SI 506: Programming I

Fall 2019

Lecture 11

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
14, 36	v1p1	Fixed typo. The function `read_file(path)` accepts a file path to be used inside the function block by the built-in function `open(path, 'r')`. Instead `open(path, 'r')` referenced the variable `source_path`, a variable set outside the `read_file(path)` function block. A variable defined in the main body of the file is considered a *global* variable, accessible throughout the file. Variables assigned within the function block are considered *local* to the function and are accessible only from within the function block. Calling the function `read_file(path)` and passing to it the value assigned to the variable `source_path` is ok; having the `read_file(open)` function reference the global variable `source_path` from inside the function block is, in most cases, a bad idea. In our case, whatever `path` value the caller passed to `read_file(path)` would be ignored (in effect overridden) by the reference to `source_path` in the function block—not the behavior we intended. Statements *inside* a function block should avoid referencing variables with global scope. Instead, the function should be "parameterized" so that it can accept from the caller whatever values it requires to perform the task assigned to it (this might also include provisioning parameters with default values if the parameters are considered optional).

UMSI

preliminaries

Class exercise

open file, read contents, write to file

Canvas Files

```
lectures/lecture_11/
    lecture_11_exercise.py
    umich_victors.txt
```

Upload to pythonanywhere.com

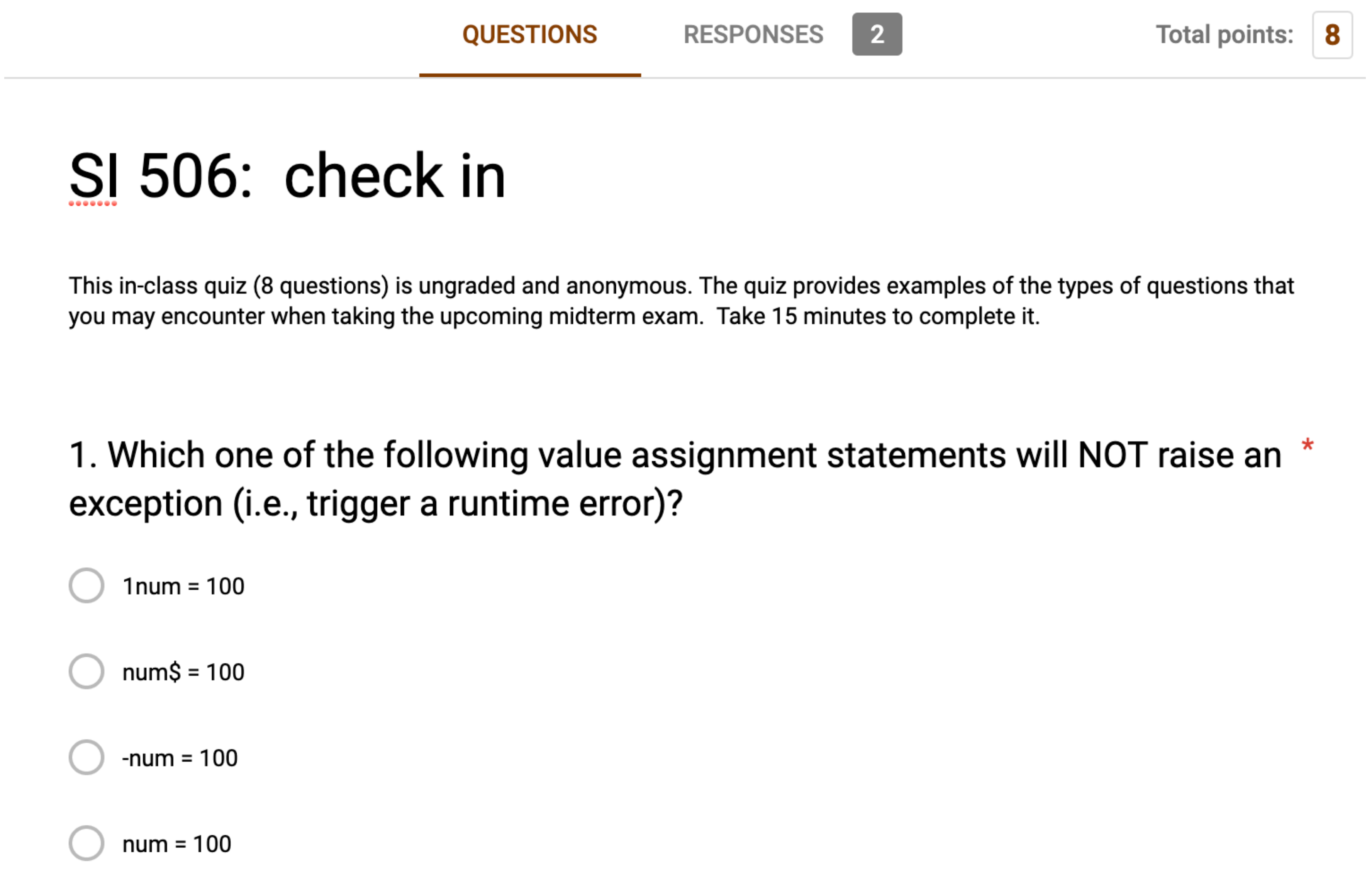
Place in same directory

quiz

Quiz

anonymous, ungraded

http://bit.ly/2AWmef7



QUESTIONS RESPONSES 2 Total points: 8

SI 506: check in

This in-class quiz (8 questions) is ungraded and anonymous. The quiz provides examples of the types of questions that you may encounter when taking the upcoming midterm exam. Take 15 minutes to complete it.

1. Which one of the following value assignment statements will NOT raise an exception (i.e., trigger a runtime error)? *

- ○ 1num = 100
- ○ num$ = 100
- ○ -num = 100
- ○ num = 100

UMSI

flashback

Way back in lecture 4: multi-line strings

use double quotations x 3

```
$ python3
>>> victors = """Hail! to the victors valiant
... Hail! to the conqu'ring heroes
... Hail! Hail! to Michigan
... the leaders and best"""
>>> print(victors)
```

Built-in function: str.splitlines()

interactive shell (type this)

```
$ python3
>>> lines = victors.splitlines()
>>> print(lines)
['Hail! to the victors valiant', "Hail! to the
conqu'ring heroes", 'Hail! Hail! to
Michigan', 'the leaders and best’]
>>> len(lines)
4
```

for loop

interactive shell (type this)

```
$ python3
>>> for line in lines:
...     print(line) # indent 4 spaces / tab
...
>>> for line in lines:
...     print(line.replace('Hail!', 'Huzzah!'))
...
```

review

lists, functions, conditional statements

Source file

umich_victors.txt

exercise 1

Function: open file, read lines, return list

using with statement and built-in open() function

```python
source_path = 'umich_victors.txt'


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


# Get file content
lyrics = read_file(source_path)
print(f"file_lyrics = {lyrics}\n")
```

← return list of strings

Compute: basic stats

total number of lines

```
# Get file content
lyrics = read_file(source_path)

# Total lines
num_lines = ??? # Fix me

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

Compute: potential gotcha

list includes blank elements

```python
# Potential gotcha (blank elements in list)
lyrics_excerpt = [
    'We hurrah, hurrah, we greet you now,',
    'Hail!',
    '',
    'Far we their praises sing'
]
```

is blank

UMSI

Compute: basic stats

total number of 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 = ??? # Fix me

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

UMSI

Compute: basic stats

total number or 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")
```

UMSI

Compute: basic stats

total number 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:
        pass # Fix me

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

Compute: basic stats

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}\n")
```

exercise II

lists: words

nested lists

```
# 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']
]
```

Functions: lines with certain words

nested lists; conditional statement, control statement

```python
def is_word_in_line(line, word):
    """Check if word is in line"""
    return word in line.lower() # to lower case


def count_lines_with_words(lyrics, words):
    """Increment count if word 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 is_word_in_line('', ''): # Fix me
                count += 1
                pass # Fix me
    return count
```

Functions: lines with certain words

call function: pass in lyrics and word list

```
# 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 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")
```

Functions: word checks with any()

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

```
def is_word_in_line(line, word):
    """Check if word is in line"""
    return word in line.lower() # to lower case


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 for word in words):
            count += 1
    return count
```

any() returns True if any element of an iterable is true. If the iterable is empty, returns False.

UMSI

exercise III

Function: frequency count of words

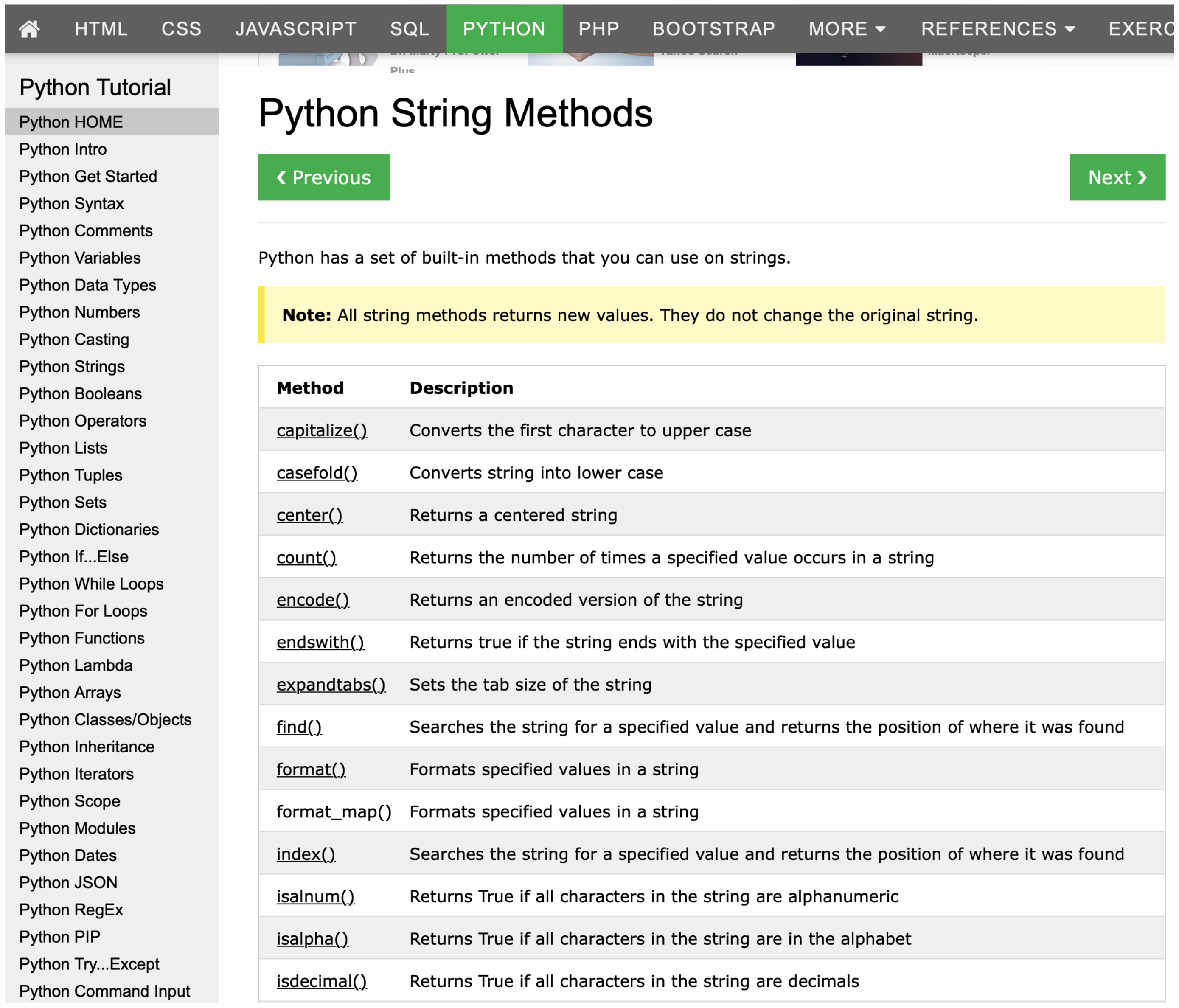
nested loops; count all instance of word in line

```python
# write function that counts the number of times
# a word appears in the lyrics.
# when incrementing count all instances of word in line
def count_word_in_lyrics(lyrics, word):
    """count number of times word appears in lyrics."""
    count = 0
    for line in lyrics:
        lower_case_line = line.lower()
        if word in lower_case_line:
            count += lower_case_line.???() # Fix me
            # count += 1 (misses multiple instances)
    return count
```

Not sure: search 'python string methods'

https://www.w3schools.com/python/python_ref_string.asp

http://bit.ly/2IxRa9S



Python String Methods

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

UMSI

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")
```

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)

Midterm exam

key concepts

files (read, write)

nested lists

functions

splitting and slicing

conditional statements

for loops (*not* while loops)

lists

strings

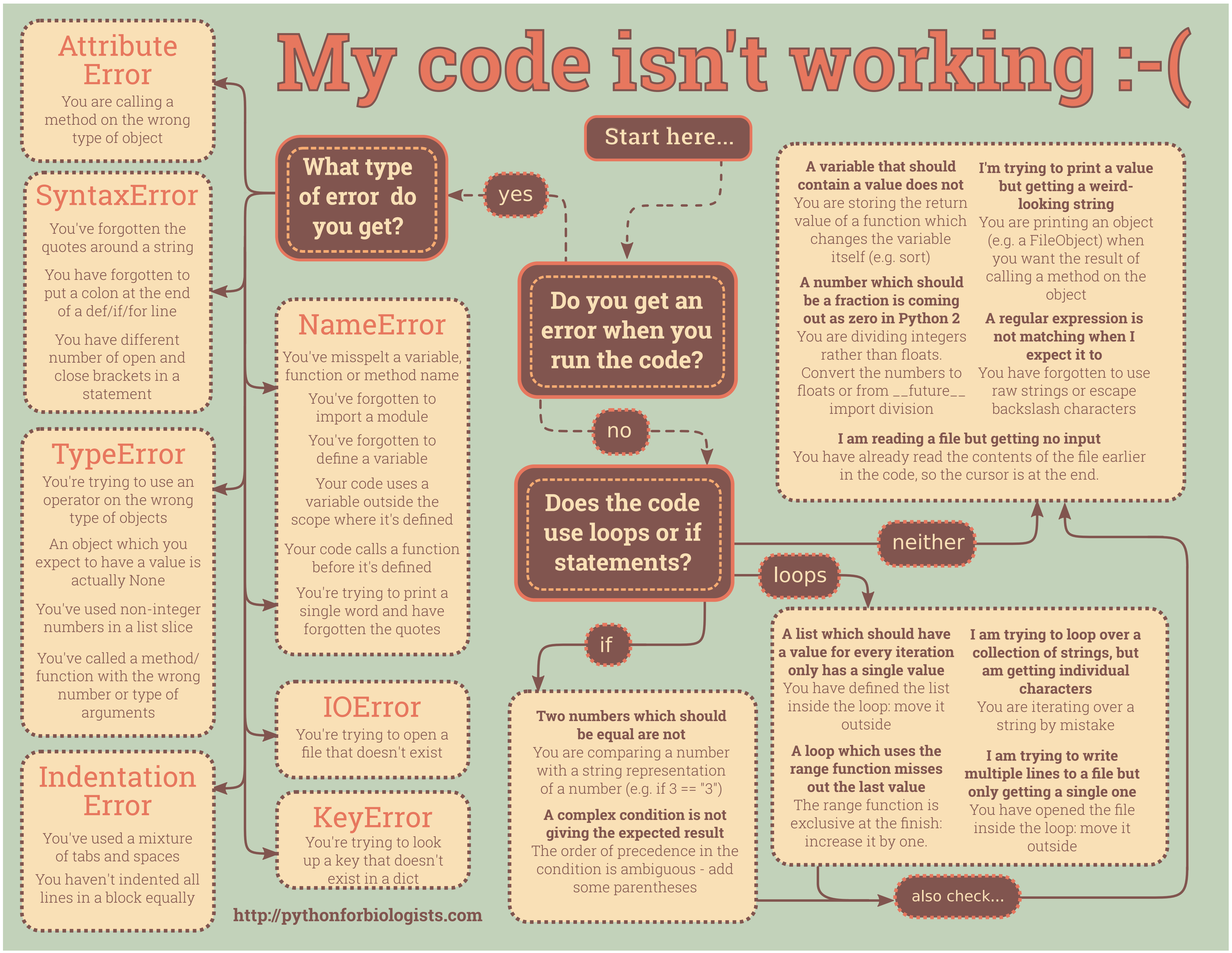
arithmetic, assignment, logical, identity, membership operators

built in functions()

objects, variables, variable assignment

When your code misbehaves

debug flowchart



My code isn't working :-(

Start here...

Do you get an error when you run the code?

What type of error do you get?

Attribute Error
You are calling a method on the wrong type of object

SyntaxError
You've forgotten the quotes around a string
You have forgotten to put a colon at the end of a def/if/for line
You have different number of open and close brackets in a statement

TypeError
You're trying to use an operator on the wrong type of objects
An object which you expect to have a value is actually None
You've used non-integer numbers in a list slice
You've called a method/ function with the wrong number or type of arguments

Indentation Error
You've used a mixture of tabs and spaces
You haven't indented all lines in a block equally

NameError
You've misspelt a variable, function or method name
You've forgotten to import a module
You've forgotten to define a variable
Your code uses a variable outside the scope where it's defined
Your code calls a function before it's defined
You're trying to print a single word and have forgotten the quotes

IOError
You're trying to open a file that doesn't exist

KeyError
You're trying to look up a key that doesn't exist in a dict

http://pythonforbiologists.com

Does the code use loops or if statements?

A variable that should contain a value does not
You are storing the return value of a function which changes the variable itself (e.g. sort)

A number which should be a fraction is coming out as zero in Python 2
You are dividing integers rather than floats. Convert the numbers to floats or from __future__ import division

I'm trying to print a value but getting a weird-looking string
You are printing an object (e.g. a FileObject) when you want the result of calling a method on the object

A regular expression is not matching when I expect it to
You have forgotten to use raw strings or escape backslash characters

I am reading a file but getting no input
You have already read the contents of the file earlier in the code, so the cursor is at the end.

Two numbers which should be equal are not
You are comparing a number with a string representation of a number (e.g. if 3 == "3")

A complex condition is not giving the expected result
The order of precedence in the condition is ambiguous - add some parentheses

A list which should have a value for every iteration only has a single value
You have defined the list inside the loop: move it outside

A loop which uses the range function misses out the last value
The range function is exclusive at the finish: increase it by one.

I am trying to loop over a collection of strings, but am getting individual characters
You are iterating over a string by mistake

I am trying to write multiple lines to a file but only getting a single one
You have opened the file inside the loop: move it outside

exercise 1

Function: open file, read lines, return list

using with statement and built-in open() function

```python
source_path = 'umich_victors.txt'


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


# Get file content
lyrics = read_file(source_path)
print(f"file_lyrics = {lyrics}\n")
```

$\leftarrow$ return list of strings

Compute: basic stats

total number of lines

```
# Get file content
lyrics = read_file(source_path)

# Total lines
num_lines = len(lyrics)

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

Compute: basic stats

Check your data

```
# Potential gotcha (blank elements in list)
lyrics_excerpt = [
    'We hurrah, hurrah, we greet you now,',
    'Hail!',
    '',
    'Far we their praises sing'
]
```

is blank

Compute: basic stats

total number of 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")
```

Compute: basic stats

total number or 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")
```

Compute: basic stats

total number of lines featuring ‘Hail!’

```python
# 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")
```

Compute: basic stats

list of line lengths

```python
# 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")
```

exercise II

lists: words

nested lists

```
# 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']
]
```

Functions: lines with words

nested lists; conditional statement, control statement

```
def is_word_in_line(line, word):
    """Check if word is in line"""
    return word in line.lower() # to lower case


def count_lines_with_words(lyrics, words):
    """Increment count if word 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 is_word_in_line(line, word):
                count += 1
                break # terminate on 1st match
    return count
```

Functions: lines with words

call function: pass in lyrics and word list

```
# 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 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”)
```

Functions: word checks with any()

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

```
def is_word_in_line(line, word):
    """Check if word is in line"""
    return word in line.lower() # to lower case


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 for word in words):
            count += 1
    return count
```

any() returns True if any element of an iterable is true. If the iterable is empty, returns False.

exercise III

Function: count word in lyrics

nested loops; str.count()

```python
# write function that counts the number of times
# a word appears in the lyrics.
# utilize str.count() when incrementing count
def count_word_in_lyrics(lyrics, word):
    """count number of times word appears in lyrics."""
    count = 0
    for line in lyrics:
        lower_case_line = line.lower()
        if word in lower_case_line:
            count += lower_case_line.count(word) # count all instances
            # count += 1 (misses multiple instances)
    return count
```

Function: count word in lyrics

nested loops; str.count()

```python
# 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']
]
```

```python
# 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")
```