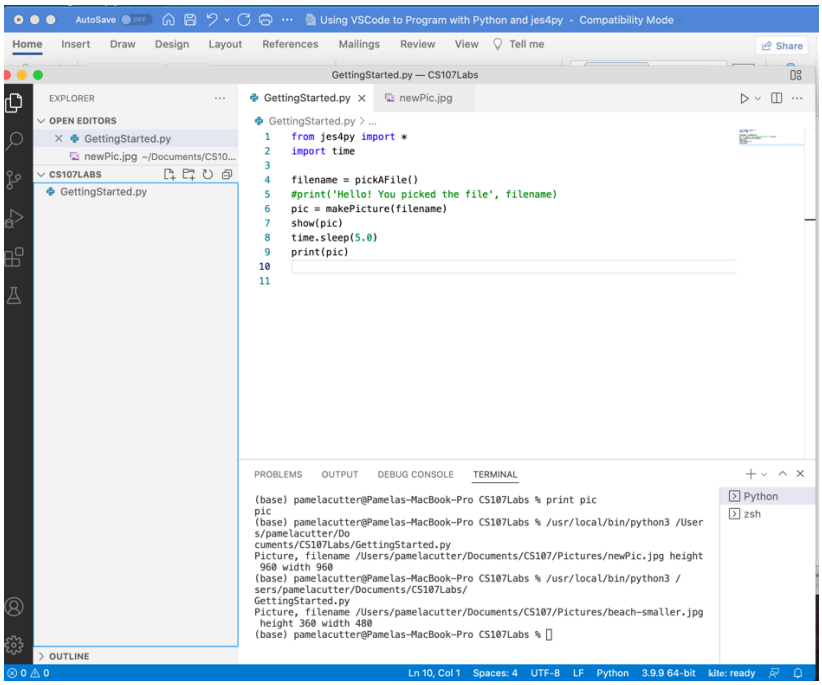
Using VSCode to Program with Python and JES4py

In this course, we will be using a program called Visual Studio Code, or VSCode, for short, to write and test Python programs. Our goal is to learn how to manipulate the digital versions of images and sounds. To do this, we are going to be using a package in Python called JES4py. This package contains functions that we will use to access and edit pixels from pictures and samples from sounds. Where does this name come from? There's a little bit of a history to it, but the short story is this: JES stands for Jython Environment for Students. But wait – didn't the instructor say we were programming in Python?! It's ok, we are. Jython is an implementation (*i.e.*, a version) of the Python programming language written in Java. It allows Python code to work easily with Java code. We should also make note that some of the media names we will work with were developed to work with JES, but are not part of a normal Jython distribution. As the Python and Java languages have changed, the original JES tool was no longer working. Hence, JES4py was developed to keep the same functionality as JES, but will work directly with Python, without requiring Jython.

Introduction to VSCode and JES4py

As we just mentioned, we will be writing our programs in VSCode and we will need to use the JES4py package. This is what VSCode looks like when you run the application:



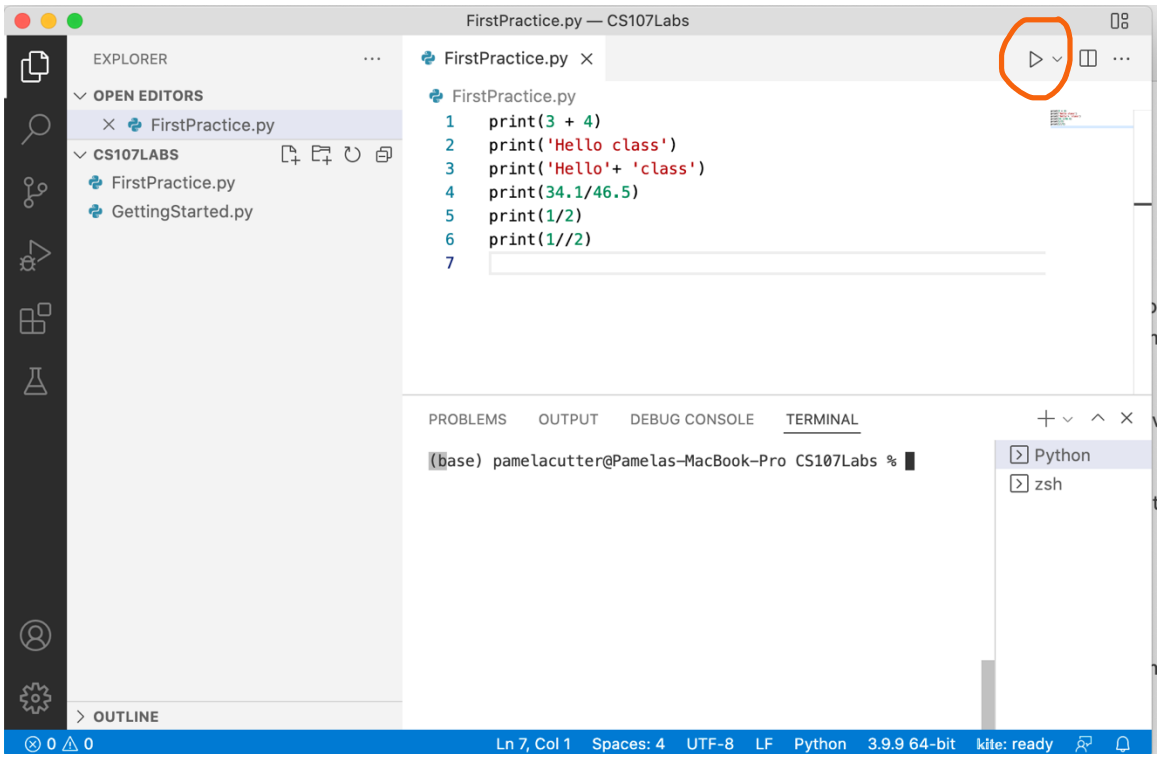
The big (white) part in the middle is the **Editor**. This is where we will write our programs. The bottom part is the **Terminal**. This is where will we see output that gets printed from our programs.

Some examples of commands we could type into the Editor are:

```
print(3 + 4)
print('Hello class')
print('Hello' + 'class')
print(34.1/46.5)
print(1/2)
print(1//2)
```

(Note: Python understands many of the standard math operators.)

Exercise: Start up VSCode. (If you are not familiar with VSCode, you may want to wait to do this at the beginning of the first mini-lab, where there will be more detailed instructions on starting your first program.) Try entering the commands from the previous example into the Editor area in VSCode. Each command should be on a different line. Your program might look something like the following:



To run this program (*i.e.,* to make the computer execute each line of code), click on the little triangle in the upper right corner of the Editor pane. (It is circled above.) Your results should show up in the Terminal pane in the bottom half of the window.

Follow-up Analysis: Were the results from this exercise what you expected? Why or why not? Was there a (small) difference between the 2nd and 3rd examples? If so, what was it? What happened in the last example? (You may need to discuss this with your instructor or class.)

We have just seen that we can use arithmetic operators and commands to make things happen in a program. When we used the `print` command, the word `print` is actually a **keyword** – a predefined word that has a specific meaning in the language. Programming languages often have a number of keywords. We will learn additional Python keywords as necessary.

In addition to using arithmetic operators and commands in a program, there are times when we will want to hold onto values for later use in a program. We will do this through the use of **constants** and **variables**. A **constant** is a value that does not change (*i.e.*, it is *constant*). Some examples of constants include 3, -15.2, "A", and "Hello". A **variable** is a value that can change throughout a program. In order to use variables, we need to give them identifiers (names). Identifiers can be almost anything (but cannot be the same name as a keyword), although good programmers try to give their variables names that reflect what the value represents. For example, if we have a variable that stores a file, we might name it *myFile*, or if we have a variable that represents a sum of numbers, we might name it *sum*. One of the features that makes Python easier to learn and use than some other languages is that you do not have to decide what type of data your variables will hold. Python takes care of those details behind the scenes. When you need to use a variable in a program, you just make up a name and use it. We do need to have some caution, though, because if you use the same name for two different values, the first value will be lost when you define the second.

Helpful Hint

When you need to use a variable in a program, you can just make up a name and use it. Be careful not to use the same name for two different values.

Aside: Rules and Conventions for Identifiers

- Names must start with a letter (usually lower case) or underscore character, but then may contain any number of letters, numbers, or other underscore characters.
- Names are case-sensitive: *Sum* and *sum* are different identifiers.
- It is a common convention that variable names that contain more than one word use underscores to separate the words, as in *my_file* or *average_of_three_numbers*, or use a capital letter at the beginning of words after the first, as in *myFile* or *averageOfThreeNumbers*.

It is now time to try out a few things. In this first mini-lab, you will learn some basics of JES and some basics for choosing and showing some pictures (and sounds).

Mini-Lab 1: Getting Started with VSCode and JES4py