**Design Document (Scrapbook Style) – Python**

Python Design Document

**Introduction**

For python the paradigm used is Procedural. The goals were successfully implementing the Slot machine whilst adhering to using as many procedural paradigms features as possible.

**Pseudo Code**

**Define Constants**

Declare constants and global variables such as **rows/cols** and **bet**

**Create Player Functions:**

Define the **get\_balance**, **add\_balance**, and **minus\_balance** functions in the **main** module to access, increase, and decrease the player's balance.

**Create Slot Machine Functions:**

Define the **print\_slot\_machine**, **create\_symbol\_list**, **create\_symbol\_grid …**  functions in the **main** module .

**Implement methods that represent a JackPot and Normal Machine:**

Implement the **get\_slot\_machine\_spin** function for both a **Normal Machine** and **Jackpot Machine** by creating a symbol grid based on the specified rows, columns, and symbols.

**Start the Game:**

In the **main** module, initialise the **balance** variable to 10.

Create a loop that continues until the user decides to exit.

In each iteration, generate a random number and prompt the user to press 's' to spin or 'e' to exit.

If the user chooses to spin, call either the **spin** or **spinLarge** function based on the random number.

Update the balance based on the winnings and total bet amount.

If the user chooses to exit, print the final balance.

**Sketches and Diagrams**

**A screenshot of a computer

Description automatically generated**

**Design Process**

Designing this in a procedural manure I knew would be the easiest. I can re-use the functions/methods I have used in C++ and Go however without object to function association. Here I develop without a player class just functions.

A screenshot of a computer

Description automatically generated

I then separated the get\_slot\_machine\_spin into more functions as shown here. We now have a separate create symbol list and create symbol grid. Which the get\_slot\_machine\_spin uses in different ways.

A screenshot of a computer screen

Description automatically generated

I then developed the check winnings function. Most of the implemtation was quite easier than others as this was my last language . So vast majority of methods were similar in approach.A screen shot of a computer

Description automatically generated

Next was the spin method. Again, as said before the implementation followed same languages as before.

A screen shot of a computer

Description automatically generated

At this point the Normal slot machine implementation was working correctly whilst demonstrating features of the Procedural paradigm however the jackPot feature was not implemented.

A screenshot of a computer

Description automatically generated

Using the random generator, I managed to switch between the methods. I only needed to create the spin function again to account for the JackPot machine variation. Code was completed.

A screenshot of a computer

Description automatically generated