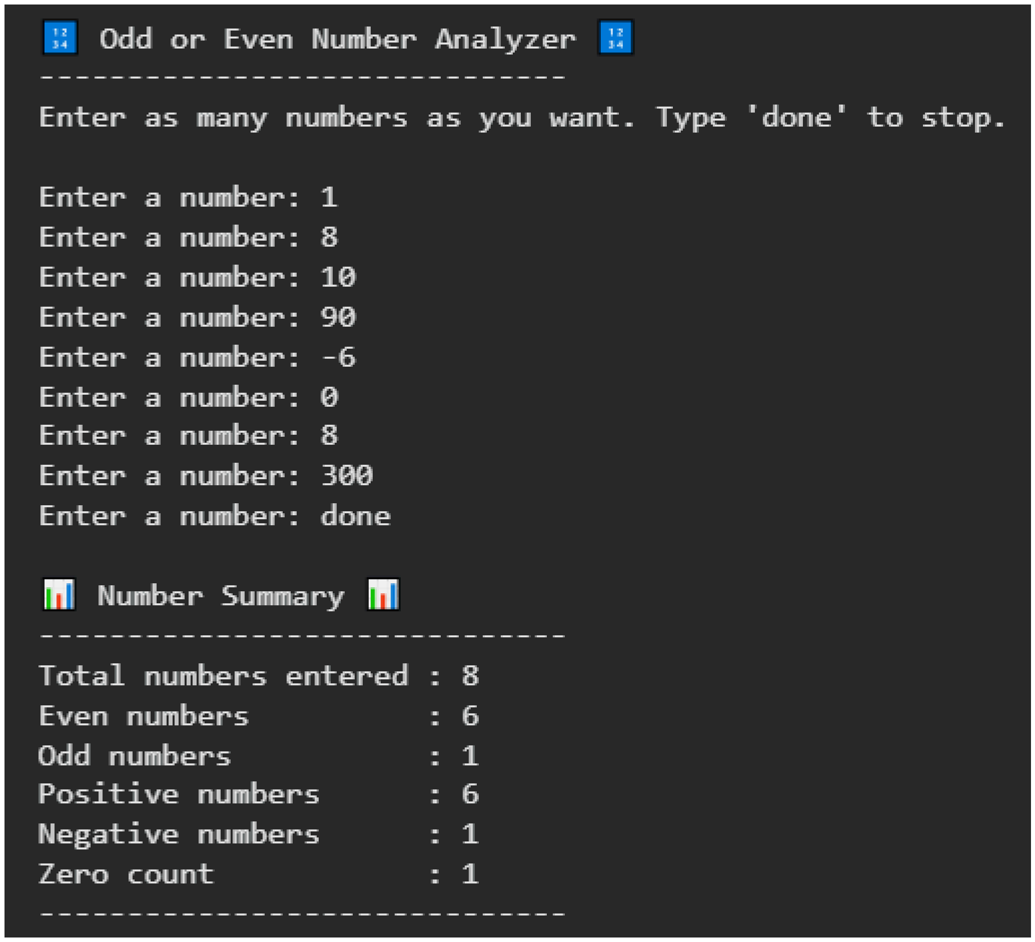
Machine Problem 1

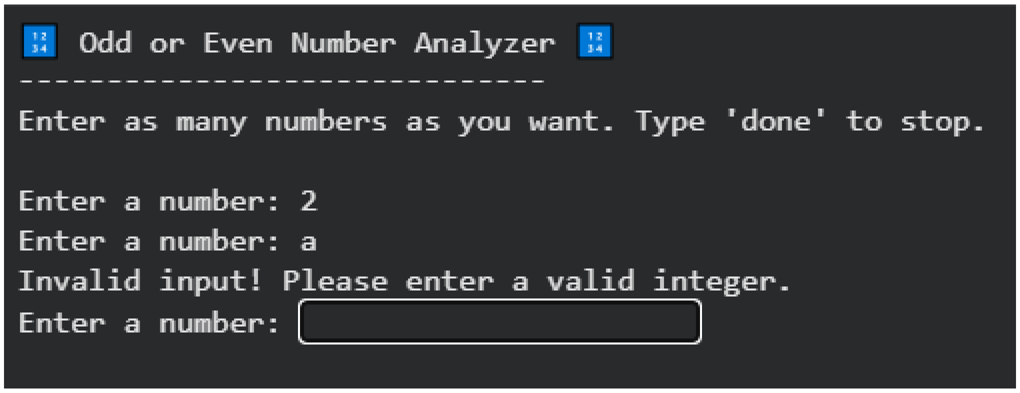
Odd or Even Number Analyzer

The Odd or Even Number Analyzer is a Python console application that allows users to enter multiple integers and analyze their characteristics. It determines how many numbers are odd, even, positive, negative, or zero using loops, conditionals, and functions. All valid inputs are stored in a list, and invalid entries are handled with an error message. The analyze_numbers() function processes the list and counts each category accordingly. Finally, the program displays a clear summary of the results for easy understanding.

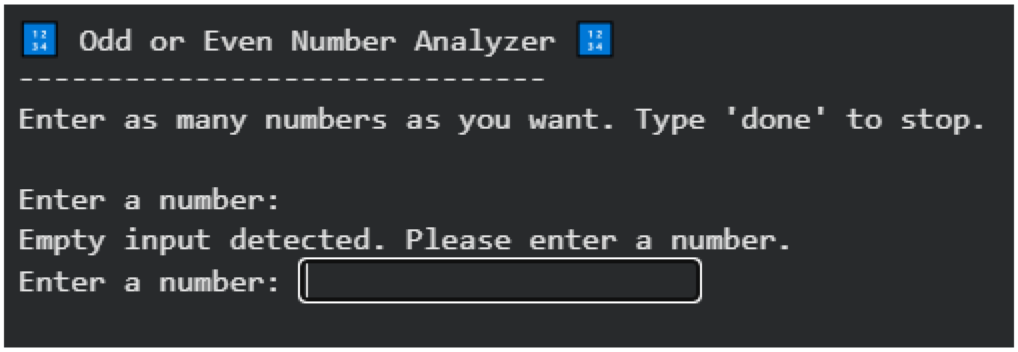
Screenshots Output:



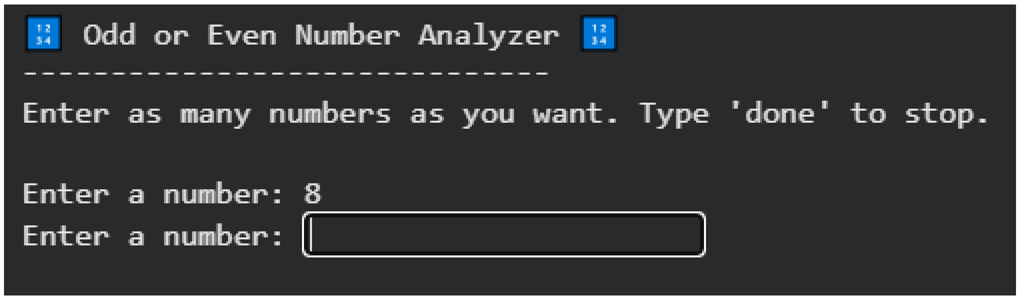
```
🔢 Odd or Even Number Analyzer 🔢
------------------------------
Enter as many numbers as you want. Type 'done' to stop.

Enter a number: 1
Enter a number: 8
Enter a number: 10
Enter a number: 90
Enter a number: -6
Enter a number: 0
Enter a number: 8
Enter a number: 300
Enter a number: done

📊 Number Summary 📊
------------------------------
Total numbers entered : 8
Even numbers          : 6
Odd numbers           : 1
Positive numbers      : 6
Negative numbers      : 1
Zero count            : 1
------------------------------
```



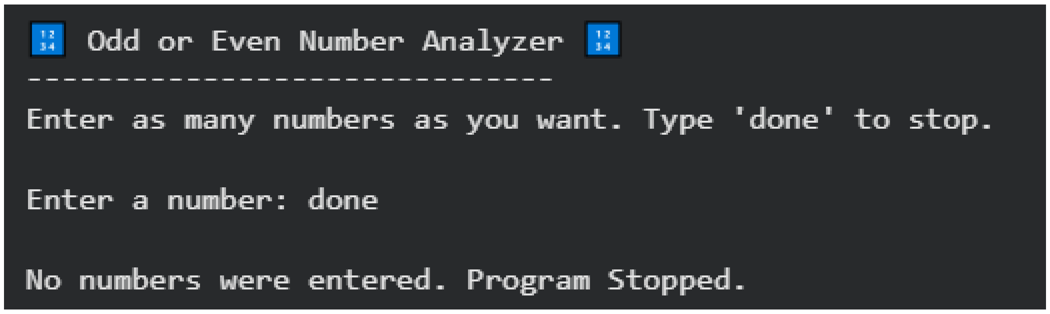
```
🔢 Odd or Even Number Analyzer 🔢
------------------------------
Enter as many numbers as you want. Type 'done' to stop.

Enter a number: 2
Enter a number: a
Invalid input! Please enter a valid integer.
Enter a number:
```



```
🔢 Odd or Even Number Analyzer 🔢
------------------------------
Enter as many numbers as you want. Type 'done' to stop.

Enter a number:
Empty input detected. Please enter a number.
Enter a number:
```



```
🔢 Odd or Even Number Analyzer 🔢
------------------------------
Enter as many numbers as you want. Type 'done' to stop.

Enter a number: 8
Enter a number:
```



```
🔢 Odd or Even Number Analyzer 🔢
------------------------------
Enter as many numbers as you want. Type 'done' to stop.

Enter a number: done

No numbers were entered. Program Stopped.
```

Code:

```
def analyze_numbers(numbers):
    summary = {
        "even": 0,
        "odd": 0,
        "positive": 0,
        "negative": 0,
        "zero": 0
    }

# number type
    for num in numbers:

        if num == 0:
            summary["zero"] += 1
        else:
            summary["positive" if num > 0 else "negative"] += 1
            summary["even" if num % 2 == 0 else "odd"] += 1

    return summary

# sumamry
def display_summary(results, total):
    print("\n📊 Number Summary 📊")
    print("-" * 30)
    print(f"Total numbers entered : {total}")
    print(f"Even numbers          : {results['even']}")
    print(f"Odd numbers           : {results['odd']}")
    print(f"Positive numbers      : {results['positive']}")
    print(f"Negative numbers      : {results['negative']}")
    print(f"Zero count            : {results['zero']}")
    print("-" * 30)

#validatioon
def main():
    numbers = []
    print("🔢 Odd or Even Number Analyzer 🔢")
    print("-" * 30)
    print("Enter as many numbers as you want. Type 'done' to stop.\n")
```

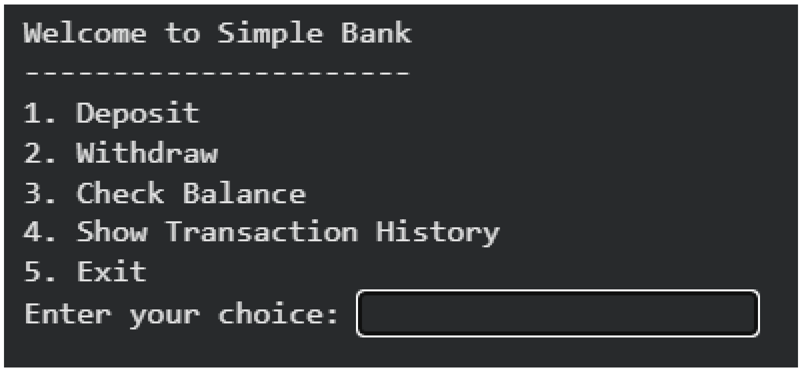
```python
    while True:
        user_input = input("Enter a number: ").strip().lower()

        if user_input == "done":
            break
        elif user_input == "":
            print("Empty input detected. Please enter a number.")
            continue

        try:
            numbers.append(int(user_input))
        except ValueError:
            print("Invalid input! Please enter a valid integer.")

    if not numbers:
        print("\nNo numbers were entered. Program Stopped.")
        return

    result = analyze_numbers(numbers)
    display_summary(result, len(numbers))


# Program entry point
if __name__ == "__main__":
    main()
```

Machine Problem 2

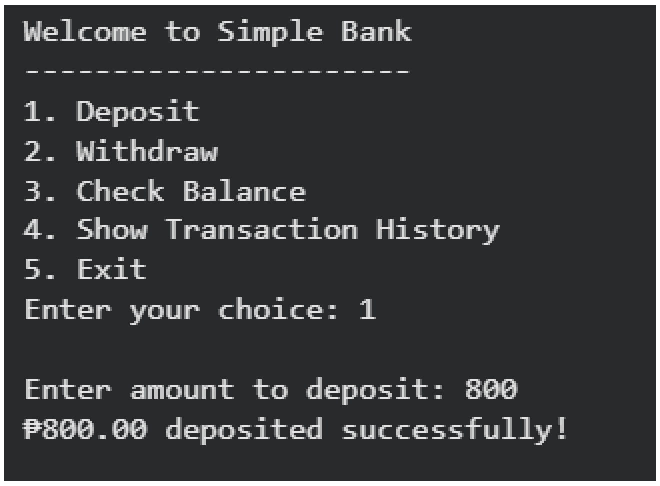
Simple Banking System

A simple banking system that allows users to check their balance, make deposits, withdraw money, and monitor transaction history. It records transactions using a list of dictionaries, handles each operation using functions, and stores the balance in variables. To ensure consistent and clear use, the system operates in a loop until the user decides to stop using it.

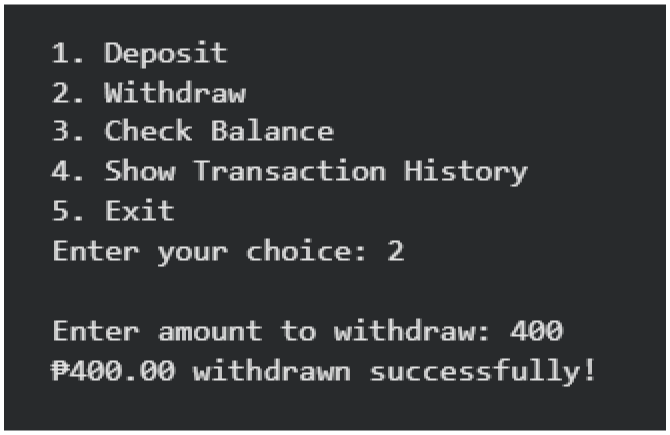
Screenshots Output:



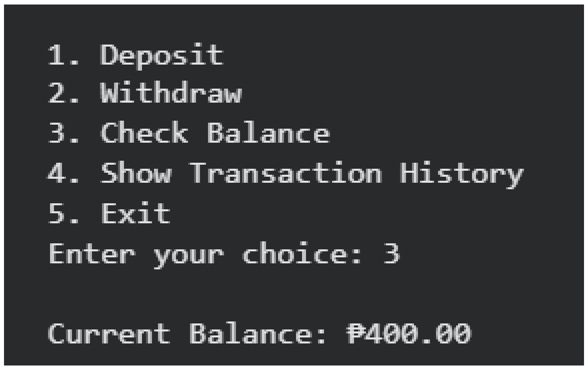
```
Welcome to Simple Bank
----------------------
1. Deposit
2. Withdraw
3. Check Balance
4. Show Transaction History
5. Exit
Enter your choice:
```



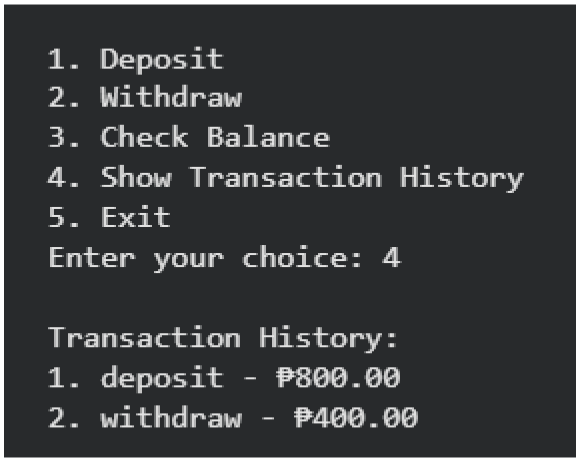
```
Welcome to Simple Bank
----------------------
1. Deposit
2. Withdraw
3. Check Balance
4. Show Transaction History
5. Exit
Enter your choice: 1

Enter amount to deposit: 800
₱800.00 deposited successfully!
```



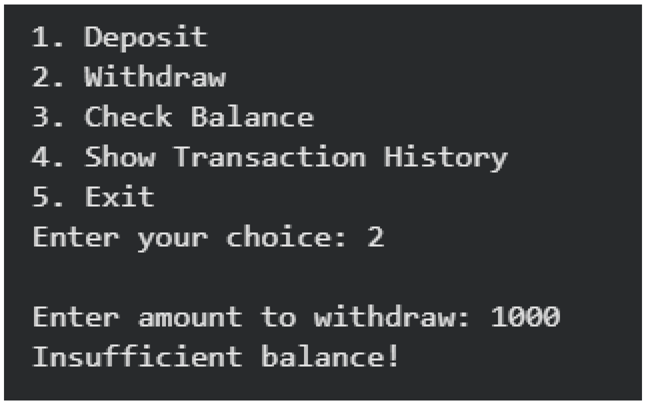
```
1. Deposit
2. Withdraw
3. Check Balance
4. Show Transaction History
5. Exit
Enter your choice: 2

Enter amount to withdraw: 400
₱400.00 withdrawn successfully!
```



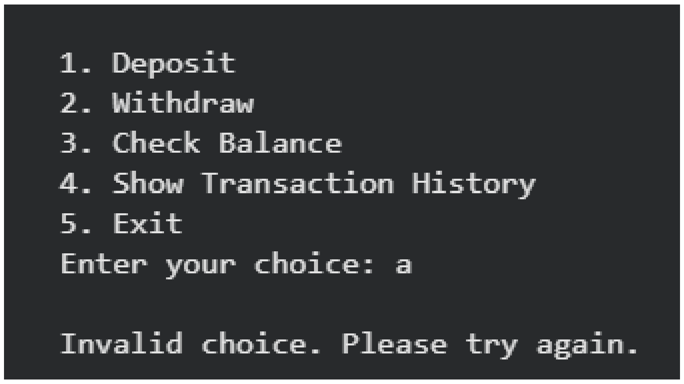
```
1. Deposit
2. Withdraw
3. Check Balance
4. Show Transaction History
5. Exit
Enter your choice: 3

Current Balance: ₱400.00
```



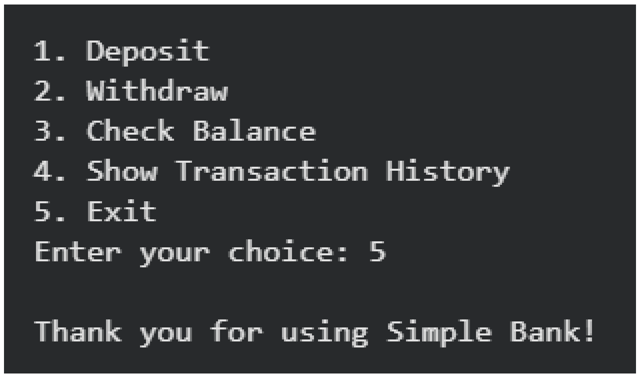
```
1. Deposit
2. Withdraw
3. Check Balance
4. Show Transaction History
5. Exit
Enter your choice: 4

Transaction History:
1. deposit - ₱800.00
2. withdraw - ₱400.00
```



```
1. Deposit
2. Withdraw
3. Check Balance
4. Show Transaction History
5. Exit
Enter your choice: 2

Enter amount to withdraw: 1000
Insufficient balance!
```



```

  1. Deposit
  2. Withdraw
  3. Check Balance
  4. Show Transaction History
  5. Exit
  Enter your choice: a

  Invalid choice. Please try again.
```



```
1. Deposit
2. Withdraw
3. Check Balance
4. Show Transaction History
5. Exit
Enter your choice: 5

Thank you for using Simple Bank!
```

Code:

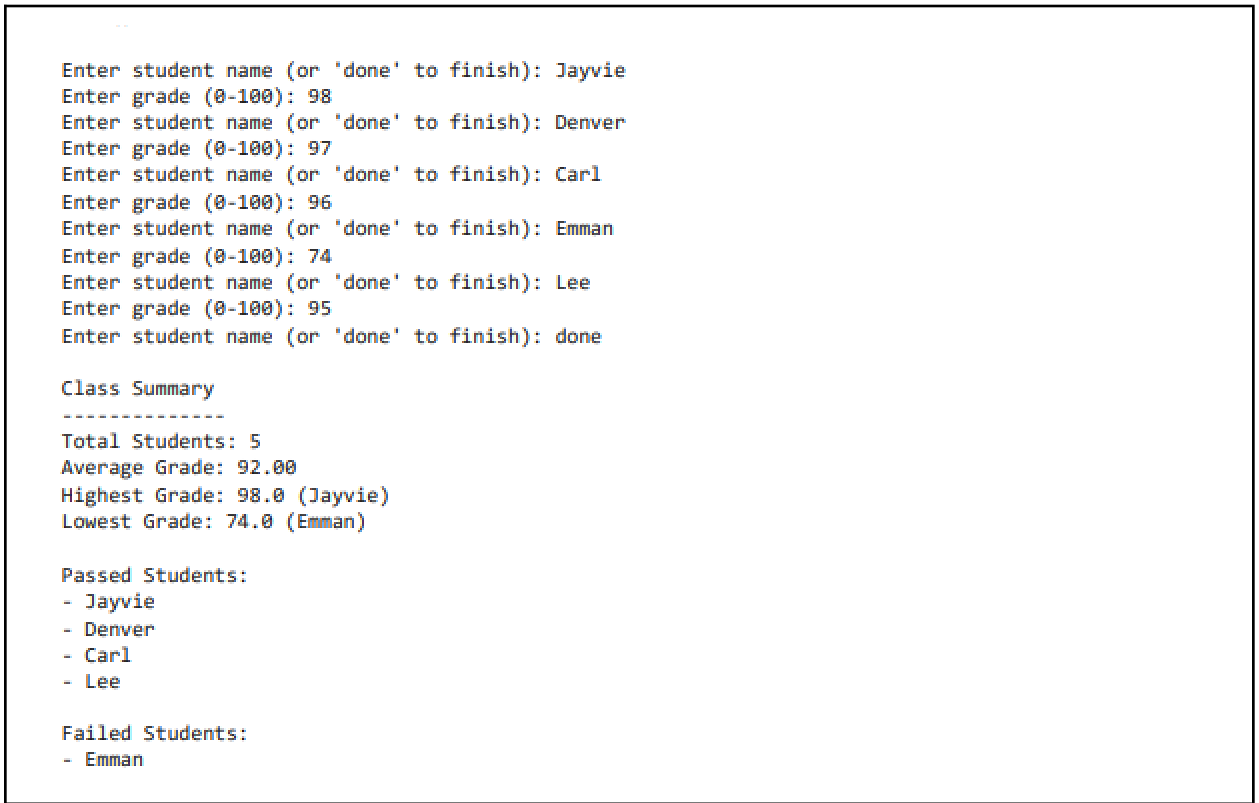
```python
def deposit(balance, transactions):
    amount = float(input("Enter amount to deposit: "))
    if amount > 0:
        balance += amount
        transactions.append({"type": "deposit", "amount": amount})
        print(f"₱{amount:.2f} deposited successfully!\n")
    else:
        print("Invalid amount. Deposit must be greater than 0.\n")
    return balance

def withdraw(balance, transactions):
    amount = float(input("Enter amount to withdraw: "))
    if amount <= 0:
        print("Invalid amount. Withdrawal must be greater than 0.\n")
    elif amount > balance:
        print("Insufficient balance!\n")
    else:
        balance -= amount
        transactions.append({"type": "withdraw", "amount": amount})
        print(f"₱{amount:.2f} withdrawn successfully!\n")
    return balance

def show_balance(balance):
    print(f"Current Balance: ₱{balance:.2f}\n")

def show_history(transactions):
    if not transactions:
        print("No transaction history.\n")
    else:
        print("Transaction History:")
        for i, t in enumerate(transactions, start=1):
            print(f"{i}. {t['type']} - ₱{t['amount']:.2f}")
        print()

def main():
    balance = 0.0
    transactions = []

    print("Welcome to Simple Bank")
    print("----------------------")

    while True:
        print("1. Deposit")
        print("2. Withdraw")
        print("3. Check Balance")
        print("4. Show Transaction History")
        print("5. Exit")

        choice = input("Enter your choice: ")
        print()

        if choice == "1":
            balance = deposit(balance, transactions)
        elif choice == "2":
            balance = withdraw(balance, transactions)
        elif choice == "3":
            show_balance(balance)
        elif choice == "4":
            show_history(transactions)
        elif choice == "5":
            print("Thank you for using Simple Bank!")
            break
        else:
            print("Invalid choice. Please try again.\n")

main()
```

Machine Problem 3

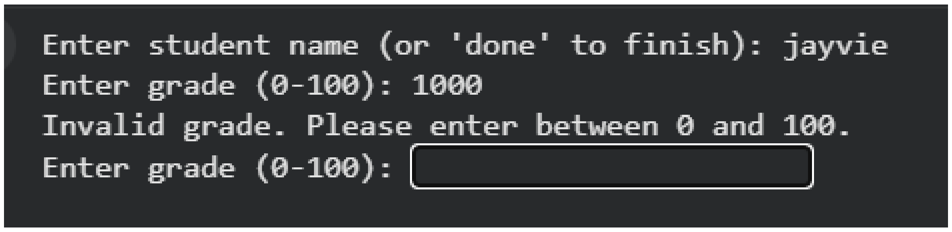
Student Performance Record System

This Python program records and analyzes student grades. It allows the user to input multiple student names and grades, validating that each grade is between 0 and 100. The program automatically determines whether each student passed or failed based on a passing grade of 75. After all entries are made, it calculates the class average, identifies the highest and lowest grades, and displays a summary report. Finally, it lists all students who passed and failed for a clear overview of class performance.

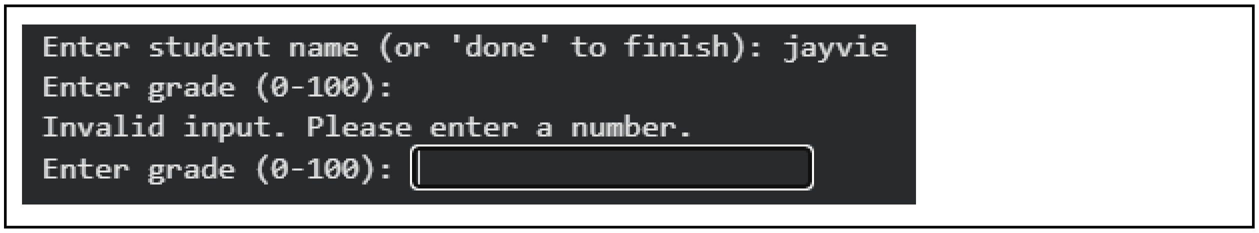
Screenshots Output:



```
Enter student name (or 'done' to finish): Jayvie
Enter grade (0-100): 98
Enter student name (or 'done' to finish): Denver
Enter grade (0-100): 97
Enter student name (or 'done' to finish): Carl
Enter grade (0-100): 96
Enter student name (or 'done' to finish): Emman
Enter grade (0-100): 74
Enter student name (or 'done' to finish): Lee
Enter grade (0-100): 95
Enter student name (or 'done' to finish): done

Class Summary
--------------
Total Students: 5
Average Grade: 92.00
Highest Grade: 98.0 (Jayvie)
Lowest Grade: 74.0 (Emman)

Passed Students:
- Jayvie
- Denver
- Carl
- Lee

Failed Students:
- Emman
```



```
Enter student name (or 'done' to finish): jayvie
Enter grade (0-100): 1000
Invalid grade. Please enter between 0 and 100.
Enter grade (0-100):
```



```
Enter student name (or 'done' to finish): jayvie
Enter grade (0-100):
Invalid input. Please enter a number.
Enter grade (0-100):
```

Code:

```python
# Student Grades Recording and Analysis

def add_student():
    name = input("Enter student name (or 'done' to finish): ")
    if name.lower() == "done":
        return None

    # input validation for grade
    while True:
        try:
            grade = float(input("Enter grade (0-100): "))
            if grade < 0 or grade > 100:
                print("Invalid grade. Please enter between 0 and 100.")
            else:
                break
        except ValueError:
            print("Invalid input. Please enter a number.")

    if grade >= 75:
        remark = "Passed"
    else:
        remark = "Failed"

    student = {"name": name, "grade": grade, "remark": remark}
    return student


def analyze_grades(students):
    total = 0
    for s in students:
        total += s["grade"]
    average = total / len(students)
    highest = max(students, key=lambda s: s["grade"])
    lowest = min(students, key=lambda s: s["grade"])

    return average, highest, lowest


def show_summary(students, average, highest, lowest):
    print("\nClass Summary")
    print("--------------")
    print("Total Students:", len(students))
    print(f"Average Grade: {average:.2f}")
    print(f"Highest Grade: {highest['grade']} ({highest['name']})")
    print(f"Lowest Grade: {lowest['grade']} ({lowest['name']})")
    print()

    passed = []
    failed = []
    for s in students:
        if s["remark"] == "Passed":
            passed.append(s["name"])
        else:
            failed.append(s["name"])

    print("Passed Students:")
    if passed:
        for p in passed:
            print("-", p)
    else:
        print("None")

    print("\nFailed Students:")
    if failed:
        for f in failed:
            print("-", f)
    else:
        print("None")
```

```python
def main():
    students = []

    while True:
        student = add_student()
        if student is None:
            break
        students.append(student)

    if len(students) > 0:
        average, highest, lowest = analyze_grades(students)
        show_summary(students, average, highest, lowest)
    else:
        print("No students entered.")


main()
```

Involvement:
Carl Justin O. Bautista: 10 - MP1, Documentation
Donn Denver Borja: 10 - MP2
Jayvie Abueg: 10 - MP3