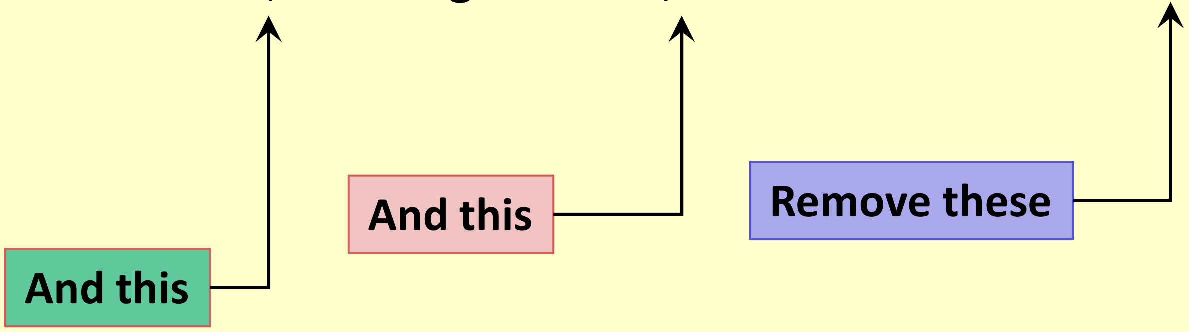
Example output — Gettysburg address

```
   7 a
   1 above
   1 add
   1 advanced
   1 ago
   1 all
   ...
   1 task
   1 testing
  13 that
  11 the
     ...
   1 work
   1 world
   1 years
------------
 138 Distinct words
```

Requirement

- Read one or more input files
- Break into individual words
- Remove punctuation *between* words …
- … but not within words

- Example
  - “But, in a larger sense, we can not dedicate --”



Requirement

- **Read one or more input files**
- **Break into individual words**
- **Remove punctuation *between* words …**
- **… but not within words**

- **Example**
  - “Bob’s hard-hearted attitude was his undoing”

Or this

But not this

How to read lines from a file

```
f = open(filename, mode)
```

- **`filename`** is a string
- Relative to current directory!
- **`mode`** should be 'r' (i.e., read)

```
for line in f:
  # process line here
```

```
f.close()  # finished with file!
```

- **Each line is a string ending in `'\n'`**

Extracting words from string

- Let `line` be the string

'brought forth on this continent, a new nation,\n'

  - (without the enclosing quotes)

- Then `line.split()` returns the list:–

```
['brought', 'forth', 'on', 'this',
'continent,', 'a', 'new', 'nation,']
```

  - I.e., partitioned at white-space

Note embedded commas

- Definition — white-space
  - Space, tab, line feed, newline, form feed, and vertical tab
  - See *Python* documentation > *Python* standard library > Text, §6.1

Note: `line.split()` method is more general
Can split at any set of characters!

Questions?

How to get rid of punctuation

- **`line.strip()` method**
  - Also `line.rstrip()`, `line.lstrip()`
  - Argument is a string of the characters to remove …
  - … from leading and trailing end!

- **Example, let `list[4]` be `'continent,'`**

- **Then**

```
list[4].strip('.,;:-?!')
```

  returns a new string with these characters stripped from the ends — i.e.,

```
'continent'
```

- **However,**

```
"Bob’s".strip('.,;:-?!')
```

  returns

```
"Bob’s"
```

Note

- `split()` first, then `strip()`!
- I.e., break into words with punctuation first, …
- … *then* remove the punctuation from ends of words, …
- … leaving contractions, possessives, hyphenated word intact!
- §11.6.3 does `strip()` first, then `split()`
- Loses internal hyphens and apostrophes!
- Produces many non-words
  - `'s'`, `'snt'`, `'t'`, `'ve'`

Questions?

You should have enough to read file and split into list (or lists) of words!

One module of your homework project!

What next?

- **Collect all words from all files into a dictionary**
- **Definition:– "Dictionary"**
  - A *Python* data type for collections, capable of storing and retrieving key-value pairs, …
    - … where keys and values can be *any* type, …
      - … data is unordered!
- **Called a *hash table* in most other languages**
  - Not a built-in data type (in those languages)
  - Also called a *map* in some languages
- **Read and study §11.6**
  - And all of Chapter 11!
- **More about Dictionaries on Monday!**

Collecting words and counts

- If word is *not* in the dictionary, add it with a count of 1

- If word is already in dictionary, increment its count

This is second module!

Short but challenging!

Third Module

- Format output and write to file
- Will discuss next time!

Questions?

Command Lines

- Windows, Macintosh, and Linux all have "command prompt" windows

- Command line format:–

```
verb arg1 arg2 arg3 ...
```

- `verb` is name of a program that carries out command action

- Each `arg` is a string
  - Delimited by spaces
  - `arg0` is the `verb`!

- Meaning:– Apply verb to the list of arguments
  - Don't return till finished!

Operating System's Responsibility

- **Pick apart command line**
  - Create a list of strings called "**`argv`**"
  - Number of items in list is "**`argc`**"
- **Load the program named verb (i.e., `arg0`) into a clean memory space.**
- **Call the function with the name `main()`, passing `argc` and `argv` as arguments**
- **Wait till it returns,**
  continue with next command line

Starting programs in a GUI

- **User "opens" a file or document**
- **OS or Window manager consults list of file types**
  - Finds program that opens the type of this file or document
  - Based on "extension" of file name
- **(Essentially) constructs a command line!**
  - As if it had been typed
  - Name of **`verb`** (i.e., program) as **`arg0`**
  - Name of file to be opened as **`arg1`**
  - Other arguments as needed
- **Calls `main()` function of the program!**

What about Python?

- **Command must be `python` or `python3`**

  Windows

  Macintosh
  Linux

- **Command line must be**

```
python HW5.py outFile InFile1 InFile2 …
```

- **Getting the arguments into *Python***

```
import sys.argv
```

`sys.argv` is a list containing the strings:–

```
['HW5.py', 'outFile', 'InFile1', 'InFile2', …]
```

Questions?

string.format()

- **A method for formatting output strings**
  - To keep columns aligned
  - To manage 'field widths'
  - To manage #'s of significant digits in floats
  - Etc.
- **Let T be a *template***
  - Structure of template to be described below
- **Then**

  `T.format(value, value, value, …)`
  - Makes a copy of **T**
  - Fills in the value arguments in the "slots" of new copy of **T**
  - Formats each value argument according to specifications in each "slot"

Template

- See §5.8.2 of textbook
- See 6.1.3 of *Python Documentation*
  - "Format String Syntax"
- Similar to formatting tools in other high-level languages
- Example:–

```
T = "Hello {0} {1}, you may have won ${2}"
```

- `T.format('Mr.', 'Smith', 1000)`

```
'Hello Mr. Smith, you may have won $1000'
```

Other formatting examples

- `T = 'left justification: {0:<5}'`
- `T.format("hi!")`

- `T = 'right justification: {0:>5}'`
- `T.format("lo!")`

- Numbers with decimals
- Decimal precisions
- Commas in numbers

- Locale-specific formats

References

- Textbook, §5.8.2
- Python 3.4.2 Documentation >
  Python Standard Library >
  Text
  - §6.1.2, 6.1.3
- Online help

Questions?