FILE PROCESSING

COMP130 – INTRODUCTION TO COMPUTING

DICKINSON COLLEGE

TRANSIENT AND PERSISTENT DATA

- Data is ***transient*** if it exists only while a program or function is running.
  - A state diagram exists only as long as the program is running.
  - A stack frame exists only until the function exits (returns).
  - Transient data (e.g. a variable) is stored in the computer's memory (RAM).
- Data is ***persistent*** if it continues to exist after the program terminates.
  - Files hold data that remains after the program terminates.
  - Files can be opened again later, shared with others and used with other programs.
  - Persistent data (e.g. a file) is stored on the computer's disk, a "thumb" (USB) drive or on a computer "in the cloud".

READING PLAIN TEXT FILES IN PYTHON

```
in_file = open('sample.txt')
line = in_file.readline()
print(line)
in_file.close()
```

To read from a file, create a `file` object and assign it to a variable.

The `readline` method to reads a line from the file as a `String`.

Except for the last line in the file, the `String` will end with a newline (`'\n'`) character

Always close the file when your program has finished reading from it.

SPLITTING STRINGS

```
file = open('sample.txt')
line = file.readline()

words = line.split()

print(len(words))
print(words[0])
print(words[1])
print(words[2])

file.close()
```

Divide the String in line into elements by breaking it apart at spaces.

Use the `len` function and the `[ ]` notation to access the individual elements.

This works the same as it did for accessing the characters in a `String`.

ITERATING OVER SPLIT STRINGS

```
file = open('sample.txt')

index = 0
while index < len(words):
    print(words[index])
    index = index + 1

file.close()
```

Use a `while` loop to iterate over the elements of the split String if the `index` is needed, or a different order of iteration is needed.

A `for in` loop is more concise and readable if iterating in order and the index is not needed.

```
file = open('sample.txt')

for word in words:
    print(word)

file.close()
```

WRITING FILES IN PYTHON

Open the file for writing.

```
out_file = open('myfile.txt', 'w')

out_file.write('Put me in a file.\n')
out_file.write('Me too!\n')
out_file.write('Hey, me three!')
out_file.write("Don't forget me.")

out_file.close()
```

Include a newline ('\n') character to end a line in the file.

Without a newline ('\n') character the next `write` appears on the same line in the file.

```
1 Put me in a file.
2 Me too!
3 Hey, me three!Don't forget me.
```

FILE PATHS

- The ***current (or present) working directory*** is the directory from which the executing program is running.
  - `pwd`
  - `cwd = os.getcwd()`
    - `/Users/braught/courses/Fall19/COMP130/GitHubMaterials/Class29-30`
- File Paths:
  - An ***absolute path starts with a /*** and specifies all of the directories to reach the file.
    - `/Users/braught/courses/Fall19/COMP130/GitHubMaterials/Class29-30/sample.txt`
  - A ***relative path does not begin with a /*** and specifies the directories and file within the current working directory.
    - `sample.txt`
    - `emissions/rcp_4.5_data.csv`

PATHS AND OPENING FILES

- The `open` function will:
  - Look in the current working directory if a relative path is given.
    - `in_file = open('sample.txt')`
    - `in_file = open('emissions/rcp_4.5_data.csv')`
      - The `rcp_4.5_data.csv` file in the `emissions` directory in the current working directory.
  - Go to the file wherever it is if an absolute path is given
    - `in_file = open('/Users/braught/docs/words.txt')`
    - `in_file = open('/Users/braught/courses/Fall19/COMP130/GitHubMaterials/Class29-30/sample.txt')`