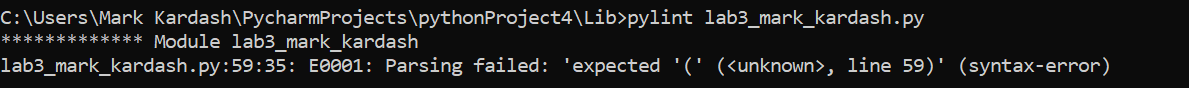
Mark Kardash

Scroll to Page 18 for revised testing from 2/23/2023

Scroll to Page 11 for revised testing from 2/11/2023

In all honesty, I had terrible difficulties with the program for Week 3, and I knew this was not going to go well. I encountered an error as soon as I launched pylint. It seemed that parsing had failed since early on in the code.



After making a small change, I was met with a plethora of errors, and a shocking score of 0/10.00

Text

Description automatically generated

I changed the ‘variable\_initiation’ function name to ‘\_\_init\_\_’. However, this did not seem to affect the score.

I eventually realized that I had imported many things incorrectly, and did not use some of my variables.

Text

Description automatically generated

Rewriting some functions gave me my first “climb”, as score of 1.35/10.00

Text

Description automatically generated

I realized that my experiment with combining elements from different types of data structures had failed. Frankly, I was very nervous, and very confused regarding the difference between lists, tuples, and dictionaries. I decided to rearrange my list to include both brackets and parentheses, as well as append each entry to the list individually. However, I knew this was going to take long, and I had less than an hour before the deadline, so I decided to ask for an extension, and decided to call it a day, with the hopes of having a clearer mind after a good night’s sleep.

Day 2:

On the second day, I saw that I still had a lot of undefined and unused variables. I realized I had overcomplicated my code without a reason. So I started cleaning it up.

Graphical user interface, text

Description automatically generated

My biggest problem at the time seemed to be that the links to the flower pictures were making a lot of lines too long. Rearranging some of the links gave me a score of 3.42/10.00

Text

Description automatically generated

I shortened some more links, and got 3.77/10.00

Text

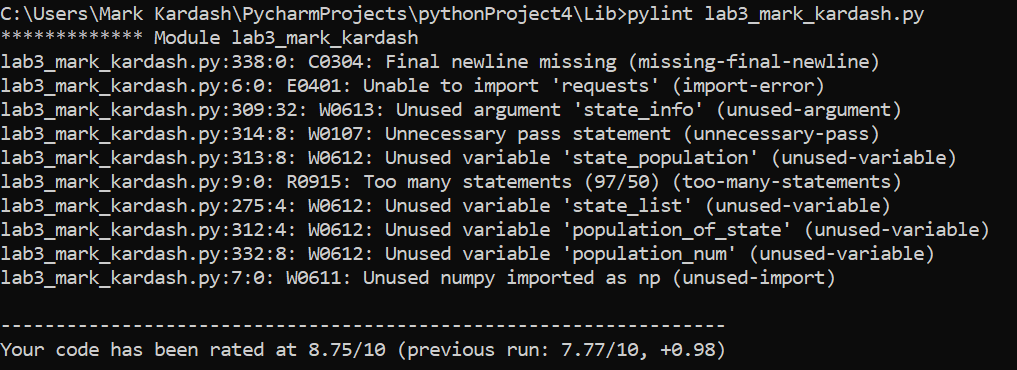
Description automatically generated

My first big breakthrough occurred when I got a 7.77/10.00, after finally fixing the last of the long lines, and removing some unused code.

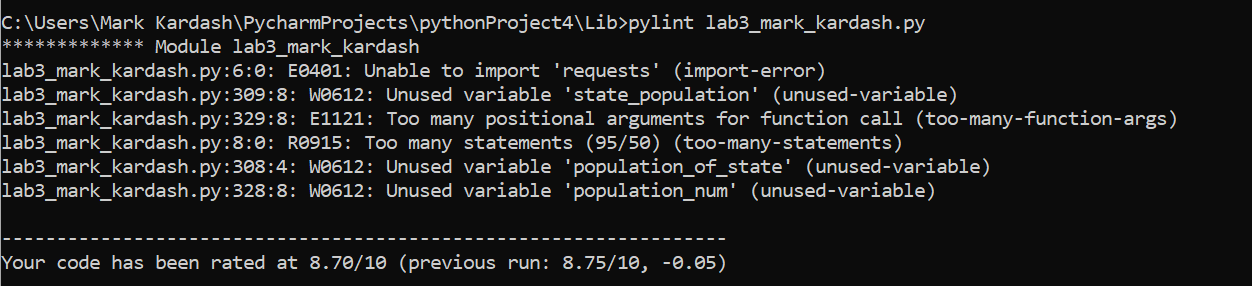
Text

Description automatically generated

My score was now 8.75/10.00



A fix of an issue created an extra unused variable, decreasing my score by 0.05.



Finally, after eliminating almost all unused variables, I got much closer to 10, with a score of 9.18/10.00

Text

Description automatically generated

All this time, I had been wondering as to why I couldn’t import the requests option. I checked the error and found out it was not built in, so I had to install it.

Text

Description automatically generated

My code was scored at 9.55, and then peaked at 9.64. All I had now was a self-assigned variable, an unused argument, an unused variable, and a problem with the option menu, which I had implemented as a function. It now said there were too many statements in it. I tried splitting it into two functions, to see if that would resolve the issue. When it gave no result, I decided to simply place the menu options as separate print statements.

Text

Description automatically generated with medium confidence

This is when the really bizarre things started happening. I received more errors than I had before, with the score decreasing to 8.89/10.00. Some of them were about missing function docstrings. However, most of them seemed to indicate that my variables had somehow been “redefined”.

Text

Description automatically generated

Fixing the missing docstrings gave me a score of 9.17, and then of 9.35. Unfortunately, this was my final score, as I did not know how to fix the “redefined” error.

Graphical user interface, text

Description automatically generated

Graphical user interface

Description automatically generated with medium confidence

Graphical user interface, text

Description automatically generated

Test Cases:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test Case: | Input: | Expected Output: | Actual Output: | Pass? |
|  | 1 | Program displays all states in alphabetical order, with capitals, populations and flowers: | Program displays only the names “State: “, “Capital: “, “Population: “, and “Flower: “, with a TypeError: ‘int’ object is not iterable. | No |
|  | 2 | Program allows user to search for a specific state, asking for the name. It then displays state data (Name, capital, etc.), along with a picture of the state flower. | Program asks for state 2 times, image of flower pops up, and, upon closing the image, program generates error “list index out of range”. | No |
|  | 3 | Program displays bar graph with overall state population of 5 most populated states. | Error: “Object ‘int’ is not iterable” | No |
|  | 4 | Program asks for a state name, gives the prompt to update population, and updates population for particular state. | Program asks for a state, prompts to update population, but then gives out ‘list’ object has no attribute ‘state\_name’ error. | No |
|  | 5 | The program exits with the message, “Thank you for using our program! Goodbye!” | The program exits with the message, “Thank you for using our program! Goodbye!” | Yes |
|  |  |  |  |  |

Test Case 1 Screenshot:

Text

Description automatically generated

Test Case 2 Screenshot:

Text

Description automatically generated

Test Case 3 Screenshot:

Text

Description automatically generated

Test Case 4 Screenshot:

Text

Description automatically generated

Test Case 5 Screenshot:

Text

Description automatically generated

Conclusions:

That…was a bust. Honestly, I struggled incredibly with this. I am EXTREMELY surprised that at least the final test case actually PASSED. I think my main problem was that I couldn’t decide which data structure to use for the project, and kept experimenting with elements from various ones. By the end of the deadline, I was too nervous and exhausted to try and reverse anything. The extra day did not do much, as I was more focused on getting a perfect pylint score rather than actually making the program work. I resolve to try better next time.

Revised Results:

Pylint:

Well, to be fair, I did MUCH better this time. My program mostly works, besides not displaying the flower image, which I didn’t figure out how to do. I purposefully went through the program before running Pylint to try and fix as many formatting issues as possible. However, my first Pylint score for the revised program was 5.98/10.00, and this was mostly due to indentation errors. Frankly, I suspected this was going to be an issue. The indentations that my PyCharm was choosing for me on this assignment seemed very odd, and did not match anything I did before, so I manually corrected them to the way I knew, because that just seemed right. It looks like now I am going to have to correct everything back to the way it was.

Text

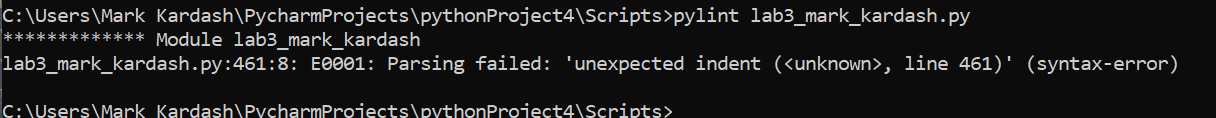
Description automatically generated

After fixing most of the mistakes, I was absolutely baffled to receive a score of 4.76/10.00, once again mostly due to bad indentation. I thought that correcting the indents to what PyCharm had suggested would fix things, but it seemed to have no effect. I was now determined to find the problem.

Text

Description automatically generated

After paying close attention to how many spaces were required for each indent, I once again corrected the code. I initially had an “unexpected indent” error.

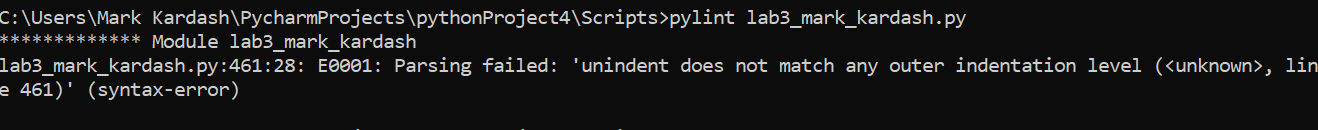


Then I had a few “Bad indentation” errors from fixing that, but still had a score of 9.17/10.00

Graphical user interface, text

Description automatically generated

Parsing error again:

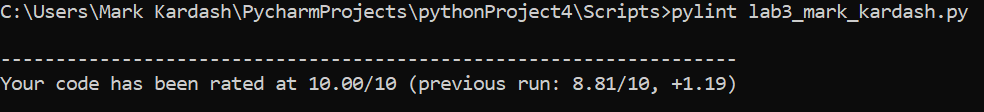


Indent errors! Again! Score of 8.81/10.00

Graphical user interface, text

Description automatically generated

Finally, after a few more tweaks, my score was a perfect one.



Revised Test Cases:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test Case: | Input (In Order): | Expected Output: | Actual Output: | Pass? |
|  | 1 | The data for all 50 states (Name, Capital, Population, and Flower, are displayed in a table. | The data for all 50 states (Name, Capital, Population, and Flower, are displayed in a table. | Yes |
|  | 2, Alabama | The program prompts the user to select a state, and then prints the data for that particular state, along with the image of the state flower. | The program prompts the user for a state, prints the header twice, then prints the corresponding data, but DOES NOT display the image, instead showing a “Keyerror” error message. | No |
|  | 3 | Program displays a graph of the 5 most populated states. | Program displays a graph of the 5 most populated states. | Yes |
|  | 4, Maryland | The program prompts the user for the state updates the population of Maryland to the new number. | Population successfully updated (Screenshots below for comparison).  However, the program does still display the image error. | Yes |
|  | 5 | “Thank you for using our program. Goodbye!” message is displayed as the program exits. | “Thank you for using our program. Goodbye!” message is displayed as the program exits. | Yes |

Test Case 1 Screenshot:

Text

Description automatically generated

Text

Description automatically generated



Test Case 2 Screenshot:

Text

Description automatically generated

Test Case 3 Screenshot:

Chart, bar chart

Description automatically generated

Test Case 4 Part A Screenshot:

Maryland population before.

Text

Description automatically generated

Test case result:

Text

Description automatically generated

Test Case 4 Part B Screenshot:

Maryland population check.

Text

Description automatically generated

Test Case 5 Screenshot:

Text

Description automatically generated

Updated Conclusion:

A much, much better. result this time. Mostly because I didn’t have a deadline that catastrophically close as I did when first submitting this. I tried a couple of ways to get the image to display, but ultimately failed. Felt I was close, but was missing something.

Revised Results (Again) (2/23/2023):

I have finally managed to get the flower images to display, making my program work perfectly. Let us start with the pylint results.

Pylint Testing:

My first official Pylint run gave me a score of 7.88/10. I had been previously using pylint while coding, so that I avoid basic mistakes, which is why it shows a previous result. My sole mistakes were lines that were too long, with the urls of the state flowers. I therefore set out to correct them.

A computer screen capture

Description automatically generated with medium confidence

This landed my code a perfect score.

Text

Description automatically generated

Test Cases:

I was not worried about the test cases at all, since I only changed the function calling the image, with the rest of the program remaining virtually unchanged. I ran the exact same test cases all over again.

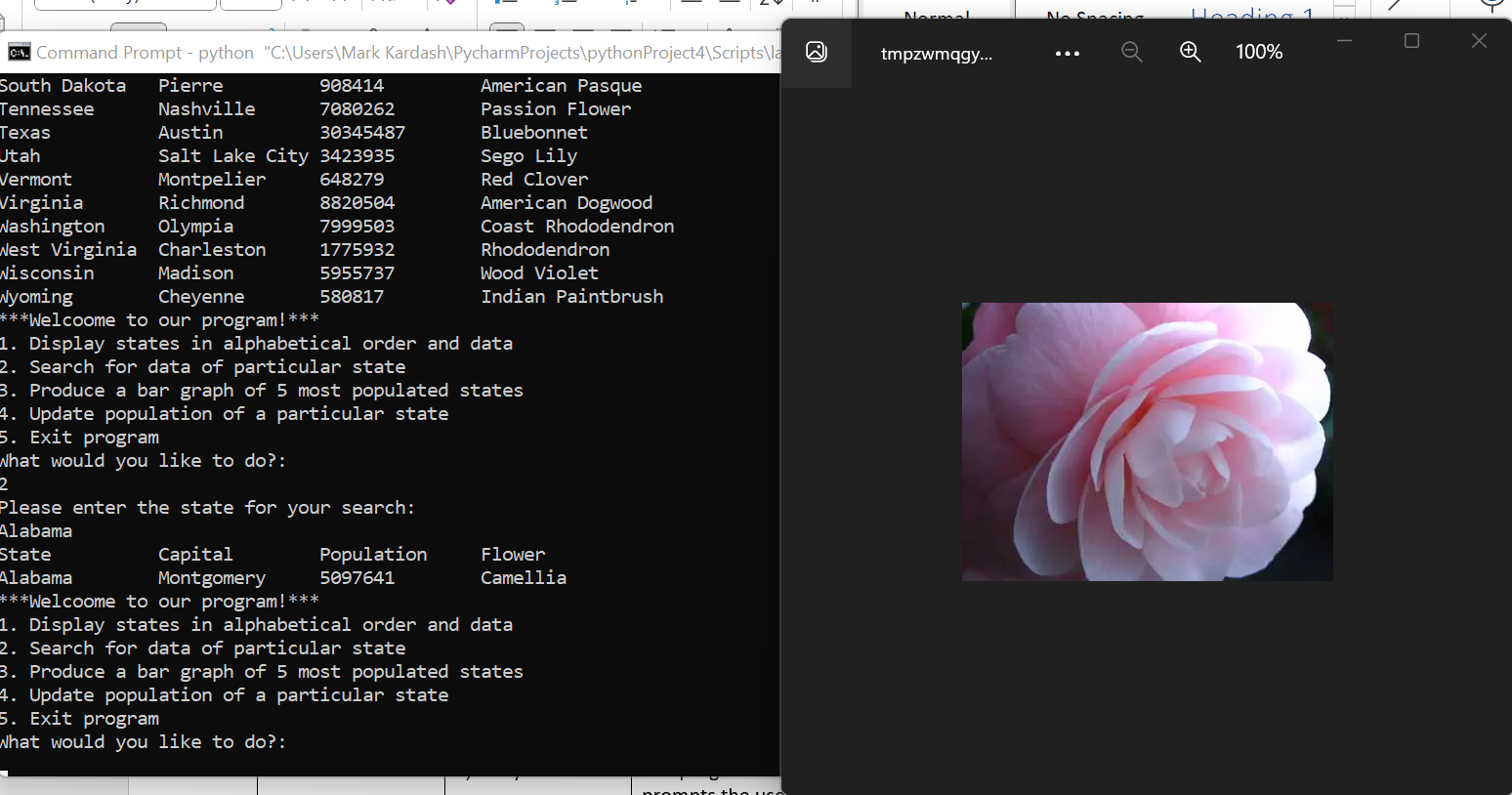
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test Case: | Input (In Order): | Expected Output: | Actual Output: | Pass? |
|  | 1 | The data for all 50 states (Name, Capital, Population, and Flower, are displayed in a table. | The data for all 50 states (Name, Capital, Population, and Flower, are displayed in a table. | Yes |
|  | 2, Alabama | The program prompts the user to select a state, and then prints the data for that particular state, along with the image of the state flower. | The program prompts the user to select a state, and then prints the data for that particular state, along with the image of the state flower. | Yes |
|  | 3 | Program displays a graph of the 5 most populated states. | Program displays a graph of the 5 most populated states. | Yes |
|  | 4, Maryland | The program prompts the user for the state updates the population of Maryland to the new number (In my case, 7,000,000) | Population successfully updated (Screenshots below for comparison). | Yes |
|  | 5 | “Thank you for using our program. Goodbye!” message is displayed as the program exits. | “Thank you for using our program. Goodbye!” message is displayed as the program exits. | Yes |

Test Case 1 Screenshot:

Text

Description automatically generated

Test Case 2 Screenshot:



Test Case 3 Screenshot:

Graphical user interface

Description automatically generated with low confidence

Test Case 4 Screenshots:

Before update:

Text

Description automatically generated

After update:

Text

Description automatically generated

Test Case 5 Screenshot:

Text

Description automatically generated

Conclusion:

It took me quite a while to figure out how to display the images on the screen, but once I did it, the program worked perfectly, without a single error. I can now say I’m officially done with it, after seeing that perseverance and enough time can get us the desired result.