SI 506: Programming I
Fall 2019
Lecture 10

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

UMSI

Slide deck revisions

errata: corrections and other changes

Slide no(s).	Fix ver.	Description
	v1p1	

Class exercise

open file, read contents, write to file

Canvas Files

```
lectures/lecture_10/
    lecture_10_exercise.py
    lecture_10_warmup.py
    noaa_whale_data_source.txt
    whale_names_source.txt
```

Upload to pythonanywhere.com

Place in same directory

preliminaries

UMSI

working with files

old school (canonical)

Files: read in data / write out data

using built-in open() function

```
source_path = 'whale_names_source.txt'
target_path = 'whale_names_target.txt'

# Create a file handle, return list
file_handle = open(source_path, 'r')
lines = file_handle.readlines()
file_handle.close()

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

# Create a file handle, write to target file
file_handle = open(target_path, 'w')
#  Get whale names, write to file
for line in lines:
    file_handle.write(line)
file_handle.close()
```

Files: read in data / write out data

using built-in open() function

```
source_path = 'whale_names_source.txt'
target_path = 'whale_names_target.txt'

# Create a file handle, return list
file_handle = open(source_path, 'r')
lines = file_handle.readlines()
file_handle.close()
```

read mode

return list

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

# Create a file handle, write to target file
file_handle = open(target_path, 'w')
#  Get whale names, write to file
for line in lines:
    file_handle.write(line)
file_handle.close()
```

write mode

write to file

UMSI

Files: open, read lines, close

return a list of lines (note: includes trailing '\n')

assignment	operation	returns	notes
file_handle =	open(path_to_file)	file object	Built in function returns "file handle" that can be used to read data out of file.
lines =	file_handle.readlines()	list	Read in the data line by line
	file_handle.close()		Close the "file handle" to free up resources.

UMSI

Files: open, read lines, close

return a list of lines (note: includes trailing '\n')

assignment	operation	returns	notes
file_handle =	open(path_to_file)	file object	Built in function returns "file handle" that can be used to read data out of file.
	for line in lines:		Iterate over the list elements.
	file_handle.write(line)		Write line to target file.
	file_handle.close()		Close the "file handle" to free up resources.

exercise 1

File: source file

whale_names_source.txt

```
lines = ['species,common_name\n',
         'Balaenoptera musculus,Blue Whale\n',
         'Eschrichtius robustus,Grey Whale\n',
         'Eubalaena glacialis,North Atlantic Right Whale\n',
         'Eubalaena japonica,North Pacific Right Whale\n',
'Physeter macrocephalus,Sperm Whale\n',
'Megaptera novaeangliae,Humpback Whale\n',
"Balaenoptera edeni,Bryde's Whale\n",
'Balaenoptera physalus,Fin Whale\n',
'Balaena mysticetus,Bowhead Whale\n',
'Balaenoptera acutorostrata,Minke Whale']
```

Note: Bryde's whale pronounced "Broodus" whale

Q: write to file only the common name?

what expression do I write?

```
target_path = 'whale_names_target.txt'

# Create a file handle, write to target file
file_handle = open(target_path, 'w')
#  Get whale names, write to file
for line in lines:
    common_name = ?
    file_handle.write(f"{common_name}")

file_handle.close()
```

working with files

with statement

Files: with statement

anatomy

```
with open(path, '<mode>') as <name>:
    for line in <name>:
        # Do something with line
        <statement(s)>
```

Advantages

Handles opening and closing the file, including open/close exceptions raised in the inner block.

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)

File: source file

noaa_whale_data_source.txt

```
species
common_name
type
max_lifespan_years
max_length_feet
max_weight_tons
```

```
species,common_name,type,max_lifespan_years,max_length_feet,max_weight_tons
Balaenoptera musculus,Blue Whale,Mysticeti (Baleen),90.0,110.0,165.0
Eschrichtius robustus,Grey Whale,Mysticeti (Baleen),70.0,49.0,45.0
Eubalaena glacialis,North Atlantic Right Whale,Mysticeti (Baleen),70.0,52.0,70.0
Eubalaena japonica,North Pacific Right Whale,Mysticeti (Baleen),70.0,64.0,100.0
Physeter macrocephalus,Sperm Whale,Odontoceti (Toothed),60.0,52.0,45.0
Megaptera novaeangliae,Humpback Whale,Mysticeti (Baleen),50.0,60.0,40.0
Balaenoptera edeni,Bryde's Whale,Mysticeti (Baleen),70.0,55.0,45.0
Balaenoptera physalus,Fin Whale,Mysticeti (Baleen),140.0,85.0,80.0
Balaena mysticetus,Bowhead Whale,Mysticeti (Baleen),200.0,60.0,100.0
Balaenoptera acutorostrata,Minke Whale,Mysticeti (Baleen),50.0,35.0,10.0
```

Files: read in data

with statement (file object is an iterable)

```
source_path = 'whale_names_source.txt'
target_path = 'whale_names_target.txt'

# Create a file handle named source, return list
with open(source_path, 'r') as source:
    lines = source.readlines()
```

OR

```
# Create a file handle named source, return list
lines = []
with open(source_path, 'r') as source:
    for line in source:
        lines.append(line)

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

iterable

UMSI

Files: loop over the file object

file object is an iterable

```
source_path = 'whale_names_source.txt'
target_path = 'whale_names_target.txt'

# Create a file handle named source
# the file object is an iterable so loop over it
lines = []
with open(source_path, 'r') as source:
    for line in source:
        lines.append(line)

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

iterable

The `for line in source` treats the file object `source` as an iterable, which automatically uses buffered I/O (input/output) and memory management when processing large files.

UMSI

Files: read in data / write out data

with statement

```
source_path = 'whale_names_source.txt'
target_path = 'whale_names_target.txt'

# Create a file handle named source, return list
lines = []
with open(source_path, 'r') as source:
    for line in source:
        lines.append(line)

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

# Create a file handle named target, write to file
with open(target_path, 'w') as target:
    for line in lines:
        target.write(line)
```

exercise II

File data: headers and whales

list indexing, slicing, strings to lists

```
# Return headers
headers = lines[?].strip().split(‘,')
print(f"headers = {headers}")

# Get whales, append to whales list,
# strip out trailing \n
whales = []
for line in lines[?]:
    whales.append(???)
print(f"whales = {whales}")
```

File data: toothed whales

use index position to select the correct element

```
# Return toothed whale(s)
toothed_whales = []
for whale in whales:
    if whale[?] == 'Odontoceti (Toothed)':
        toothed_whales.append(whale)

print(f"toothed whales = {toothed_whales}")
```

File data: baleen whales

use headers.index() to select the correct element

```
# Return Baleen whales
baleen_whales = []
for whale in whales:
    if whale[?] == 'Mysticeti (Baleen)':
        baleen_whales.append(whale)

print(f"baleen whales = {baleen_whales}\n")
```

File data: whales by max length

use negative index position to select the correct element

```
# Return whales with max length >= 80 feet
long_whales = []
for whale in whales:
    if whale[?] >= 80.0:
        long_whales.append(whale)

print(f"whales max length >= 80 ft = {long_whales}\n")
```

good enough?

List data: gotcha

decimal values expressed as strings; what to do?

```
                                          max_lifespan_years
                                          max_length_feet
                                          max_weight_tons
[
    ['…', 'Blue Whale', '…', '90.0', '110.0', '165.0'],
    ['…', 'Grey Whale', '…', '70.0', '49.0', '45.0'],
    . . .
]
```

List data: gotcha

use built-in functions to recast string/number values

```
[
    ['…', 'Blue Whale', '…', '90.0', '110.0', '165.0'],
    ['…', 'Grey Whale', '…', '70.0', '49.0', '45.0'],
    . . .
]
```

int(): returns integer

float(): returns a floating point number

```
max_lifespan_years = float(whale[3])
max_length_feet = float(whale[4])
max_weight_tons = float(whale[-1])
```

File data: whales by max length

use negative index position to select the correct element

```
# Return whales with max length >= 80 feet
long_whales = []
for whale in whales:
    if ???(whale[?]) >= 80.0:
        long_whales.append(whale)

print(f"whales max length >= 80 ft = {long_whales}\n")
```

File data: whales by weight

use index position; recast value

```
# Return whales that weigh between 40 and 70 tons
# (exclusive)
mid_weight_whales = []
for whale in whales:
    if 40.0 < ???(whale[?]) < 70.0:
        mid_weight_whales.append(whale)

print(f"mid-weight whales = {mid_weight_whales}\n")
```

File data: whales by lifespan

select open() parameter mode; use index position; recast value

```
target_path = 'noaa_whale_data_target.txt'

def format_max_lifespan_str(whale):
    """ Return '<common_name> max lifespan = <max_lifespan> years\n' """
    lifespan = f"{?} max lifespan = {str(?)} years\n"
       return lifespan
```

```
# Write whales with a max lifespan >= 90 years to target file
# Convert list back to string
with open(target_path, '?') as target:
    for whale in whales:
        if float(whale[?]) >= 90.0:
            target.write(format_max_lifespan_str(?))
```

File: target file

noaa_whale_data_target.txt

```
Blue Whale max lifespan = 90.0 years
Fin Whale max lifespan = 140.0 years
Bowhead Whale max lifespan = 200.0 years
```

finis

directors cut

UMSI

List slicing: problem 1

extract indices using range() and index position

```
regions = ['Eastern Africa', 'Western Africa', 'Southern Africa']

countries_regions = ['Botswana, Southern Africa', 'Kenya, Eastern Africa',
                     'Ghana, Western Africa', 'Uganda, Eastern Africa',
                     'Nigeria, Western Africa']

# PROBLEM 1
# Extract indices of Eastern Af      tries from
# countries_regions list (source) and store in list
# named eastern_africa       (target

eastern_african_indices = []

for index in range(len(countries_regions)):
    if countries_regions[index].split(', ')[1] == regions[0]:
        eastern_african_indices.append(index)

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

split

position

index

UMSI

List slicing: problem 2

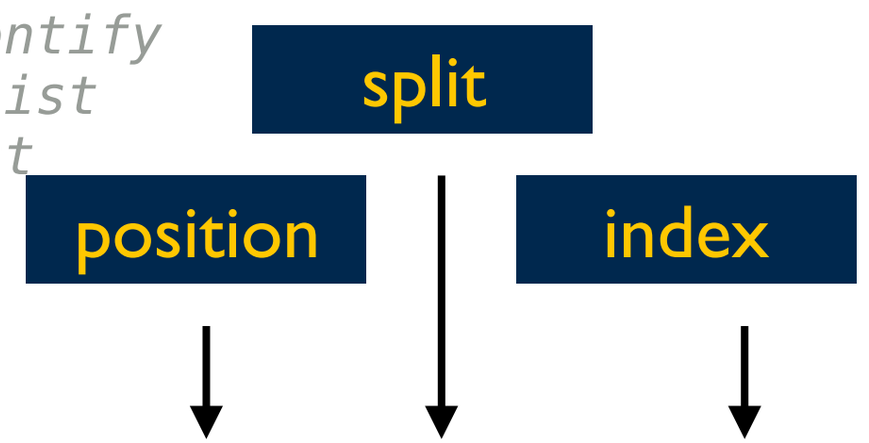
use indices to identify East African countries

```
countries_regions = ['Botswana, Southern Africa', 'Kenya, Eastern Africa',
                     'Ghana, Western Africa', 'Uganda, Eastern Africa',
                     'Nigeria, Western Africa']

eastern_african_indices = [1, 3] # derived from problem 1


# PROBLEM 2
# Use the indices in eastern_african_indices to identify
# East African countries in the countries_regions list
# and then store the country name (only) in the list
# eastern_african_countries.

eastern_african_countries = []

for index in eastern_african_indices:
    eastern_african_countries.append(countries_regions[index].split(', ')[0])

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

List slicing: problem 2

use indices to identify East African countries

```
countries_regions = ['Botswana, Southern Africa', 'Kenya, Eastern Africa',
                     'Ghana, Western Africa', 'Uganda, Eastern Africa',
                     'Nigeria, Western Africa']

eastern_african_indices = [1, 3] # derived from problem 1


# PROBLEM 2
# Use the indices in eastern_african_indices to identify
# East African countries in the countries_regions list
# and then store the country name (only) in the list
# eastern_african_countries.

eastern_african_countries = []

for index in eastern_african_indices:
    eastern_african_countries.append(countries_regions[index].split(', ')[0])

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



Python console

write/execute Python code (only)

Python3.7 console 13351686

Share with others

```
Python 3.7.0 (default, Aug 22 2018, 20:50:05)
[GCC 5.4.0 20160609] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> import json
>>> console = 'command line interpreter'
>>> purpose = 'accept user input in the form of Python code and attempt to execute it.'
>>> use = 'typically used for quick prototyping and exploration of the language (i.e., teaching).'
>>> data = {}
>>> data['console'] = console
>>> data['purpose'] = purpose
>>> data['use'] = use
>>> json_data = json.dumps(data)
>>> print(json_data)
{"console": "command line interpreter", "purpose": "accept user input in the form of Python code and attempt to execute i
t.", "use": "typically used for quick prototyping and exploration of the language (i.e., teaching)."}
>>>
```


UMSI

Unix shell (Bash)

interact with operating system, issue commands, run scripts

Bash console 13351749

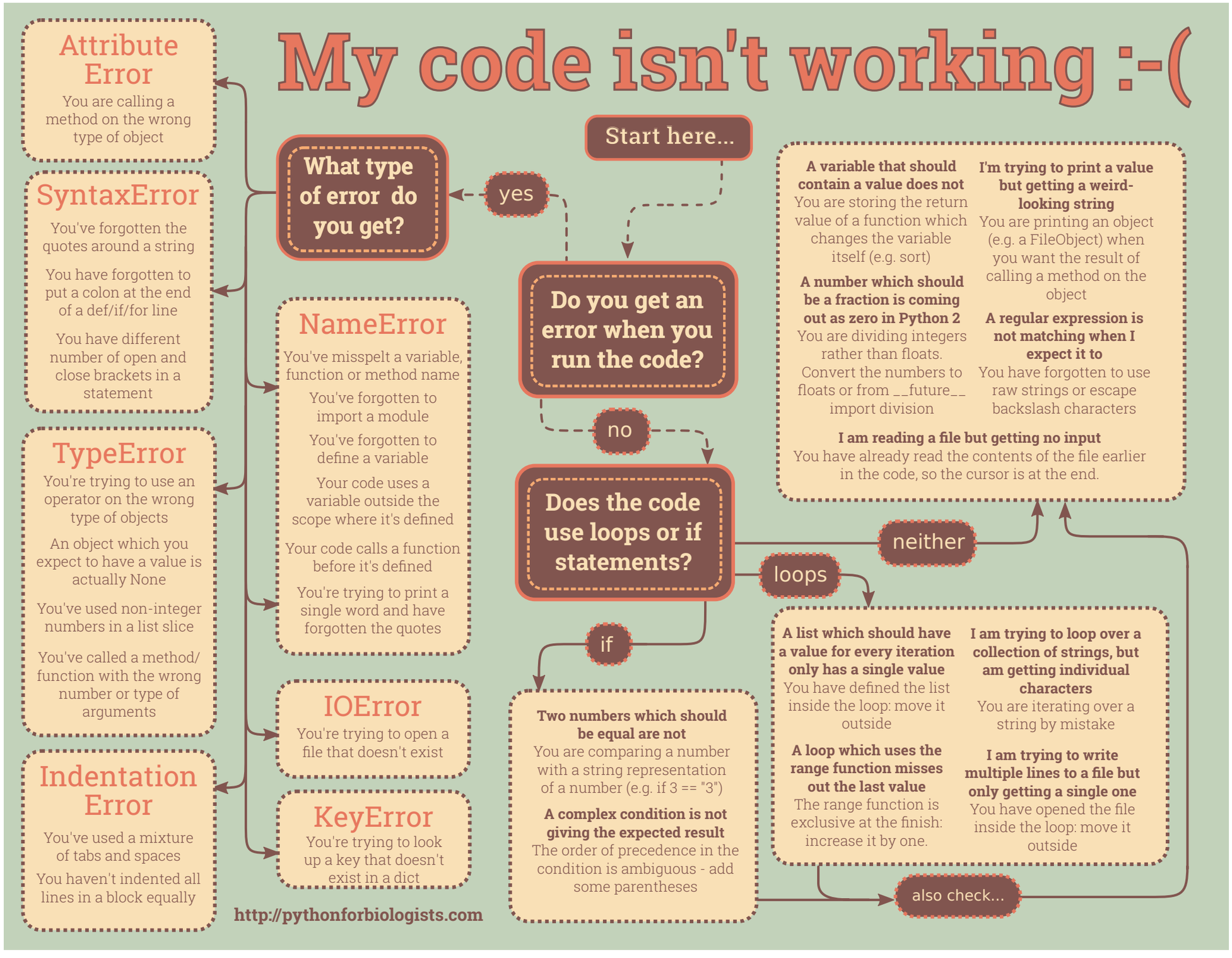
Share with others

```
01:43 ~ $ pwd
/home/arwhyte
01:43 ~ $ ls
README.txt  SI506
01:43 ~ $ cd SI506
01:44 ~/SI506 $ ls -la
total 16
drwxrwxr-x 4 arwhyte registered_users 4096 Sep  5 04:14 .
drwxrwxr-x 5 arwhyte registered_users 4096 Sep  5 22:01 ..
drwxrwxr-x 2 arwhyte registered_users 4096 Sep  5 02:28 lab_exercises
drwxrwxr-x 2 arwhyte registered_users 4096 Sep  2 00:43 problem_sets
01:44 ~/SI506 $ cd lab_exercises
01:44 ~/SI506/lab_exercises $ ls -la
total 12
drwxrwxr-x 2 arwhyte registered_users 4096 Sep  5 02:28 .
drwxrwxr-x 4 arwhyte registered_users 4096 Sep  5 04:14 ..
-rw-rw-r-- 1 arwhyte registered_users 1483 Sep  5 02:28 si506_lab_01.py
01:44 ~/SI506/lab_exercises $ python3 si506_lab_01.py arwhyte
Huzzah! arwhyte writes first Python program at 2019-09-11T21:44:51.572295-04:00
01:44 ~/SI506/lab_exercises $
```


UMSI

When your code misbehaves

debug flowchart



UMSI

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

UMSI