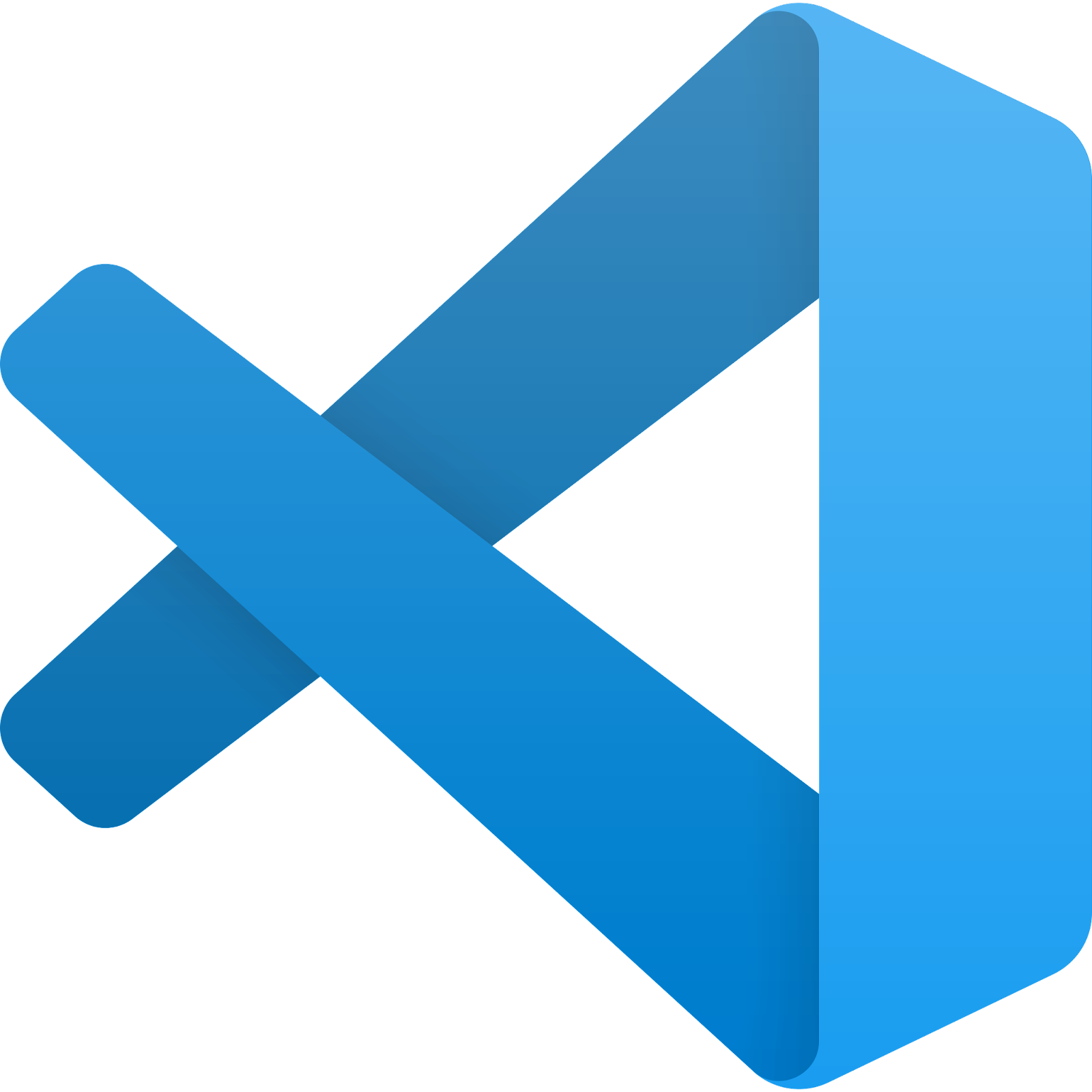
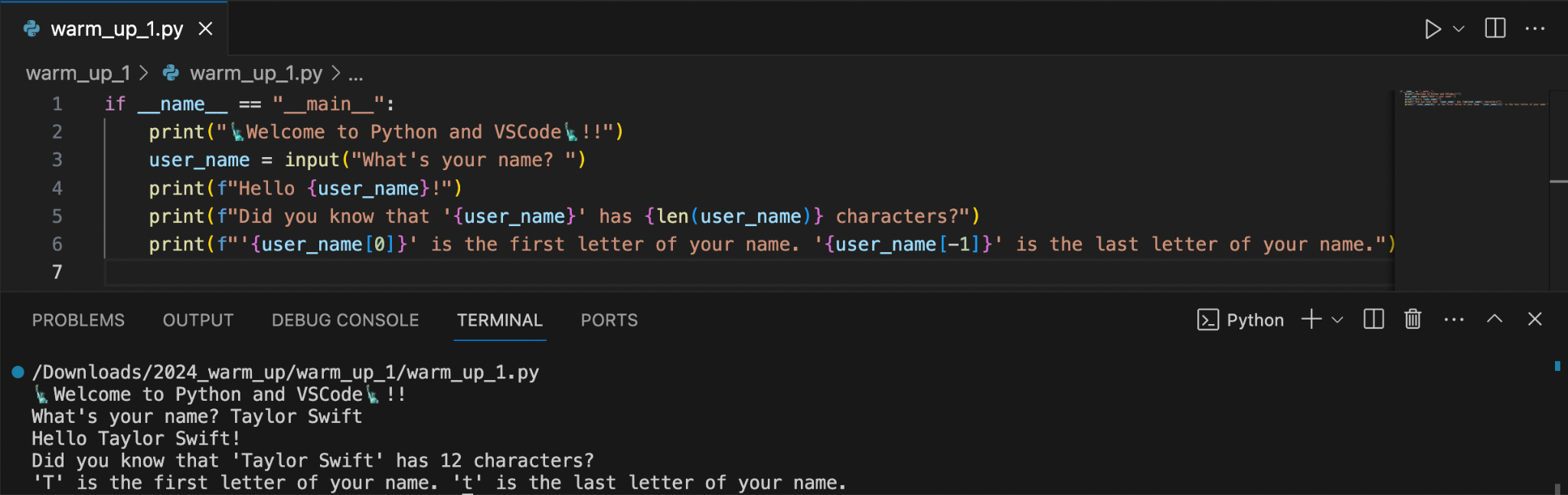
# **Python Warm-Up #1: Hello Python**

| **Download the Python Warm-Up folder.** |
| --- |

1. Create an **AMP\_CS** folder on your desktop.
2. Download a zipped version of the **warm\_up** folder.
3. Unzip the zipped version of the **warm\_up** folder.
4. Move the unzipped **warm\_up** folder to the **AMP\_CS** folder on your desktop.

| **2. Open and read** [**warm\_up\_1.py**](https://drive.google.com/file/d/1yRkcwCHq6HBOOx4tRSSpQp4vYFUKW2w8/view?usp=drive_link) **in the VS Code editor.** |
| --- |

1. Open the VS Code editor: 
2. Use VS Code to open the unzipped **warm\_up** folder
3. Click on **warm\_up\_1.py**
   1. You’ll see the Python code for **warm\_up\_1.py** open in a new VSCode window.



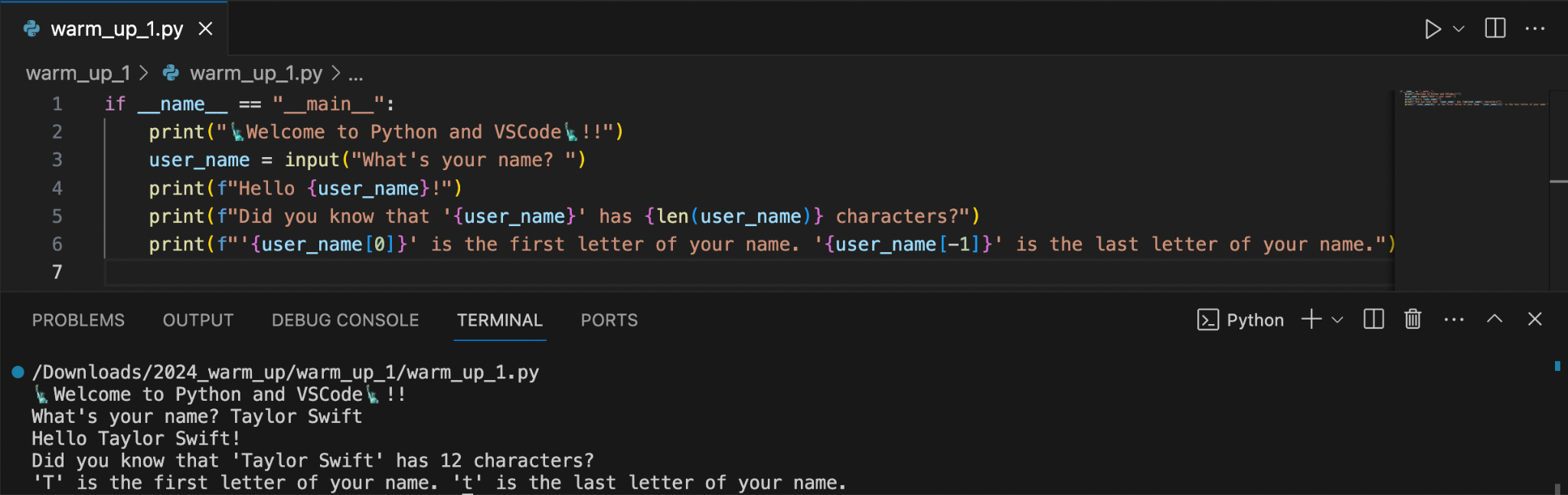
1. Take a minute to read through the code in warm\_up\_1.py.
   1. Python was designed to prioritize human-readability. **What do you think will happen when you run this Python program?**

| **3. Use the VSCode editor to run the Python code in warm\_up\_1.py.** |
| --- |

1. Click on the Play button to run the **warm\_up\_1.py** Python file.

* If you don’t see this button appear in the upper right corner of the VSCode window, you haven’t successfully installed the VSCode Python extension. See instructions included in the Welcome to AMP:CS email

1. Enter your name in the **Terminal** window that appears after clicking the Play button:



| **Understanding warm\_up\_1.py** |
| --- |

### **Line #1:** if \_\_name\_\_ == "\_\_main\_\_":

### **Provides a starting point for the Python interpreter**

* While not strictly necessary, this line provides an explicit context for the Python interpreter
  + "\_\_main\_\_"indicates that the code should be run when the file is being used as a stand-alone program
* Code in this section will be executed first, even if other code precedes it
  + In later warm-up exercises you'll write function definitions which precede this line.
  + There are other important considerations for including this line in your Python files, some of which we'll explore later.
* Indentation is very important in Python.
* Lines 2 - 6 are consistently indented to indicate that they are part of the "\_\_main\_\_" section.
* The Python interpreter will throw an error if your programs exhibit inconsistent indentation.

### **Line #2:** print("🗽Welcome to Python and VSCode🗽!!")

### **Outputs a text-based message to the Terminal window**

* A built-in Python function called print is used to display output to the terminal.
  + In this example, the print function takes a single string literal as a parameter.
* Python string literals can be delimited using several different quotation marks:

| "This is a string." 'This is also a string.' 'You said, "That's right!"' | """  This is  a multi-line  string. """ | '''  This is also  a multi-line string. ''' |
| --- | --- | --- |

#### 

### **Line #3:** user\_name = input("What's your name? ")

### **Prompts the user to enter their name, then stores the data in a variable**

* A built-in Python function called input is used to gather input from the terminal
  + A string parameter for the input function is shown as a prompt for the user
  + Data entered by the user is always returned as a string
* Variables store a reference to data for future access
* Python variable names:
  + must start with a letter or the underscore character, and contain only: A-z, 0-9, and \_
  + are case-sensitive
  + should be self-documenting
* **snake\_case** is a variable naming convention using lowercase letters and \_ to separate words
  + Our projects will use **snake\_case** for most variable/function names

### 

### **Line #4:** print(f"Hello {user\_name}!")

### **Uses f-string notation to interweave contents of the** user\_name **variable into a string literal:**

* There are many ways to include the content of variables in Python output. For consistency, we will use Python **f-strings.**
* Placing an 'f' just before a Python string literal indicates that f-string notation will be used.
* f-string notation uses { }'s to indicate that the name of the variable should be replaced by its’ contents

### **Line #5:** print(f"Did you know that '{user\_name}' has {len(user\_name)} characters?")

### **Creates a string literal with double-quotes, while using single-quotes inside of the string with no error:**

* The built-in function called len is used to determine the length of a string.
  + We’ll see later that the len function can also be used to determine the length of a list.
* Notice how f-string notation permits function calls

### **Line #6:** print(f"{user\_name[0]} is the first letter of your name. {user\_name[-1]} is the last letter of your name.")

### **Uses index notation to reference individual letters in a string variable**

* In Python, String indexing is similar to array/list indexing
  + [ ]s are used to indicate a specific letter/index in the string
* Notice how string indexing uses 0 to refer to the first letter of a string.
* Python permits negative indexing, which subtracts from the length of the string/list.

| **Warm-Up #1 Exercises -** |
| --- |

These exercises require you to submit to the Warm-Up #1 assignment on Gradescope. Gradescope is our autograding platform. You should have received instructions for joining Gradescope via an email from your instructor.

### **Exercise 1.1: Introduce an indentation error to see how VS Code handles error feedback.**

Make a small change the last line in warm\_up\_1.py to break the consistent indentation:

| **if \_\_name\_\_ == "\_\_main\_\_":  print("Welcome to 🗽 for AMP 2024!!")  user\_name = input("What's your name? ")  print(f"Hello {user\_name}!")  print(f"Did you know that '{user\_name}' has {len(user\_name)} characters?")  print(f"'{user\_name[0]}' is the first letter of your name. '{user\_name[-1]}' is the last letter of your name.")** |
| --- |

Save your changes, then run the Python code. What feedback does VS Code give you on your mistake?

### **Exercise 1.2: Write your own Python input and output statements.**

Undo your changes from Exercise 1.1, then extend the original code in **warm\_up\_1.py** to ask the user their favorite show for streaming. Store their response in an appropriately named variable, then output to the terminal:

| **"Don't spend so much time watching '\_\_\_\_\_\_\_\_\_\_\_\_' that you leave your HOMEWORK to the last minute!"** |
| --- |