Python Programming: Scripting

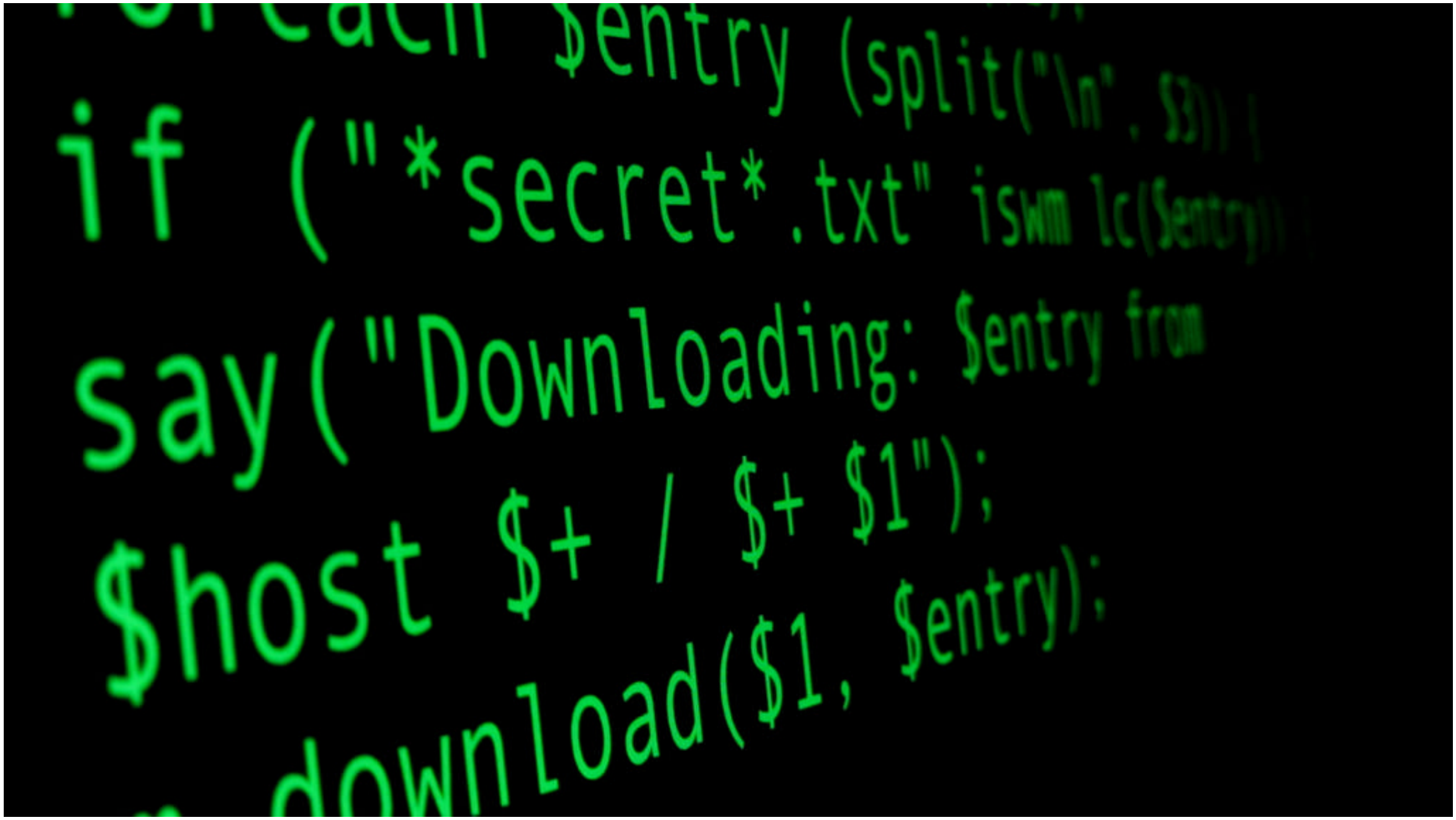
Lesson Objectives

After this lesson, you will be able to…

- Explain the uses of scripting.
- Write scripts that perform file I/O.
- Write scripts that take user input.

Introduction

Discussion: What comes to mind when you hear the word “scripting”?



What's a Scripting Language?

There are only two types of programming languages in the world: **scripting languages** or **compiled languages**.

All languages, like Python, are one of these two categories.

Scripting languages:

- One is Python!
- Write code, then immediate run it: `python my_file.py`
- Executes statements in order.
- Find a bug? Fix it, run it, repeat.

What's a Scripting Language?

Compiled languages:

- Compile means "build".
- *We can't immediate run code - the computer can't just read the code and needs to translate it to something it understands first.*
- Write code, *then compile it (not quick!),* then run it.
- Find a bug? Fix it, *wait for the code to compile,* run it, repeat.

You don't need to memorize this - just know that there's a difference, and Python is scripting.

What do you think a *script* is?

What is a Script?

Just some code that does something.

- Usually written in a scripting language.
- Can be as simple or as complex as needed!

Let's write a script:

- Create a file called `my_script.py`
- Open the file in `Atom`.
- Type the line

```
print("hello world!")
```

CONGRATS: You now have a script!

Look familiar? You've been scripting since day 1!

Scripting, Commonly

When people say scripts, though, they usually mean code that:

- Takes input.
- Gives output.
- Reads or writes to a file.
- Performs a task.

We have “perform a task” down!

Quick Review

Script:

- Just code that does something. You've written dozens of scripts in Python so far!

Scripting Language:

- A language where you can immediately run code. Python is one!
- Write -> Run.

Compiled Language:

- Compile means build! We can't immediately run code.
- Write -> Build -> Run.

We're only working with Python, so we can just write and run our code!

Next Up: Playing with files in Python.

Scripting, Part 1: Files

Let's further our programming toolkit.

On your computer, you can:

- Create or open a file (text, jpg, Word doc…).
- Read it.
- Edit it.
- Close it.

These are pretty basic actions. Can we do it in Python?

We Do: Let's Read a File!

With files, there are three key points.

1. Tell Python to open the file: `my_file = open(<file name>)`
2. Do something with the file! (Read it, edit it, etc).
3. Close the file when you're done: `my_file.close()`

First, let's check out **read**: View, but not change, the contents, with `read()`.

Let's try. On your Desktop, create a file called "hello.txt" with the word "hi" in it.

1. Now, also on your Desktop, create a file, `first_reading.py`. Fill it with: `python my_file = open("hello.txt") print(my_file.read()) my_file.close()`
2. Run it!

Note: The file must exist already!

What About Editing Files?

In programming, “edit” is referred to as “write”, short for “write to.” How do we write a file?

`open(<file name>)` has optional parameters: `open(<file name>, <mode>)`

- Mode: “What do you want to do with the file?” The default is “read.” Use `w` for “write”:

```
# To read a file:

my_file = open("hello.txt")

print(my_file.read()) ## We want this to be write, not read!

my_file.close()


# To write a file:

my_file = open("hello.txt", "w")

## Write some stuff

my_file.close()
```

Important: Write *overwrites* the current file!

We Do: Writing Files

Let's try this. Change your script. We're going to make it a little more complex - since we're programming, we can use variables!

```
# Open the file hello.txt
my_file = open("hello.txt", "w")


# Write some content to my_file.txt
my_file.write("Hello world")
my_text = "Apple juice is delicious." # Use the variable!
my_file.write(my_text) # Writes "Apple juice is delicious."
my_file.write("Have a nice day!")


# Always close the file
my_file.close()
```

Run it!

Open the file to check.

Thought: How could you make new lines?

Discussion: Writing Complex Strings

What happens if we try to `write` multiple strings?

```
# But it doesn't  work with write.

my_file = open("a_file.txt", "w")

my_text = "Apple juice is delicious."

my_file.write(my_text, "Don't you think?") # Error! Write takes 1 argument (



my_file.close()
```

Error! `write` only takes one argument. We need to concat the strings. *Always just pass one argument to file.write()*.

```
my_file = open("a_file.txt", "w")

my_text = "Apple juice is delicious."

string_to_write = my_text + "Don't you think?" # Make one string here!

my_file.write(string_to_write)

my_file.close()
```

We Do: Creating Files

What if the file doesn't exist yet?

Write to the rescue!

- Write opens a file for writing…
- But it also creates it if need be!

At the bottom of your script, add:

```
# Open OR create file totally_new_file.txt
my_new_file = open("totally_new_file.txt", "w")


# Write some content to totally_new_file.txt
my_new_file.write("Content goes here")


# Always close the file
my_new_file.close()
```

Check your desktop after running it!

You Do: Create a File

Now, try it yourself. Write a new script:

- `open()`, in read mode, your existing `a_file.txt`.
- `.read()` the file and save the contents into a variable, `file_contents`.
- Using `.write()`, create a new file called `b_file.txt`.
- Write `file_contents` to `b_file.txt`.

Don't forget to `close()` your files!

Create a File: Solution

```
my_file = open("a_file.txt", "r")

file_contents = my_file.read()

my_file.close()


my_file_script = open("b_file.txt", "w")

my_file_script.write(file_contents)

my_file_script.close()
```

Quick Review

You can open, read, and write files with Python.

Write will create the file if it doesn't already exist.

Always close your files!

```
file_to_read = open("a_file.txt")

file_to_write = open("my_file_script.txt", "w")



string_to_write = file_to_read.read()

file_to_write.write(string_to_write)



file_to_read.close()

file_to_write.close()
```

Next up: More advanced file options.

Other File Modes

What if we want to read AND write a file? Or write to the end of a file instead of overwriting what's there?

`open` has a few other modes.

Value	Mode	Purpose
`r`	Reading	Read only. The default!
`w`	Write	Use to change (and create) file contents
`a`	Append	Use to write to the end of a file
`r+`	Read Plus	Can do both read and write

Don't memorize this; just know it's there. A lot of programming is understanding your options and then Googling the syntax! The biggest thing for you to learn is the concepts that Python can do.

I Do: The With Keyword

Always remembering to close a file can be hard. There's another way to open files so Python closes it for us!

```
# Instead of:

file_object = open("my_file.txt", "w")

file_object.write("Hello World!")

file_object.close()


# We can say:

with open("my_file.txt", "w") as file_object: # This line replaces the open

  file_object.write("Hello World!") # This line is the same; note the indent
```

What Else is in File?

These are just for reference - we won't be using them!

- Do you have a list that you want to write on multiple lines? Use `my_file.writelines([<your list>])`
- Does your file have things on multiple lines you want to read into a list variable? Use `list_contents = my_file.readlines()`
- Separating some written lines? Add `\n` to the `write()`

Quick Review:

File has a lot of advanced options.

- You can write a list across multiple lines, or read a file with multiple lines into a list variable.
- Write only takes one argument, so concat your strings!
- You can open files using `with` to automatically close them.

```
# Instead of:
file_object = open("my_file.txt", "w")
file_object.write("Hello World!")
file_object.close()


# We can say:
with open("my_file.txt", "w") as file_object: # This line replaces the open
  file_object.write("Hello World!") # This line is the same; note the indent
```

Next up: User Input!

What about User Input?

We've just done a lot with file I/O (in/out).

We can prompt users for information, too.

You've seen this a few times (remember the error checking, with the try/catch?)! It's very common.

```
# Prompts with "input"

# Saves result in user_name

user_name = input("Please type your name:")
```

You Do: Bring it all Together!

1. Create a file called `about_script.py`.
2. In it, prompt the user for their name. Then, prompt them for their favorite food.
3. Using write, create a file called `about_me.txt`.
4. In `about_me.txt`, write out the name and favorite food in a sentence.

Bonus: Use `format` for forming your sentence!

Bring it all Together, Solution

```
user_name = input("Please type your name: ")

user_food = input("Please type your favorite food: ")


file = open("about_me.txt", "w")

file.write("My name is " + user_name +  " and my favorite food is " +  user_
```

Summary and Q&A

Scripting language vs compiled language.

- Scripting languages: Write -> Run.
- Compiled languages: Write -> Build -> Run.

Script:

- Just some code!

Summary and Q&A

File I/O:

- `my_file = open("a_file.txt", "w")`
- `my_file.write("Some content")`
- `my_file.write(my_text)`
- `my_file.close()`

User input

- `user_name = input("Please type your name:")`

Additional Resources

- Socratica Video: Text Files
- Executing a Python Script
- Reading and Writing Files
- File Object Documentation
- Binary vs Text Files