**CS506 Programming for Computing**

**HOP04A Working with Files**

03/16/2020 Developed by Amrutha Vaidyanathan

10/25/2020 Reviewed by Kim Nguyen

Center for Information Assurance (CIAE) @City University of Seattle (CityU)

**Before You Start**

* The directory path shown in screenshots may be different from yours.
* Some steps are not explained in the tutorial**.** If you are not sure what to do:
  1. Consult the resources listed below.
  2. If you cannot solve the problem after a few tries, ask a TA for help.

**Learning Outcomes**

* Learn ways to pass information to functions
* How to write certain functions whose primary job is to display information and other functions designed to process data and return a value or set of values.
* Learn to store functions in separate files called modules to help organize your main program files.
* How to use debugger tools

**Resources**

Matthes, E. (2019). [Python Crash Course: A Hands-On, Project-Based Introduction to Programming, 2nd Edition](https://login.proxy.cityu.edu/sso/skillport?context=146803). No Starch Press. (ISBN 9781593279288)

**Preparation**

In Visual Studio Code, open the private repository generated when you accepted the HOP04 assignment (If you cannot find that repository in your machine, you might have not cloned the repo, if so, please do before proceeding).

**Working with Files**

Files can be a persist way to store your data even though your program has finished. Also, you can even check whether your files or flash drives exist or attached to computer.

Basic steps to reading or writing file in Python

* Call the open() function to return a File object.
* Call the read() or write() method on the File object.
* Close the file by calling the close() method on the File object.

1. Create a text file **hello.txt**under **Module4** directory and type **Hello World**.

A black sign with white text

Description automatically generated

1. Create **Files.py** under the **Module4** directory and type the following. You will see the output is identical as the string inside the **hello.txt**

A picture containing drawing

Description automatically generated

In the terminal type **python3 Files.py** to check the output of the code.



The open and read function is explained with reference to the next code.

There is another method for opening and reading the file. The below code will explain it.

1. Create a file **Files\_other.py** and type the following.

A black sign with white text

Description automatically generated

In the terminal run the program by typing **python3 Files\_other.py**. It will display the similar output as before.



The open() function needs one argument: the name of the file you want to open. Python looks for this file in the directory where the program that's currently being executed is stored. In this example, “File\_other.py” is currently running, so Python looks for “hello.txt” in the directory where “File\_other.py”  is stored. The open() function returns an object representing the file. Here, open(“hello.txt”) returns an object representing “hello.txt”. Python assigns this object to file\_object.

The keyword with closes the file once access to it is no longer needed. Notice how we call open() in this program but not close(). You could open and close the file by calling open() and close(), but if a bug in your program prevents the close() method from being executed, the file may never close.

**Difference between the output and contents inside the original file:**

The only difference between the output from running the file and the original file is the extra blank line at the end of the output.

The blank line appears because read() returns an empty string when it reaches the end of the file; this empty string shows up as a blank line. If you want to remove the extra blank line, you can use rstrip() in the call to print()

You don’t have to type the below code.

A black sign with white text

Description automatically generated

**Append & Write mode**

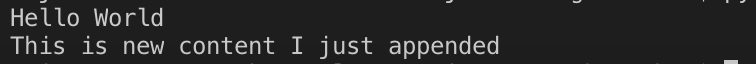
1. Create a file **File\_mode.py** and type the below code.

Text

Description automatically generated

**Note**: In this case, we use **append mode** because the content already exists. Using **write mode** will overwrite the existing file with new value. Both append and write mode will create new file if it does not exist.

In the terminal type **python3 File\_mode.py** to check the output.



After running **File\_mode.py***,* You will get new text file called **world.txt** with new content and the **hello.txt** with updated content.

**Making a List of Lines from a File**

1. Create a file **Files\_list.py** and type the following code.

A close up of text on a black background

Description automatically generated

When you use with, the file object returned by open() is only available inside the with block that contains it. If you want to retain access to a file's contents outside the with block, you can store the file's lines in a list inside the block and then work with that list.

The readlines() method takes each line from the file and stores it in a list. This list is then assigned to lines, which we can continue to work with after the with block ends.

In the terminal type **python3 Files\_list.py** to see the output for the code

