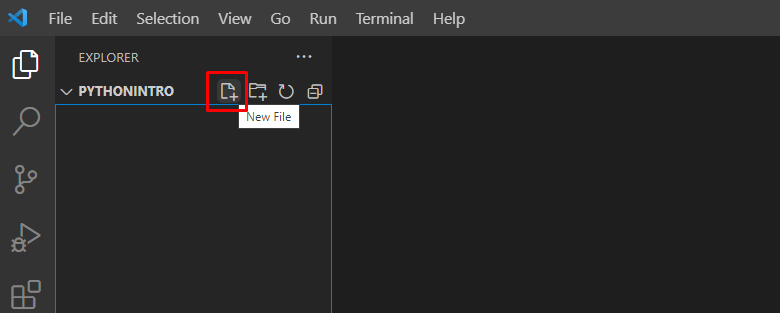
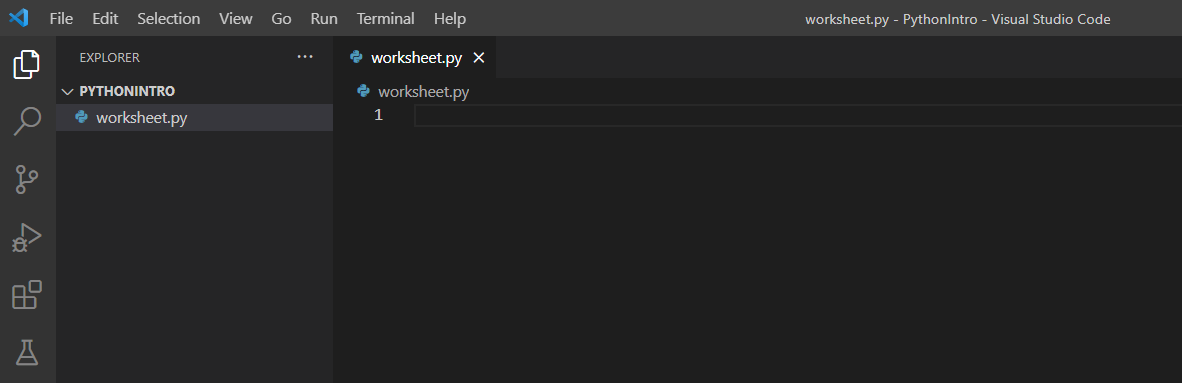
**Python in Action Worksheet**

* Inside your projects folder, create a folder called ‘python\_in\_action’
* Open Visual Studio Code, and select ‘File > Open Folder’ at the top to open the python\_in\_action folder. In the Explorer, click the ‘new file button’ and create a file called “worksheet.py”. Applying the correct extension to the file name (.py) will create this file asa Python file.
* 
* What a successful version looks like:
* 

**Variables (slides 3-6)**

A Variable is the appropriate tool any time we want our program to remember a value so that it can be used later. Once a variable has been declared, it can later be given a new value to reflect the changing state of the application.

1. **Days of the Week**
   1. Create a variable called **day\_of\_week** assign it a value of the string ‘Monday’.
   2. Print this variable to the console
   3. On a following line, change the value of **day\_of\_week** so that it has the value ‘Friday’.
   4. Print the statement ‘I can’t wait for Friday!” ***using the day\_of\_week variable in an f-string***
   5. Put down a breakpoint on the first line of your code and run your code with the debugger. Use the Step Into button to go from line to line. Hover your mouse over the **day\_of\_week** variable every place it is used to see its current value change from Monday to Friday before it is printed.
2. **User Input**

The Python **input()** function is similar to **print()** in that it will display a message in the terminal. However, when called, the **input()** function will **return** whatever is typed in the terminal after the message is displayed, and save it to a variable.

* 1. Create a variable called **animal\_input** and set it equal to the input function call with the phrase ‘What is your favorite animal?’
  2. Create another variable called **color\_input** and set it equal to the input function call with the phrase ‘What is your favorite color?’
  3. Combine these two variables within an **f-string** so you can print the phrase “I’ve never seen a <color> <animal>!”

One of the most important things when creating a new variable is to think of a descriptive name that explains exactly what this variable represents in your code. If someone with no knowledge of your code were to jump in, they should be able to figure out what your variable represents.

Some examples of great variable names:

**todays\_date**

**player\_one**

**user\_input**

**highest\_number**

**customer\_from\_database**

Some examples of bad variable names:

**x**

**num**

**thing**

**p1**

**item**

**edit\_cust**

Try not to abbreviate when creating variable names. The few keystrokes you save will not be worth the potential for confusion!

**Conditionals (Slides 7-8)**

Before starting the next section, think of your favorite breakfast, your favorite lunch, and your favorite dinner. Write these three things in some comments in your code. Remember to start a line with # to designate it as a comment

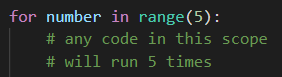
1. **Meals**
   1. Create a variable called **time\_of\_day** and give it the value 1300
   2. Below that, create another variable. This time, you choose the name. This variable will end up holding the value of one of the meals you have previous chosen. But right now, you don’t know which it will hold! What would be a good name to use for a variable that might end up with any of those three values?
   3. Now create a conditional with three parts: if **time\_of\_day** has a value of less than 1200, assign your own variable the value of the breakfast you chose.
   4. If **time\_of\_day** is between 1200 and 1700, assign your variable the lunch you chose.
   5. If **time\_of\_day** is greater than 1700, assign your variable the dinner you chose
   6. After these conditionals, print your variable to the console. What value do you expect it to show?
   7. Go back to the line where you created **time\_of\_day** and change its value twice so that you are able to see the other two meals print when you run the code.
   8. Now try to ‘break’ this section of code by giving **time\_of\_day** a value that will cause an error. Think about how the variable is being used and what types of values would not make sense to be used that way. Run the code again to see the error get thrown.
   9. After you have successfully caused this section of code to throw an error, place a breakpoint on the first line of code in the file and debug the program. Watch how it will run all previous code until it encounters the line where the error occurs.
2. **Random Number**
3. Declare a variable to store a random number between 0 and 10. You will need to do some research to determine how to generate a random number in Python.
   1. A good search term to use: “random number Python”
4. If the number is between 0 and 2, print to the console “Beatles”
5. If the number is between 3 and 5, print to the console “Stones”
6. If the number is between 6 and 8, print to the console “Floyd”
7. If the number is equal to 9 or 10, print to the console “Hendrix”

**Loops (Slides 9-11)**

**For Loops**

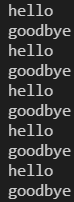
In Python, for loops are primarily used for iterating over collections like lists. However, you can also modify a for loop to act similar to a JavaScript for loop, so that it will run a predefined number of times.

**Note:** The following method is ONLY appropriate when you want a block of code to execute a certain amount of times. If you are meaning to iterate over a collection like a list, this style of for loop is not recommended. See the Lists section below to see the preferred approach of iterating over collections.



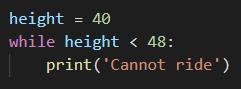
The above line of code creates a scope where any included code would be repeated 5 times. Additionally, the variable **number** that is created will increment from 0 to 4 with each loop.

1. Write your own for loop that will display the message “Python is cool!” 7 times.
2. Write your own for loop that will display the numbers 0-10, one number per loop. You will need to utilize a variable for this!
3. Write a for loop that will print the following sequence:



**While Loops**

While loops allow iteration to continue as long as a conditional holds true. Because of this, it is quite easy to end up with an infinite loop.



Even this small section of code would result in an infinite loop if there is no mechanism in place to change the value of **height.**

When using a while loop, you always want to think about how you plan to have the loop end, and this is typically done by changing a variable used in the conditional so that the condition is no longer true.

What could you change about this while loop so that it will loop 8 times instead of infinitely?

Add this while loop to your code and try it out!

**If you get caught in an infinite loop, you can hit Ctrl-C in the terminal to stop the process.**

**Magic Number**

This exercise will bring together all the pieces you have learned about so far!

1. Create a variable called **magic\_number** and give it an initial value of 50
2. Create a variable called **guess** and give it an initial value of 0
3. Create a while loop that will allow a user to continue to input a guess until they correctly choose the magic number.
4. If the number guessed is lower than the magic number, it should print “Too low!”
5. If the number guessed is higher than the magic number, it should print “Too high!”
6. If the number is within 10 of the magic number in either direction, it should print the appropriate message and additionally print “Getting warmer!”
7. If the number is correctly guessed, instead of “Getting warmer!”, print the magic number along with a congratulations message and break the loop.
8. Once you have tested and determined it works, go back to the beginning and change **magic\_number** to be assigned a random number between 0 and 100 instead of the hardcoded value.
9. Play your new guessing game a few times! Try to beat your own score
10. After playing a few times, put down a breakpoint at the start of the while loop and run it with the debugger. Hover over the **magic\_number** variable to see how much easier it is to determine the value of your variables by debugging this way!

**Functions (Slides 12-16)**

1. **Favorite Movie (void functions)**
   1. Create an appropriately-named variable and assign it the value of the name of your favorite movie.
   2. On the next line, print this variable to the console. Run your code to test it out.
   3. Now, indent the previous two lines of code you wrote and add the following line ABOVE them. This will move them into a function named ‘**print\_movie\_name**’



* 1. Run your code again. Why does the movie name no longer print? (See slide 15)
  2. Now do what slide 15 states is required for a function to run so you can test it out!

1. **Favorite Band (return functions)**
   1. Write a function that asks a user to enter their favorite band
   2. On the next line, the function should return the user’s response
   3. When you call the function, save the returned result as a new variable.
   4. After the function is called, print the returned result.
2. **Concert (parameters)**
   1. Create a new function called **concert\_display.** This function should have one parameter called **musical\_act**
   2. Inside the function, ask the user to enter the street they live on and save it as a variable called **my\_street**.
   3. Print an f-string that combines the variable that holds the user’s response along with the **musical\_act** parameter. The message should end up saying “It would be great if <**musical\_act**> played a show on <**my\_street**>!”
   4. Now when you call the **concert\_display** function, pass in variable that got returned from the Favorite Band section.

**Lists (Slides 17-19)**

1. **Desktop Items**
   1. Create a new variable called **desktop\_items**. Establish this variable as a list by setting it equal to a pair of square bracket [].
   2. When creating a list literal like this, we can insert some initial values at the same time it is created.
   3. Between the brackets, add a few strings to describe three items you see around you in your workspace. ‘desk’, ‘lamp’, etc. Make sure to separate the items with commas OUTISIDE the quote marks.
   4. What index values do these 3 items have?
   5. Print to the console the item that resides in index 1.
   6. On a new line, use the python list method that allows you to add new things to an existing list to add the string ‘Infinity Gauntlet’ to your **desktop\_items** list.
   7. Now, utilize a for loop to iterate over your list and print out each item one at a time.