

Intro to Coding with Python—Working with Files

Dr. Ab Mosca (they/them)

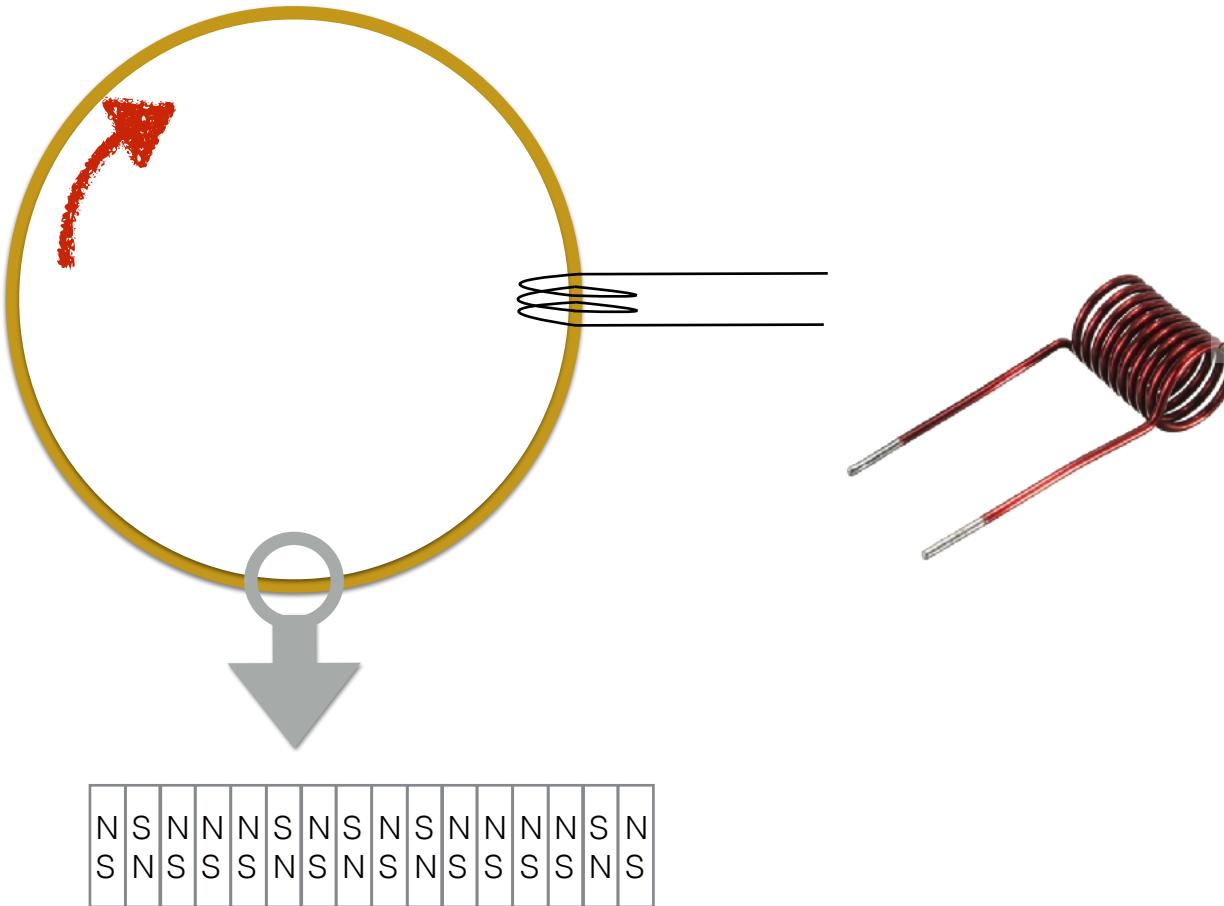
Plan for Today

- What is a file
- Read data in
- Writing data out

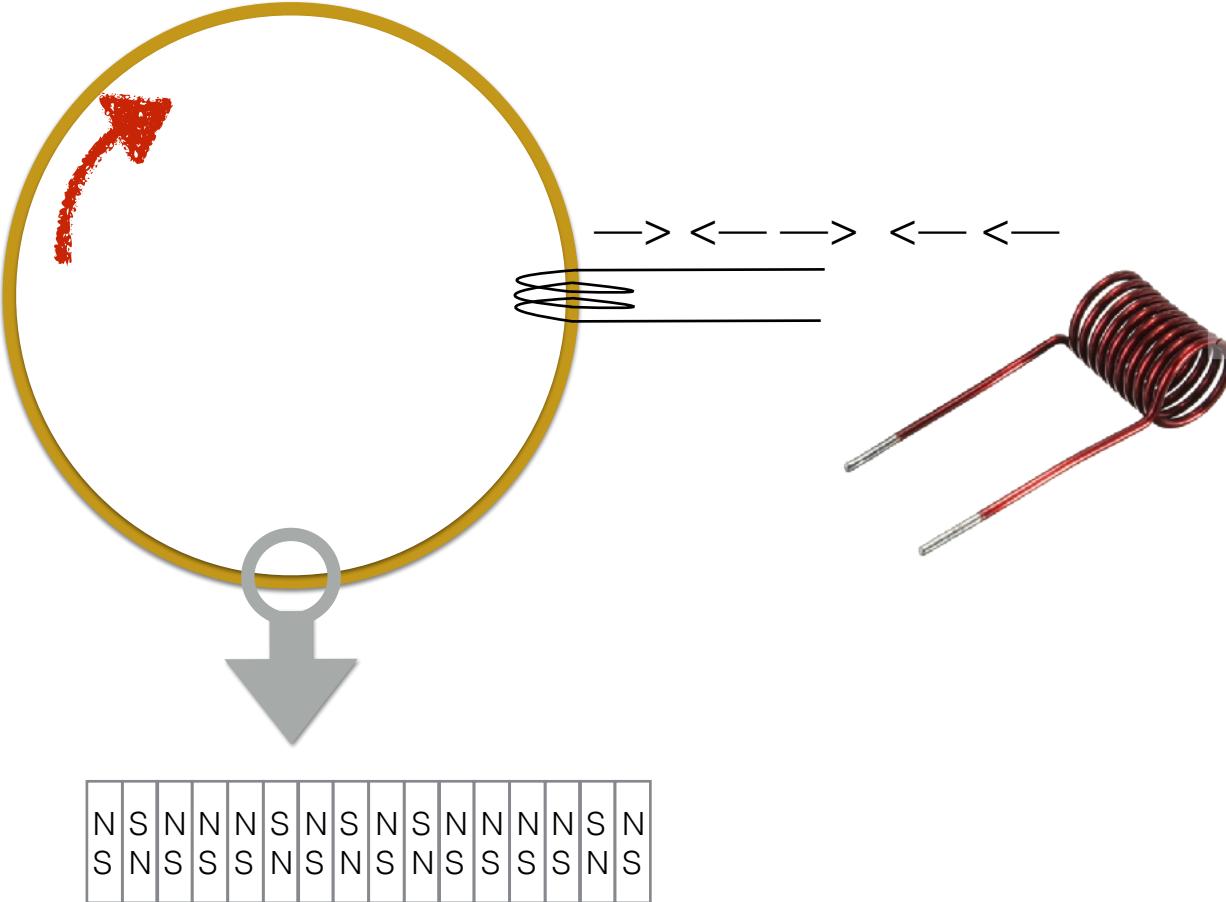
RECAP: how
does a hard
drive work?



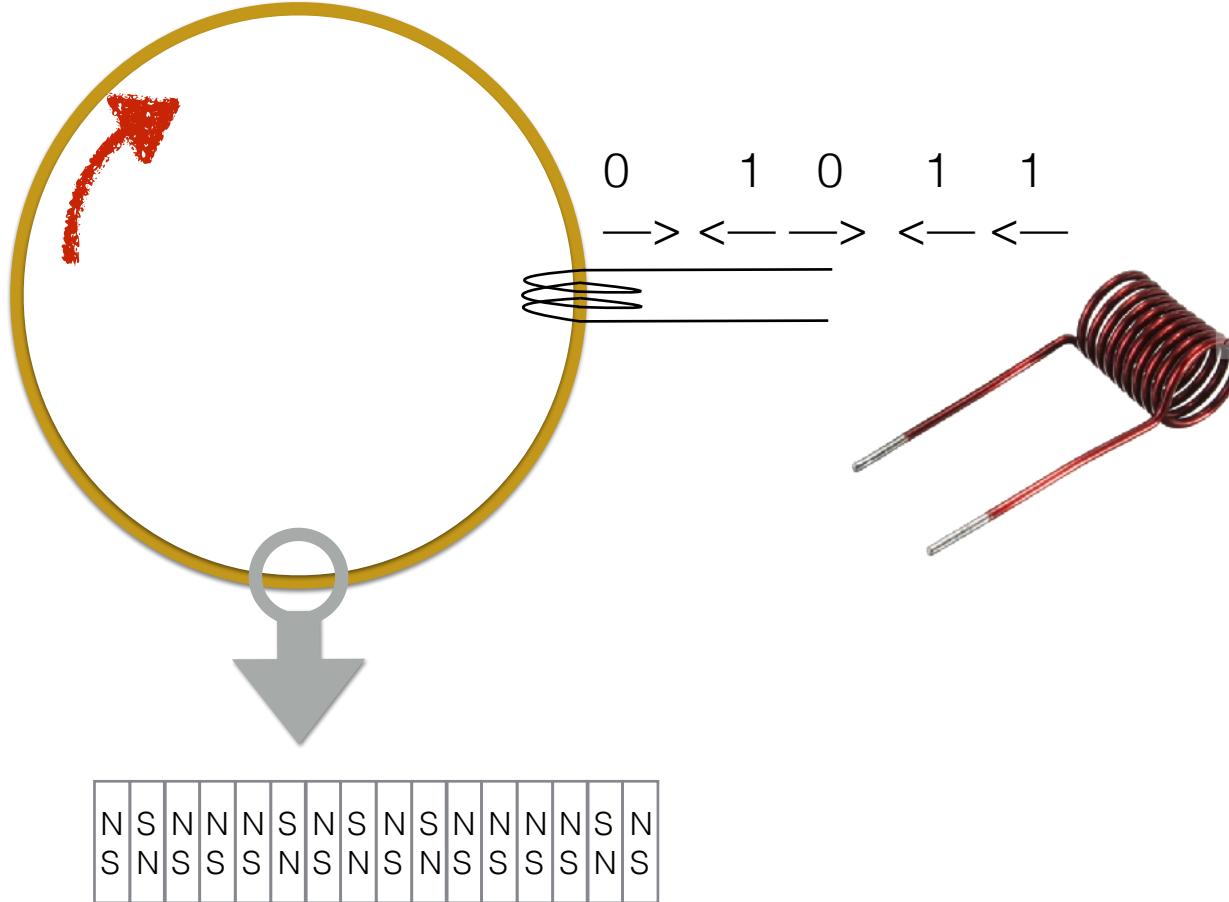
RECAP: how does a hard drive work?



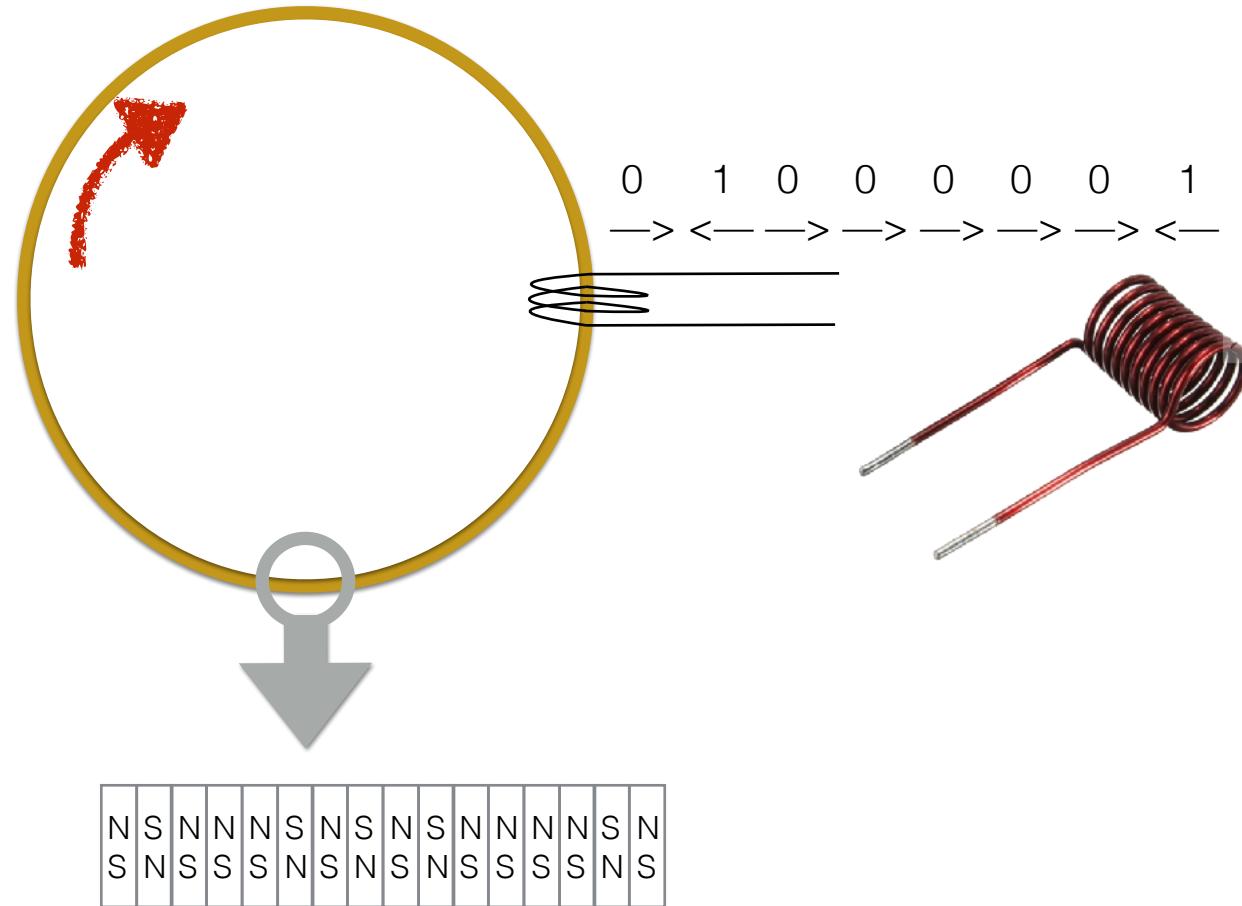
RECAP: how does a hard drive work?



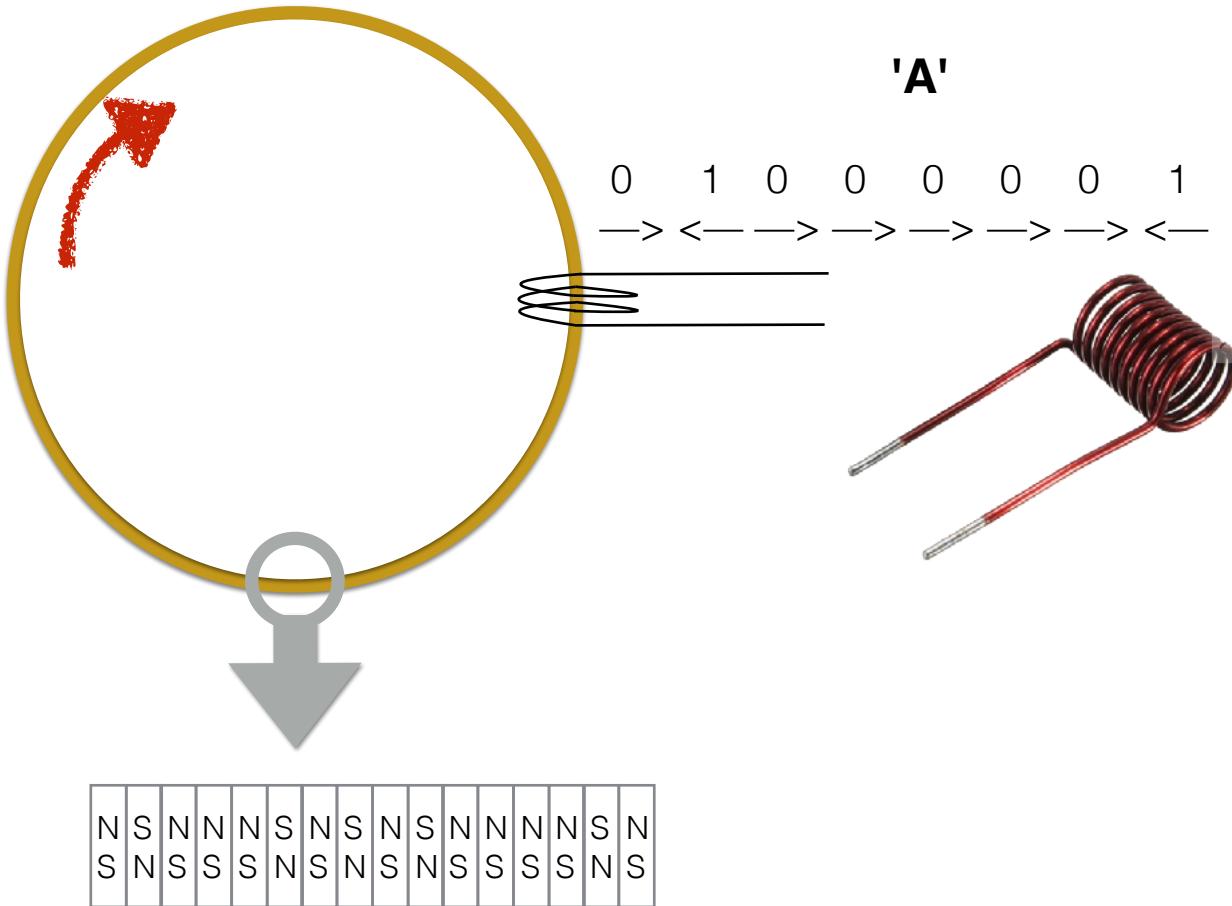
RECAP: how does a hard drive work?



RECAP: how does a hard drive work?



RECAP: how does a hard drive work?



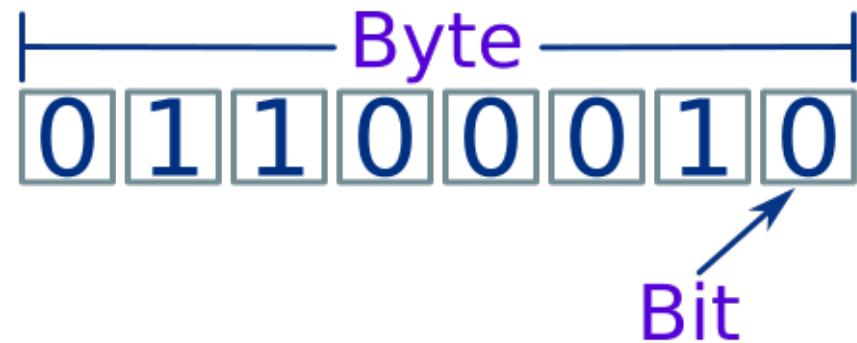
What is a file?

The screenshot shows a Mac OS X Finder window with the title bar "SCS-Noonan-CSC". The window displays a list of files and folders in a table format. The columns are "Name", "Date Modified", "Size", and "Kind". The "Name" column includes icons representing the file type (e.g., PDF, HTML, RMD, YAML). The "Date Modified" column shows the last update time for each item. The "Size" column indicates the file size, and the "Kind" column identifies the file type. The "Favorites" sidebar on the left lists "jcrouser", "Desktop", "Google Drive", "Dropbox", "Applications", "Documents", and "Downloads". The "Devices" and "Shared" sections are empty. The "Tags" section contains a single entry: "All...".

| | Name | Date Modified | Size | Kind |
|--|--|------------------------|-----------|-------------|
| | _config.yml | Jan 23, 2018, 2:20 PM | 29 bytes | TextWr...un |
| | _site.yml | Yesterday, 12:12 AM | 1 KB | TextWr...un |
| | build_site.R | May 17, 2018, 11:04 PM | 280 bytes | R Source F |
| | Course Syllabus - I...CS Summer 2018 | Jun 7, 2018, 10:32 AM | 134 KB | Adobe...cu |
| | Course Syllabus - I...mmer 2018 v2.pdf | Jun 7, 2018, 10:37 AM | 132 KB | Adobe...cu |
| | index.html | Yesterday, 11:02 AM | 8 KB | HTML |
| | index.Rmd | Jun 29, 2018, 8:40 AM | 1 KB | R Mark...wr |
| | labs | Yesterday, 10:03 AM | -- | Folder |
| | lectures | Today, 8:04 PM | -- | Folder |
| | README.html | Yesterday, 11:02 AM | 8 KB | HTML |
| | README.md | Jan 23, 2018, 2:20 PM | 1 KB | Markd...cur |
| | Sample Course Syllabus Outline | Jun 7, 2018, 10:31 AM | 34 KB | Micros...(d |
| | schedule.html | Yesterday, 11:02 AM | 11 KB | HTML |
| | schedule.Rmd | Today, 7:29 PM | 3 KB | R Mark...wr |
| | SCS Noonan - Intr...te Biography.gform | Jun 11, 2018, 9:22 AM | 176 bytes | Google for |
| | site_libs | Yesterday, 11:02 AM | -- | Folder |
| | style.css | Jan 23, 2018, 2:20 PM | 26 bytes | TextWr...un |
| | syllabus.html | Yesterday, 11:02 AM | 10 KB | HTML |
| | syllabus.Rmd | Jun 7, 2018, 10:38 AM | 3 KB | R Mark...wr |

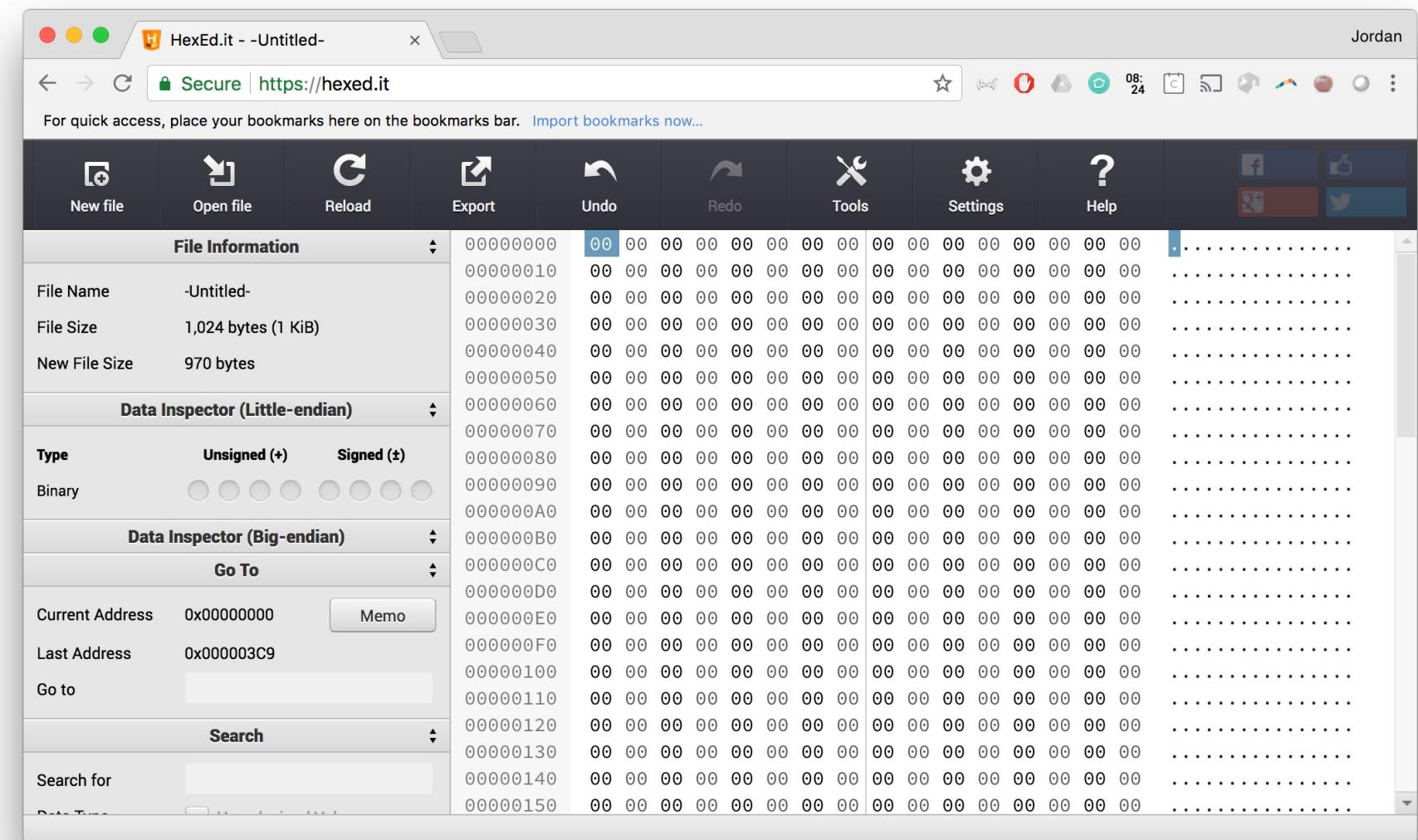
Working with files

- Containers of **bits**, organized into **bytes**



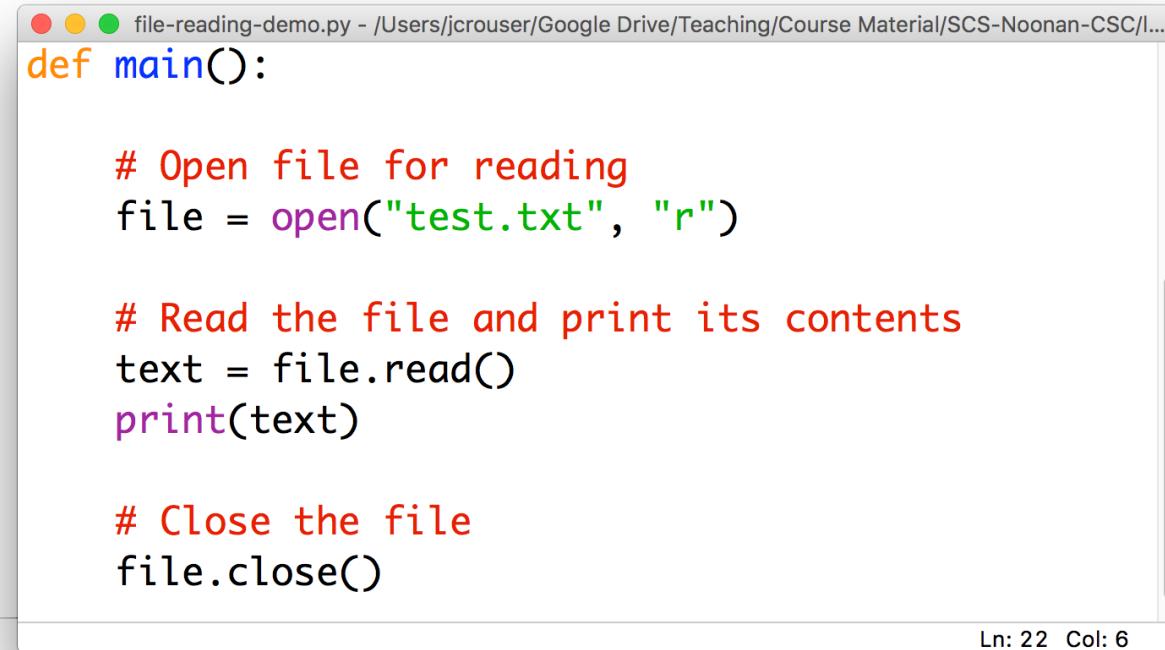
- Could represent text, images, music, movies, programs, applications, list of files (folders)... but underneath, they're all the same: os and 1s
- We'll start by playing with text files

Demo: looking inside a text file



Reading a text file

- In order to bring data stored in a text file into a Python program, we need to **read** it:



A screenshot of a code editor window titled "file-reading-demo.py". The code is as follows:

```
def main():

    # Open file for reading
    file = open("test.txt", "r")

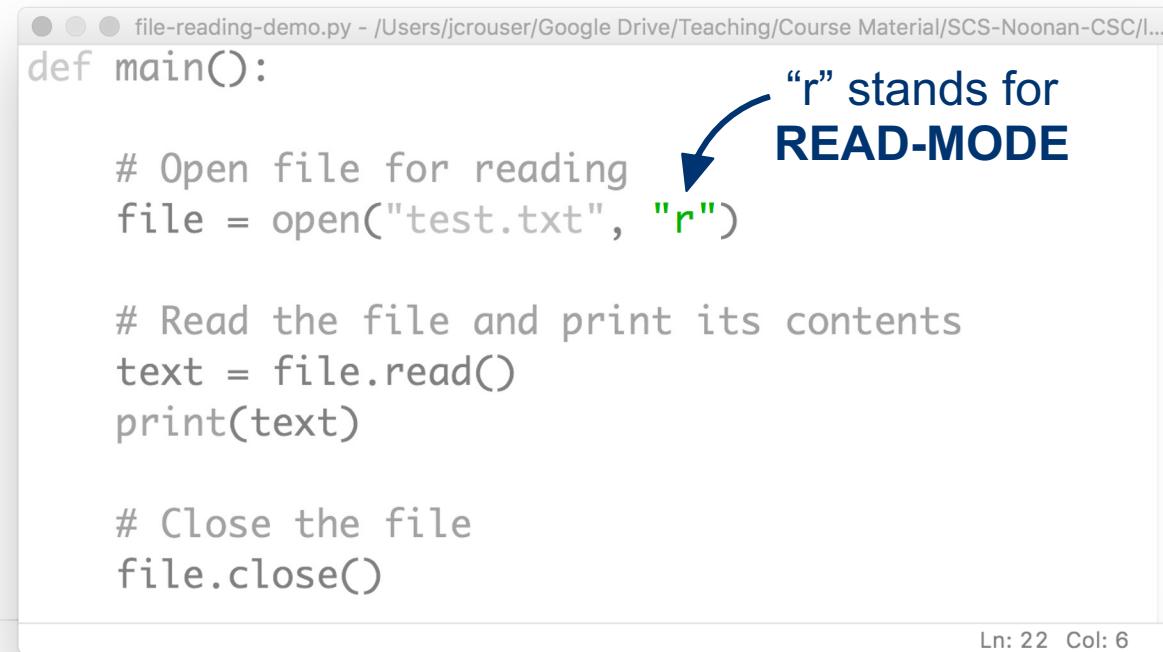
    # Read the file and print its contents
    text = file.read()
    print(text)

    # Close the file
    file.close()
```

The status bar at the bottom right shows "Ln: 22 Col: 6".

Reading a text file

- In order to bring data stored in a text file into a Python program, we need to **read** it:



The image shows a screenshot of a code editor window titled "file-reading-demo.py". The code is as follows:

```
def main():
    # Open file for reading
    file = open("test.txt", "r")

    # Read the file and print its contents
    text = file.read()
    print(text)

    # Close the file
    file.close()
```

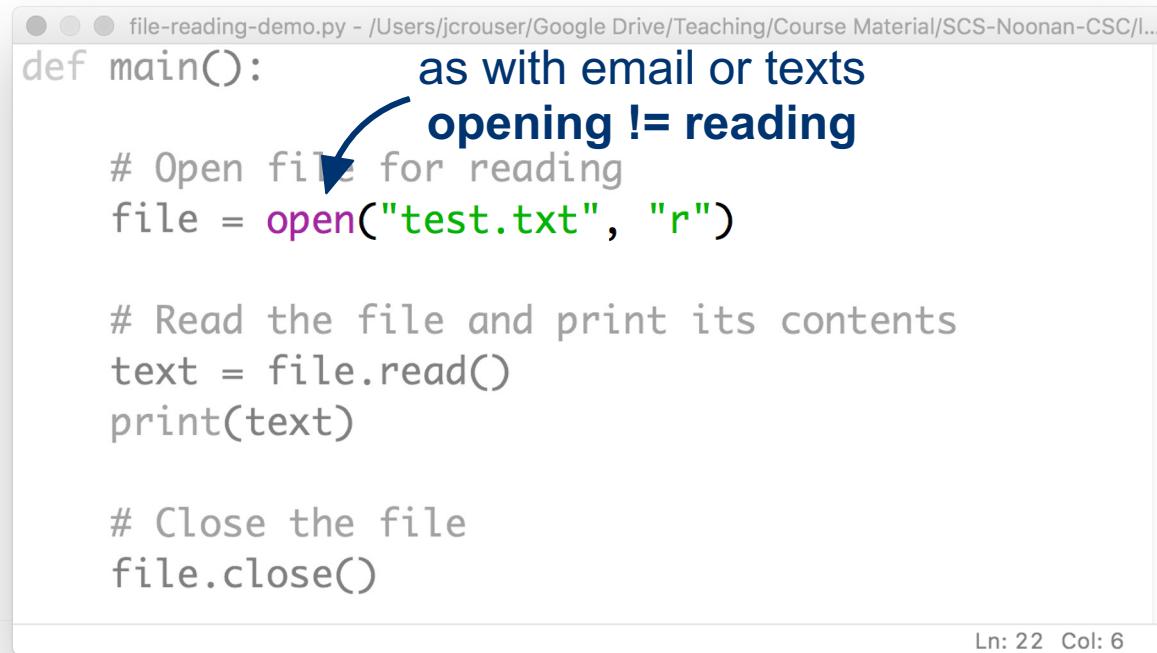
A green arrow points from the text "r" in the line `file = open("test.txt", "r")` to the text "READ-MODE" in the annotation.

“r” stands for
READ-MODE

Ln: 22 Col: 6

Reading a text file

- In order to bring data stored in a text file into a Python program, we need to **read** it:



The image shows a screenshot of a code editor window titled "file-reading-demo.py". The code is as follows:

```
def main():
    # Open file for reading
    file = open("test.txt", "r")

    # Read the file and print its contents
    text = file.read()
    print(text)

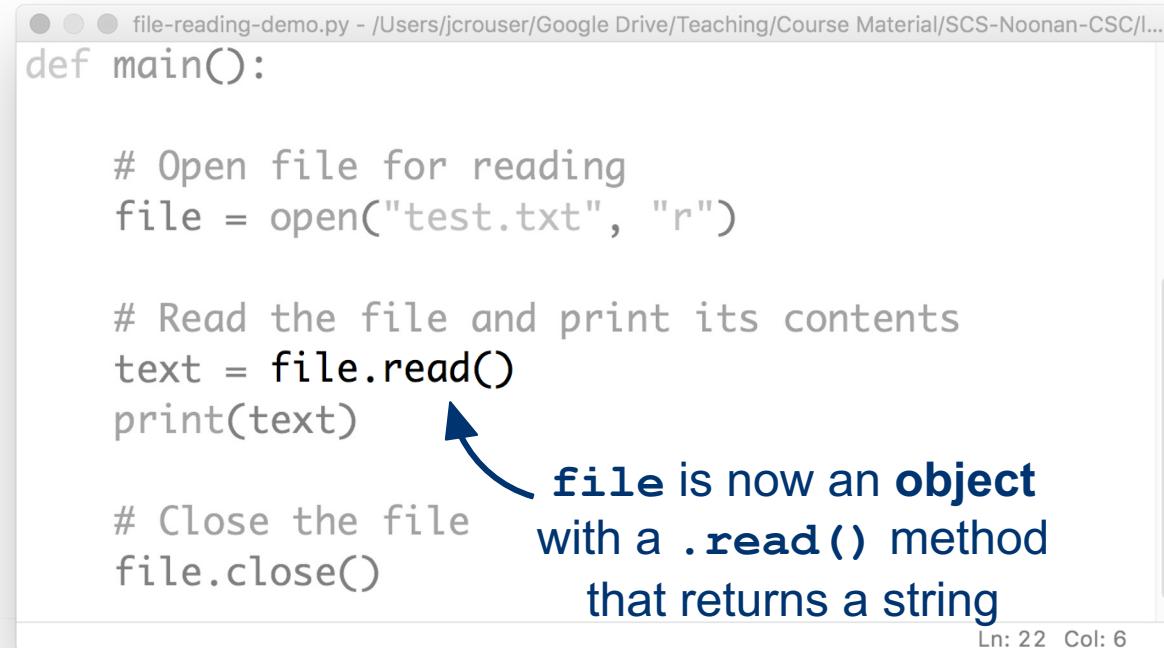
    # Close the file
    file.close()
```

A blue annotation has been added to the code. It consists of the text "as with email or texts" in blue, followed by "opening != reading" in blue, with a blue arrow pointing from the word "opening" to the opening parenthesis of the `open` function call.

Ln: 22 Col: 6

Reading a text file

- In order to bring data stored in a text file into a Python program, we need to **read** it:



A screenshot of a code editor window titled "file-reading-demo.py". The code defines a main() function that opens a file named "test.txt" in read mode, reads its contents into a variable named "text", and then prints "text". Finally, it closes the file. A blue arrow points from the explanatory text below to the line "text = file.read()".

```
def main():

    # Open file for reading
    file = open("test.txt", "r")

    # Read the file and print its contents
    text = file.read()
    print(text)

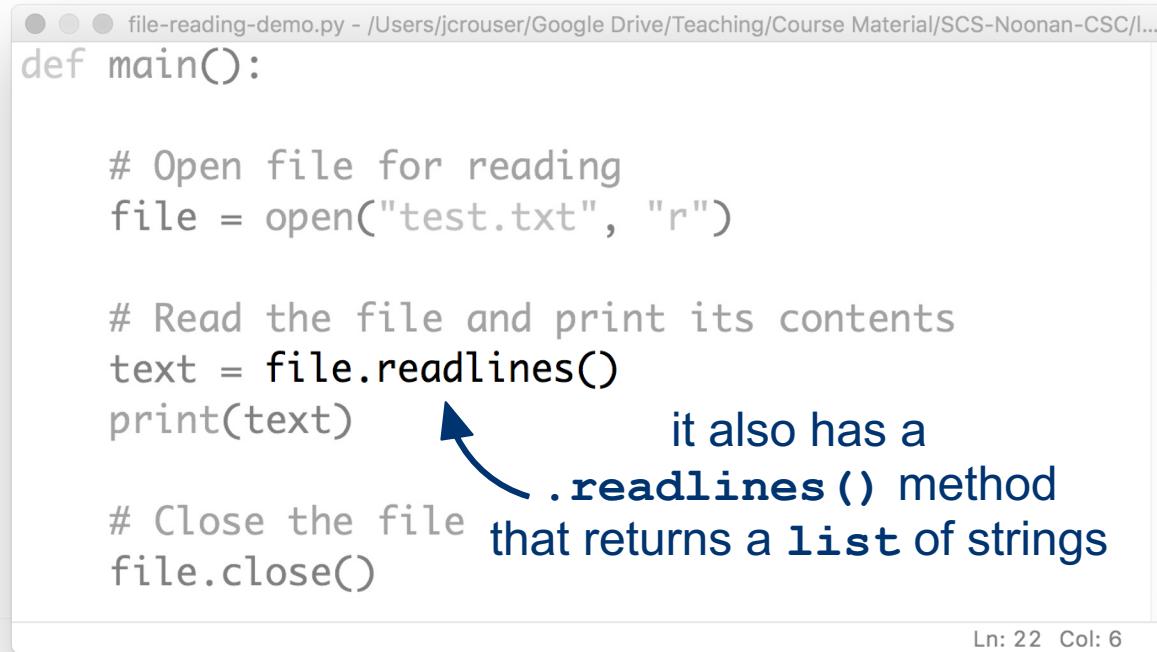
    # Close the file
    file.close()
```

file is now an **object** with a **.read()** method that returns a string

Ln: 22 Col: 6

Reading a text file

- In order to bring data stored in a text file into a Python program, we need to **read** it:



A screenshot of a code editor window titled "file-reading-demo.py". The code is as follows:

```
file-reading-demo.py - /Users/jcrouser/Google Drive/Teaching/Course Material/SCS-Noonan-CSC/I...
def main():

    # Open file for reading
    file = open("test.txt", "r")

    # Read the file and print its contents
    text = file.readlines()
    print(text)

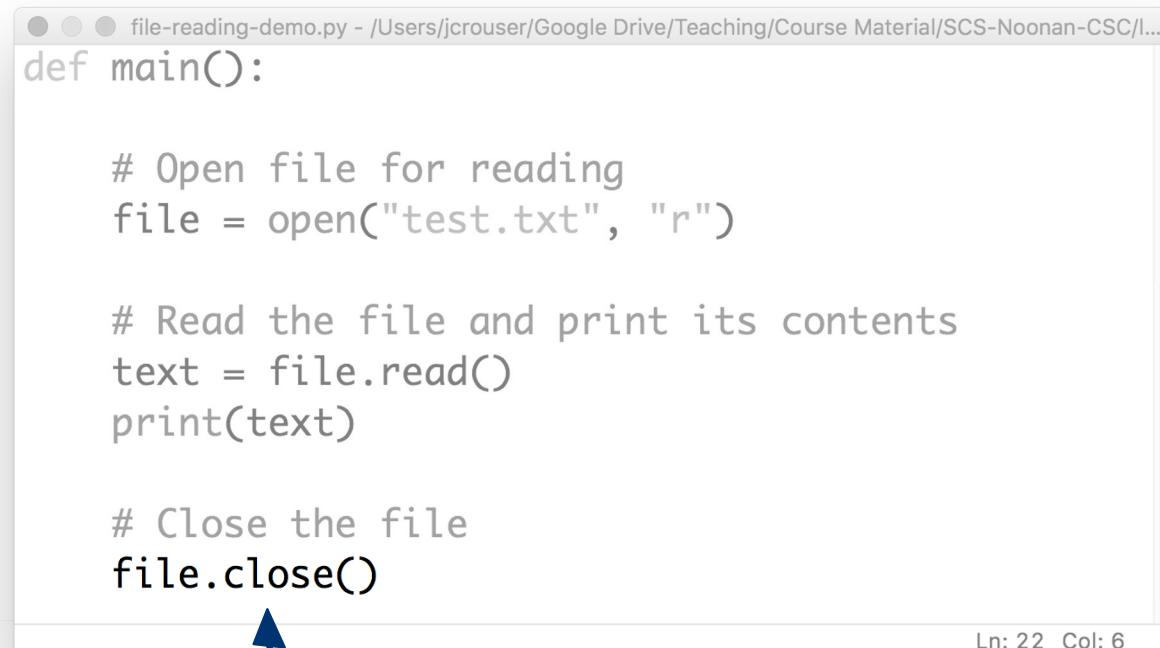
    # Close the file
    file.close()
```

The line `text = file.readlines()` is highlighted in blue. A blue arrow points from the text "it also has a .readlines() method that returns a list of strings" to the ".readlines()" part of the highlighted line.

Ln: 22 Col: 6

Reading a text file

- In order to bring data stored in a text file into a Python program, we need to **read** it:



A screenshot of a code editor window titled "file-reading-demo.py". The code is as follows:

```
def main():

    # Open file for reading
    file = open("test.txt", "r")

    # Read the file and print its contents
    text = file.read()
    print(text)

    # Close the file
    file.close()
```

The line "file.close()" is highlighted with a blue arrow pointing to it from the explanatory text below.

after you **.read()** or **.readlines()** a file, remember to **.close()** it

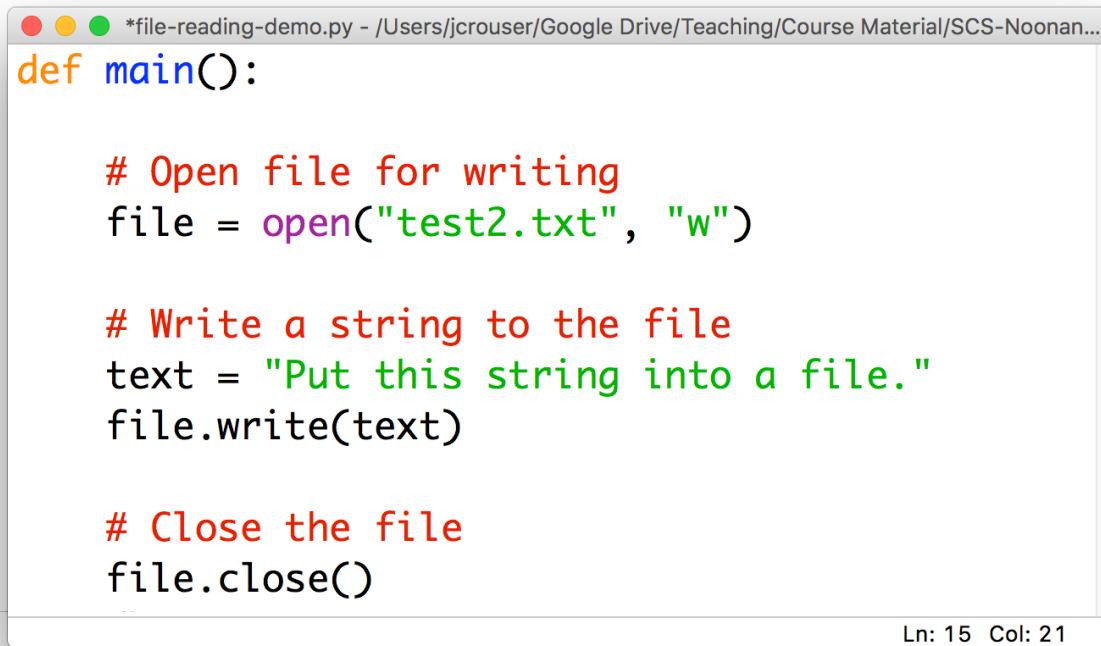
Key points for reading files

- Three-step process:
 1. `.open()`
 2. `.read()` or `.readlines()`
 3. `.close()`
- All three steps, always in that order
- If you want to `.read()` a file multiple times, you have to repeat the whole process

```
file = open("test.txt", "r")
text1 = file.read()
text2 = file.read() # Empty!
file.close()
```

Writing data to a text file

- The process looks very similar when we want to **write** data to a file:



A screenshot of a Python code editor window titled "*file-reading-demo.py - /Users/jcrouser/Google Drive/Teaching/Course Material/SCS-Noonan...". The code defines a main function that opens a file named "test2.txt" in write mode, writes a string to it, and then closes the file.

```
def main():

    # Open file for writing
    file = open("test2.txt", "w")

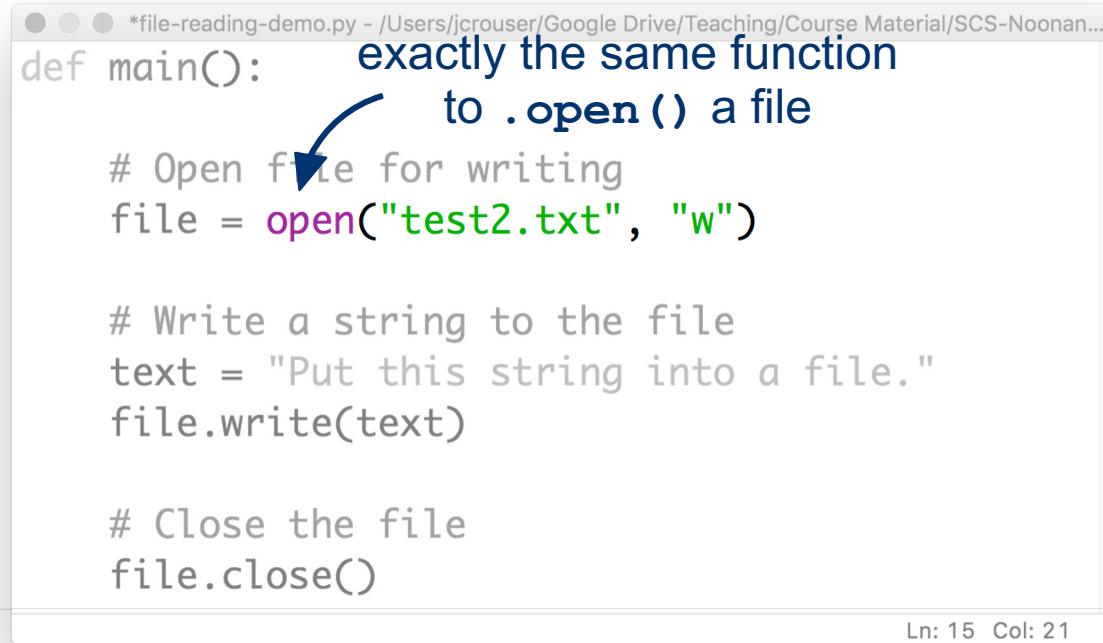
    # Write a string to the file
    text = "Put this string into a file."
    file.write(text)

    # Close the file
    file.close()

Ln: 15 Col: 21
```

Writing data to a text file

- The process looks very similar when we want to **write** data to a file:



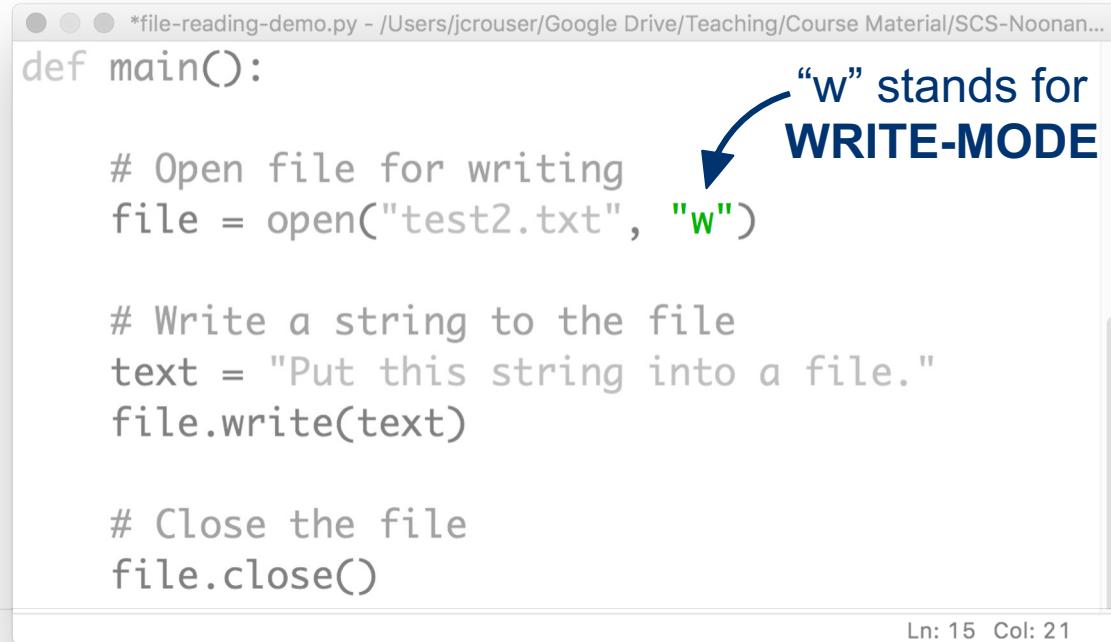
A screenshot of a Python code editor window titled '*file-reading-demo.py - /Users/jcrouser/Google Drive/Teaching/Course Material/SCS-Noonan...'. The code defines a function 'main()' that opens a file for writing, writes a string to it, and then closes the file. A blue annotation with an arrow points from the word 'file' in the 'open()' call to the word 'file' in the 'write()' call, with the text 'exactly the same function to .open() a file' written above the arrow. The status bar at the bottom right shows 'Ln: 15 Col: 21'.

```
def main():    exactly the same function
                  to .open() a file
    # Open file for writing
    file = open("test2.txt", "w")
    # Write a string to the file
    text = "Put this string into a file."
    file.write(text)
    # Close the file
    file.close()
```

Ln: 15 Col: 21

Writing data to a text file

- The process looks very similar when we want to **write** data to a file:



```
*file-reading-demo.py - /Users/jcrouser/Google Drive/Teaching/Course Material/SCS-Noonan...
def main():

    # Open file for writing
    file = open("test2.txt", "w")

    # Write a string to the file
    text = "Put this string into a file."
    file.write(text)

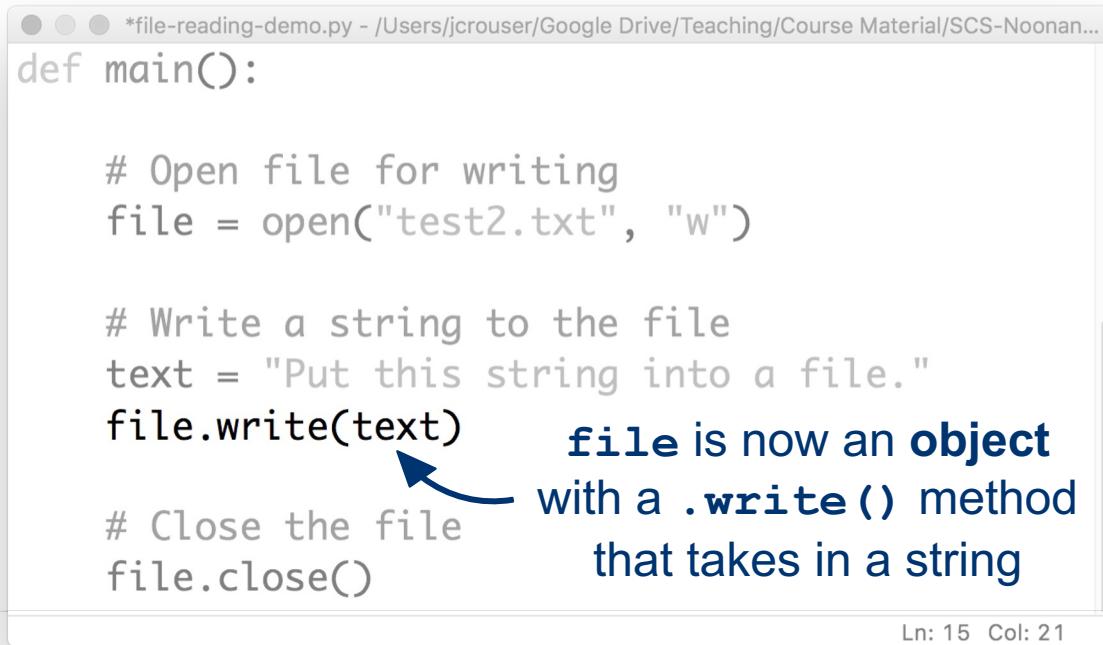
    # Close the file
    file.close()

Ln: 15 Col: 21
```

“w” stands for **WRITE-MODE**

Writing data to a text file

- The process looks very similar when we want to **write** data to a file:



```
*file-reading-demo.py - /Users/jcrouser/Google Drive/Teaching/Course Material/SCS-Noonan...
def main():

    # Open file for writing
    file = open("test2.txt", "w")

    # Write a string to the file
    text = "Put this string into a file."
    file.write(text)

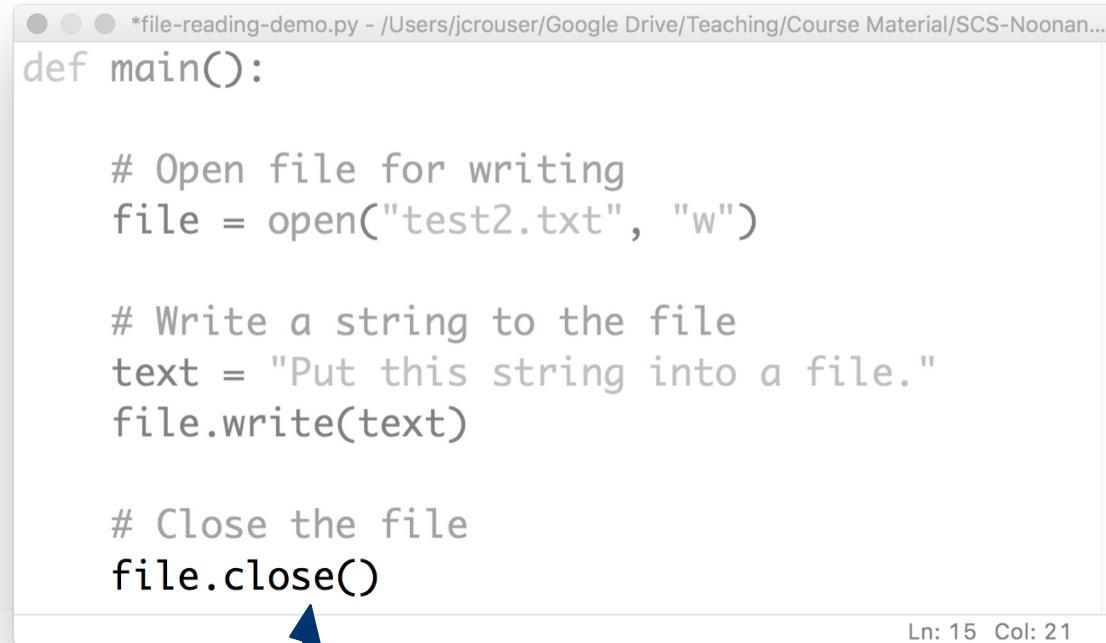
    # Close the file
    file.close()
```

Ln: 15 Col: 21

file is now an **object** with a **.write()** method that takes in a string

Writing data to a text file

- The process looks very similar when we want to **write** data to a file:



A screenshot of a code editor window titled '*file-reading-demo.py - /Users/jcrouser/Google Drive/Teaching/Course Material/SCS-Noonan...'. The code is as follows:

```
def main():

    # Open file for writing
    file = open("test2.txt", "w")

    # Write a string to the file
    text = "Put this string into a file."
    file.write(text)

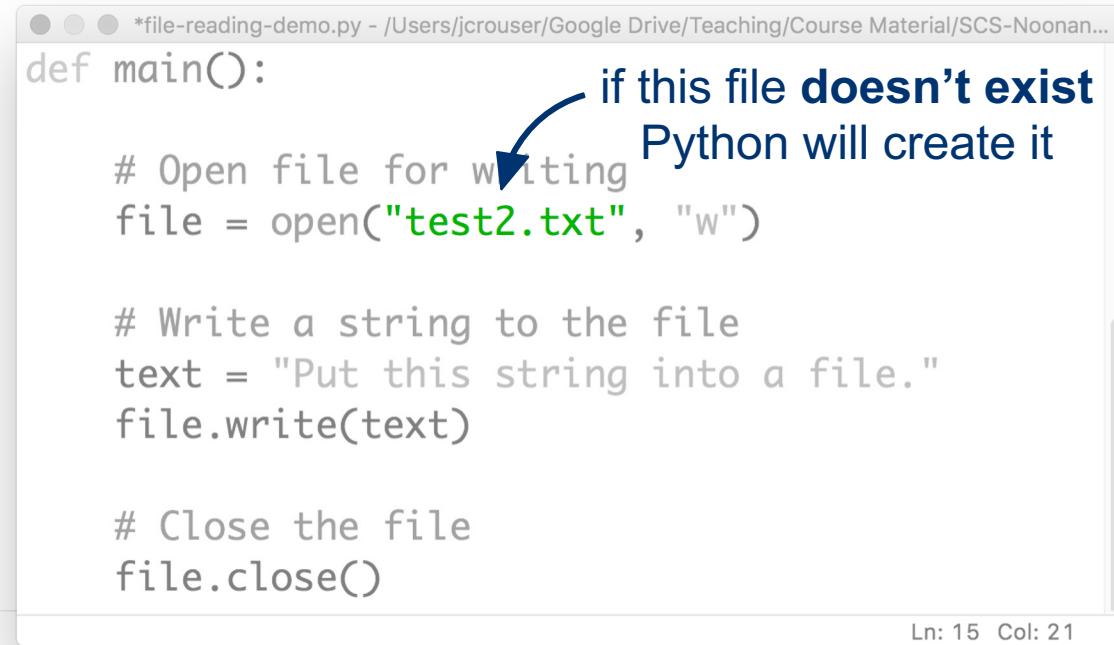
    # Close the file
    file.close()
```

Ln: 15 Col: 21

as before, after you `.write()` to a file
remember to `.close()` it

Writing data to a text file

- Some quirks to be aware of when **writing** to files:



The image shows a screenshot of a Python code editor window. The title bar reads "*file-reading-demo.py - /Users/jcrouser/Google Drive/Teaching/Course Material/SCS-Noonan...". The code in the editor is as follows:

```
def main():
    # Open file for writing
    file = open("test2.txt", "w")

    # Write a string to the file
    text = "Put this string into a file."
    file.write(text)

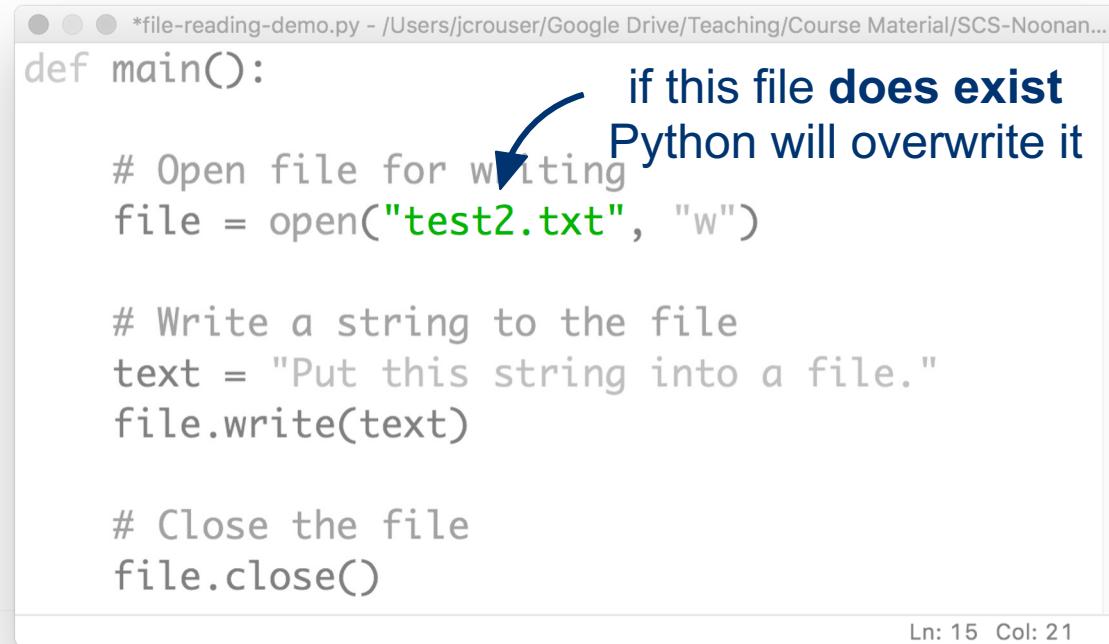
    # Close the file
    file.close()
```

A blue callout bubble with an arrow points from the text "if this file doesn't exist" to the file path "test2.txt" in the code. The text "if this file doesn't exist" is in blue, and "Python will create it" is in dark blue.

Ln: 15 Col: 21

Writing data to a text file

- Some quirks to be aware of when **writing** to files:



The screenshot shows a code editor window with the following Python script:

```
def main():
    # Open file for writing
    file = open("test2.txt", "w")

    # Write a string to the file
    text = "Put this string into a file."
    file.write(text)

    # Close the file
    file.close()
```

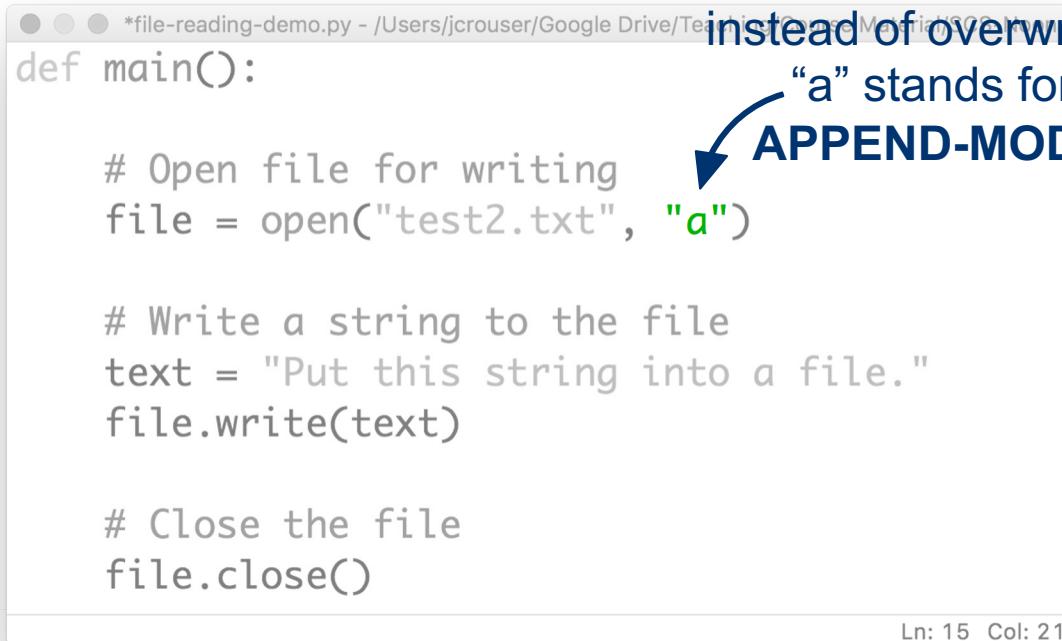
A blue arrow points from the text "if this file does exist" to the line "file = open("test2.txt", "w")".

if this file **does exist**
Python will overwrite it

Ln: 15 Col: 21

Writing data to a text file

- Some quirks to be aware of when **writing** to files:



```
*file-reading-demo.py - /Users/jcrouser/Google Drive/Teaching/Python/Material/07-File I/O
def main():

    # Open file for writing
    file = open("test2.txt", "a")

    # Write a string to the file
    text = "Put this string into a file."
    file.write(text)

    # Close the file
    file.close()

Ln: 15 Col: 21
```

if you want to
add to an existing file
instead of overwrite it

“a” stands for
APPEND-MODE

A blue arrow points from the word "a" in the code to the explanatory text "“a” stands for APPEND-MODE".

Key points for writing to files

- Three-step process:
 1. `.open()`
 2. `.write()`
 3. `.close()`
- Unlike `.read()`, you can `.write()` to an `.open()` file as many times as you want (appending each time)

```
file = open("test2.txt", "w")
file.write("Hello")
file.write("there!")
file.close()
```

- If you want a new line, you have to add it yourself! (`\n`)

15-minute exercise: read and write

Open the files.py demo on repl.it

Write a program that:

1. reads the file **horizontal.txt**
2. breaks it into individual words
3. and writes the words to a new file **vertical.txt**, each one on its own line

Discussion

What did you come up with?