Report On

# Slotz : A Betting Game

Submitted in partial fulfillment of the requirements of the Course Project for Skill Based Laboratory of Python Programming in Semester IV of Second Year Artificial Intelligence & Data Science Engineering


by

Yash Nilesh Kasare (Roll No. 24 )
Rushikesh Masal (Roll No.32)
Gaurav Nevrekar (Roll No.36)


**Under the guidance**

**Mr. Raunak Joshi**


**University of Mumbai**

**Vidyavardhini's College of Engineering & Technology**

**Department of Artificial Intelligence and Data Science**


**(A.Y. 2024-25)**

**Vidyavardhini's College of Engineering and Technology**

Department of Artificial Intelligence & Data Science

# CERTIFICATE

This is to certify that the project entitled "Slotz : A Python Betting Game" is a bonafide work of Yash Kasare (Roll No.24), Gaurav Nevrekar (Roll No.36), Rushikesh Masal (Roll No.32)" submitted to the University of Mumbai in partial fulfillment of the requirement for the Course project in semester IV of Second Year Artificial Intelligence and Data Science engineering.

Prof Raunak Joshi
Guide

## Abstract

This project is a terminal-based online betting game developed using Python that simulates a slot machine experience. The game allows users to deposit a balance, select the number of lines (up to three) they wish to bet on, and place a wager for each line within predefined limits. The core functionality of the game revolves around generating a 3x3 slot machine grid filled with randomly selected symbols, each with assigned frequencies and payout values, thus creating a realistic betting scenario. Players win when identical symbols align across the selected lines, and their earnings are calculated based on the symbol's value and the bet amount. The game continuously updates the user's balance, ensuring transparency and maintaining the excitement of real-time gameplay. Robust input validation ensures the user enters valid data for deposits, number of lines, and betting amounts, enhancing user experience and minimizing errors. This project serves as a strong foundation for understanding fundamental Python programming concepts such as modular coding, loops, conditionals, lists, and randomization, and it can be expanded further with features like user accounts, GUI interfaces, or multiplayer capabilities.

# Table of Contents

## 1.Introduction

The evolution of digital entertainment has led to a surge in the popularity of online games, with betting and casino-style games being a prominent category. This project focuses on the development of a **simple online betting game** using Python, designed to simulate the mechanics of a classic slot machine. The primary objective of the game is to provide an engaging, interactive experience where users can place bets, spin a virtual slot machine, and earn winnings based on symbol alignment.

This terminal-based game incorporates basic elements of betting such as balance deposit, line selection, wagering per line, and payout calculations. By using Python's built-in functionalities and libraries like `random`, the game ensures unpredictability and fairness in outcomes. The design emphasizes user interaction through clear prompts and input validation, ensuring a smooth and error-free gameplay experience.

This project not only showcases how fundamental programming concepts such as loops, functions, conditionals, and list manipulation can be applied in game development but also provides a practical understanding of how real-world betting systems operate at a basic level. It serves as an ideal foundation for beginners interested in game development, and it opens doors for future enhancements such as graphical interfaces, score tracking, and multiplayer features.

## 2.Problem statement:

The lack of simple, educational, and risk-free platforms for understanding the core logic of betting games poses a challenge for learners and developers interested in game design and probability-based systems. Most existing betting platforms are either too complex, require real financial transactions, or lack transparency in how the underlying logic works. There is a need for a basic, interactive slot machine simulation that can help users experience the essential features of a betting system—such as placing bets, random outcomes, calculating winnings, and managing virtual balance—without any real-world consequences. This project aims to solve this problem by developing a Python-based terminal betting game that replicates the mechanics of a traditional slot machine, allowing users to place line-based bets, spin for random results, and receive payouts based on matching symbols, all within a controlled, text-based environment.

## 3. Proposed System

### 3.1 Architecture Overview

The proposed system is a **console-based slot machine betting game** developed using Python. It follows a **modular architecture**, with different functions handling specific aspects of the game

such as input collection, balance tracking, symbol generation, result evaluation, and user interaction. The overall system architecture can be broken down into the following components:

- **User Input Module:** Handles user deposits, number of lines selection, and bet amount per line with validation.
- **Slot Machine Logic Module:** Randomly generates symbols based on predefined probabilities using Python's `random` library.
- **Winnings Evaluation Module:** Compares the outcome across selected lines to determine winnings using symbol value logic.
- **Display Module:** Presents the slot machine grid and winnings in a structured and readable format.
- **Game Loop Controller:** Controls the flow of the game, updates user balance, and manages the session until the user exits.

## 4. Implementation Plan Details

The system was implemented in the following phases:

- **Phase 1: Requirements Definition & Game Design**
  - Defined game rules: 3x3 slot machine grid, max 3 betting lines, and symbol-based rewards.
  - Assigned probabilities and values to symbols (A, B, C, D).
- **Phase 2: Functional Module Development**
  - Created individual functions for deposit, input handling, slot machine spin generation, winnings calculation, and display.
- **Phase 3: Integration and Game Loop**
  - Integrated all modules into a main game loop that supports continuous gameplay until the user quits.
- **Phase 4: Testing and Debugging**
  - Tested for various input scenarios, invalid data entries, and edge cases (e.g., zero balance, max/min bets).

## 5.Implementation Result and Analysis

The final implementation successfully created a fully functional slot machine simulation. Key features included:

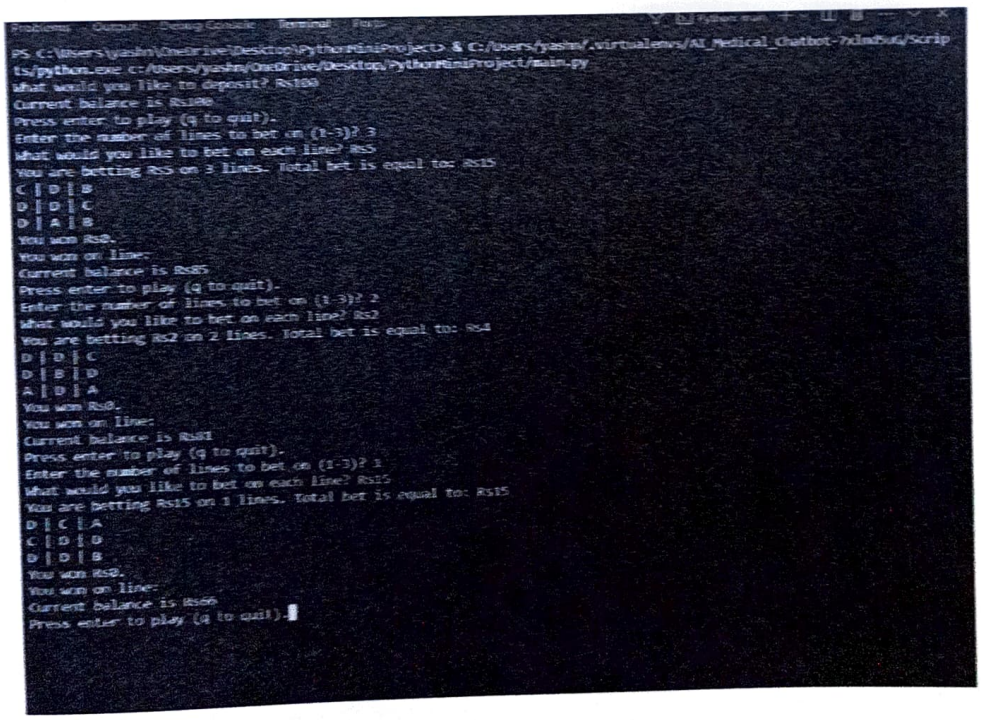
- Dynamic symbol generation using realistic frequencies.
- Bet validation and line selection within user-defined limits.
- Clear output display showing spin results, winnings, and updated balance.
- Accurate calculation of winnings based on matching symbols across bet lines.

Users were able to interact with the game smoothly via the terminal, with real-time feedback on their bets and results. The code structure was modular and easy to modify for future enhancements such as GUI integration or expanded game logic.

The implemented system effectively meets its objectives by offering a simple, engaging, and educational simulation of a betting game. It reinforces concepts like probability, user input handling, and game logic implementation in Python. Through modular function design, the code ensures clarity, reusability, and maintainability. The terminal interface, although basic, serves as an ideal environment for learning and understanding game logic without the need for complex graphics.

From a learning and development standpoint, the project demonstrates how basic programming techniques can be applied to model real-world systems in a fun and interactive way. Future improvements may include graphical UI using libraries like Tkinter or PyGame, user account systems, and expanded betting options to enhance user engagement and experience.

## Implementation:



## 6. Conclusion

The development of this Python-based online betting game successfully demonstrates how core programming concepts can be applied to create an interactive and engaging terminal application. By simulating a slot machine, the project provides users with a simplified yet realistic betting experience, including deposit handling, line-based betting, symbol randomization, and reward calculation. Throughout the implementation, focus was placed on user input validation, modular code design, and a smooth gameplay loop to ensure reliability and user satisfaction.

This project not only achieved its goal of modelling a basic betting system but also served as a practical exercise in problem-solving, logic building, and understanding the fundamentals of game development. It highlights the versatility of Python in handling real-world simulations and offers a strong foundation for future enhancements such as graphical interfaces, account management, and expanded betting logic. Overall, the project was a valuable learning experience in both programming and game design, providing insights into the structure and behaviour of simple casino-style applications.

## 7. Code

**main.py**

```
import random

MAX_LINES = 3
MAX_BET = 100
MIN_BET = 1

ROWS = 3
COLS = 3

symbol_count = {
    "A": 2,
    "B": 4,
    "C": 6,
    "D": 8
}

symbol_value = {
    "A": 5,
    "B": 4,
```

```
    "C": 3,
    "D": 2
}


def check_winnings(columns, lines, bet, values):
    winnings = 0
    winning_lines = []
    for line in range(lines):
        symbol = columns[0][line]
        for column in columns:
            symbol_to_check = column[line]
            if symbol != symbol_to_check:
                break
        else:
            winnings += values[symbol] * bet
            winning_lines.append(line + 1)

    return winnings, winning_lines


def get_slot_machine_spin(rows, cols, symbols):
    all_symbols = []
    for symbol, symbol_count in symbols.items():
        for _ in range(symbol_count):
            all_symbols.append(symbol)

    columns = []
    for _ in range(cols):
        column = []
        current_symbols = all_symbols[:]
        for _ in range(rows):
            value = random.choice(current_symbols)
            current_symbols.remove(value)
            column.append(value)

        columns.append(column)

    return columns


def print_slot_machine(columns):
    for row in range(len(columns[0])):
        for i, column in enumerate(columns):
            if i != len(columns) - 1:
                print(column[row], end=" | ")
```

```
            else:
                print(column[row], end="")

        print()


def deposit():
    while True:
        amount = input("What would you like to deposit? Rs")
        if amount.isdigit():
            amount = int(amount)
            if amount > 0:
                break
            else:
                print("Amount must be greater than 0.")
        else:
            print("Please enter a number.")

    return amount


def get_number_of_lines():
    while True:
        lines = input(
            "Enter the number of lines to bet on (1-" + str(MAX_LINES) + ")? ")
        if lines.isdigit():
            lines = int(lines)
            if 1 <= lines <= MAX_LINES:
                break
            else:
                print("Enter a valid number of lines.")
        else:
            print("Please enter a number.")

    return lines


def get_bet():
    while True:
        amount = input("What would you like to bet on each line? Rs")
        if amount.isdigit():
            amount = int(amount)
            if MIN_BET <= amount <= MAX_BET:
                break
            else:
                print(f"Amount must be between Rs{MIN_BET} - Rs{MAX_BET}.")
```

```
        else:
            print("Please enter a number.")

    return amount


def spin(balance):
    lines = get_number_of_lines()
    while True:
        bet = get_bet()
        total_bet = bet * lines

        if total_bet > balance:
            print(
                f"You do not have enough to bet that amount, your current balance
is: Rs{balance}")
        else:
            break

    print(
        f"You are betting Rs{bet} on {lines} lines. Total bet is equal to:
Rs{total_bet}")

    slots = get_slot_machine_spin(ROWS, COLS, symbol_count)
    print_slot_machine(slots)
    winnings, winning_lines = check_winnings(slots, lines, bet, symbol_value)
    print(f"You won Rs{winnings}.")
    print(f"You won on line:", *winning_lines)
    return winnings - total_bet


def main():
    balance = deposit()
    while True:
        print(f"Current balance is Rs{balance}")
        answer = input("Press enter to play (q to quit).")
        if answer == "q":
            break
        balance += spin(balance)

    print(f"You left with ${balance}")


main()
```

## References

1. Python Software Foundation. *Python Documentation*. Retrieved from: https://docs.python.org/3/
2. W3Schools. *Python Tutorial*. Retrieved from: https://www.w3schools.com/python/
3. GeeksforGeeks. *Python Random Module*. Retrieved from: https://www.geeksforgeeks.org/python-random-module/
4. Real Python. *Python Functions and Loops*. Retrieved from: https://realpython.com/
5. Stack Overflow. *Community discussions and code examples*. Retrieved from: https://stackoverflow.com/
6. Programiz. *Python Input, Output and Import*. Retrieved from: https://www.programiz.com/python-programming
7. Invent With Python. *Making Games with Python & Pygame*. Retrieved from: https://inventwithpython.com/pygame/