

Lecture 30:

WORKING WITH FILES

CSC111: Introduction to CS through Programming

R. Jordan Crouser

Assistant Professor of Computer Science

Smith College

Announcements: Smithies in CS events

Alumni in CS Discuss Identity & Tech 6 pm - 7pm

REGISTER WITH QR CODE OR LINK BELOW

ASIAN SMITHIES IN CS
NOV 12TH, 2020



LGBTQ+ SMITHIES IN CS
NOV 13TH, 2020



[HTTPS://TINYURL.COM/Y5Y72HDR](https://tinyurl.com/y5y72hdr)

[HTTPS://TINYURL.COM/Y56VVHRJ](https://tinyurl.com/y56vvhrj)

BLACK SMITHIES IN CS
NOV 5TH, 2020



[HTTPS://TINYURL.COM/Y5NQTDL](https://tinyurl.com/y5njqtld)

1ST GEN SMITHIES IN CS
NOV 6TH, 2020



[HTTPS://TINYURL.COM/Y5ELYQ5X](https://tinyurl.com/y5elyq5x)

LATINX SMITHIES IN CS
NOV 19TH, 2020



[HTTPS://TINYURL.COM/Y6GHULMO](https://tinyurl.com/y6ghulmo)

INTERNATIONAL SMITHIES IN
CS
NOV 20TH, 2020



[HTTPS://TINYURL.COM/Y397Y2ZW](https://tinyurl.com/y397y2zw)

NON-TRADITIONAL SMITHIES IN CS
NOV 23RD, 2020



[HTTPS://TINYURL.COM/Y5UV0JK7](https://tinyurl.com/y5uv0jk7)



Announcements: auto-extension on A6

- This week is a rough one for a lot of us
- A6 now due **Monday 11/9** for everyone (+3 days)
- Take care of yourselves: remember that you have a free from of one lab and one assignment

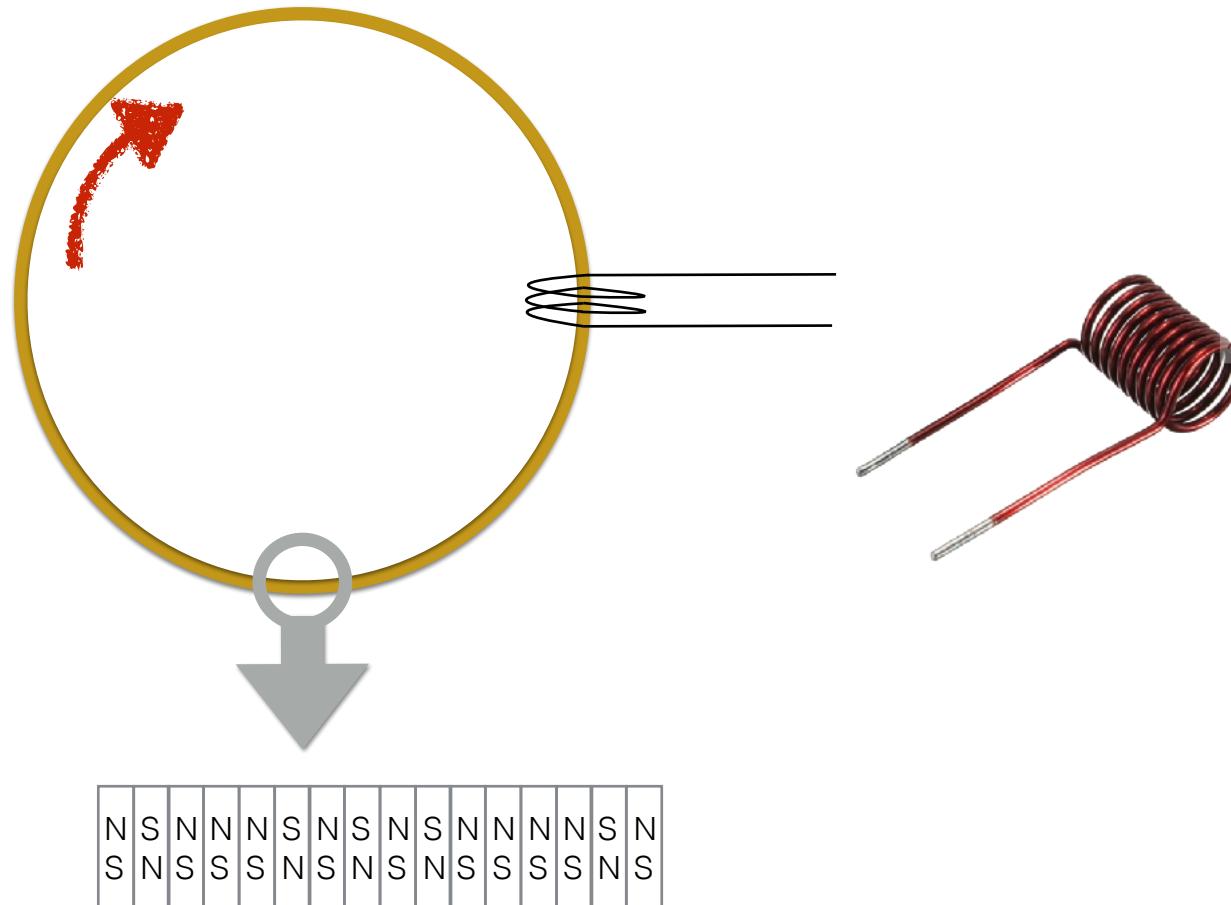
Outline

- ✓ Announcements
- Working with files
 - what is a file?
 - reading data in
 - writing data out
 - *some details about the final project*
- Lab: Sorting
- Life Skill #5: Code Diagrams
- Useful python packages

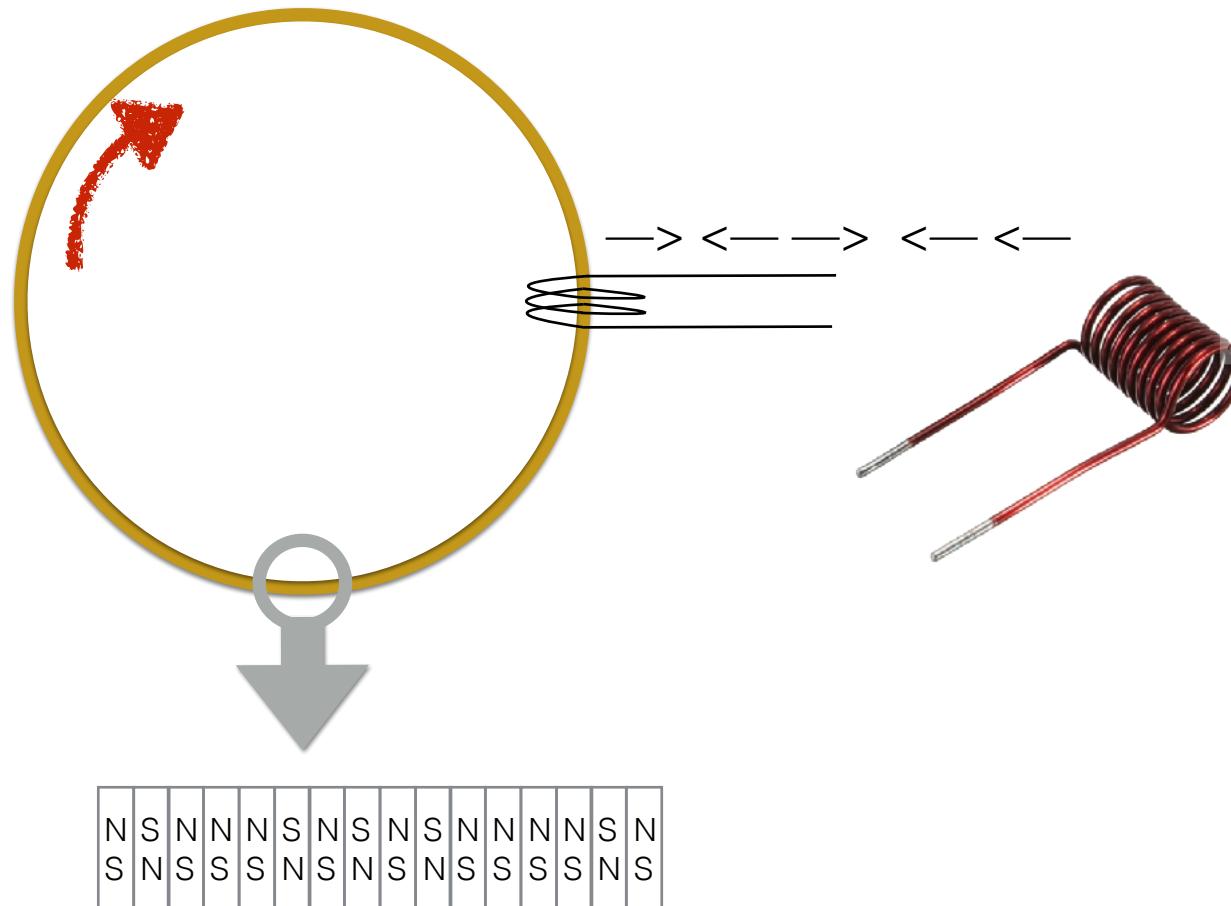
RECAP: how does a hard drive work?



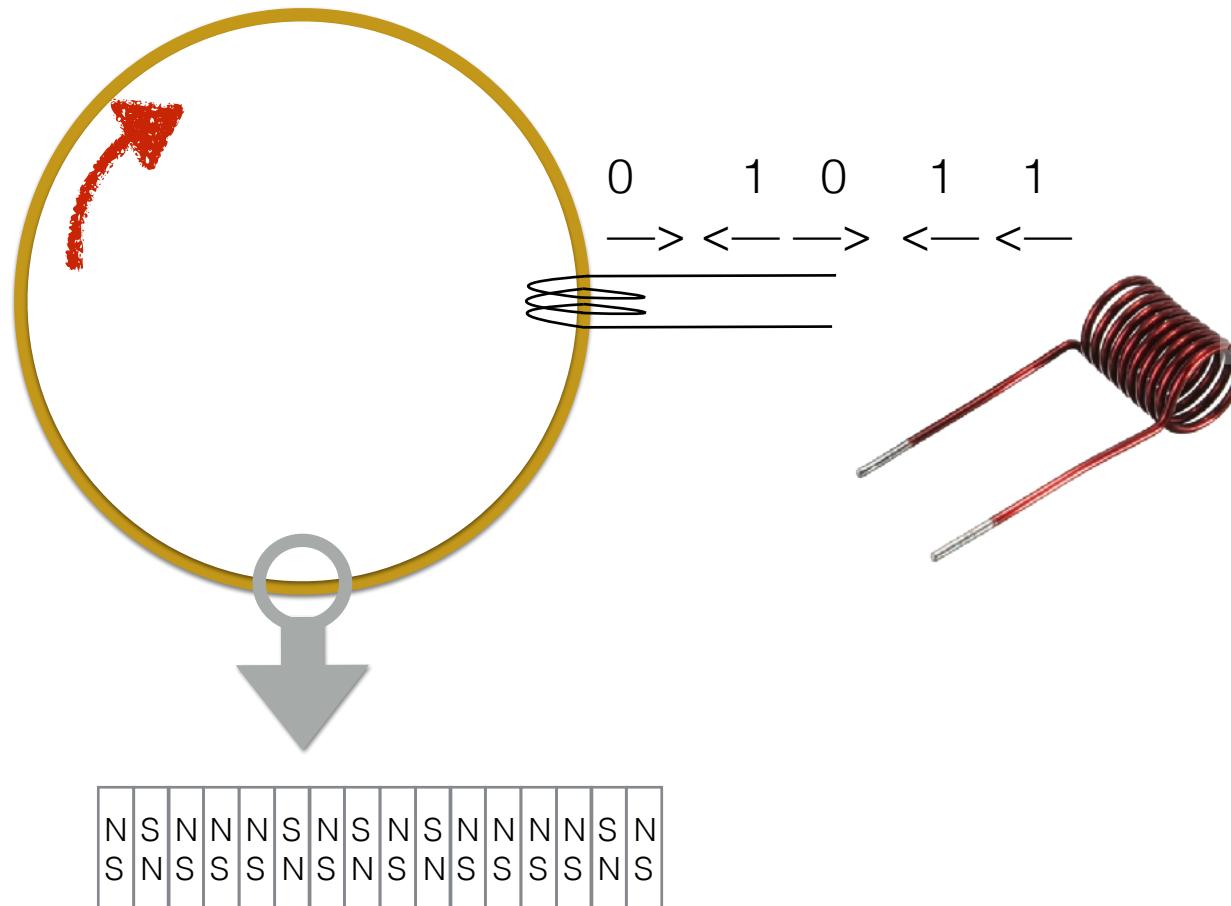
RECAP: how does a hard drive work?



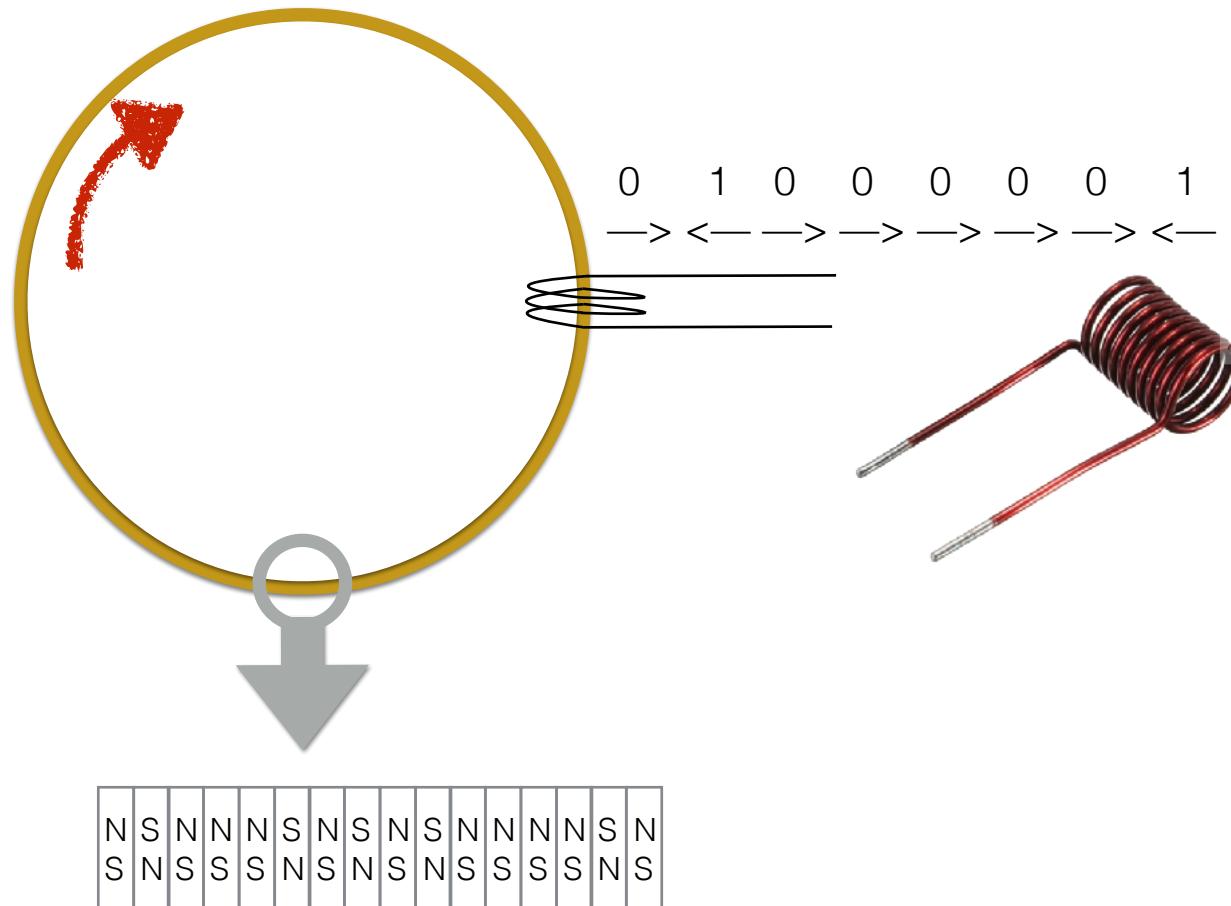
RECAP: how does a hard drive work?



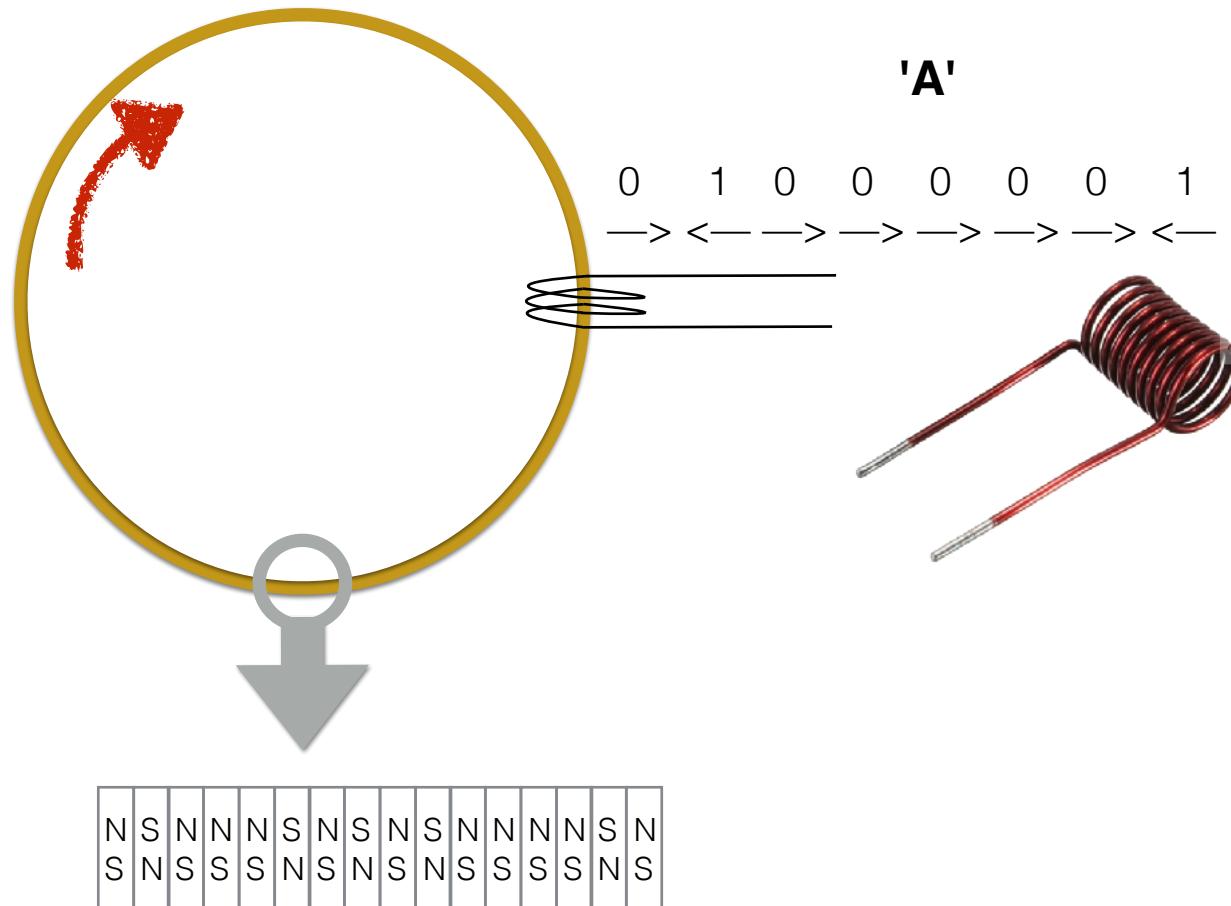
RECAP: how does a hard drive work?



RECAP: how does a hard drive work?



RECAP: how does a hard drive work?

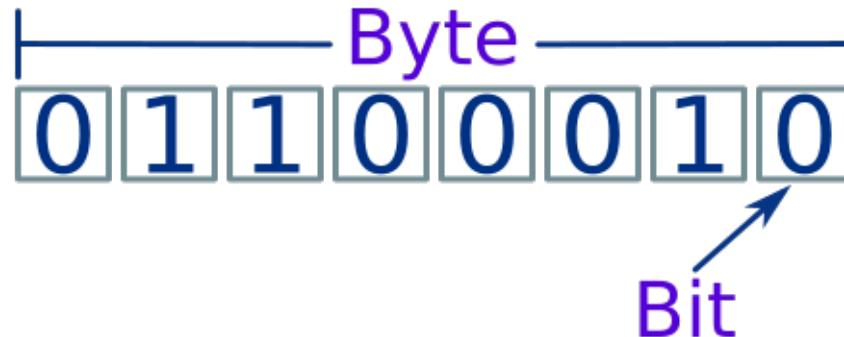


What is a file?

| SCS-Noonan-CSC | | | | |
|----------------|--|------------------------|-----------|-------------|
| Favorites | Name | Date Modified | Size | Kind |
| jcrouser | _config.yml | Jan 23, 2018, 2:20 PM | 29 bytes | TextWr...un |
| Desktop | _site.yml | Yesterday, 12:12 AM | 1 KB | TextWr...un |
| Google Drive | build_site.R | May 17, 2018, 11:04 PM | 280 bytes | R Source F |
| Dropbox | Course Syllabus - I...CS Summer 2018 | Jun 7, 2018, 10:32 AM | 134 KB | Adobe...cur |
| Applications | Course Syllabus - I...mmer 2018 v2.pdf | Jun 7, 2018, 10:37 AM | 132 KB | Adobe...cur |
| Documents | index.html | Yesterday, 11:02 AM | 8 KB | HTML |
| Downloads | 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 |
| Devices | Sample Course Syllabus Outline | Jun 7, 2018, 10:31 AM | 34 KB | Micros...(d |
| Shared | schedule.html | Yesterday, 11:02 AM | 11 KB | HTML |
| All... | schedule.Rmd | Today, 7:29 PM | 3 KB | R Mark...wr |
| Tags | SCS-Noonan-CSC | | 176 bytes | Google for |
| | syllabus.Rmd | | -- | Folder |
| | | | 26 bytes | TextWr...un |
| | | | 10 KB | HTML |
| | | | 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: 0s and 1s
- We'll start by playing with text files

Demo: looking inside a text file

The screenshot shows a hex editor window titled "HexEd.it - -Untitled-". The toolbar includes icons for New file, Open file, Reload, Export, Undo, Redo, Tools, Settings, and Help. The address bar shows "Secure | https://hexed.it" and the user "Jordan". A message at the top says, "For quick access, place your bookmarks here on the bookmarks bar. Import bookmarks now...".

File Information

| | |
|---------------|---------------------|
| File Name | -Untitled- |
| File Size | 1,024 bytes (1 KiB) |
| New File Size | 970 bytes |

Data Inspector (Little-endian)

| Type | Unsigned (+) | Signed (±) |
|--------|--------------|------------|
| Binary | • • • • • • | |

Data Inspector (Big-endian)

Go To

| | | |
|-----------------|------------|------|
| Current Address | 0x00000000 | Memo |
| Last Address | 0x000003C9 | |
| Go to | | |

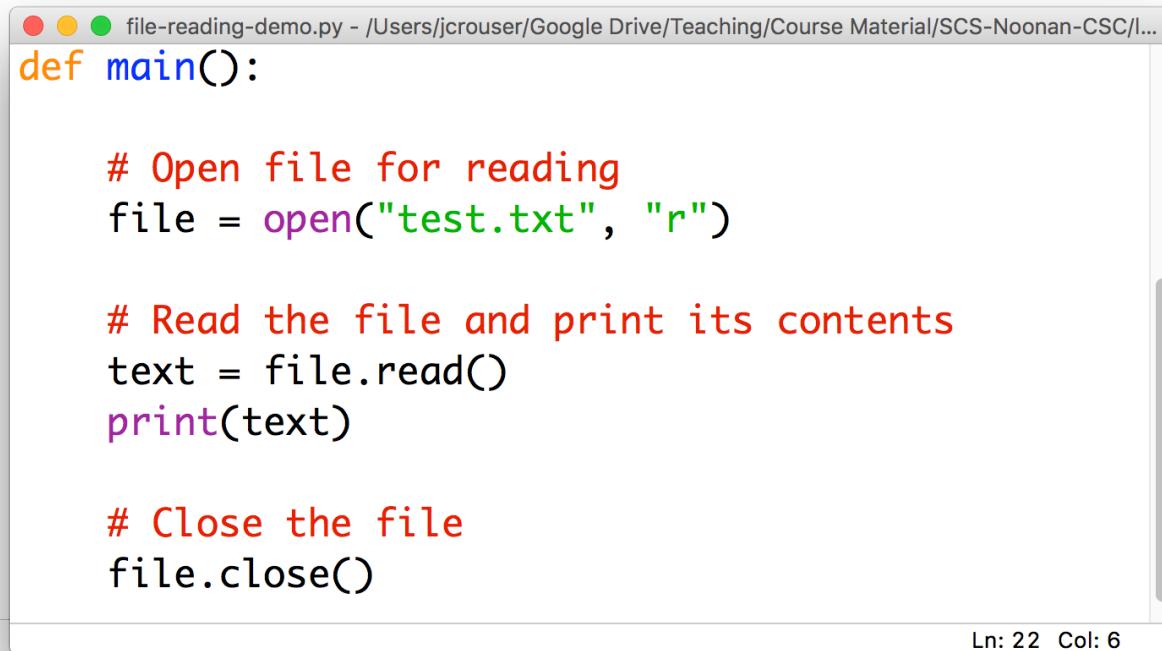
Search

| | |
|------------|--|
| Search for | |
|------------|--|

The main pane displays memory starting at address 0x00000000. The first few bytes are highlighted in blue: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00. The rest of the memory is shown as a series of zeros.

Reading a text file

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



The screenshot shows a code editor window with a title bar "file-reading-demo.py - /Users/jcrouser/Google Drive/Teaching/Course Material/SCS-Noonan-CSC/I...". The main area contains the following Python code:

```
def main():

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

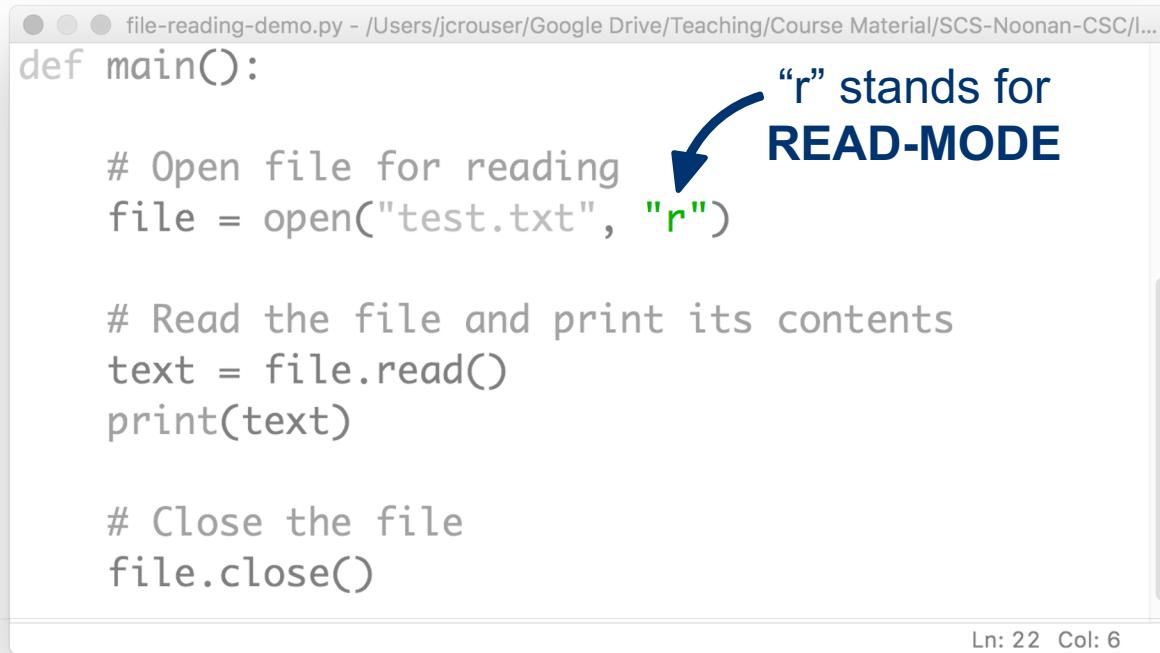
    # Read the file and print its contents
    text = file.read()
    print(text)

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

At the bottom right of the editor, there is a status bar with "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:

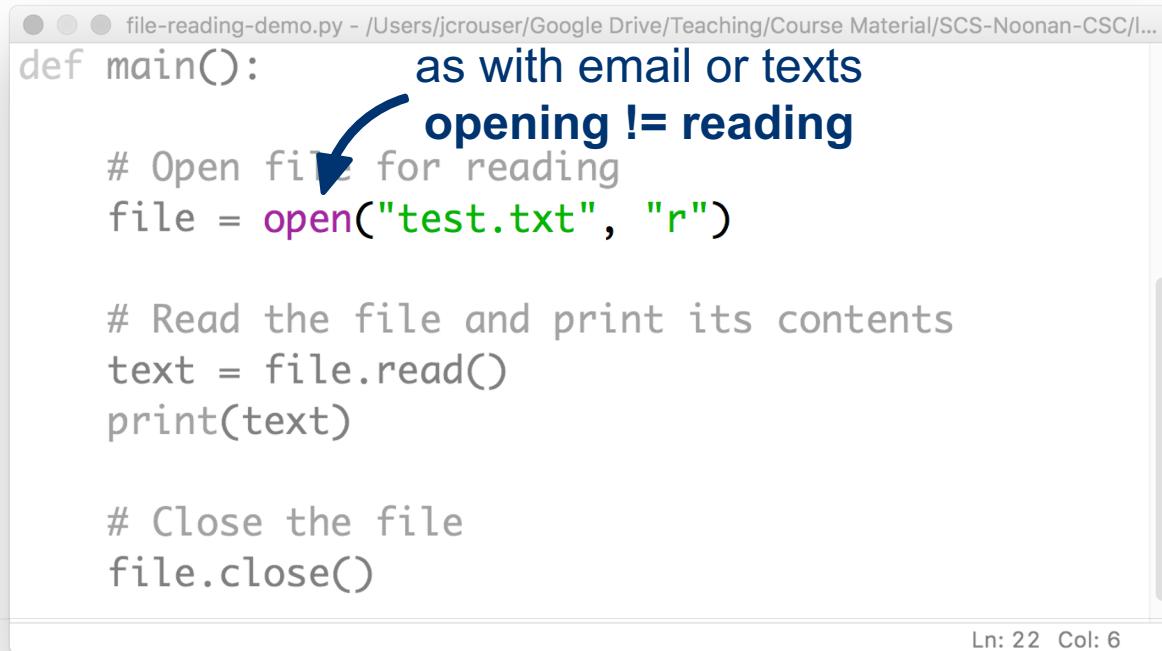


A screenshot of a code editor window titled "file-reading-demo.py". The code defines a main function that opens a file named "test.txt" in read mode ("r"), reads its contents, and prints them. It then closes the file. A green arrow points from the text "r" in the open command to a callout box containing the text "'r' stands for READ-MODE".

```
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()  
  
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**:



```
file-reading-demo.py - /Users/jcrouser/Google Drive/Teaching/Course Material/SCS-Noonan-CSC/I...
def main():
    as with email or texts
    opening != reading
    # Open file for reading
    file = open("test.txt", "r")

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

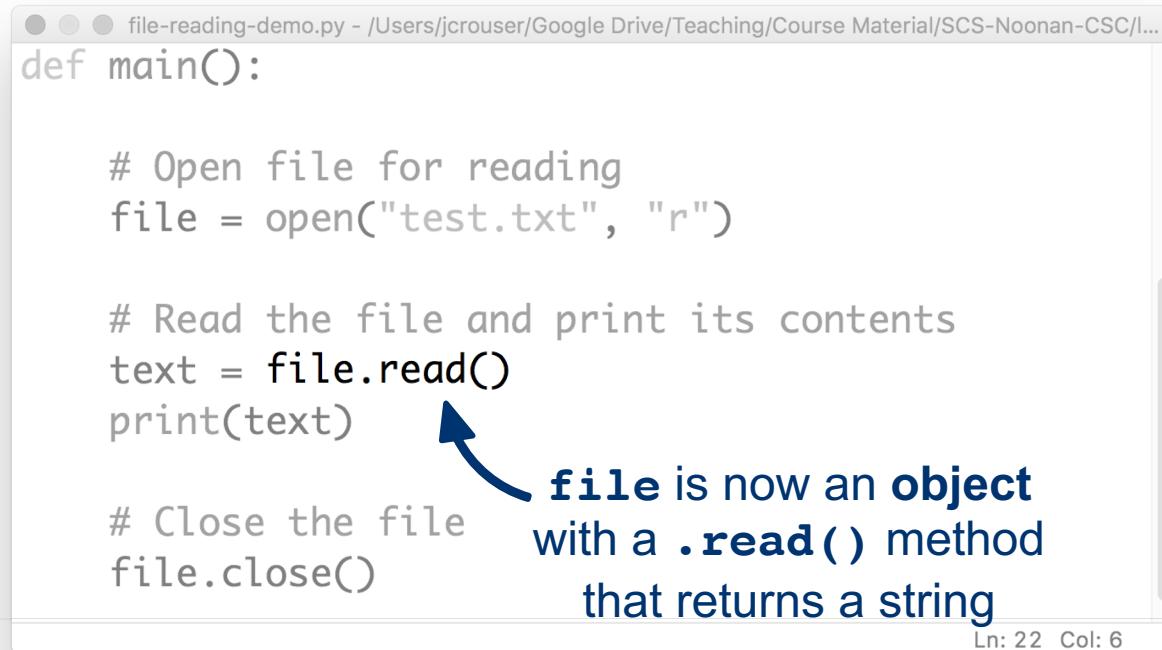
    # Close the file
    file.close()

Ln: 22 Col: 6
```

The screenshot shows a code editor window with the title 'file-reading-demo.py'. The code defines a main function that opens a file named 'test.txt' in reading mode ('r'). It then reads the file's contents and prints them. Finally, it closes the file. A blue arrow points from the explanatory text 'as with email or texts' to the 'open' function call.

Reading a text file

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



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

```
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.read()
    print(text)

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

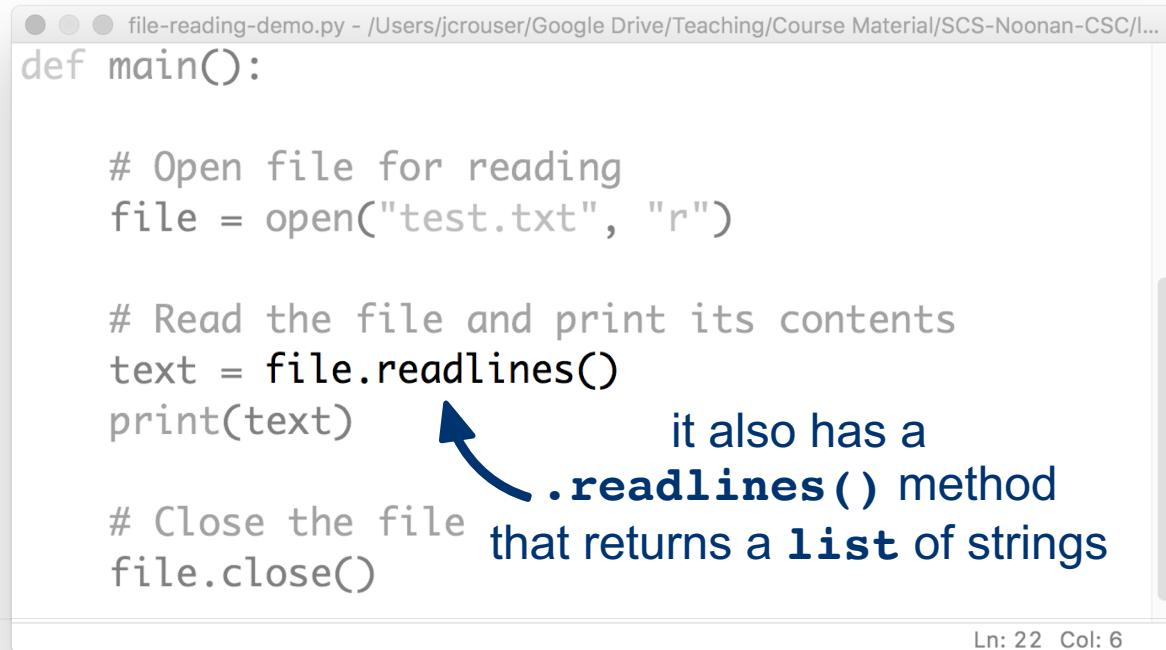
A blue arrow points from the explanatory text below to the `file.read()` method call.

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**:



The screenshot shows a code editor window with a Python script named `file-reading-demo.py`. The script contains the following code:

```
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()
```

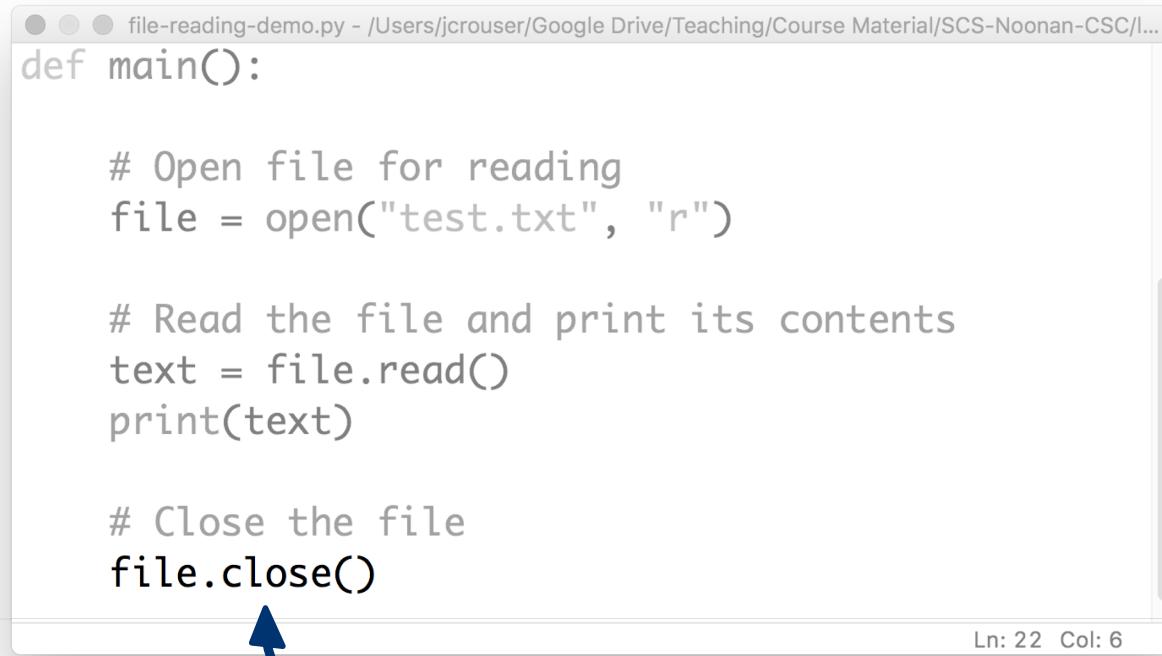
A blue arrow points from the text `it also has a .readlines() method` to the `readlines()` method call in the script.

it also has a
.readlines() method
that returns a **list** of strings

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

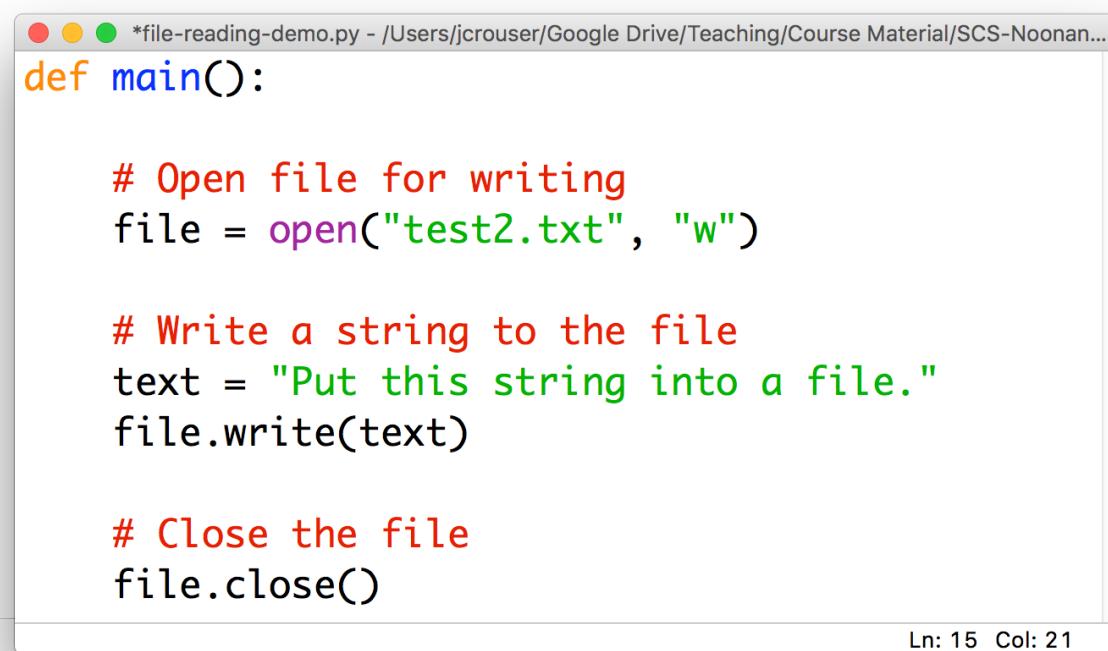
Key points for reading files

- 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 code editor window titled "*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")

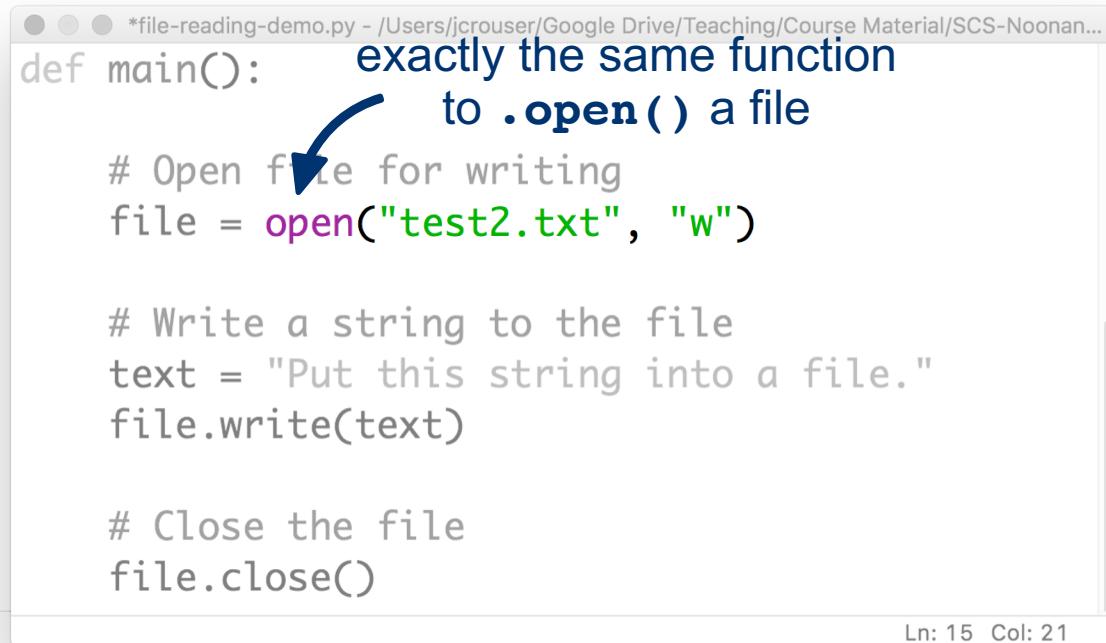
    # Write a string to the file
    text = "Put this string into a file."
    file.write(text)

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

The status bar at the bottom right of the editor window shows "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():
    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
```

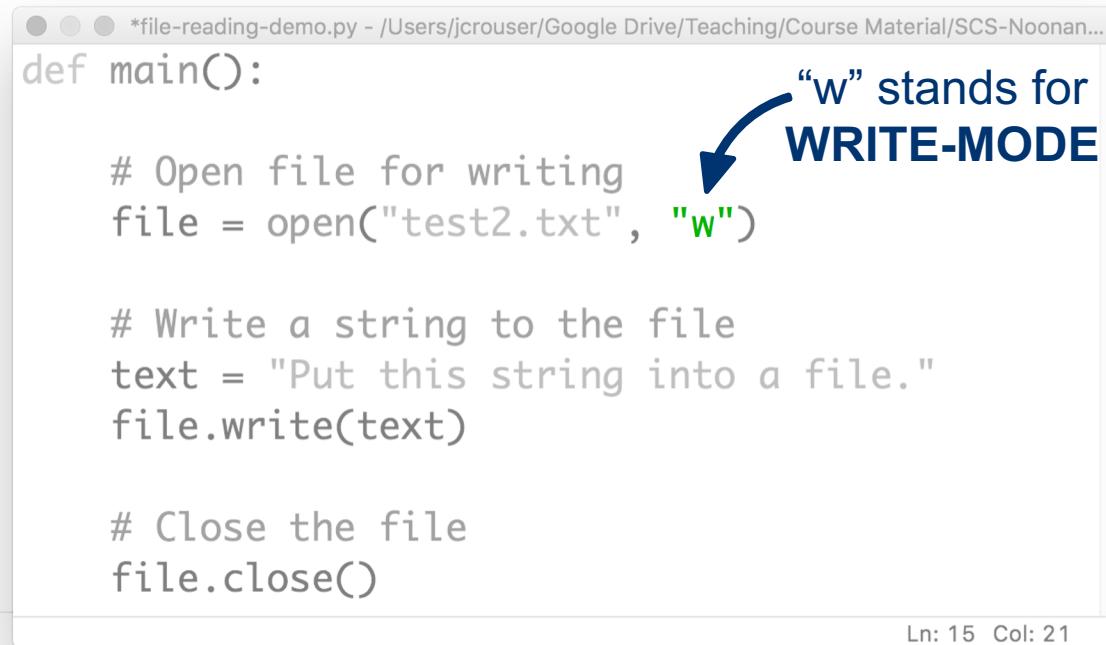
The screenshot shows a code editor window with a Python script titled '*file-reading-demo.py'. The script contains the following code:

```
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 red arrow points to the word 'file' in the second line of the code, specifically underlining it. Above this line, the text 'exactly the same function' is written in blue, followed by 'to .open()' in blue, and 'a file' in blue. The status bar at the bottom of the editor shows '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 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()
```

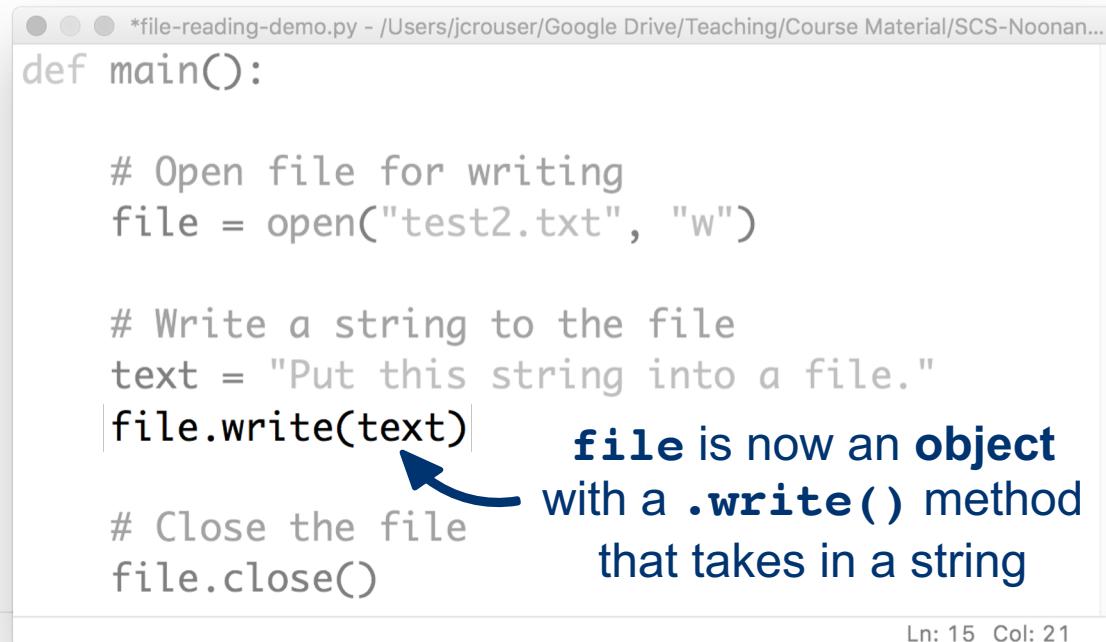
A green arrow points from the text "“w” stands for WRITE-MODE" to the character "w" in the line `file = open("test2.txt", "w")`.

“w” stands for
WRITE-MODE

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 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)

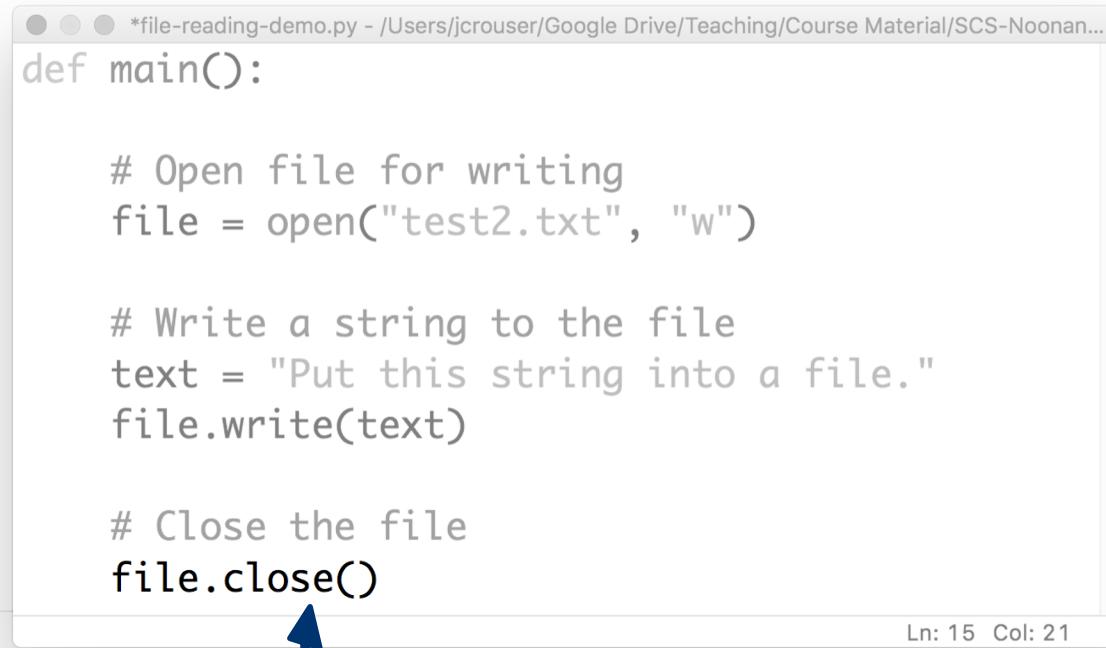
    # Close the file
    file.close()
```

An annotation with a blue arrow points from the text "file is now an object with a .write() method that takes in a string" to the line `file.write(text)`.

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 code editor window titled '*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()
```

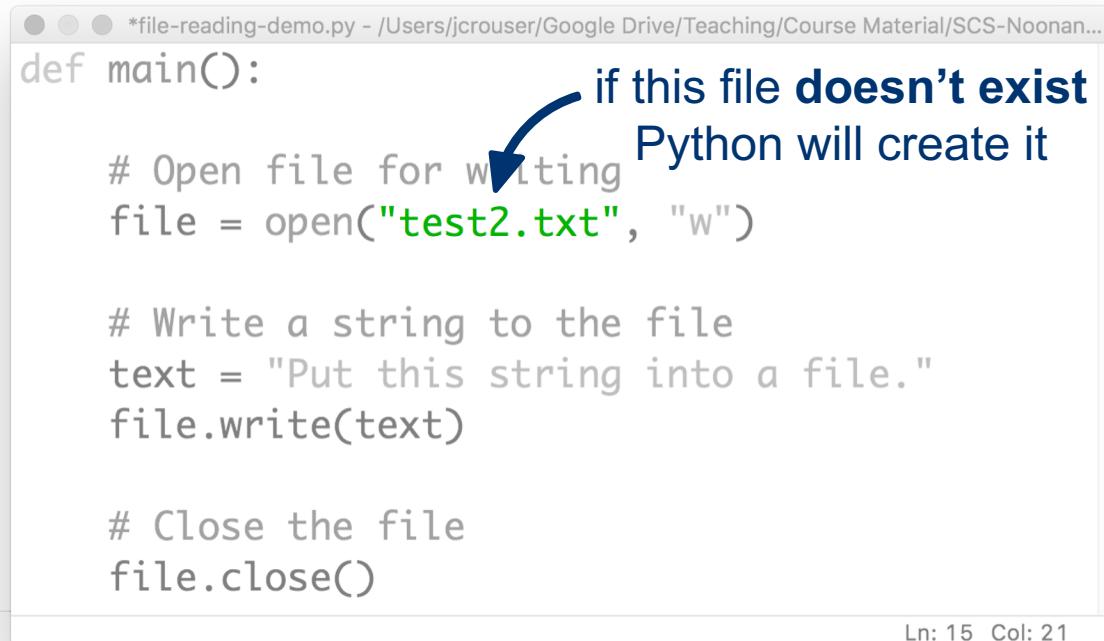
In the bottom right corner of the editor window, there is a status bar with 'Ln: 15 Col: 21'.



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

```
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()
```

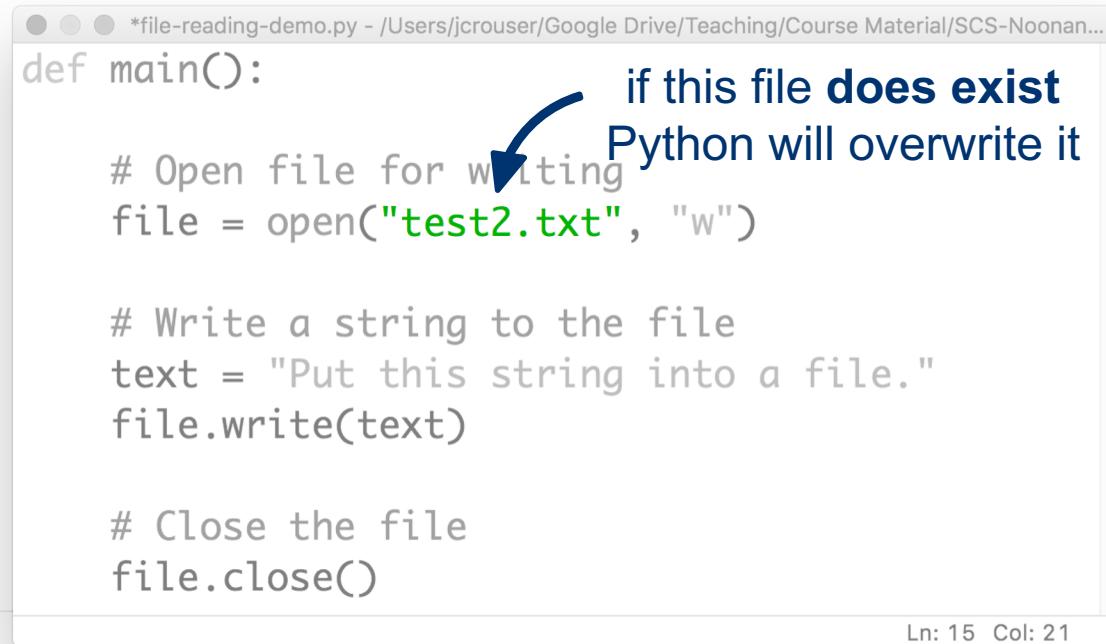
A blue arrow points from the word "writing" in the first line of the code to the explanatory text "if this file doesn't exist Python will create it".

if this file doesn't exist
Python will create it

Ln: 15 Col: 21

Writing data to a text file

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



The image shows 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()
```

A blue arrow points from the word "writing" in the first line of code to the explanatory text "if this file does exist Python will overwrite it" located to the right of the code.

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 code:

```
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()
```

A callout box with a blue arrow points to the letter 'a' in the line `file = open("test2.txt", "a")`. The text inside the callout box reads:

if you want to
add to an existing file
instead of overwrite it
“a” stands for
APPEND-MODE

Ln: 15 Col: 21

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

Fork: <https://repl.it/@JordanCrouser/vertical-horizontal>

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?



About the final project

- **Four weeks of class left!** (time to talk about the final)
- **Goal of the project:** apply the techniques we've learned in this class to something **you care about**
- **Ideas:**
 - an computer-generated animation
 - a custom game
 - a tool to help plan your path through the major
 - a automatic poem generator
 - a graphing calculator program
 - **anything else you can think of!**

Final project details and deliverables

- **Recommendation:** work in teams of 2-5 people
- **Deliverables:**
 - Friday Nov 13th: **Final Project Proposal**
 - Friday November 20th: **Prototype I**
 - Monday November 30th: **Prototype II**
 - Thursday December 17th: **Final Write-Up** (solo or group)

FP1: final project proposal

1. Names of **people** working on this project
2. What's the **big idea** behind this project?
3. What are the (major) **building blocks** the project will need to be successful?
4. Which do **you know how to build already**, and which ones **do you still need to figure out**?
5. Are there any **potential roadblocks**?

Coming up

- ✓ Announcements
- ✓ Working with files
 - ✓ what is a file?
 - ✓ reading data in
 - ✓ writing data out
 - ✓ *some details about the final project*
- Lab: Sorting
- Life Skill #5: Code Diagrams
- Useful python packages