Mark Kardash

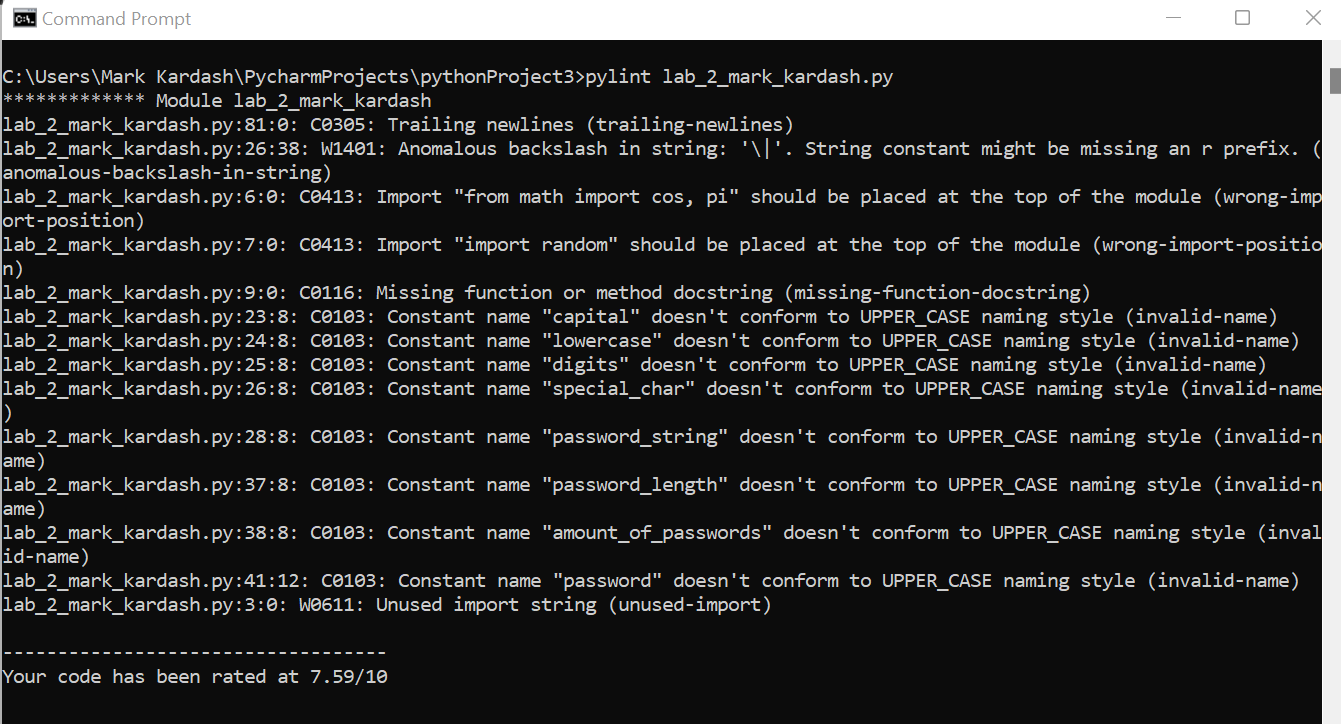
SDEV300:

(Scroll to page 4 for revised results from 02.08.2022)

Initial Results:

Pylint Grading Results:

As always, I am starting from the Pylint results. My first score was 7.59. There were mostly capitalization errors, along with some placing mistakes, and a “trailing newlines” warning. I decided to deal with the capitalization errors first since they were the easiest to repair.



My program now had much fewer errors, but received a new grading of 7.89, which was still quite low. I proceeded to fix several overlooked capitalization errors, and add a doc string.

Text

Description automatically generated

Now getting the hopeful score of 9.82, my program had only one error left: trailing newlines. Luckily, having dealt with this problem last week, I quickly figured out how to tackle it.

Text

Description automatically generated

Finally, I received the desired score of 10.00/10

Text

Description automatically generated

Test Cases:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test Case: | Input: | Expected output: | Actual Output: | Pass? |
| 1. | a | Program generates 10 random passwords with letters, numbers, and other characters. | Program generates passwords infinitely. | No |
| 2. | c | “There are 892 days remaining until July 4, 2025” | “TypeError: can only concatenate str (not “datetime.date” to str” | No |
| 3. | f | “Thank you for using our program. Have a great day!” |  | No (Stuck at previous error message). |

Test Case 1 Screenshot:

Text

Description automatically generated with medium confidence

Test Case 2 Screenshot:

Text

Description automatically generated

Test Case 3:

Text

Description automatically generated

Conclusions:

My program has failed. There seems to be some fundamental error(s) within it, as none of the test cases passed. The one that came the closest to doing so is Test Case 1, as the passwords were actually generated, albeit not meeting the limit. In the second test case, there seems to be a datatype issue, as “str” datatype is shown to not be compatible with the “datetime.date” function. Finally, I was not even able to perform the final test case, as the program seemed to remain at the output error of the previous one. My final conclusion is that the program needs serious repairs.

Revised Results:

After having been given the chance to resubmit the program, I did a few tweaks, and fixed my mistakes. In doing so, I realized that most of my mistakes were quite small and silly. I believe the only reason I didn’t correct them was because I was racing against time to submit my work, with mere minutes remaining. I now am much more confident that the program will function as intended. As usual, let’s start with the Pylint score.

Pylint Results:

My first new Pylint score was lower than the goal of 10, but, admittedly, not that bad. The 8.33 score seems to mostly be produced by errors in the naming style, and lines that were too long.

Graphical user interface, text

Description automatically generated

After correcting the capitalization errors, I was only left with a bunch of “statement has no effect” warnings, and a score of 9.34/10.00.

A picture containing text

Description automatically generated

I proceeded to remove some statements, but that seemed to affect the program’s functionality, so I reversed my previous corrections of capitalization. That gave me a score of 9.22/10.00, but I was now determined to discover the root of the problem.

A picture containing text

Description automatically generated

This time, I had much more success, getting a score of 9.84/10.00, with my program still maintaining proper functionality. I was, however, very confused by the single error that kept me from getting a 10. It said there was an “unnecessary elif statement” on line 23, when in fact there was no such statement anywhere in that area.

A computer screen capture

Description automatically generated with low confidence

I encountered that error several more times.

Graphical user interface, text

Description automatically generated with medium confidence

Eventually, I decided that, since there are “break” statements, I could replace all the outer “elif” statements with “if” statements, without much change in functionality. My score rose to 9.38/10.00. Now, I had to figure out how to make the names adhere to the naming style without ruining the program’s functionality.

Text

Description automatically generated

The problem was eventually solved, as I decided to capitalize every single mention of “UPPER\_CASE”, “LOWER\_CASE”, “SPECIAL\_CHAR”, and digits. I also inserted the symbol (“”) for every time the user selects ‘N’ on a particular feature. This finally landed me a perfect score.

Graphical user interface, text

Description automatically generated

Test Cases:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test Case: | Input (In order): | Expected Output: | Actual Output: | Pass? |
|  | a, 10, Y, N, Y, N | Program outputs a 10-character password with only lowercase letters and special characters. | Your password is:  }!rb{\_)gx | Yes |
|  | b, 100, 20 | 20% of 100 is 20 | 20% of 100 is 20 | Yes |
|  | c | There are 877 days remaining until July 4th, 2025 | There are 877 days remaining until July 4th, 2025 | Yes |
|  | d, 9, 12, 67 | The length of the leg is 13.700190540528524 | The length of the leg is 13.700190540528524 | Yes |
|  | e, 7, 2.1 | The volume of the cylinder is 318369.7284423933 | The volume of the cylinder is 318369.7284423933 | Yes |
|  | f | Thank you for using our program. Have a great day! | Thank you for using our program. Have a great day! | Yes |

Test Case 1 Screenshot:

Text

Description automatically generated

Test Case 2 Screenshot:

A computer screen capture

Description automatically generated with medium confidence

Test Case 3 Screenshot:

Text

Description automatically generated

Test Case 4 Screenshot:

Text

Description automatically generated

Test Case 5 Screenshot:

Text

Description automatically generated

Test Case 6 Screenshot:

Text

Description automatically generated

Conclusions:

Upon redesign, my program passed with flying colors. As I said before, most of my mistakes were minor shortcomings that I had simply overlooked in my hurry to get things done. I could say the lesson I learned from this is to pay more attention to my code, and re-check it before running my programs.