

SI 506: Programming I

Fall 2019

Lecture 12

Anthony Whyte <arwhyte@umich.edu>
Lecturer III, School of Information
715 N. University Ave, Ann Arbor, MI 48109
Roumanis Square, 2nd floor (“the loft”)

Slide deck revisions

errata: corrections and other changes

Slide no(s).	Fix ver.	Description
--------------	----------	-------------

	v1p1	
--	------	--

preliminaries

Class exercise

open file, read contents, write to file

Canvas Files

```
lectures/lecture_12/  
    lecture_12_exercise.py  
    umich_victors_with_title.txt
```

Upload to pythonanywhere.com

Place in same directory

Midterm

topic coverage

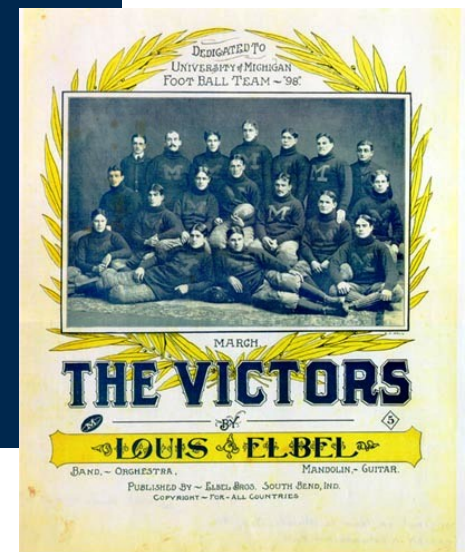
- values, types, and variables
- statements and expressions
- operators (select list)
 - arithmetic (+, -, *, /, //)
 - assignment (=, +=)
 - comparison (==, !=, >, <, >=, <=)
 - logical (and, or, not)
 - membership (in, not in)
- built-in functions (input(), int(), float(), len(), open(), range(), sorted(), str(), type())
- strings, string methods
- lists, list methods
- indexing and slicing
- loops (for, while)
 - use of counters in loops (i = 0 ... i +=1)
 - control statements (continue, break)
- conditional statements (if, else)
 - truth value testing (test object for truth value in if or while statements)
- functions
 - parameters, optional parameters
 - return statements
- reading from and writing to files
 - using the with statement

review

lists, functions, conditional statements

Source file

umich_victors_with_title.txt



challenge I

Function: open file, read lines, return list

using with statement and built-in open() function

```
source_path = 'umich_victors_with_title.txt'
```



```
def read_file(path):  
    """Read file line by line, return list."""  
    file_lines = []  
    with open(path, 'r') as file_obj:  
        for file_line in file_obj:  
            file_lines.append(file_line.strip())  
    return file_lines
```

```
# Get file content
```

```
lines = read_file(source_path)  
print(f"lines = {lines}\n")
```



return list of strings

List: lyrics

get lines from file, work with lyrics only

```
# Get file content
```

```
lines = read_file(source_path)
```

```
print(f"lines = {lines}\n")
```

```
# Get lyrics only
```

```
lyrics = lines[0] # Fix me
```

```
print(f"lyrics = {lyrics}\n")
```

challenge II

List: lyrics

list includes blank elements

Potential gotcha (blank elements in list)

```
lyrics_excerpt = [  
    'We hurrah, hurrah, we greet you now,',  
    'Hail!',  
    '',  
    'Far we their praises sing'  
]
```

is blank

Loop: truth value test, counter

return count of non-blank lines

```
# Get file content
```

```
lyrics = read_file(source_path)
```

```
# Count non-blank lines
```

```
num_non_blank_lines = 0
```

```
for line in lyrics:
```

```
    if line: # truthful (non blank line)
```

```
        num_non_blank_lines = ??? # Fix me
```

```
print(f"num_non_blank_lines = {num_non_blank_lines}\n")
```

Loop: truth value test, counter

return count of blank lines

```
# Get file content
```

```
lyrics = read_file(source_path)
```

```
# Total blank lines
```

```
num_blank_lines = 0
```

```
for line in lyrics:
```

```
    if line: # Fix me (fails to identify blank line)
```

```
        num_blank_lines += 1
```

```
print(f"blank_lines = {num_blank_lines}\n")
```

Loop: if statement, membership operator

return count of lines featuring 'Hail!'

```
# Get file content
```

```
lyrics = read_file(source_path)
```

```
# Get count of lines featuring 'Hail!'
```

```
num_hail_lines = 0
```

```
hail = 'Hail!'
```

```
for line in lyrics:
```

```
    pass # Fix me (add conditional statement, counter)
```

```
print(f"num_hail_lines = {num_hail_lines}\n")
```

Loop: get line lengths

list of line lengths

Get file content

```
lyrics = read_file(source_path)
```

Get length of each line, add to list

```
line_lengths = []
```

```
for line in lyrics:
```

```
    line_lengths.append(line) # Fix me
```

```
print(f"line_lengths = {line_lengths}")
```

```
print(f"line_lengths_sorted = {sorted(line_lengths)}\n")
```


Loop: if statement, comparison operator

append to list lines 28 characters long

```
# Get file content
```

```
lyrics = read_file(source_path)
```

```
twenty_eight_chars = []
```

```
for line in lyrics:
```

```
    pass # Fix me: test for 28 char lines
```

```
        # Fix me: add line to list
```

```
print(f"twenty_eight_chars = {twenty_eight_chars}\n")
```

challenge III

List: words

nested lists

```
# Word lookup list  
# greetings,  
# applause  
# honorifics (the nouns of winners)  
# superlative adjectives  
words = [  
    ['hail'],  
    ['cheer', 'hurrah'],  
    ['champions', 'heroes', 'leaders', 'victors'],  
    ['best', 'stalwart', 'triumphant', 'valiant']  
]
```

Function: count lines with certain words

nested lists; membership check, counter, control statement

```
def count_lines_with_words(lyrics, words):  
    """Increment count if any word in word list  
    is found in line. If match found, terminate  
    inner loop to avoid inflating count."""  
    count = 0  
    for line in lyrics:  
        for word in words:  
            if . . . # FIX ME: define condition  
                # FIX ME: need counter  
                pass # FIX ME terminate on 1st match  
    return count
```

Functions: lines with certain words

call function: pass in lyrics and word list

```
# Word lookup list
# greeting
# applause
# honorifics (the nouns of winners)
# superlative adjectives
words = [
    ['hail'],
    ['cheer', 'hurrah'],
    ['champions', 'heroes', 'leaders', 'victors'],
    ['best', 'stalwart', 'triumphant', 'valiant']
]

# Test 4 (superlatives list)
superlative_lines_count = count_lines_with_words(lyrics, []) # Fix

print(f"superlative_lines_count = {superlative_lines_count}")
print(f"superlative lines/total lines = {round(superlative_lines_count/num_lines, 2)}\n")

# Test 5 (new list, one element = 'victors')
victors_lines_count = count_lines_with_words(lyrics, []) # Fix

print(f"victors_lines_count = {victors_lines_count}\n")
print(f"victors lines/total lines = {round(victors_lines_count/num_lines, 2)}\n")
```

Function: count lines with certain words

refactor: use built-in function `any()` [NOT part of midterm]

Built-in `any()` function returns `True` if any element of an iterable is true. If the iterable is empty, returns `False`.

```
def count_lines_with_words(lyrics, words):  
    """Increment count if word is found in line.  
    If any match found, increment counter."""  
    count = 0  
    for line in lyrics:  
        if any(word in line.lower() for word in words):  
            count += 1  
    return count
```

challenge IV

Function: frequency count of words

nested loops; count all instances of a particular word

```
# Finally, write 'leaders and best' chorus to target file  
# Append 'Go Blue!' to chorus then write to file  
# Remember to add trailing newline \n to line string  
leaders_chorus = lyrics[17:??] # Fix me  
chorus_with_go_blue = ???
```

```
print(f"leaders_chorus = {leaders_chorus}")  
print(f"go_blue = {go_blue}\n")
```

```
# Add counter to print statement (debug friend)
```

```
i = 1
```

```
with open(target_path, 'w') as target:  
    for line in leaders_chorus:  
        print(f"line {i} = {line}")  
        target.write(f"{line}\n")  
        i += 1
```


Not sure: search 'python string methods'

w3schools.com

<http://bit.ly/2IxRa9S>

[Home](#) [HTML](#) [CSS](#) [JAVASCRIPT](#) [SQL](#) [PYTHON](#) [PHP](#) [BOOTSTRAP](#) [MORE](#) [REFERENCES](#) [EXERCISES](#)

Python Tutorial

- Python HOME
- Python Intro
- Python Get Started
- Python Syntax
- Python Comments
- Python Variables
- Python Data Types
- Python Numbers
- Python Casting
- Python Strings
- Python Booleans
- Python Operators
- Python Lists
- Python Tuples
- Python Sets
- Python Dictionaries
- Python If...Else
- Python While Loops
- Python For Loops
- Python Functions
- Python Lambda
- Python Arrays
- Python Classes/Objects
- Python Inheritance
- Python Iterators
- Python Scope
- Python Modules
- Python Dates
- Python JSON
- Python RegEx
- Python PIP
- Python Try...Except
- Python Command Input

Python String Methods

[< Previous](#)[Next >](#)

Python has a set of built-in methods that you can use on strings.

Note: All string methods returns new values. They do not change the original string.

Method	Description
capitalize()	Converts the first character to upper case
casefold()	Converts string into lower case
center()	Returns a centered string
count()	Returns the number of times a specified value occurs in a string
encode()	Returns an encoded version of the string
endswith()	Returns true if the string ends with the specified value
expandtabs()	Sets the tab size of the string
find()	Searches the string for a specified value and returns the position of where it was found
format()	Formats specified values in a string
format_map()	Formats specified values in a string
index()	Searches the string for a specified value and returns the position of where it was found
isalnum()	Returns True if all characters in the string are alphanumeric
isalpha()	Returns True if all characters in the string are in the alphabet
isdecimal()	Returns True if all characters in the string are decimals

Function: frequency count of words

nested loops; str.count()

```
# Word lookup list
# words[0] greeting
# word[1] applause
# words[2] honorifics (the nouns of winners)
# words[3] superlative adjectives
words = [
    ['hail'],
    ['cheer', 'hurrah'],
    ['champions', 'heroes', 'leaders', 'victors'],
    ['best', 'stalwart', 'triumphant', 'valiant']
]
```

Test 1

```
hail_count = count_word_in_lyrics(lyrics, ' ') # Fix me
print(f

# hail_count = {hail_count}\n

)
```

Test 2

```
cheer_count = count_word_in_lyrics(lyrics, ' ') # Fix me
print(f

# cheer_count = {cheer_count}\n

)
```

challenge V

File: write leaders and best chorus to file

list slicing, insert, append

```
target_path = 'umich_victors_leaders_refrain.txt'
```

```
# write 'leaders and best' chorus to target file  
# First, slice the list to get the 'leaders and best' chorus  
# Second, insert the original title and copyright and blank line before  
chorus  
# Third, append 'Go Blue!' to chorus  
# Finally, write to file  
# Remember to add trailing newline \n to line string
```

```
leaders_chorus = lyrics[17:0] # Fix me
```

```
i = 0  
for line in lines[0]: # Fix me  
    leaders_chorus.???? # Fix me with a list method  
    i += 1  
leaders_chorus.append('Go Blue!')
```

```
i = 1 # Debug: add line number to print statement  
with open(target_path, 'w') as target:  
    for line in leaders_chorus:  
        print(f"line {i} = {line}") # Debug line  
        target.write(f"{line}\n")  
        i += 1
```

Not sure: search 'python list methods'

w3schools.com

<http://bit.ly/2Oytje1>

[Home](#) [HTML](#) [CSS](#) [JAVASCRIPT](#) [SQL](#) [PYTHON](#) [PHP](#) [BOOTSTRAP](#) [MORE ▾](#) [REFERENCES ▾](#) [EXERCISES](#)

MySQL Insert

MySQL Select

MySQL Where

MySQL Order By

MySQL Delete

MySQL Drop Table

MySQL Update

MySQL Limit

MySQL Join

Python MongoDB

MongoDB Get Started

MongoDB Create Database

MongoDB Create Collection

MongoDB Insert

MongoDB Find

MongoDB Query

MongoDB Sort

MongoDB Delete

MongoDB Drop Collection

MongoDB Update

MongoDB Limit

Python Reference

Python Overview

Python Built-in Functions

Python String Methods

Python List Methods

Python Dictionary Methods

Python Tuple Methods

Python List/Array Methods

[< Previous](#)[Next >](#)

Python has a set of built-in methods that you can use on lists/arrays.

Method	Description
append()	Adds an element at the end of the list
clear()	Removes all the elements from the list
copy()	Returns a copy of the list
count()	Returns the number of elements with the specified value
extend()	Add the elements of a list (or any iterable), to the end of the current list
index()	Returns the index of the first element with the specified value
insert()	Adds an element at the specified position
pop()	Removes the element at the specified position
remove()	Removes the first item with the specified value
reverse()	Reverses the order of the list
sort()	Sorts the list

Note: Python does not have built-in support for Arrays, but Python Lists can be used instead.

finis

directors cut

File: optional parameter modes

open()

```
file_handle = open(path, '<mode>')
```

‘r’: read

‘w’: write

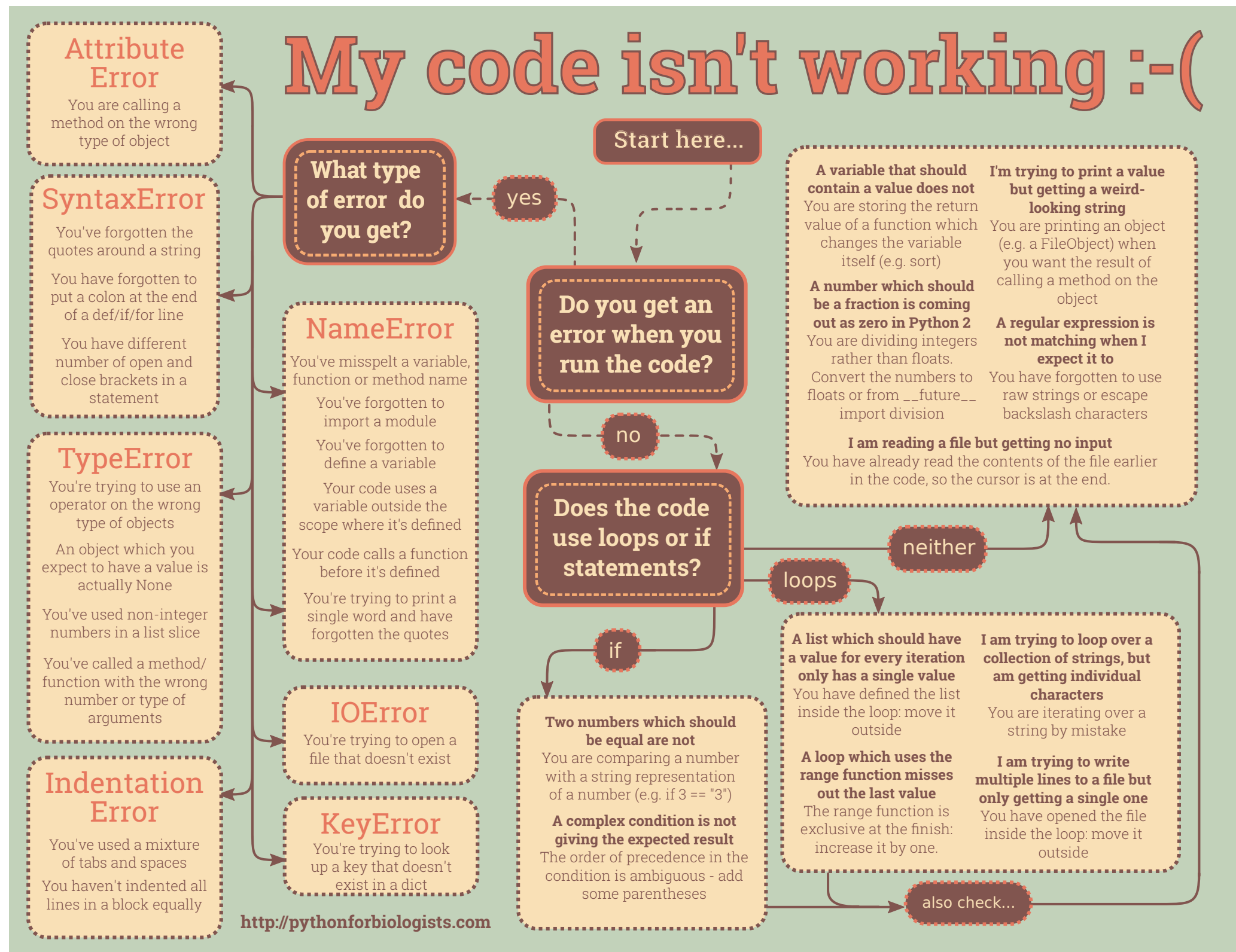
‘x’: create, write (new file)

‘a’: append (existing file)

‘r+’: read, write (same file)

When your code misbehaves

debug flowchart



Function: open file, read lines, return list

using with statement and built-in open() function



```
source_path = 'umich_victors_with_title.txt'
```

```
def read_file(path):  
    """Read file line by line, return list."""  
    file_lines = []  
    with open(path, 'r') as file_obj:  
        for file_line in file_obj:  
            file_lines.append(file_line.strip())  
    return file_lines
```

```
# Get file content
```

```
lines = read_file(source_path)  
print(f"lines = {lines}\n")
```



return list of strings

List: lyrics

get lines from file, work with lyrics only

```
# Get file content
```

```
lines = read_file(source_path)
```

```
print(f"lines = {lines}\n")
```

```
# Get lyrics only
```

```
lyrics = lines[3:]
```

```
print(f"lyrics = {lyrics}\n")
```

List: lyrics

compute length (count of all lines)

```
# Get file content
```

```
lyrics = read_file(source_path)
```

```
# Total lines
```

```
num_lines = len(lyrics)
```

```
print(f"num_lines = {num_lines}\n")
```

Loop: truth value test, counter

return count non blank lines

```
# Get file content
```

```
lyrics = read_file(source_path)
```

```
# Total non-blank lines
```

```
num_non_blank_lines = 0
```

```
for line in lyrics:
```

```
    if line: # truthy (non blank line)
```

```
        num_non_blank_lines += 1
```

```
print(f"num_non_blank_lines = {num_non_blank_lines}\n")
```

Loop: truth value test, counter

return count of blank lines

```
# Get file content
```

```
lyrics = read_file(source_path)
```

```
# Total blank lines
```

```
num_blank_lines = 0
```

```
for line in lyrics:
```

```
    if not line: # falsy (blank line)
```

```
        num_blank_lines += 1
```

```
print(f"blank_lines = {num_blank_lines}\n")
```

Loop: if statement, membership operator

return count of lines featuring 'Hail!'

```
# Get file content
```

```
lyrics = read_file(source_path)
```

```
# Get count of lines featuring 'Hail!'
```

```
num_hail_lines = 0
```

```
hail = 'Hail!'
```

```
for line in lyrics:
```

```
    if hail in line:
```

```
        num_hail_lines += 1
```

```
print(f"num_hail_lines = {num_hail_lines}\n")
```

Loop: get line lengths

list of line lengths

Get file content

```
lyrics = read_file(source_path)
```

Get length of each line, add to list

```
line_lengths = []
```

```
for line in lyrics:
```

```
    line_lengths.append(len(line))
```

```
print(f"line_lengths = {line_lengths}\n")
```


Loop: if statement, comparison operator

append to list lines 28 characters long

```
# Get file content
```

```
lyrics = read_file(source_path)
```

```
twenty_eight_chars = []
```

```
for line in lyrics:
```

```
    if len(line) == 28:
```

```
        twenty_eight_chars.append(line)
```

```
print(f"twenty_eight_chars = {twenty_eight_chars}\n")
```

List: words

nested lists

```
# Word lookup list  
# greetings  
# applause  
# honorifics (the nouns of winners)  
# superlative adjectives  
words = [  
    ['hail'],  
    ['cheer', 'hurrah'],  
    ['champions', 'heroes', 'leaders', 'victors'],  
    ['best', 'stalwart', 'triumphant', 'valiant']  
]
```

Function: count lines with certain words

nested lists; membership check, counter, control statement

```
def count_lines_with_words(lyrics, words):  
    """Increment count if any word in word list  
    is found in line. If match found, terminate  
    inner loop to avoid inflating count."""  
    count = 0  
    for line in lyrics:  
        for word in words:  
            if word in line.lower():  
                count += 1  
                break # terminate on 1st match  
    return count
```

Function: count lines with certain words

call function: pass in lyrics and word list

```
# Word lookup list
# greetings
# applause
# honorifics (the nouns of winners)
# superlative adjectives
words = [
    ['hail'],
    ['cheer', 'hurrah'],
    ['champions', 'heroes', 'leaders', 'victors'],
    ['best', 'stalwart', 'triumphant', 'valiant']
]

# Test 4 (superlatives list)
superlative_lines_count = count_lines_with_words(lyrics, words[-1])

print(f"superlative_lines_count = {superlative_lines_count}")
print(f"superlative lines/total lines = {round(superlative_lines_count/num_lines, 2)}\n")

# Test 5 (new list, one element = 'victors')
victors_lines_count = count_lines_with_words(lyrics, [words[2][3]])

print(f"victors_lines_count = {victors_lines_count}\n")
print(f"victors lines/total lines = {round(victors_lines_count/num_lines, 2)}\n")
```

Function: count lines with certain words

refactor: use built-in function `any()` [NOT part of midterm]

Built-in `any()` function returns `True` if any element of an iterable is true. If the iterable is empty, returns `False`.

```
def count_lines_with_words(lyrics, words):  
    """Increment count if word is found in line.  
    If any match found, increment counter."""  
    count = 0  
    for line in lyrics:  
        if any(word in line.lower() for word in words):  
            count += 1  
    return count
```

Function: frequency count of words

nested loops; count all instances of a particular word

```
def count_word_in_lyrics(lyrics, word):  
    """count number of times word appears in lyrics."""  
    count = 0  
    for line in lyrics:  
        if word in line.lower():  
            count += line.lower().count(word)  
            # count += 1 (misses multiple instances)  
    return count
```

Function: count word in lyrics

nested loops; str.count()

```
# Word lookup list  
# words[0] greeting  
# word[1] applause  
# words[2] honorifics (the nouns of winners)  
# words[3] superlative adjectives
```

```
words = [  
    ['hail'],  
    ['cheer', 'hurrah'],  
    ['champions', 'heroes', 'leaders', 'victors'],  
    ['best', 'stalwart', 'triumphant', 'valiant']  
]
```

Test 1

```
hail_count = count_word_in_lyrics(lyrics, words[0][0])  
print(f

## hail_count

 = {hail_count}\n")
```

Test 2

```
cheer_count = count_word_in_lyrics(lyrics, words[1][0])  
print(f

## cheer_count

 = {cheer_count}\n")
```

File: write leaders and best chorus to file

list slicing, insert, append

```
target_path = 'umich_victors_leaders_refrain.txt'
```

```
# write 'leaders and best' chorus to target file  
# First, slice the list to get the 'leaders and best' chorus  
# Second, insert the original title and copyright and blank line before  
chorus  
# Third, append 'Go Blue!' to chorus  
# Finally, write to file  
# Remember to add trailing newline \n to line string
```

```
leaders_chorus = lyrics[17:21]
```

```
i = 0  
for line in lines[:3]:  
    leaders_chorus.insert(i, line)  
    i += 1  
leaders_chorus.append('Go Blue!')
```

```
i = 1 # Debug: add line number to print statement  
with open(target_path, 'w') as target:  
    for line in leaders_chorus:  
        print(f"line {i} = {line}")  
        target.write(f"{line}\n")  
        i += 1
```