Krishnan_Siranjevi_21274501_ISE_Report

Numerology Application Report

Table of Contents

Overview

1. Cover Page
2. Introduction
3. Module Descriptions
4. Modularity
5. Black Box Testing
6. White Box Testing
7. Test Implementation and Test Execution
8. Summary
9. Version Control
10. Discussion

Introduction

This report is based on the Numerology Application, a software tool that performs various numerology calculations and operations based on a person's date of birth. The application has two main scenarios:

1. **Scenario A**: Calculating the Life Path Number (LPN) from a given date of birth.
2. **Scenario B**: Determining the generation a person belongs to based on their date of birth.

The report covers module descriptions, modularity principles, black box and white box testing, test implementation, and test execution for both scenarios.

Module Descriptions

Scenario A: Life Path Number Calculation

Submodule: lpn(dob) Imports: datetime Exports: num (int)

This module takes a date of birth (dob) as input and calculates the Life Path Number (LPN) using the following steps: 1. Convert the date of birth to a string in the format "YYYYMMDD". 2. Sum the individual digits of the string. 3. If the sum is greater than 9 and not a master number (11, 22, or 33), continue summing the digits until a single-digit number or a master number is obtained. 4. Return the final number as the LPN.

Submodule: is_master(n) Imports: n (int) Exports: Boolean

This module checks if a given number `n` is a master number (11, 22, or 33) and returns `True` if it is, and `False` otherwise.

Submodule: lucky_color(n) Imports: n (int) Exports: color (str)

This module takes a Life Path Number `n` as input and returns the corresponding lucky color based on the following mapping:

- 1: Red
- 2: Orange
- 3: Yellow
- 4: Green
- 5: Blue
- 6: Indigo
- 7: Violet
- 8: Purple
- 9: Gold
- 11: Silver
- 22: Platinum
- 33: Diamond

If the input number is not in the mapping, it returns "Unknown".

Scenario B: Determine Generation

Submodule: get_gen(dob) Imports: datetime Exports: gen (str)

This module takes a date of birth (dob) as input and determines the generation the person belongs to based on the following mapping:

- 1901 - 1927: Greatest Generation
- 1928 - 1945: Silent Generation
- 1946 - 1964: Baby Boomers
- 1965 - 1980: Generation X
- 1981 - 1996: Millennials
- 1997 - 2012: Generation Z
- 2013 - 2024: Generation Alpha

If the date of birth falls outside of the defined ranges, it returns "Unknown Generation".

Modularity

Description of Module Usage

Scenario A: Life Path Number Calculation

1. `lpn(dob)` is used to calculate the Life Path Number from the given date of birth.
2. `is_master(n)` is used to check if the calculated Life Path Number is a master number (11, 22, or 33).
3. `lucky_color(n)` is used to determine the lucky color associated with the calculated Life Path Number.

Scenario B: Determine Generation

1. `get_gen(dob)` is used to determine the generation a person belongs to based on their date of birth.

Functioning of the Program

To run the program, execute the following command:

```
git clone https://github.com/SiranjeviKrishnan/Krishnan_Siranjevi_21274501_ISErepo.git
cd Krishnan_Siranjevi_21274501_ISErepo
python3 main.py
```

The program will present a menu with options to perform various numerology calculations and operations. Follow the prompts to enter the required inputs and view the results.

Cohesion

The program follows the principles of cohesion by grouping related functionalities into separate modules. For example, the `lpn` module contains functions related to calculating the Life Path Number, while the `get_gen` module handles the determination of the generation based on the date of birth.

Redundancy

The program minimizes redundancy by using dictionaries or tuples to store the mapping between Life Path Numbers and lucky colors, as well as the mapping between date ranges and generations.

Coupling

The program exhibits low coupling between modules, as each module has a specific responsibility and can be tested and modified independently.

Control Flags

Control flags are not used in this program. Instead, it relies on conditional statements and function returns to control the flow of execution.

Black Box Testing

Black box testing techniques, such as Equivalence Partitioning (EP) and Boundary Value Analysis (BVA), have been applied to test the program's functionality.

Scenario A: Life Path Number Calculation

1. `lpn(dob)` (Equivalence Partitioning)

Category	Test Data	Expected Result
Valid Date	1990-04-15	11
Valid Date	1985-12-30	11
Valid Date	2000-01-01	4
Valid Date	2011-11-11	8
Valid Date	2020-03-14	3
Valid Date	1988-08-08	6
Invalid Date	2024-02-30	Error
Invalid Date	1900-13-01	Error

2. `is_master(n)` (Equivalence Partitioning)

Category	Test Data	Expected Result
Master Number	11	True
Master Number	22	True
Master Number	33	True
Non-Master Number	4	False
Non-Master Number	9	False

3. `lucky_color(n)` (Equivalence Partitioning)

Category	Test Data	Expected Result
Valid LPN	1	Red
Valid LPN	2	Orange
Valid LPN	3	Yellow
Valid LPN	4	Green
Valid LPN	5	Blue
Valid LPN	6	Indigo
Valid LPN	7	Violet
Valid LPN	8	Purple
Valid LPN	9	Gold
Master Number	11	Silver
Master Number	22	Platinum
Master Number	33	Diamond
Invalid LPN	99	Unknown

Scenario B: Determine Generation

1. `get_gen(dob)` (Equivalence Partitioning)

Category	Test Data	Expected Result
Greatest Generation	1925-06-15	Greatest Generation
Silent Generation	1930-06-15	Silent Generation
Baby Boomers	1950-06-15	Baby Boomers
Generation X	1975-06-15	Generation X
Millennials	1990-06-15	Millennials
Generation Z	2000-06-15	Generation Z
Generation Alpha	2020-06-15	Generation Alpha
Unknown Generation (Before Range)	1900-06-15	Unknown Generation
Unknown Generation (After Range)	2025-06-15	Unknown Generation
Invalid Date	2024-02-30	Error

White Box Testing

lpn Module Test Cases

Path	Test Data	Expected Results
Pass into the def function and calculate LPN for a valid date	1990-04-15	11
Pass into the def function and calculate LPN for a valid date	2000-01-01	4
Pass into the def function and calculate LPN for an invalid date	2024-02-30	Error
Pass into the def function and calculate LPN for an invalid date format	“01/01/2000”	Error

get_gen Module Test Cases

Path	Test Data	Expected Results
Pass into the def function and determine generation for a valid date within the Greatest Generation range	1925-06-15	Greatest Generation
Pass into the def function and determine generation for a valid date within the Silent Generation range	1930-06-15	Silent Generation
Pass into the def function and determine generation for a valid date within the Baby Boomers range	1950-06-15	Baby Boomers

Path	Test Data	Expected Results
Pass into the def function and determine generation for a valid date within the Generation X range	1975-06-15	Generation X
Pass into the def function and determine generation for a valid date within the Millennials range	1990-06-15	Millennials
Pass into the def function and determine generation for a valid date within the Generation Z range	2000-06-15	Generation Z
Pass into the def function and determine generation for a valid date within the Generation Alpha range	2020-06-15	Generation Alpha

Path	Test Data	Expected Results
Pass into the def function and determine generation for a date before the defined ranges	1900-06-15	Unknown Generation
Pass into the def function and determine generation for a date after the defined ranges	2025-06-15	Unknown Generation
Pass into the def function and determine generation for an invalid date	2024-02-30	Error

Test Implementation and Test Execution

The project includes a test suite written using the `unittest` framework. To run the tests, execute the following command in the project directory:

```
python3 test.py
```

This will run all the test cases defined in the test suite.

Test Execution

1. Type `python main.py` to run the Numerology Application.
2. Type `python test.py` to run the test suite.

Discussion

This project provided a great opportunity to understand the basics of software development and witness the real-time use of various concepts and principles. It served as a valuable learning experience in applying modular design, testing methodologies, and version control practices to a practical application. While the current implementation fulfills the project requirements, there is always room for improvement and future enhancements. One area for potential development would be to incorporate a user-friendly graphical user interface (GUI) with improved user experience (UX) and interactive features.