**Systems programming Assignment 1 Report**

**Chart, histogram

Description automatically generatedOverview and Functionality of the Program**

**Chart, line chart, histogram

Description automatically generated**Written for this assignment is essentially a dice rolling analysis program used to simulate rolling die/dice numerous times, processing the data, and displaying it visually to the user. When the user runs the program, they are given a main menu with 5 different options to plot a histogram, KDE plot, print out analysis, load examples and to exit the program. The user can enter integers 1 to 5 to choose which option they wish to select. To run the histogram functionality the user is prompted to enter 1. They can then proceed to enter all the information about the dice they desire to roll (number of dice, number of rolls, number of sides for each dice). The user can fully customise what arrangement of dice are simulated. The program will then produce raw data and a histogram which can both be saved into a file by the user. The range for x and y axis and number of histogram bars is automatically calculated producing readable and clear histograms regardless of the data entered. If the user decided to enter 2 and create a KDE plot, they will be prompted to again provide all the required information about the die/dice. It will then produce a KDE plot or kernel distribution estimation plot which is essentially a continuous and smoothed histogram represented using a line. If the user wishes to print out an analysis of the dice rolling simulation it will display to the user the mean, median, mode, maximum, minimum and deviation for the results. The user can then also choose to generate pre-loaded examples for histogram, KDE plot and analysis. They then have a choice of 3 examples to return the results for. Once the user has completed their analysis it will keep returning to the main menu until the user enters 5 to quit the program.

Figure 2: KDE plot

Figure 1: Histogram Plot

**Design of the Program**

When designing the programme several libraries had to be imported. This included importing the matplotlib library that is used to generate the plots and their visuals. Another library used in the code was the statistics library which was used to generate the mean, median, mode and deviation of the data. Seaborn library was also imported which is used to generate the KDE plot for the program. Finally, the randint function from the random library was imported. This function was crucial in designing the program and forms the basis for simulating the dice. To simulate the dice in my program a class was created. This allowed me to create multiple objects to represent the dice. Each object has the attribute of the number of sides and a list of results which stores the outcome for rolling the dice. The class has one method which is used to generate random integers based upon the number of sides for the dice; it then appends these to the result attribute. As I wanted the user to continuously return to the main menu until they wish to quit. I nested the code in my main function in a while loop which while true will keep repeating until the break function is called when the user enters 5. To receive input from the user I realised that much of the inputs needed are integers greater than 0. I created a function called getint which contains all the error handling and using the built-in input function to take the user’s input. This function is used throughout the entire code and helps save repeating unnecessary lines of code. Another design element implanted into the code would be use of dictionaries to store the data for the example plots. Using dictionaries allowed me to store data with a key and corresponding value. The data included the type of plot, the dice, and the number of rolls. This allows for easy look up and the correct plot can be made depending on which dictionary is used. Use of this is much more efficient than alternative data types.

**Possible Improvements**

When looking at the code there is improvement to be made. One feature that could be implemented would be to have dice that are imbalanced/unfair. This could be set by the user and would add a further element of functionality to the code. Another way the code could be improved is with a more interactive UI in which the number of dice, rolls and sides could be changed on the fly. The plot displaying the data will instantly react to any of these changes making a more interactive and easier to use program. The test file currently only tests getint and analysis functions. more functions within the code should be tested and the test should be more thorough to ensure robustness of the functions. Currently my program has no limit on how many rolls or dice can be entered by the user. This needs to be resolved to prevent overflow error. Given more time some of these features could be implemented but as it stands, I am happy with the functionality of the code and what I’ve learnt writing it.