

- Please, put cellphones away from your desk
- Feel free to interrupt with a question
- Raise hand to respond to a question

- Homework

- o HW3 due Monday, end of the day
- o Correction sent out

Practice from Handout 7, is due on Thursday

Registration is coming up -- my recommendations

Check out :

- ma346 - Data Science
- cs370 - Intro to Machine Learning
- ft370 - Application of Natural Language Processing to Financial Investments



lecture7

Handout 7 CS230 – Introduction to Programming with Python–Spring'24 Page 1 of 4

file Handout 7 Working with text files.

There are two types of files: **text** and **binary**. A text file is a file object able to read and write str objects; text files are essentially strings on disk.

OS PACKAGE

The **os** and **os.path** modules (must be imported if used) have many useful functions for manipulating files and directories (a.k.a. folders).

- The terms **folder** and **directory** are synonyms.
- Files and folders form a hierarchy starting with a root.
- Root is specified differently depending on the operating system: {C:\ vs /}
- Path indicates location.
- Paths can be specified as **absolute** (starting from the root) or **relative** (to the current working directory)
 - o **./** depending on the operating system, indicates a subfolder
 - o **..** refers to the parent folder
 - o **.** refers to the current folder

Read more about OS and OS.PATH in <https://automatetheboringstuff.com/chapter8/>

Method(params): returns	Description
os.listdir(path): list	takes a pathname path and returns a list of its contents (names of files and directories)
os.getcwd(): str	Returns the str specifying the current working directory
os.chdir(path)	Change current working directory to one specified by path
os.mkdir(path)	Create a directory named path.
os.path.exists(strpath): bool	Returns True if strpath represents an existing directory, and False otherwise
os.path.join(dirname, [dirname], [filename]): str	Returns a path, combining the dirname(s) and filename Note: this only creates a path string, does not verify its correctness, i.e. that the folders/files actually exist.

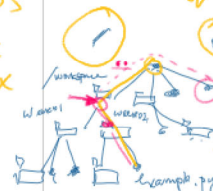
''' Demonstrating files and folders.
Requires that there be a subfolder myfiles in the program folder
with file sampleFile.txt'''

```
import os

# find current working directory - a path
cwd = os.getcwd()
print("This program is working from this folder path:", cwd)
```

Mac OS
Unix
Linux

Windows (DOS)



absolute - full address

C:\workspace\week02\example.py

relative path - from some location

week02/example.py

os.path.join('week02', 'example.py')

Windows → week02\example.py
Mac OS → week02/example.py

```

# get all files in this folder
lstContent = os.listdir(cwd) # or os.listdir()
print("This folder has the following content:")
print("\t", lstContent)

lstSubfolderContent = os.listdir("myfiles")
print("\nthe myfiles subfolder has the following content:")
print("\t", lstSubfolderContent)

# let's create a relative path for a subfolder file
pathToFileInSubfolder = os.path.join("myfiles", "sampleFile.txt")
print("\nA relative path for a subfolder file:")
print("\t", pathToFileInSubfolder)

```

PRACTICE PROBLEMS

- Perform the following operations using python code
 - List the content of the parent directory
- Using a list comprehension generate a list with all file extensions of files found in a given folder.

READ AND WRITE FROM/TO A FILE

Create a file object that is associated with a physical file, a.k.a. opening a file:

```
file = open(filename, mode)
```

Mode is one of the following (The default mode is 'r'):

'r'	Open a file for <u>reading only</u> .
'r+'	Read and Write.
'x'	Create and open a file for writing only; fails if the file exists.
'w'	Create and open a file for writing only; erases old content if the file exists

Function(params):returns

open(filename, mode): file Open a file in a specified mode.

In the following, file refers to a file object obtained after opening the file

Method(params):returns

Method(params):returns	Description
file.read():str	Returns the <u>specified number</u> of characters from the file. <u>If the argument is omitted, the entire</u> remaining contents are read.
file.readline(): str	Returns the <u>next</u> line of file as a string.
file.write(s, str): int	Writes the <u>string</u> to the file. <u>Returns number of chars written</u>
file.close(): None	Closes the file, <u>releasing resources</u> to the operating system.

Note that reading and writing operations move the file object's position in the file to the end of the read/written portion.

0 1 2 3



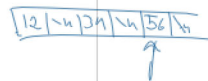
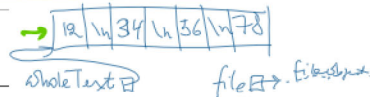
text



Seek(pos)

Examples. Suppose `measures.txt` in `myfiles` folder has the following content

12
34
56
78



`''' readFileDemo.py: Iterating over a file line by line
Requires that there be a subfolder myfiles in the program folder
with file measures.txt
'''`

`import os`

`print("Read the file as a whole into the string variable")
file = open(os.path.join("myfiles", "measures.txt"))
wholeText = file.read()
file.close()
print(wholeText)`

`print("Read line by line using WHILE loop")
file = open(os.path.join("myfiles", "measures.txt"))
line = file.readline()
while line: # line != ''
 print(line, end="")
 line = file.readline()
file.close()`

`print("Read line by line using for loop, requires NO read_line()")
namesfile = open(os.path.join("myfiles", "measures.txt"), 'r')
for line in namesfile:
 print(line, end="")`

for item in sequence:

PRACTICE PROBLEMS

3. Add code to the program `readFileDemo.py` above to compute the total sum of the numbers in the file. Write the total sum into a new file called `total.txt`.
4. File `weeklyTasks.txt` in `myfiles` folder has day names followed by task descriptions, e.g.

Mon	-- office hours
Tue	-- meeting
Wed	-- classes
Thu	-- office hours
Fri	-- meetings

Create a program that will ask the user to specify the day, then retrieve and print the day's task according to the records in the file.

EXCEPTIONS

An exception is a runtime error. To prevent the program from breaking and stopping because of an error, wrap the code that might produce an error into a try block, and include the error handling code into the except block, e.g.

```
'''Demo exception handling'''

import os

print("Demonstrate exception handling try-except block")
path = os.path.join("myfiles", "mmmeasures.txt")
try: #if an exception (i.e. error) occurs in try block...
    file = open()
    wholeText = file.read(path)
    file.close()
    print(wholeText)
except: # . . .program will not break, instead going to this part
    print("\nAn exception occurred when\n opening, reading or closing "
          "file", path )
    print("You can put more error handling code here")

print("Bye")
```

PRACTICE PROBLEMS

5. Modify programs you developed to handle exceptions associated with reading the files.