



# Computer Programming I

## Reading and Writing with Files

# The Need for Files

- What happens to the data stored in variables in RAM after a program stops running?
  - Data disappears once the program stops running
- How do you retain the data?
  - Save the data in a file so it can be retrieved later
- Sometimes the data needed in a program is stored in a file
  - Instead of waiting for a user to type data with a keyboard the program should read the data from a file that resides in permanent storage.
- Python can open files for reading and writing
  - Python assumes files contain text

# Input Files

- The process of retrieving data from a file is known as **reading data**
- When data is read from a file the data is copied from the file into a variable in RAM
- **Input file** is the term used to describe a file from which data is read
  - The program gets input from the file



# Output Files

- The process of saving data in a file is known as **writing data**
- When data is written to a file the data is copied from a variable in RAM to the file
- **Output file** is a file to which data is written
  - The program stores output in it

# Working with I/O Files

- When a file is used in a program we must:
  1. Open the file
    - Create a connection between the file and the program
  2. Process the file
    - Read data from or write data to the file
  3. Close the file
    - Disconnect the file from the program



# File Names

- Files are identified by a **filename**
- A filename consist of a **base name** and an extension
- A file **extension** indicates the type of data stored in a file
  - A short sequence of characters that appear at the end of a file name, preceded by a period (for example: .py, .cpp, .dat, .pdf, .txt, .jpg, .wav)



clap.wav



Pizza.png



data.txt



salaryIncrease.cpp



palindrome.py

# Directories

- Files exist in *directories* (sometimes called *folders*)
- Directories can contain files or other directories.
- There is a base directory on your computer, sometimes called the *root directory*
  - Example: C:\ (in a Windows device)
- A complete description of what directories to visit to get to a file is called a *path*
- The path separator for Windows is \
  - C:\CS171\loops.py
- The path separator for Mac is /
  - /Users/myMac/Documents/CS171/loops.py



# Opening files

- To open a file we will use the built-in `open()` function
- This function opens a file and returns a **file object** that you use to manipulate the file in the program:

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

- ***filename***: a string with the name (and path) of the file to be opened
- If no path is provided, the file is assumed to be in the same directory as the program you are writing/executing.



# Opening files

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

- **mode**: is a string that says what you want to do with the file:
  - **r** : “open file for reading only” – default mode
  - **w** : “open file for writing only” – overwrite any existing content.
  - **a** : “open file for writing only, with appending”
  - **r+** : update mode that allows for both reading and writing at the same time
  - **rb**: read in binary
  - **wb**: write in binary

# Opening files

- Examples:

```
#opening for reading
```

```
inFile = open("data.txt")
```

```
myFile = open("C:\\CS171\\readme.txt", "r")
```

```
#opening for writing
```

```
myfile = open("results.txt", "w")
```

```
outFile = open("/Users/myMac/CS171/out.txt", "w")
```

- **NOTE:**

- The path separator for Windows is **\\**
- The path separator in a Mac is **/**



# Closing files

- To close a file we will use the method `close()` :

*`FileObject.close()`*

- The file cannot be manipulated by the program until it's opened again.
- Example: `inFile.close()`  
`outFile.close()`

# Reading text from a file

- **read()** reads the whole file as a single string.
  - Returns a string.
- **readlines()** reads the whole file into a list where each element is a single line (a string).
  - Returns a list of strings.
  - Each line ends with the new line character (`\n`)
- **readline()** reads one line of the file at the time.
  - Returns a string.
- **Note:** **read()** and **readlines()** read the file content until the end of the file (EOF) is found.
  - can only be used once without closing and reopening the file.



# Examples

```
>>> infile = open('C:\\Users\\CS171\\news.txt')
>>> content = infile.read()
>>> content #a string
```

```
'The next house you buy may already have someone living in it: Alexa.\nAmazon and home-builder Lennar are opening staged-homes filled to the brim with Alexa-powered smart-home products. Anyone can visit these new "Amazon Experience Centers" to see what it's like to yell at a speaker to open the blinds, or push a button to order more toilet paper from Amazon Prime. The companies are launching the houses in 15 cities across the United States to start.\n\nLennar already has Amazon's Alexa smart assistant pre-installed in all of its new homes. The company includes an Echo Show and an Echo Dot so homeowners can lock doors, change temperatures, and turn off lights with voice commands.\n\nIts houses have a number of other smart-home gadgets to get people started, including a Sonos One speaker, a Ring video doorbell, smart locks, a connected thermostat, and a Samsung SmartThings hub.\n\nLennar announced the Echo integration last summer as part of the its "everything included" approach to selling houses.\n\nLennar is the largest home builder in the United States, and made almost 30,000 homes in 2017. It has long had fully staged model homes all around the country. Now it is letting Amazon transform some of them with even more of its products, from Fire TVs and security cameras, to Dash Buttons and Prime Music services.\n\nLennar homes may have Echos on the counters, but most of the other included devices will work on any smart-home platform. The company sells all its houses with wireless internet pre-installed, including commercial grade access points and coverage in every corner of the house.\n\nIn 2016, the company announced it would sell houses designed to work with connected devices and Home Kit, Apple's smart-home platform.\n\n"We realized very quickly that while the devices worked well and the technology was really compelling, it was also very scary," said David Kaiserman, president of Lennar Ventures.\n\nAmazon offered something Apple didn't: customer service professionals who can come to your home and install or fix all these futuristic gadgets through Amazon Home Services.\n\n"We think that's a necessary ingredient just to get people started and comfortable," said Kaiserman.\n\nThe smart-home market is expected to hit $53 billion in 2022, according to Zion Market Research. Major hardware chains like Lowes now stock connected-home devices, or have their own lines. And the big technology companies including Amazon, Google, and Apple are all fighting to be the primary system used in the home.\n\nAmazon hopes its new experience centers will give it an edge. Potential customers can test out a futuristic home soaked in Amazon products and services, managed by a roommate named Alexa who never leaves dirty dishes in the sink.'
```

```
>>> infile.close()
```



# Examples

```
>>> infile = open('/Users/myMac/CS171/news.txt')
>>> lines = infile.readlines()
>>> lines      #a list of strings
```

```
['The next house you buy may already have someone living in it: Alexa.\n', 'Amazon and home-builder Lennar are opening staged-homes filled to the brim with Alexa-powered smart-home products. Anyone can visit these new "Amazon Experience Centers" to see what it\'s like to yell at a speaker to open the blinds, or push a button to order more toilet paper from Amazon Prime. The companies are launching the houses in 15 cities across the United States to start.\n', '\n', "Lennar already has Amazon's Alexa smart assistant pre-installed in all of its new homes. The company includes an Echo Show and an Echo Dot so homeowners can lock doors, change temperatures, and turn off lights with voice commands.\n", '\n', 'Its houses have a number of other smart-home gadgets to get people started, including a Sonos One speaker, a Ring video doorbell, smart locks, a connected thermostat, and a Samsung SmartThings hub.\n', '\n', 'Lennar announced the Echo integration last summer as part of the its "everything included" approach to selling houses.\n', '\n', 'Lennar is the largest home builder in the United States, and made almost 30,000 homes in 2017. It has long had fully staged model homes all around the country. Now it is letting Amazon transform some of them with even more of its products, from Fire TVs and security cameras, to Dash Buttons and Prime Music services.\n', '\n', 'Lennar homes may have Echos on the counters, but most of the other included devices will work on any smart-home platform. The company sells all its homes with wireless internet pre-installed, including commercial grade access points and coverage in every corner of the house.\n', '\n', "In 2016, the company announced it would sell houses designed to work with connected devices and Home Kit, Apple's smart-home platform.\n", '\n', '"We realized very quickly that while the devices worked well and the technology was really compelling, it was also very scary," said David Kaiserman, president of Lennar Ventures.\n', '\n', "Amazon offered something Apple didn't: customer service professionals who can come to your home and install or fix all these futuristic gadgets through Amazon Home Services.\n", '\n', '"We think that\'s a necessary ingredient just to get people started and comfortable," said Kaiserman.\n', '\n', 'The smart-home market is expected to hit $53 billion in 2022, according to Zion Market Research. Major hardware chains like Lowes now stock connected-home devices, or have their own lines. And the big technology companies including Amazon, Google, and Apple are all fighting to be the primary system used in the home.\n', '\n', 'Amazon hopes its new experience centers will give it an edge. Potential customers can test out a futuristic home soaked in Amazon products and services, managed by a roommate named Alexa who never leaves dirty dishes in the sink.']
```

```
>>> infile.close()
```



# Examples

```
>>> fileName = input('Enter the name of the file to read: ')
Enter the name of the file to read: news.txt
>>> infile = open(fileName)
>>> line1 = infile.readline()
>>> line1    #first line in the file
'The next house you buy may already have someone living in
it: Alexa.'

>>> infile.close()
```

# Loops and files

- If you need to process the content of a file, the **for** loop is very helpful.
- Example:

```
infile = open(fileName)
lines = infile.readlines()
for line in lines :
    newLine = line.replace('Alexa', 'assistant' )
    print(newline)
infile.close()
```

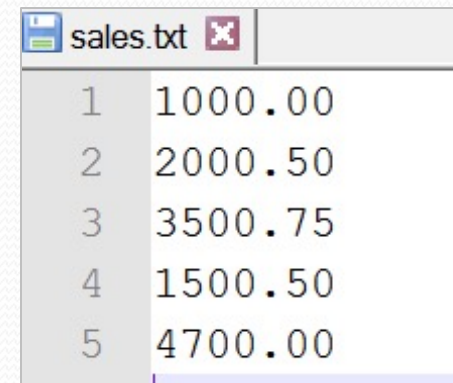


# Reading files with numeric values

- Python file methods for reading, read everything as strings (text)
- If you are working with numbers:
  - You need to convert them from string format to numeric format using the `int` and `float` functions.

## Example:

```
>>> infile = open('C:\\Users\\CS171\\sales.txt')
>>> values = infile.readlines()
>>> for item in values :
        total = total + float(item)
>>> total
12701.75
```



sales.txt	
1	1000.00
2	2000.50
3	3500.75
4	1500.50
5	4700.00

# Loop over files

- **for** loops can be used to loop over the lines of a file.

```
infile = open("words.txt", "r")
palindromes = []

for line in infile:
    line = line.rstrip() #remove trailing spaces

    #Compare line to its reverse
    if line == line[::-1] : #if equal add to list
        palindromes.append(line)

# display results
print("Found:", len(palindromes), "palindromes:")
for word in palindromes:
    print(word)
```



# Loop over files

```
>>> %Run palindrome.py
```

```
Found: 89 palindromes.
```

```
a
```

```
aha
```

```
ama
```

```
ana
```

```
anna
```

```
bb
```

```
bib
```

```
bob
```

```
. . . .
```

# Possible errors

- If the file we are trying to open does not exist an error will be raised.
- Example:

```
>>> infile = open('myData.txt')
```

```
Traceback (most recent call last):
```

```
File "<pyshell>", line 1, in <module>
```

```
FileNotFoundError: [Errno 2] No such file or  
directory: 'myData.txt'
```



# Possible errors

- If the file we are trying to read (as text) does not contain plain text, an error will be raised.
- Example:

```
>>> infile = open('/CS171/square.png')
```

```
>>> infile.readlines()
```

```
Traceback (most recent call last):
```

```
  File "<pyshell>", line 1, in <module>
```

```
    File
```

```
"/Applications/Thonny.app/Contents/Frameworks/Python.framework/Versions/3.7/Resources/Python.app/Contents/MacOS/../../../../../../../../Python.framework/Versions/3.7/lib/python3.7/codecs.py", line 322, in decode
```

```
    (result, consumed) = self._buffer_decode(data, self.errors, final)
```

```
UnicodeDecodeError: 'utf-8' codec can't decode byte 0x89 in position 0: invalid start byte
```

# How to open a file for writing

We must explicitly indicate that the file will be open for writing:

`open(filename, mode)`

- *filename*: the name (and path) of the file to be opened
- *mode* is a string that says what you want to do with the file.
  - **w** is for overwriting the file
  - **a** is for appending content to the file

Examples:

```
outFile = open('results.txt' , 'w')
```

```
outFile = open('results.txt' , 'a')
```

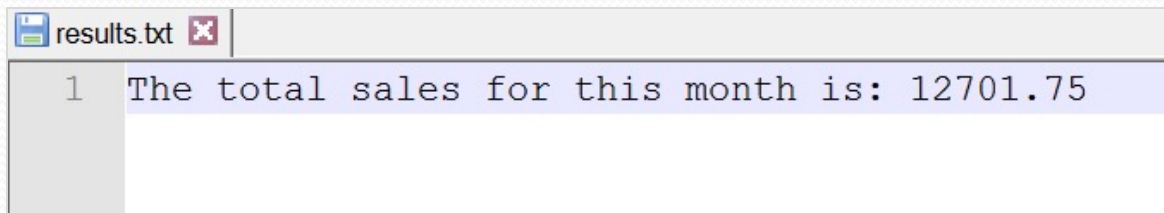


# Writing into Files

- To write into a file we use the `write(myString)` method
- This method only writes a string to the file.
- If you are writing numbers (numeric values such as integers or floating-point numbers) to the file, then you need to convert them from the numeric format to the string format using the `str` function.

# Examples

```
>>> fileOut = open('C:\\Users\\CS171\\results.txt', 'w')
>>> fileOut.write('The total sales for this month is: ')
>>> fileOut.write(str(total))
>>> fileOut.close()
```





# The OS Module

- The OS module offers a number of powerful capabilities for dealing with files, e.g., renaming files, finding out when a file was last modified, and so on.
- We start accessing the OS module by typing:  
`import os`
- To avoid portability issues (Windows vs. Mac/Linux) we can use `os.path` functions to handle file paths.

# The OS Module

- `os.path.join()` concatenates the arguments using the correct path separator for the current operating system.
- Example:

```
>>> path = os.path.join('CS171', 'Examples', 'loops.py')
>>> path
'CS171\\Examples\\loops.py' #in Windows
'CS171/Examples/loops.py'  #in Mac / Linux
```
- In Windows, if you need the drive (as a letter) in a full path, you must include the separator with the letter:
- Example:

```
>>> path = os.path.join('C:\\', 'CS171', 'Examples', 'loops.py')
>>> path
'C:\\CS171\\Examples\\loops.py'
```



# The OS Module

- `os.path.sep` can be used in combination with the string method `split()` to separate a path into its individual components (tokens)
  - The result is a list of strings, with the components of the path
  - `os.path.sep` stores the path separator for the operating system in use.
- Example:

```
>>> path = 'C:\\CS171\\Examples\\loops.py'
>>> tokens = path.split(os.path.sep)
>>> tokens
['C:', 'CS171', 'Examples', 'loops.py']
```

# More os.path functions

- `os.path.split(aPath)` splits *aPath* into a 2-item tuple (head, tail), where tail is the last token in the *aPath* string and head is everything else.
- `os.path.exists(aPath)` returns **True** if *aPath* exists, else returns **False**.
- `os.path.isfile(aPath)` returns **True** if *aPath* is an existing file, and **False** otherwise (e.g., *aPath* is a directory).



# Examples

```
>>> path = r'C:\Users\Desktop\CS 171'
>>> dirAndFile = os.path.split(path)
>>> dirAndFile
('C:\\Users\\Desktop', 'CS 171')
>>> pathExists = os.path.exists(path)
>>> pathExists
True
>>> isAFile = os.path.isfile(path)
>>> isAFile
False
>>> isAFile = os.path.isfile(r'C:\Users\Desktop\CS 171\Loops.py')
>>> isAFile
True
```

# The `listdir()` function

- `os.listdir(path)` returns a list with the base filename and suffix of the files in the input directory

- Example:

```
>>> files = os.listdir(r'C:\Desktop\CS 171')
>>> files
['bmiWithFunctions.py', 'functionsDemo.py',
 'Loops.py', 'selection.py']
```



# The `getcwd()` function

- `os.getcwd()` This function returns a string which represents the current working directory.
- Example:

```
>>> import os
>>> currentDir = os.getcwd()
>>> print(currentDir)
/Users/Documents/CS171/Fall2020/Week7
```

# Checking that a file exists

```
import os
currentDir = os.getcwd()
if 'sales.txt' in os.listdir(currentDir):
    print ('File is here')
else :
    print ('Error: file does not exist')
```



# Working with binary files

```
# open a binary file for reading
fileName = input("Enter name of a non-text file: ")

# open a binary file for reading
binFile = open(fileName, "rb")
counter = 1
byte = binFile.read(1)  #read 1st byte

# read 1 byte at the time, until there are no bytes left to read
while byte != b"" :
    byte = binFile.read(1)
    print(str(counter) + ": " + str(byte))
    counter += 1

# close file
binFile.close()
```

# Working with binary files

```
>>> %Run binary_read.py
```

```
Enter name of file: word_doc.docx
```

```
1: b'K'
```

```
2: b'\x03'
```

```
3: b'\x04'
```

```
4: b'\x14'
```

```
5: b'\x00'
```

```
6: b'\x06'
```

```
7: b'\x00'
```

```
8: b'\x08'
```

```
. . . . .
```

```
11792: b'\x00'
```

```
11793: b'\x00'
```

```
11794: b''
```