Video Script: "Functions & File I/O"

Video Length Estimate: 10-11 minutes

(0:00 - 0:25) Intro Music & Title

[VISUAL: SLIDE 1 - Title Slide]

(Music fades to background)

Host: "Hello and welcome back! In our last video, we mastered loops. Today, we're continuing our journey through Python's fundamentals by tackling two incredibly important topics: Functions and File I/O."

(0:25 - 2:30) Part 1: Functions - Packaging Your Code

[VISUAL: SLIDE 2 - Functions Introduction]

Host: "Let's start with functions. The tagline here is 'Package Your Power'. A function is a reusable block of code that performs a specific task. Think of it like a recipe: you define it once, and you can use it over and over again."

[VISUAL: SLIDE 3 - Why Use Functions?]

Host: "Why bother? Three key reasons: Reusability, which saves you from rewriting code; Organization, which helps you break down big problems; and Readability, which makes your code much easier for you and others to understand."

[VISUAL: SLIDE 4 - Anatomy of a Function]

Host: "The syntax is straightforward. You use the def keyword, followed by a function name, parentheses, and a colon. The code that belongs to the function is indented underneath. To run it, you 'call' the function by its name."

[ACTION: Switch screen to Jupyter Notebook - Section 2.1]

Host: "Let's see this in action. In our Week 2 notebook, under section 2.1, I'll define a simple function."

(Host types and executes the code)

```
def greet():
    print("Hello from a function!")

greet()
```

Host: "See? The print statement didn't run until we explicitly called greet()."

(2:30 - 4:30) Functions with Parameters & Return Values

[VISUAL: SLIDE 5 - Parameters vs. Arguments]

Host: "Now, let's make our functions more useful. We can pass data into them using parameters. A parameter is the placeholder variable in the function definition. The actual value you pass in is called an argument."

[ACTION: Switch to Jupyter Notebook - Section 2.2, first cell]

(Host types and executes the code)

```python
def greet_by_name(name):
    print(f"Hello, {name}! Welcome to the course.")

greet_by_name("Charlie")
```

Host: "Here, name is the parameter, and 'Charlie' is the argument. This makes our function dynamic."

[VISUAL: SLIDE 6 - The return Statement]

Host: "But what if we want to get data out of a function? For that, we use the return statement. It sends a value back, which we can then store in a variable."

[ACTION: Switch to Jupyter Notebook - Section 2.2, second cell]

(Host types and executes the code)

Python

```python
def add_numbers(x, y):
    result = x + y
    return result

sum_result = add_numbers(10, 5)
print(f"The returned result is: {sum_result}")
```

Host: "Our function did the calculation and returned the value, which we captured in sum_result. This is fundamental for building any kind of data processing pipeline."

(4:30 - 7:00) Part 2: File I/O - Saving Your Work

[VISUAL: SLIDE 7 - I/O Introduction]

Host: "Okay, let's switch gears to I/O, which stands for Input and Output. The tagline is 'Interacting with Users & Files'. This is all about how our Python program communicates with the outside world."

[VISUAL: SLIDE 8 - Two Types of Input]

Host: "There are two main ways to get data into our program. First, Direct Input, where we get information live from the user at their keyboard. This should be familiar to you from week 1 where you used the input function. The second way is via File Input, where we read data that's already been saved to a file. This more advanced form of I/O is what we're going to focus on here."

[VISUAL: SLIDE 9 - User Input]

Here is a quick recap of the input() function where you can enter a message to be displayed for context and get an input back. Recall that the input() function always returns a string and so if you need it in any other data type, you will need to perform a typecast i.e., convert the string to an int, float or whatever is the appropriate data type for your program's context.

[VISUAL: SLIDE 10 - Why Use File I/O?]

Host: "Now, for bigger data, asking the user to type it all is not practical. That's where File I/O comes in. Everything in your program's memory is temporary. File I/O is how you save your results and, crucially for this course, how you read datasets."

[VISUAL: SLIDE 11 - The with Statement]

Host: "The best practice for handling files is the with statement. It's your safety net—it automatically and safely closes the file for you, which is critical to prevent data loss."

[VISUAL: SLIDE 12 - File Modes]

Host: "When you open a file, you tell Python the 'mode'. The main ones are 'r' for reading, 'w' for writing, and 'a' for appending."

[ACTION: Switch to Jupyter Notebook - Section 3.2]

Host: "Let's write to a file. In section 3.2, I'll create hello.txt."

(Host types and executes the code for writing a file)

Python
OBJ
```
with open('hello.txt', 'w') as f:
    f.write("This is the first line.\n")
    f.write("Hello from Python!")
```

Host: "And just like that, the file is created."

(8:30 - 9:30) Reading from a File & Conclusion

[ACTION: Switch to Jupyter Notebook - Section 3.3]

Host: "Now let's read it back in section 3.3."

(Host types and executes the code for reading a file)

Python
OBJ
```
with open('hello.txt', 'r') as f:
    content = f.read()

print(content)
```

Host: "And there you have it! You now know how to package code with functions, get live data from a user with input(), and save and load data with File I/O."

(8:30 - 9:30) Practice & Conclusion

[ACTION: Scroll down in Jupyter Notebook to the practice questions for Functions and File I/O]

Host: "That covers the essentials of functions and file I/O! You now know how to package your code for reuse and how to save and load data."

Host: "As always, the best way to learn is by doing. In the notebook, you have two practice questions: one to build a function that calculates the area of a rectangle, and another to save a list of your favorite movies to a file. Please take some time to work through them."

[VISUAL: SLIDE 13 - Thank You slide]

Host: "Fantastic work. You're building a really solid foundation in python for the exciting machine learning topics to come."

(Outro music fades in)