# SI 506: Programming I Fall 2019

Lecture 09

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





#### Class exercise

open file, read contents, write to file

Canvas Files
lectures/lecture\_09/
lecture\_08\_exercise.py
whale\_names.txt

Upload to pythonanywhere.com
Place in same directory





## preliminaries





### 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





#### Next week

Lectures, lab exercise, problem set

## all review





## working with files

old school





### Files: open, read, close

open existing file, return file handle, read content, close

```
path = 'whale_names.txt'

# Create a file handle, read content
file_handle = open(path)
whale_names = file_handle.read()
file_handle.close()

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





### Files: open, read, close

open existing file, return file handle, read content, close

```
path = 'whale_names.txt'

# Create a file handle, read content tent

file_handle = open(path)
whale_names = file_handle.read()
file_handle.close() ← you must close

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





### Files: gotcha

once closed, you must (re)open file, or trigger a traceback

```
path = 'whale_names.txt'
# Create a file handle, read contents
file_handle = open(path)
whale names = file handle.read()
file handle.close()
names = file_handle.read()
print(f"names = {names}")
```





### Files: close() means closed

once closed, you must (re)open file, or trigger a traceback

```
Traceback (most recent call last):
    File ".../whales.py", line 18, in <module>
        names = file_handle.read()
ValueError: I/O operation on closed file.
```





### Files: open, read line, close

readline() reads one line at a time; advances to next line

```
path = 'whale_names.txt'
# Create a file handle, read 2 lines
file handle = open(path)
header = file handle.readline()
whale = file handle.readline()
file handle.close()
                    advances to next line
print(f"header = {header}")
print(f"whale = {whale}\n")
```





### Files: gotcha

one read operation limit; call different type returns empty value

```
path = 'whale names.txt'
# Create a file handle
file handle = open(path)
whale names = file handle.read()
line = file_handle.readline()
file handle.close()
print(f"whale_names = {whale_names}")
print(f"line = {line}\n")
                blank string
```





### Files: open, read lines, return list, close

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





#### Files: iterate over each line; create list

loop over file\_handle (note: includes trailing '\n')

```
path = 'whale names.txt'
# Create a file handle, return list
lines = []
file_handle = open(path)
for line in file handle:
    lines.append(line)
file handle.close()
print(f"lines = {lines}\n")
```





#### File: whale list returned

using readlines() or for loop with file\_handle

```
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']
```

newline escape char appended to each line

double quotes surround single quote

Note: Bryde's whale pronounced "Broodus" whale





## 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)





## Files: open, write line, close anatomy

```
# Create a file handle, return list
path = 'whale_lines.txt'
file_handle = open(path, 'w')
# Get whale names, write to file
for line in lines:
    file_handle.write(f"{line}")
file_handle.close()
```

warning: if file does not exist it will be created; if file exists it will be overwritten.





#### Files: open, write line, close

anatomy

optional mode parameter 'w' (write)

```
# Create a file handle, return list
path = 'whale_lines.txt'
file_handle = open(path, 'w')
# Get whale names, write to file
for line in lines:
    file handle.write(f"{line}")
file handle.close()
```

always close

warning: if file does not exist it will be created; if file exists it will be overwritten.





#### File: exercise

#### write whale common names to file

```
# Create a file handle, return list
path = 'whale names.txt'
file_handle = open(path)
lines = #FIX ME
file handle.close()
# Extract headers (first row)
headers = lines[0].rstrip().split(',')
# Get whales only (skip header)
whales = []
for line in lines[#FIX ME]:
    whales.append(#FIX ME)
# Get common name by index position lookup
def get_common_name(names):
    """Return common name,"""
    return names[headers.index(#FIX ME)]
# Open new file in write mode, get file handle
# Iterate over whales list, call function to return common name, write to file
path = 'whale common names.txt'
file handle = open(path, '#FIX ME')
# Get whale names, write to file
for whale in whales:
    common name = #FIX ME
    file handle.write(f"{#FIX ME}\n")
file_handle.close()
```





### File: output file

whale\_common\_names.txt

Blue Whale Grey Whale North Atlantic Right Whale North Pacific Right Whale Sperm Whale Humpback Whale Bryde's Whale Fin Whale Bowhead Whale Minke Whale





#### In class exercise

#### fix broken file: open file, read lines, write lines new file

- Ia. Use open() to open whale\_names.txt and obtain a file handle.
- Ib. Return a list of lines using readlines(); assign to list named lines.
- 1c. Close file handle.
- 2. Assign first element of lines to a list named headers; use str.split(',') to return list of species and common names; use str.rstrip() to strip out newline ('\n'); syntax: headers = lines[?].rstrip().split(',')
- 3a. Create an empty list named whales = []
- 3b. Use a for loop to iterate over line elements containing *only* whales (use list slicing to skip header line); split the string on the comma returning a list; use str.rstrip() to remove '\n'
- 4. In the function named get\_common\_name(names) reference the header index position associated with common\_name in the return statement in order to return the common name in the evaluated list.
- 5a. Use open () with optional mode parameter 'w' to create a file named whale\_common\_names.txt and obtain a file handle.
- 5b. Use a for loop to iterate over the whales list; use the file handle to write to the file each whale's common name, calling the get\_common\_name(names) function to return the common name of each whale. Add newline '\n' escape char to string you create.
- 5c. Close the file handle.



### Files: open, write line, close

anatomy

```
optional mode parameter 'w' (write)
```

warning: if file does not exist it will be created; if file exists it will be overwritten.





## finis





## directors cut





### Functions: quick review

anatomy

```
'definition' keyword
                                    default value
                             arguments
            name
def create_email_address(name, domain=umich_domain):
    """Combine local part and domain to form an email address.
    return ''.join([name, '@', domain])
                                                docstring
      return statement gives back a value
```

code block (indented)





### Functions: quick review

have: uniqnames; need: U-M email addresses

```
umich_domain = 'umich.edu' # default domain value
uniqnames = ['arwhyte', 'csev', 'nantin'] # source
umich email addresses = [] # target
def create_email_address(name, domain=umich_domain):
    """Combine local part & domain.""
    return ''.join([name, '@', domain])
# Loop over uniquames list and call create_email_address
for name in uniquames:
    umich_email_addresses.append(create_email_address(name))
print(f"uniqnames = {uniqnames}\n")
print(f"umich_email_addresses = {umich_email_addresses}\n")
```



### List slicing: problem 1

extract indices using range() and list indexing operation

```
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 African countries from
# countries_regions list (source) and store in list
# named eastern_african_indices (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")
```





### 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
                                 split
# Extract indices of Eastern Ai
                                          tries from
# countries_regions list (source) and store in list
# named eastern_africa
                                 target
                                          index
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")
```





### 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
                                                               split
# East African countries in the countries_regions list
# and then store the country name (only) in the list
# eastern african countries.
                                                      position
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)."}
>>> ■
```





### Unix shell (Bash)

interact with operating system, issue commands, run scripts

```
Bash console 13351749
```





```
01:43 \sim \$ 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 $
```





#### When your code misbehaves debug flowchart

#### 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

### My code isn't working :-(

Start here...

Do you get an

error when you

run the code?

Does the code

use loops or if

statements?

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

What type of error do you get?

#### 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

#### A variable that should contain a value does not

You are storing the return value of a function which You are printing an object 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 weirdlooking string

(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.

neithei

loops

#### 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

also check.





### Files: open, write, close

#### write whale common names to file

```
# Create a file handle, return list
path = 'whale names.txt'
file handle = open(path)
lines = file handle.readlines()
file handle.close()
# Extract headers (first row)
headers = lines[0].rstrip().split(',')
# Get whales only (skip header)
whales = []
for line in lines[1:]:
    whales.append(line.rstrip().split(','))
# Get common name by index position lookup
def get_common_name(names):
    """Return common name,"""
    return names[headers.index('common_name')]
# Open new file in write mode, get file handle
# Iterate over whales list, call function to return common name, write to file
path = 'whale common names.txt'
file_handle = open(path, 'w')
# Get whale names, write to file
for whale in whales:
    common_name = get_common_name(whale)
    file_handle.write(f"{common_name}\n")
file_handle.close()
```



