# Name: Dikshyanta Uprety

# Student number: 20045364

# Answers to question 1

In the table below, the errors in the code, the nature of the error, their lines, the correct code and remarks are noted

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Serial Number** | **Line number in the editor** | **The line of code** | **Error raised** | **Corrected line of code** | **Remarks** |
| 1 | 17 | randomChar = chr(randint(MIN\_ASCII,Max\_ASCII)) | NameError: name 'Max\_ASCII' is not defined | randomChar = chr(randint(MIN\_ASCII,MAX\_ASCII)) | The variable 'Max\_ASCII' used in the original code is not defined.  Therefore, must be a typing error while trying to refer to the variable 'MAX\_ASCII'  defined in line 7 |
| 2 | 21 | retun result | SyntaxError: invalid syntax | return result | The line is at the end of the randomPassword() function. This is where a return statement is usually located.  Therefore, 'retun' must be a typing mistake while trying to use the 'return' keyword. |
| 3 | 36 and  37 | elif ch >= "0" and ch <= "9": has\_num = True | IndentationError: expected an indented block | elif ch >= "0" and ch <= "9": has\_num = True | Python expects an indentation in the statement following an elif clause. The indentation is missing on line 37 after elif clause on line 36. |
| 4 | 48 | if checkPassword(): | TypeError: checkPassword() missing 1 required positional argument: 'password' | if checkPassword(p): | The function checkPassword takes a input password and determines if it is strong. Therefore, a password must be provided as a parameter so that it can be checked by the function. |
| 5 | 53 | print("Your suggested password is:", randomPassword) | Your suggested password is: function randomPassword at 0x000002202E7BC5E0 | print("Your suggested password is:", randomPassword()) | In order to call a function in the code, the name of the function has to be followed by a open and close bracket '()'. The brackets are missing while calling the function at line 53. |

# Methodology for finding the bugs

Firstly, entirety of the code was copied into visual studio code (editor of choice) in a python file. The editor immediately showed the first three errors(syntax) whereas two errors were unidentified by the editor (semantics). The methodology to find each of the error is described below:

1. The first error was identified by the editor and was the use of undefined variable 'Max\_ASCII'. Upon inspecting the code, an unused variable 'MAX\_ASCII' was declared beforehand. Therefore, it was assumed that the former was a typing error while trying to use the latter.
2. The second error was identified by the editor and was the use of 'retun' at the end of a function. Since 'return' is conventionally at the end of a function to specify what value to return when the function is invoked, this was also assumed to be a mistype while trying to use 'return' keyword like the first error.
3. The third error identified by the editor was missing indentation after an elif statement. Since python does not use curly brackets, like some programming languages such as java, to combine a body of text, an indentation is required for the body of an elif clause.
4. The fourth error which was a semantic error, not immediately identified by the editor, was a missing argument in the checkPassword() function. This was identified by comparing the function definition in line 25 which required a parameter, and the function invoking on line 48, which did not require any.
5. The fifth error, also a semantics error, unidentified by the editor was the missing'()' while invoking a function. As a result of this, the function itself was returned instead of the content specified by the return keyword within the function. This was identified when the output was not expected (a function was returned in the place of a string).

**Answers to Question 2**

**The result of the simulations from a run is given below:**

**For 4 cooks, the result of the 10 simulations and their average is listed below:**

Instance: 0, number of cars: 94, average wait time: 252.41 minutes.

Instance: 1, number of cars: 98, average wait time: 283.65 minutes.

Instance: 2, number of cars: 107, average wait time: 321.98 minutes.

Instance: 3, number of cars: 85, average wait time: 186.13 minutes.

Instance: 4, number of cars: 99, average wait time: 269.57 minutes.

Instance: 5, number of cars: 123, average wait time: 376.38 minutes.

Instance: 6, number of cars: 87, average wait time: 208.0 minutes.

Instance: 7, number of cars: 95, average wait time: 251.32 minutes.

Instance: 8, number of cars: 82, average wait time: 222.46 minutes.

Instance: 9, number of cars: 100, average wait time: 266.58 minutes.

The Average wait time: 263.85 minutes.

**For 6 cooks, the result of the 10 simulations and their average is listed below:**

Instance: 0, number of cars: 97, average wait time: 59.84 minutes.

Instance: 1, number of cars: 96, average wait time: 55.13 minutes.

Instance: 2, number of cars: 97, average wait time: 66.69 minutes.

Instance: 3, number of cars: 99, average wait time: 47.23 minutes.

Instance: 4, number of cars: 89, average wait time: 38.62 minutes.

Instance: 5, number of cars: 103, average wait time: 37.75 minutes.

Instance: 6, number of cars: 98, average wait time: 28.87 minutes.

Instance: 7, number of cars: 91, average wait time: 37.84 minutes.

Instance: 8, number of cars: 87, average wait time: 23.64 minutes.

Instance: 9, number of cars: 90, average wait time: 33.9 minutes.

The Average wait time: 42.95 minutes.

**Part (B)**

**Conclusion**

The average wait time while operating with 4 cooks is 263.85 minutes per car whereas the average wait time while operating with 6 cooks is 42.95 minutes per car. When the number of cooks is increased by 50%, the wait times is reduced by 84% (approximately).

**Part (C)**

**What is the risk of running a single simulation and why do you need to run multiple simulations?**

In the output provided, for 4 cooks the average wait time for instance 2 and 6 is 321.98 minutes and 208.0 minutes respectively. The difference between them and their average difference to the collective average is 113.98 and 57 minutes. Because the simulation uses random values, some instances of the output can be extremely small or big (outlier). Using outliers will give us exaggerated or diminished understanding of the data and will deter in further computation and decision making. Therefore, we use the average of multiple executions instead of one instance’s output to minimize the effect of such outliers.

* Word count: 94 words

**Explain the choice of data structures and algorithms for your code.**

**Data Structure**

The data structure used for the simulation is list. The wait times for each car is calculated and appended to the list. The list is used to compute the average wait time.

List is chosen because it is a simple and powerful data structure and values can be added easily using the append command.

* Word count: 54 words

**Algorithm**

The most fundamental algorithm for the solution is:

* Every minute there is a 20% chance that a car comes
* A car will order between 3 to 15 items (randomly) with equal probabilities

The use of such calculated random factors is to make the simulation as close to real life as possible.

* Word count: 51
* Collective word count(C): 199 words

**Part (D)**

**Discuss issues relating to data protection and GDPR that the csv file’s users should be concerned about.**

According to the European Union’s General Data Protection Regulation (GDPR) regulation personal data includes all data which are or can be assigned to a person in any kind of way. Some examples are telephone, credit card or personnel number of a person, account data, number plate, appearance, customer number or address, etc. (Intersoft Consulting, 2021)

From above, we can see that the number plate information of a person’s car, among other information is a personal data belonging to that person and can be used to identify them. The CSV file displays the number plate information of the cars alongside the orders. Therefore, any person who has access to the CSV file can potentially use the data for sinister purposes unintended by the producer of the csv file. Such action is called data breach. Any breach of personal data is punishable by huge financial penalty, both to the party writing the CSV and the party using it. (Intersoft Consulting, 2021)

The number plate information of the cars does not have any effect on the simulation designed. Therefore, a possible solution could be to omit the information completely from the csv file. If the car’s number plate would be useful for computing values and analysis outside the scope of the simulation, the data could be encrypted to facilitate better security. Encrypting the data reduces the probability of data breach greatly while still providing the data to intended parties. It can be used to ensure that only the intended parties are allowed to use the data and can be used to monitor the usage.

* Word count: 261 words

**Answers to Question 3**

**Part III**

For arrays of dimension 100\* 100, the 10 instances along with their running times (for both bubble and merge sort) are given below. An average for these 10 instances has been calculated and will be used to compare the two sorts.

**For Bubble sort:**

Instance 0, time taken: 0.061 seconds.

Instance 1, time taken: 0.059 seconds.

Instance 2, time taken: 0.059 seconds.

Instance 3, time taken: 0.059 seconds.

Instance 4, time taken: 0.059 seconds.

Instance 5, time taken: 0.061 seconds.

Instance 6, time taken: 0.06 seconds.

Instance 7, time taken: 0.058 seconds.

Instance 8, time taken: 0.059 seconds.

Instance 9, time taken: 0.059 seconds.

The average time for the above instances is 0.059 seconds.

**For merge sort:**

Instance 0, time taken: 0.017 seconds.

Instance 1, time taken: 0.018 seconds.

Instance 2, time taken: 0.018 seconds.

Instance 3, time taken: 0.019 seconds.

Instance 4, time taken: 0.019 seconds.

Instance 5, time taken: 0.018 seconds.

Instance 6, time taken: 0.02 seconds.

Instance 7, time taken: 0.018 seconds.

Instance 8, time taken: 0.018 seconds.

Instance 9, time taken: 0.019 seconds.

The average time for the above instances is 0.018 seconds.

**A brief commentary on the result:**

The time taken by bubble sort is about 3.3 times the time taken by merge sort.

**Are the results expected?**

The premise is that the time complexity of bubble sort is O(n^2) and that of merge sort is O(n\*log(n)), where n is the number of inputs. For any positive value of n, the value of n\*log(n) will always be less than (n^2). Therefore, for the same number of inputs, the time taken by merge sort should be less than bubble sort. Since that statement is justified by the results (0.059 seconds for bubble sort and 0.018 seconds for merge sort), I will conclude that the results are expected.

* Word count: 88

**Part IV**

For arrays of dimension 500\* 500, the 10 instances along with their running times (for both bubble and merge sort) are given below. An average for these 10 instances has been calculated and will be used to compare the two sorts.

**For bubble sort:**

Instance 0, time taken: 7.743 seconds.

Instance 1, time taken: 7.782 seconds.

Instance 2, time taken: 7.876 seconds.

Instance 3, time taken: 7.746 seconds.

Instance 4, time taken: 7.706 seconds.

Instance 5, time taken: 7.75 seconds.

Instance 6, time taken: 8.018 seconds.

Instance 7, time taken: 7.873 seconds.

Instance 8, time taken: 7.767 seconds.

Instance 9, time taken: 7.691 seconds.

The average time for the above instances is 7.795 seconds.

**For merge sort:**

Instance 0, time taken: 0.531 seconds.

Instance 1, time taken: 0.549 seconds.

Instance 2, time taken: 0.547 seconds.

Instance 3, time taken: 0.535 seconds.

Instance 4, time taken: 0.549 seconds.

Instance 5, time taken: 0.565 seconds.

Instance 6, time taken: 0.547 seconds.

Instance 7, time taken: 0.534 seconds.

Instance 8, time taken: 0.533 seconds.

Instance 9, time taken: 0.547 seconds.

The average time for the above instances is 0.543 seconds.

**A brief commentary on the result:**

The time taken by bubble sort is about 14.4 times the time taken by merge sort.

**Are the results expected?**

We already know that time complexity of bubble sort is O(n^2) and that of merge sort is O(n\*log(n)), where n is the number of inputs. As the value of n increases, the value of (n^2) increases more sharply than the value of (n\*log(n)). Therefore, when number of inputs are increased significantly, the time taken by the two algorithms should show greater difference. For an array of dimension (100\*100), the time taken by bubble sort was 3.3 times greater whereas for an array of dimension (500\*500), it is 14.4 times greater. Since the difference has increased significantly, the results are expected.

* Word count: 100

**References**

Intersoft Consulting (2021) GDPR Personal Data. Available from: https://gdpr-info.eu/issues/personal-data/ [Accessed 28 April 2022].

Intersoft Consulting (2021) Fine and Penalties. Available from: https://gdpr-info.eu/issues/fines-penalties/) [Accessed 28 April 2022].