CHANDIGARH UNIVERSITY
Discover. Learn. Empower.
NAAC GRADE A+
Accredited University

Subject Code: 21CAP677

Experiment No. 1

Student Name: Akhand Pratap Singh **UID: 21MCI1068**
Section/Group: 21MAM-2_A **Semester: 2nd**
Date of Submission: 17/02/2022 **Course: MCA (AI/ML)**

Aim/Overview of the practical: Program flow control in Python

Question Write a program to create Loan Payment Schedule

Payment Calculation

$$payment = \frac{interest * loan\ value}{1 - (1 + interest)^{-number\ of\ payments}}$$

- Interest is the rate you pay towards the loan, here it will be used as a decimal
- (so for example: 5% is .05).
- The number of payments will be determined by how many years the loan is for multiplied by 12 (one payment per month).
- If your loan is for less than a year, you can use a fraction (i.e. 9 months is .75)
- Interest can be calculated by multiplying the interest rate and the current loan value
- Each month this number will be different, so we include it in the loop.
- Principle is the actual amount of the payment that goes towards the loan balance and is found by subtracting the calculated interest from the payment
- Our loop condition states that we are going to loop as long as the value of the loan is greater than 0.
- The easiest check here is to see whether the next payment will cause the loan to be less than 0, and if so, then the principle payment of this month will be equal to the loan value.
- Finally, the loan value is updated by subtracting the current months principle payment.
- Compute the loan payment schedule over the lifetime of the loan using BeautifulTable
- `pip install beautifultable` (https://pypi.org/project/beautifultable/)

Solution:

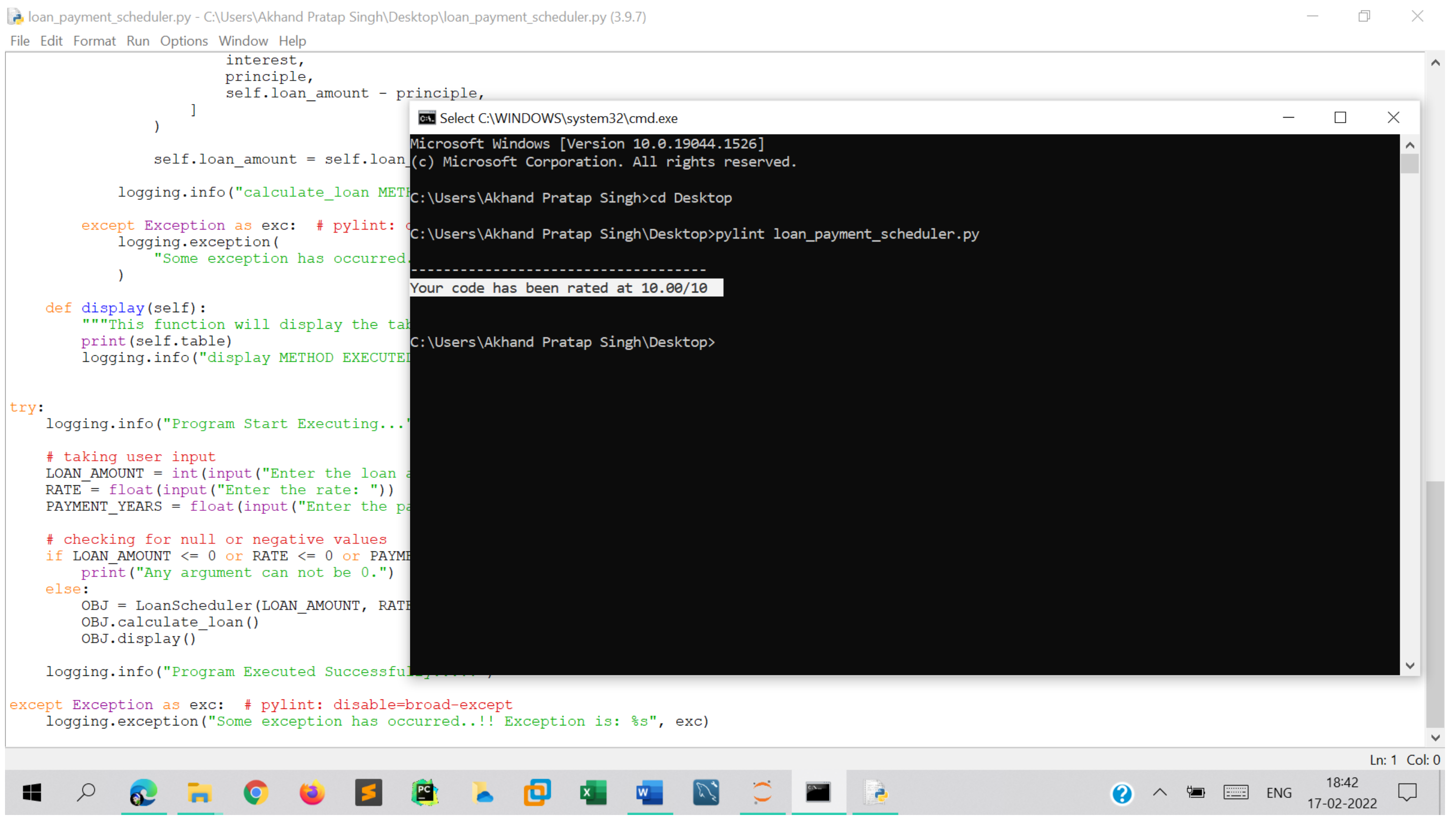
1. **Code for experiment/practical:**

```
"""Importing logging and beautifultable package"""
import logging
from beautifultable import BeautifulTable

# creating log file loans.log and storing logs into it
logging.basicConfig(
    filename="loans.log",
    level=logging.DEBUG,
    format="%(asctime)s %(levelname)s %(message)s",
)


class LoanScheduler:
    """
    This class consist of calculate_loan method which will
    schedule the loan payments according to year
    and customer will have to pay that number of
    payments
    """

    def __init__(self, loan_amount, rate, payment_years):
        """
        This will initialize the class variables and
        create the table schema.
        """
        logging.info("START EXECUTING __init__ METHOD")

        # intializing class variables
        self.loan_amount = loan_amount
        self.rate = rate
        self.payment_years = payment_years

        # createing table and schema of table with column names
        self.table = BeautifulTable()
        self.table.columns.header = [
            "MONTH",
            "LOAN AMOUNT",
            "PAYMENT",
            "INTEREST",
            "PRINCIPLE",
            "NEW LOAN AMOUNT",
        ]

        logging.info("__init__ METHOD EXECUTED")

    def calculate_loan(self):
        """
        This function will schedule the loan amount in number of payments customers have
to pay.
        """
        try:
            logging.info("START EXECUTING calculate_loan METHOD")

            # calculating interest rate, total number of payments
            # and calculating payment amount
            sr_no = 0
            interest_rate = self.rate / 12.0
            total_payments = self.payment_years * 12
            payment = (interest_rate * self.loan_amount) / (
                1 - ((1 + interest_rate) ** (-total_payments))
            )

            # calculating princile, interest and amount payable in each month
            # appending each month record into table
            while self.loan_amount > 0:
                sr_no += 1
                interest = self.loan_amount * interest_rate
                principle = payment - interest

                if self.loan_amount - payment < 0:
                    principle = self.loan_amount

                self.table.rows.append(
                    [
                        sr_no,
                        self.loan_amount,
                        payment,
                        interest,
                        principle,
                        self.loan_amount - principle,
                    ]
                )

                self.loan_amount = self.loan_amount - principle

            logging.info("calculate_loan METHOD EXECUTED")

        except Exception as exc:  # pylint: disable=broad-except
            logging.exception(
                "Some exception has occurred..!! Exception is: %s", exc
            )

    def display(self):
        """This function will display the table with total payments information"""
        print(self.table)
        logging.info("display METHOD EXECUTED")


try:
    logging.info("Program Start Executing...")

    # taking user input
    LOAN_AMOUNT = int(input("Enter the loan amount: "))
    RATE = float(input("Enter the rate: "))
    PAYMENT_YEARS = float(input("Enter the payment years: "))

    # checking for null or negative values
    if LOAN_AMOUNT <= 0 or RATE <= 0 or PAYMENT_YEARS <= 0:
        print("Any argument can not be 0.")
    else:
        OBJ = LoanScheduler(LOAN_AMOUNT, RATE, PAYMENT_YEARS)
        OBJ.calculate_loan()
        OBJ.display()

    logging.info("Program Executed Successfully...!!")

except Exception as exc:  # pylint: disable=broad-except
    logging.exception("Some exception has occurred..!! Exception is: %s", exc)
```

2. Code analyser and formatter tool(s) used: Name the tools with screenshots and score of the code

Pylint Code Analyser:-

Score of Pylint Software is: 10 out of 10



loan_payment_scheduler.py - C:\Users\Akhand Pratap Singh\Desktop\loan_payment_scheduler.py (3.9.7)

```
                        interest,
                        principle,
                        self.loan_amount - principle,
                    ]
                )

                self.loan_amount = self.loan

            logging.info("calculate_loan METH

        except Exception as exc:  # pylint: 
            logging.exception(
                "Some exception has occurred
            )

    def display(self):
        """This function will display the tab
        print(self.table)
        logging.info("display METHOD EXECUTED


try:
    logging.info("Program Start Executing...

    # taking user input
    LOAN_AMOUNT = int(input("Enter the loan 
    RATE = float(input("Enter the rate: "))
    PAYMENT_YEARS = float(input("Enter the pa

    # checking for null or negative values
    if LOAN_AMOUNT <= 0 or RATE <= 0 or PAYME
        print("Any argument can not be 0.")
    else:
        OBJ = LoanScheduler(LOAN_AMOUNT, RATE
        OBJ.calculate_loan()
        OBJ.display()

    logging.info("Program Executed Successful

except Exception as exc:  # pylint: disable=broad-except
    logging.exception("Some exception has occurred..!! Exception is: %s", exc)
```

Select C:\WINDOWS\system32\cmd.exe

```
Microsoft Windows [Version 10.0.19044.1526]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Akhand Pratap Singh>cd Desktop

C:\Users\Akhand Pratap Singh\Desktop>pylint loan_payment_scheduler.py

-----------------------------------
Your code has been rated at 10.00/10


C:\Users\Akhand Pratap Singh\Desktop>
```

3. PEP-8 Rules used to write the code:

(i) Docstrings :- Modules and Packages docstring, class docstring, function docstring

(ii) Using Logging and Exception Handling

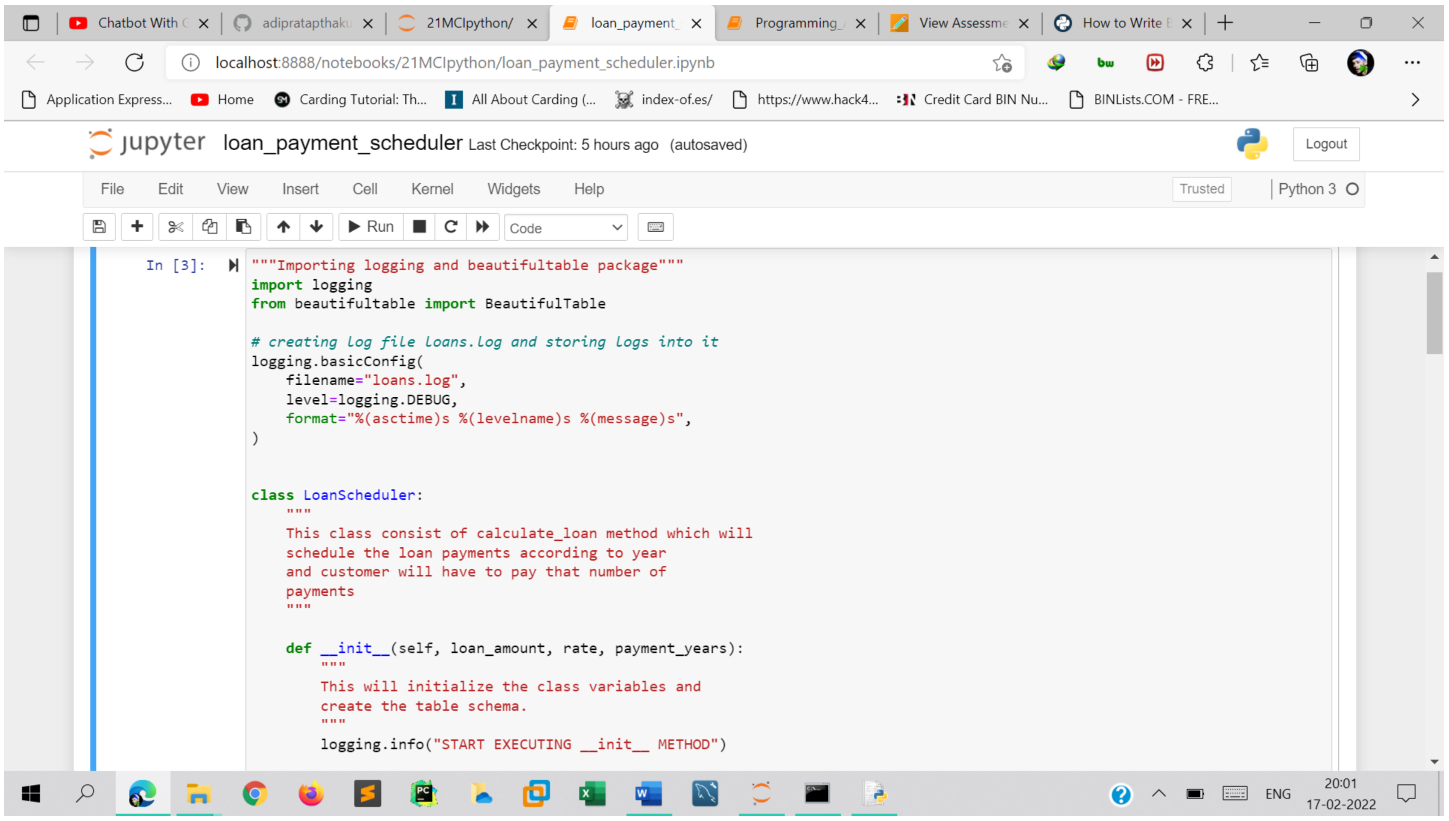
(iii) Using naming convention of variables, methods, classes and constants.

- Variables: Use a lowercase single letter, word, or words. Separate words with underscores.
- Methods: Use a lowercase word or words. Separate words with underscores.
- Classes: Start each word with a capital letter. Do not separate words with underscores. This style is called camel case.
- Constants & Objects: Use an uppercase single letter, word, or words. Separate words with underscores.

(iv) Using line continuation, So that lines should be limited to 79 characters.

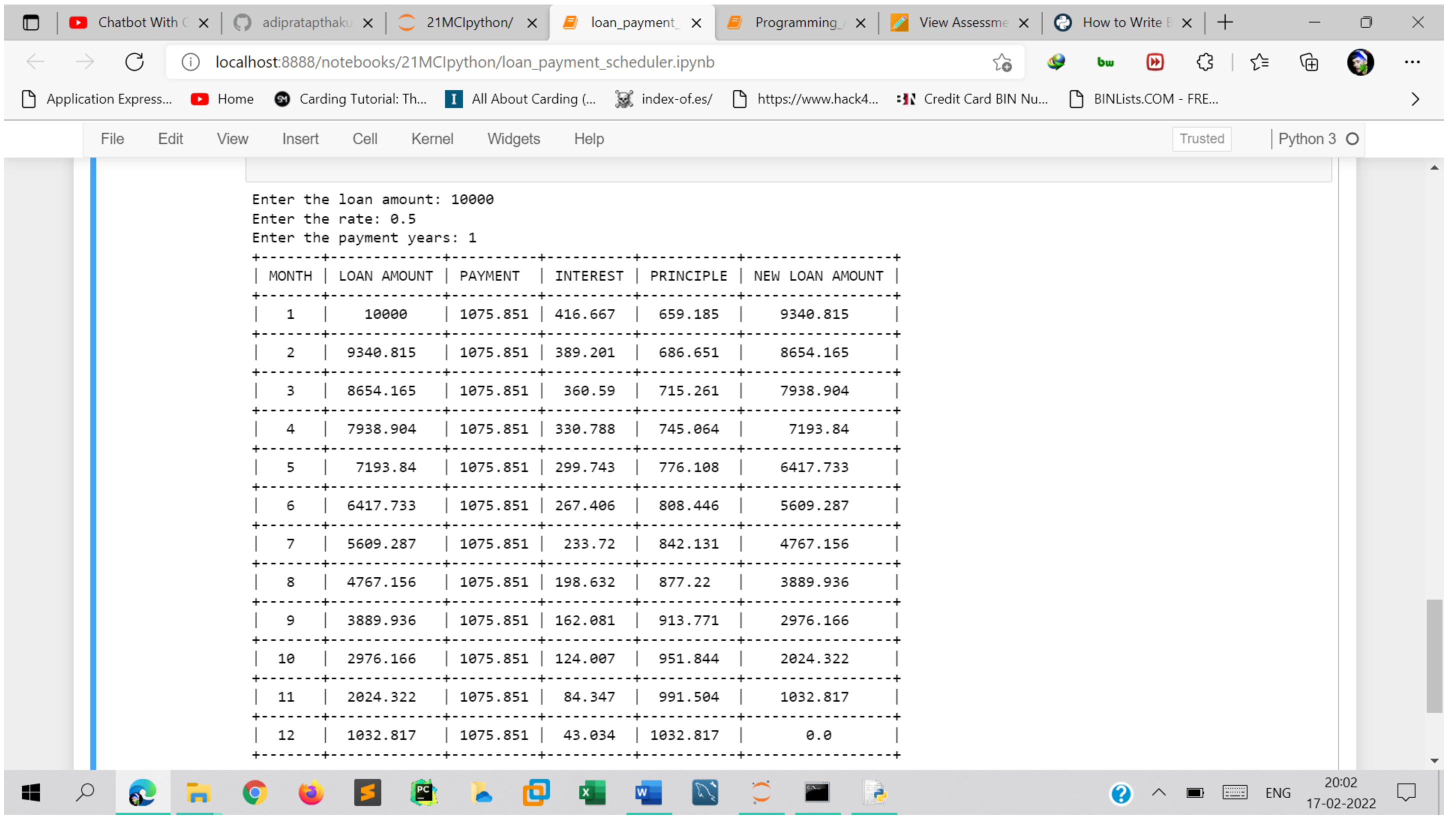
4. Result/Output:

Code Screenshot:-



```
In [3]:  """Importing logging and beautifultable package"""
         import logging
         from beautifultable import BeautifulTable

         # creating log file loans.log and storing logs into it
         logging.basicConfig(
             filename="loans.log",
             level=logging.DEBUG,
             format="%(asctime)s %(levelname)s %(message)s",
         )


         class LoanScheduler:
             """
             This class consist of calculate_loan method which will
             schedule the loan payments according to year
             and customer will have to pay that number of
             payments
             """

             def __init__(self, loan_amount, rate, payment_years):
                 """
                 This will initialize the class variables and
                 create the table schema.
                 """
                 logging.info("START EXECUTING __init__ METHOD")
```

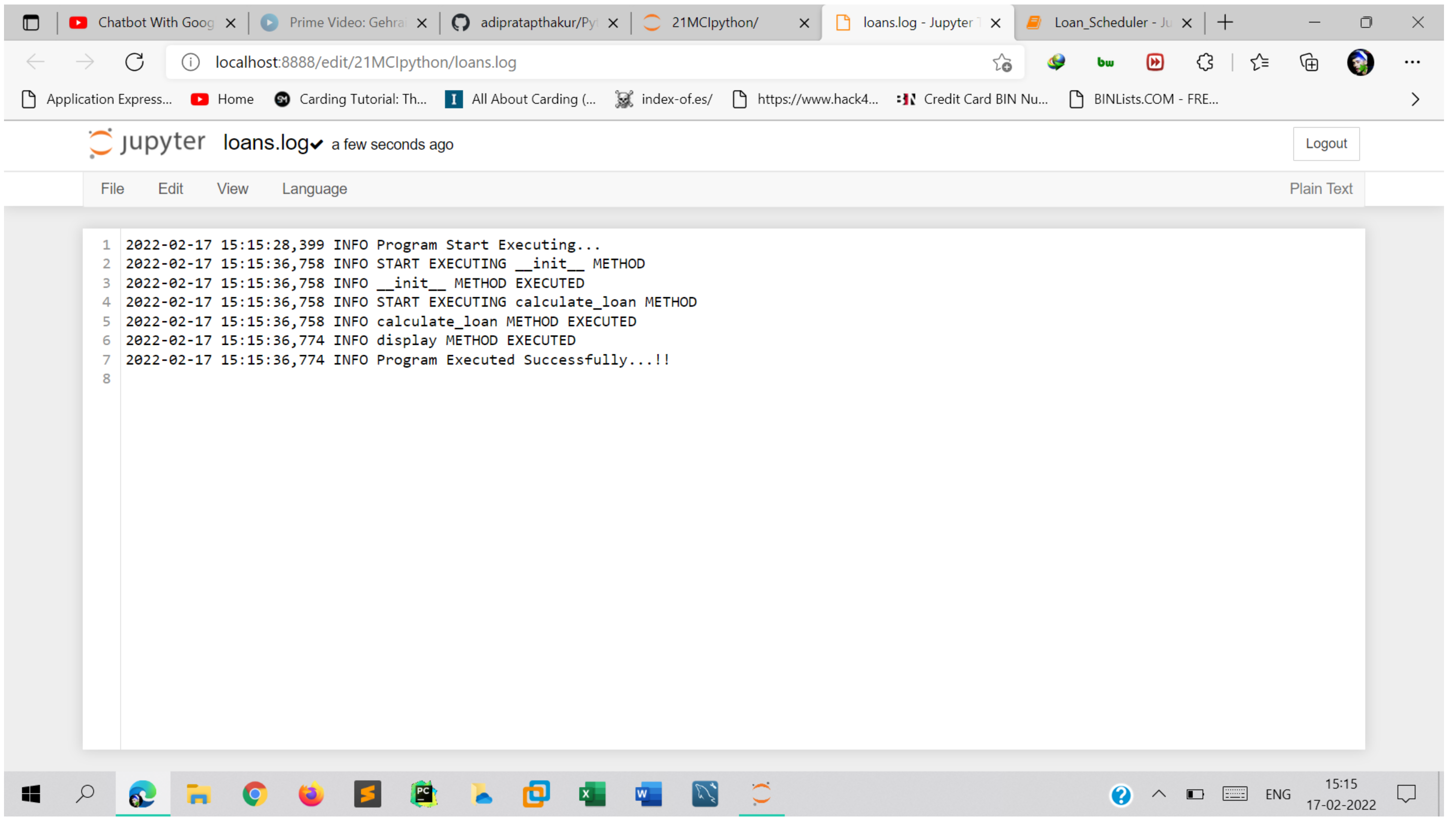
Program Output :-



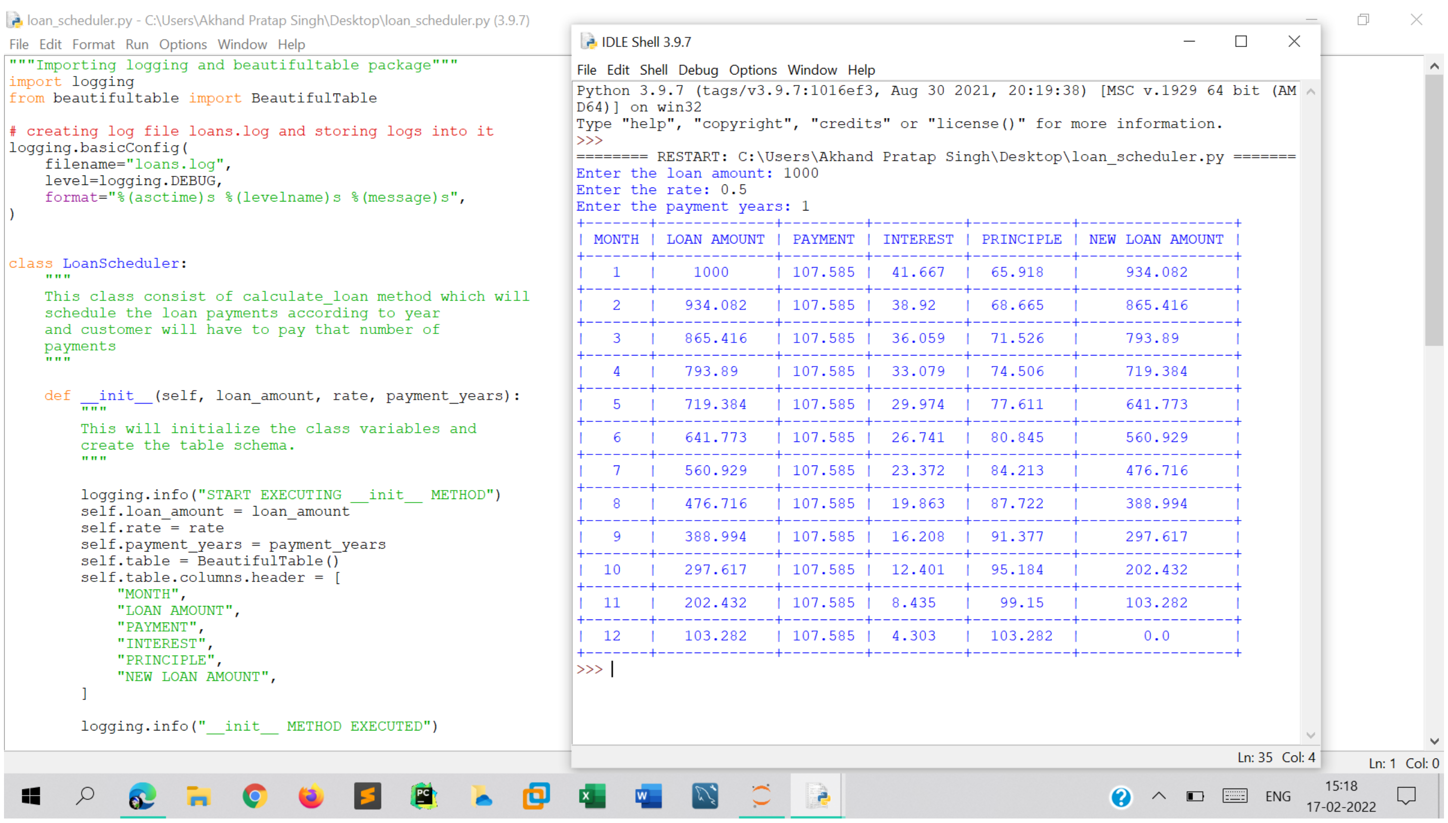
```
Enter the loan amount: 10000
Enter the rate: 0.5
Enter the payment years: 1
+-------+-------------+----------+----------+-----------+-----------------+
| MONTH | LOAN AMOUNT | PAYMENT  | INTEREST | PRINCIPLE | NEW LOAN AMOUNT |
+-------+-------------+----------+----------+-----------+-----------------+
|   1   |    10000    | 1075.851 | 416.667  |  659.185  |    9340.815     |
+-------+-------------+----------+----------+-----------+-----------------+
|   2   |  9340.815   | 1075.851 | 389.201  |  686.651  |    8654.165     |
+-------+-------------+----------+----------+-----------+-----------------+
|   3   |  8654.165   | 1075.851 |  360.59  |  715.261  |    7938.904     |
+-------+-------------+----------+----------+-----------+-----------------+
|   4   |  7938.904   | 1075.851 | 330.788  |  745.064  |     7193.84     |
+-------+-------------+----------+----------+-----------+-----------------+
|   5   |   7193.84   | 1075.851 | 299.743  |  776.108  |    6417.733     |
+-------+-------------+----------+----------+-----------+-----------------+
|   6   |  6417.733   | 1075.851 | 267.406  |  808.446  |    5609.287     |
+-------+-------------+----------+----------+-----------+-----------------+
|   7   |  5609.287   | 1075.851 |  233.72  |  842.131  |    4767.156     |
+-------+-------------+----------+----------+-----------+-----------------+
|   8   |  4767.156   | 1075.851 | 198.632  |  877.22   |    3889.936     |
+-------+-------------+----------+----------+-----------+-----------------+
|   9   |  3889.936   | 1075.851 | 162.081  |  913.771  |    2976.166     |
+-------+-------------+----------+----------+-----------+-----------------+
|  10   |  2976.166   | 1075.851 | 124.007  |  951.844  |    2024.322     |
+-------+-------------+----------+----------+-----------+-----------------+
|  11   |  2024.322   | 1075.851 |  84.347  |  991.504  |    1032.817     |
+-------+-------------+----------+----------+-----------+-----------------+
|  12   |  1032.817   | 1075.851 |  43.034  | 1032.817  |       0.0       |
+-------+-------------+----------+----------+-----------+-----------------+
```

CHANDIGARH UNIVERSITY Discover. Learn. Empower. NAAC GRADE A+ Accredited University

Log Files Screenshot:-


localhost:8888/edit/21MClpython/loans.log
jupyter loans.log a few seconds ago
Logout
File Edit View Language
Plain Text
15:15
17-02-2022

```
2022-02-17 15:15:28,399 INFO Program Start Executing...
2022-02-17 15:15:36,758 INFO START EXECUTING __init__ METHOD
2022-02-17 15:15:36,758 INFO __init__ METHOD EXECUTED
2022-02-17 15:15:36,758 INFO START EXECUTING calculate_loan METHOD
2022-02-17 15:15:36,758 INFO calculate_loan METHOD EXECUTED
2022-02-17 15:15:36,774 INFO display METHOD EXECUTED
2022-02-17 15:15:36,774 INFO Program Executed Successfully...!!
```


loan_scheduler.py - C:\Users\Akhand Pratap Singh\Desktop\loan_scheduler.py (3.9.7)
File Edit Format Run Options Window Help
IDLE Shell 3.9.7
File Edit Shell Debug Options Window Help
Ln: 35 Col: 4
Ln: 1 Col: 0
15:18
17-02-2022

```python
"""Importing logging and beautifultable package"""
import logging
from beautifultable import BeautifulTable

# creating log file loans.log and storing logs into it
logging.basicConfig(
    filename="loans.log",
    level=logging.DEBUG,
    format="%(asctime)s %(levelname)s %(message)s",
)


class LoanScheduler:
    """
    This class consist of calculate_loan method which will
    schedule the loan payments according to year
    and customer will have to pay that number of
    payments
    """

    def __init__(self, loan_amount, rate, payment_years):
        """
        This will initialize the class variables and
        create the table schema.
        """

        logging.info("START EXECUTING __init__ METHOD")
        self.loan_amount = loan_amount
        self.rate = rate
        self.payment_years = payment_years
        self.table = BeautifulTable()
        self.table.columns.header = [
            "MONTH",
            "LOAN AMOUNT",
            "PAYMENT",
            "INTEREST",
            "PRINCIPLE",
            "NEW LOAN AMOUNT",
        ]

        logging.info("__init__ METHOD EXECUTED")
```

```
Python 3.9.7 (tags/v3.9.7:1016ef3, Aug 30 2021, 20:19:38) [MSC v.1929 64 bit (AM
D64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
======== RESTART: C:\Users\Akhand Pratap Singh\Desktop\loan_scheduler.py =======
Enter the loan amount: 1000
Enter the rate: 0.5
Enter the payment years: 1
+-------+-------------+---------+----------+-----------+-----------------+
| MONTH | LOAN AMOUNT | PAYMENT | INTEREST | PRINCIPLE | NEW LOAN AMOUNT |
+-------+-------------+---------+----------+-----------+-----------------+
|   1   |    1000     | 107.585 |  41.667  |  65.918   |     934.082     |
+-------+-------------+---------+----------+-----------+-----------------+
|   2   |   934.082   | 107.585 |  38.92   |  68.665   |     865.416     |
+-------+-------------+---------+----------+-----------+-----------------+
|   3   |   865.416   | 107.585 |  36.059  |  71.526   |     793.89      |
+-------+-------------+---------+----------+-----------+-----------------+
|   4   |   793.89    | 107.585 |  33.079  |  74.506   |     719.384     |
+-------+-------------+---------+----------+-----------+-----------------+
|   5   |   719.384   | 107.585 |  29.974  |  77.611   |     641.773     |
+-------+-------------+---------+----------+-----------+-----------------+
|   6   |   641.773   | 107.585 |  26.741  |  80.845   |     560.929     |
+-------+-------------+---------+----------+-----------+-----------------+
|   7   |   560.929   | 107.585 |  23.372  |  84.213   |     476.716     |
+-------+-------------+---------+----------+-----------+-----------------+
|   8   |   476.716   | 107.585 |  19.863  |  87.722   |     388.994     |
+-------+-------------+---------+----------+-----------+-----------------+
|   9   |   388.994   | 107.585 |  16.208  |  91.377   |     297.617     |
+-------+-------------+---------+----------+-----------+-----------------+
|  10   |   297.617   | 107.585 |  12.401  |  95.184   |     202.432     |
+-------+-------------+---------+----------+-----------+-----------------+
|  11   |   202.432   | 107.585 |  8.435   |   99.15   |     103.282     |
+-------+-------------+---------+----------+-----------+-----------------+
|  12   |   103.282   | 107.585 |  4.303   |  103.282  |       0.0       |
+-------+-------------+---------+----------+-----------+-----------------+
>>>
```

Learning outcomes (What I have learnt):

1. I have learnt about the PEP-8 guidelines to write the code so that it increases readability and consistency of Python Code.

2. I have learnt about the code analyzer software to check the rating of our Python code.

3. I have learnt how to maintain logs of our programs and debug our code.

4. I have learnt about the exception handling.

5. I have learnt about the concept of basic control flow of python program and about the classes and objects.

Evaluation Grid:

Sr. No.	Parameters	Marks Obtained	Maximum Marks
1.	Demonstration and Performance (Pre Lab Quiz)		5
2.	Worksheet		10
3.	Post Lab Quiz		5