ASSIGNMENT-2

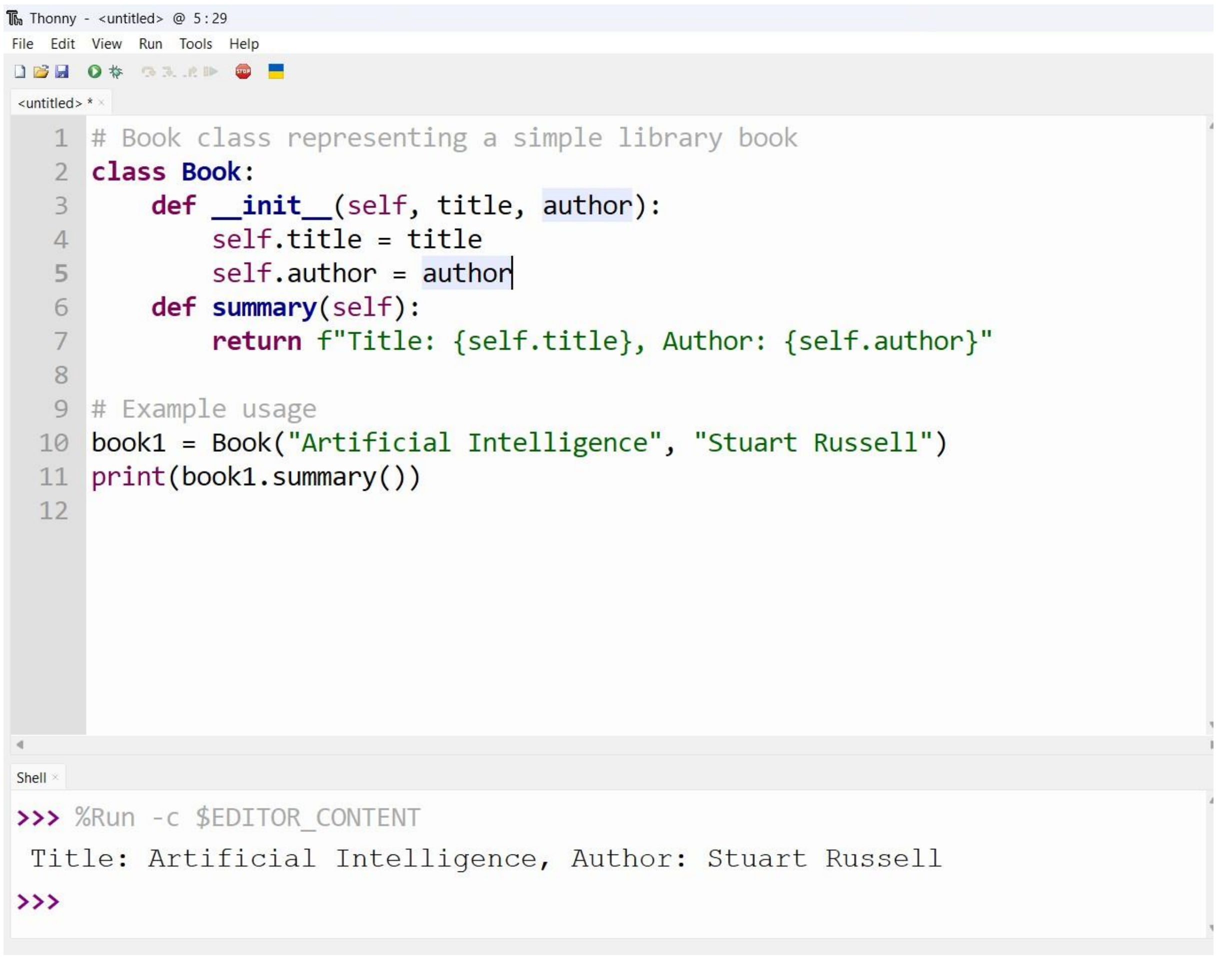
NAME:G.VARSHITHA

BATCH-16

HTNO:-2303A51053

Task1:BookClassGeneration(UsingCursorAI) Python

Code: Book Class

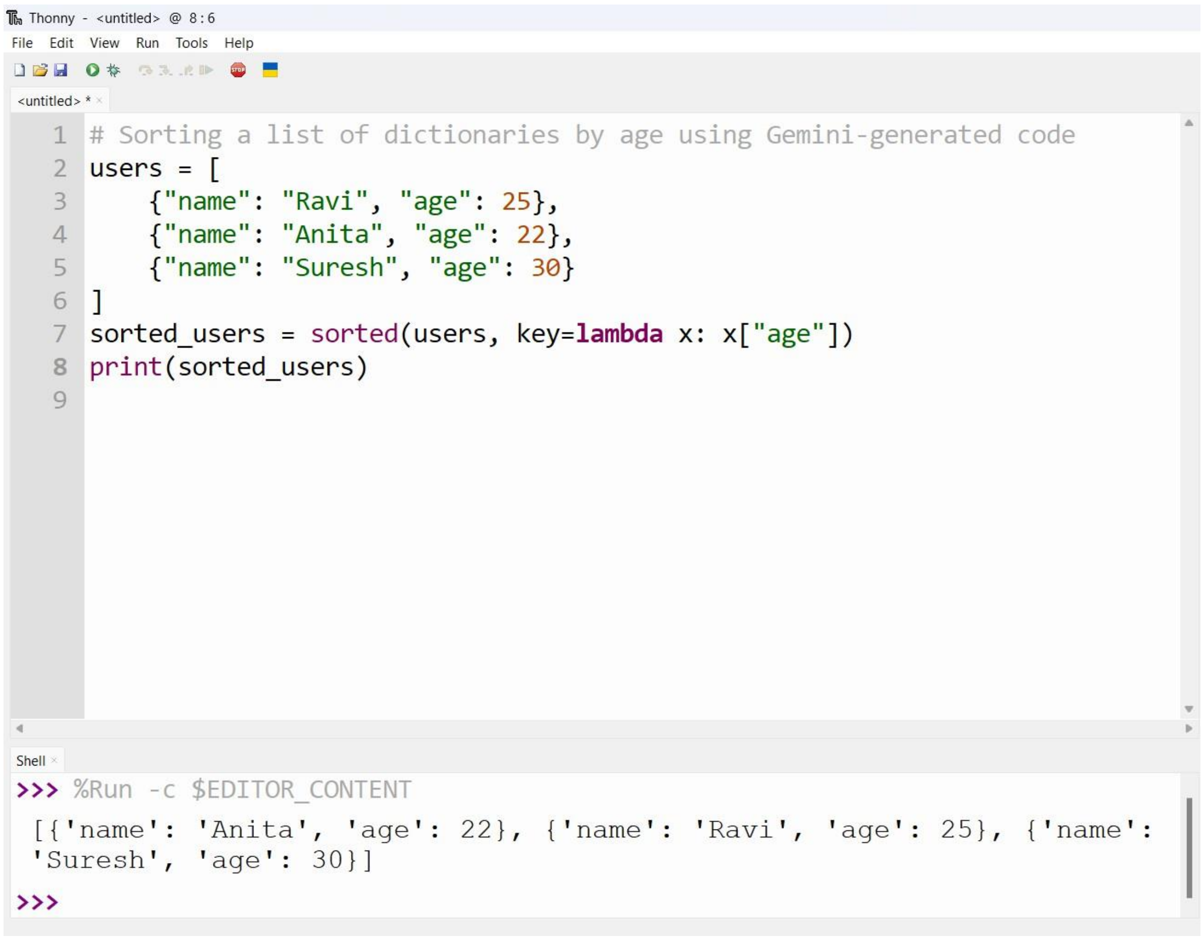


```
# Book class representing a simple library book
class Book:
    def __init__(self, title, author):
        self.title = title
        self.author = author
    def summary(self):
        return f"Title: {self.title}, Author: {self.author}"

# Example usage
book1 = Book("Artificial Intelligence", "Stuart Russell")
print(book1.summary())
```

```
>>> %Run -c $EDITOR_CONTENT
 Title: Artificial Intelligence, Author: Stuart Russell
>>>
```

Task2:SortingDictionarieswithAI

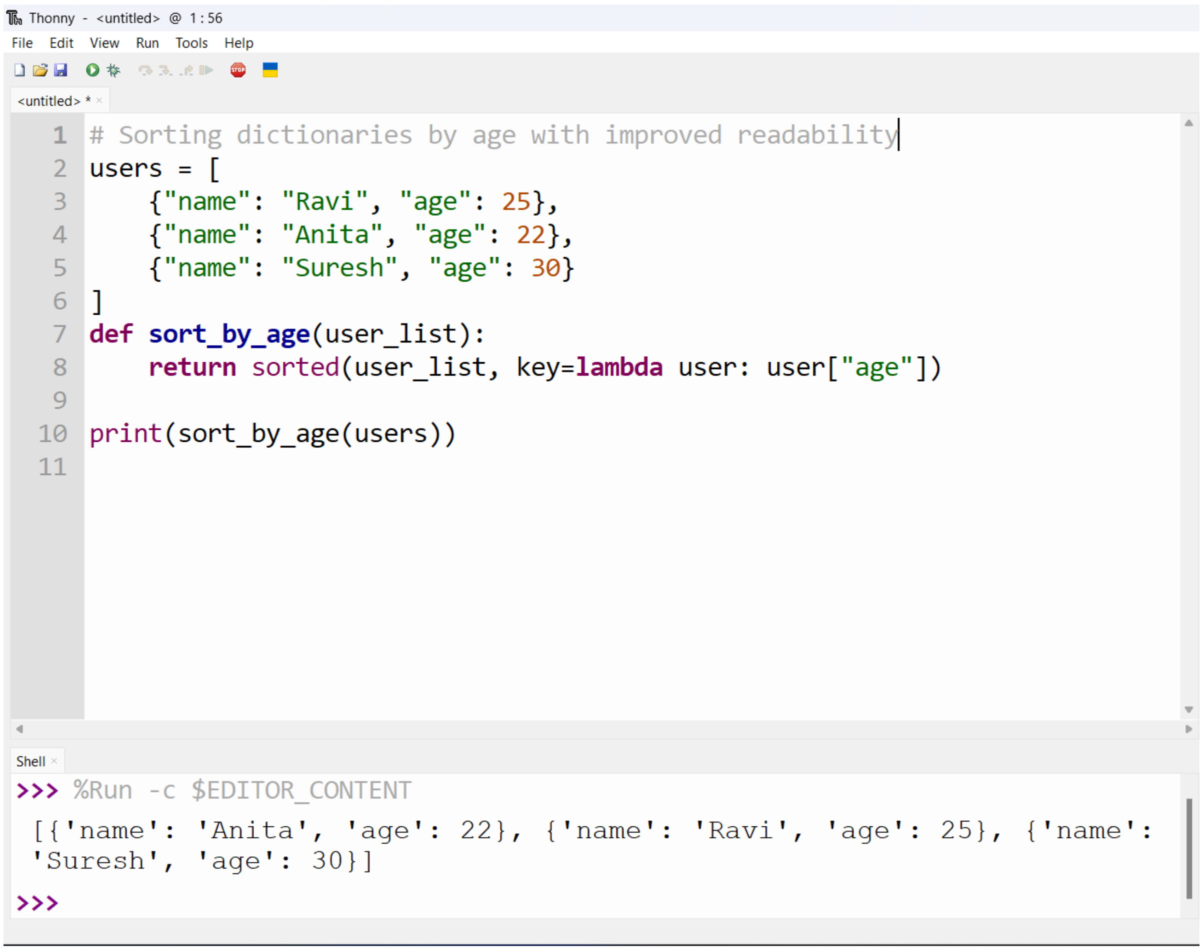
◆ **UsingGoogleGemini**



```python
# Sorting a list of dictionaries by age using Gemini-generated code
users = [
    {"name": "Ravi", "age": 25},
    {"name": "Anita", "age": 22},
    {"name": "Suresh", "age": 30}
]
sorted_users = sorted(users, key=lambda x: x["age"])
print(sorted_users)
```

```
>>> %Run -c $EDITOR_CONTENT
 [{'name': 'Anita', 'age': 22}, {'name': 'Ravi', 'age': 25}, {'name':
 'Suresh', 'age': 30}]
>>>
```

UsingCursor AI



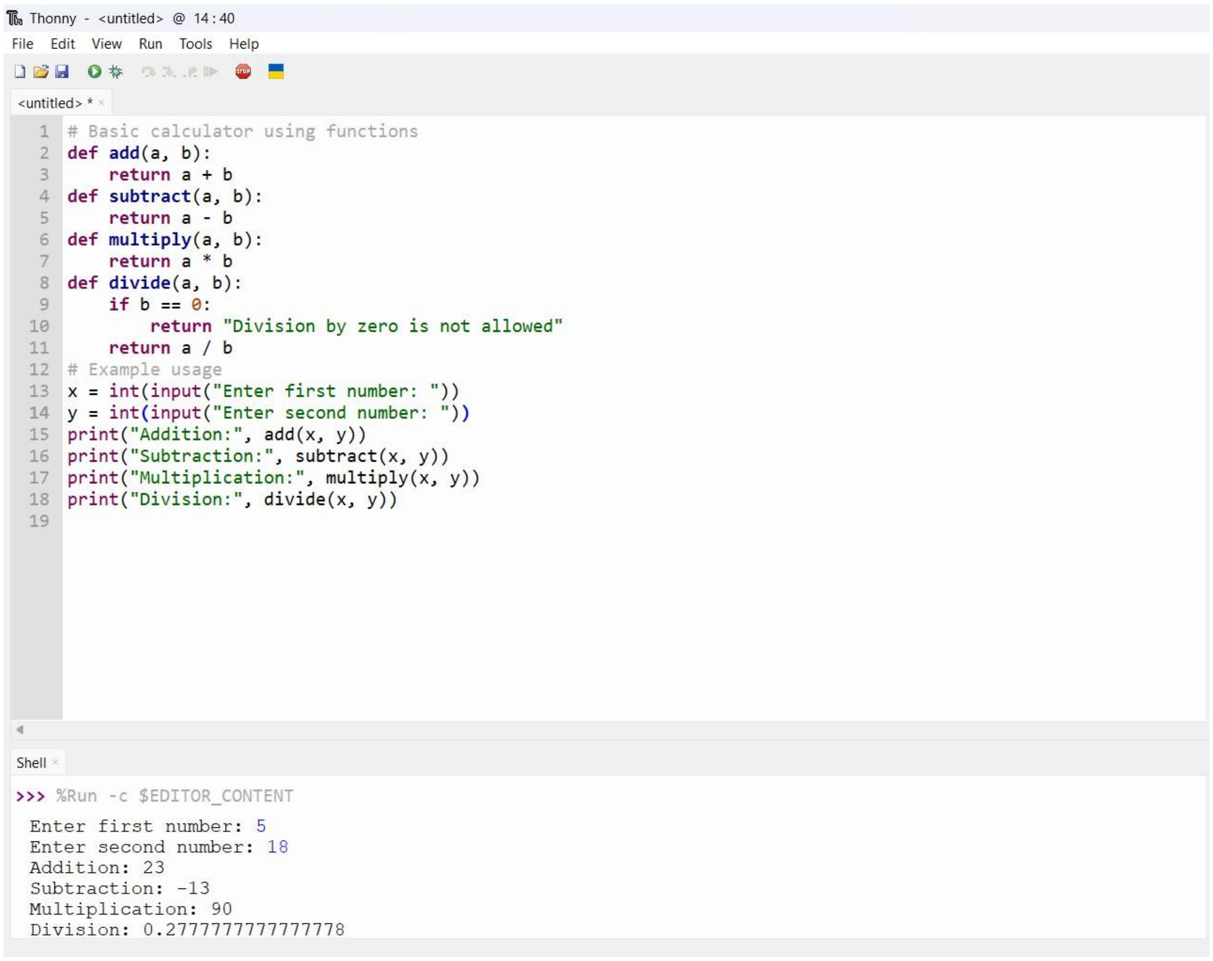
```python
# Sorting dictionaries by age with improved readability
users = [
    {"name": "Ravi", "age": 25},
    {"name": "Anita", "age": 22},
    {"name": "Suresh", "age": 30}
]
def sort_by_age(user_list):
    return sorted(user_list, key=lambda user: user["age"])

print(sort_by_age(users))
```

```
>>> %Run -c $EDITOR_CONTENT
 [{'name': 'Anita', 'age': 22}, {'name': 'Ravi', 'age': 25}, {'name':
 'Suresh', 'age': 30}]
>>>
```

Comparison(ShortNote):
The solutiongeneratedbyGemini is short and direct,making it quick to understand for simple tasks.CursorAI,ontheotherhand,focusedmoreoncleanstructurebyusingafunction,which improves readability and makes the code reusable. While both approaches have similar performance,theCursorAIversionismoresuitableforlargerprojectswheremaintainabilityand clarity are important.

Task3:CalculatorUsingFunctions(Gemini) Calculator

Code

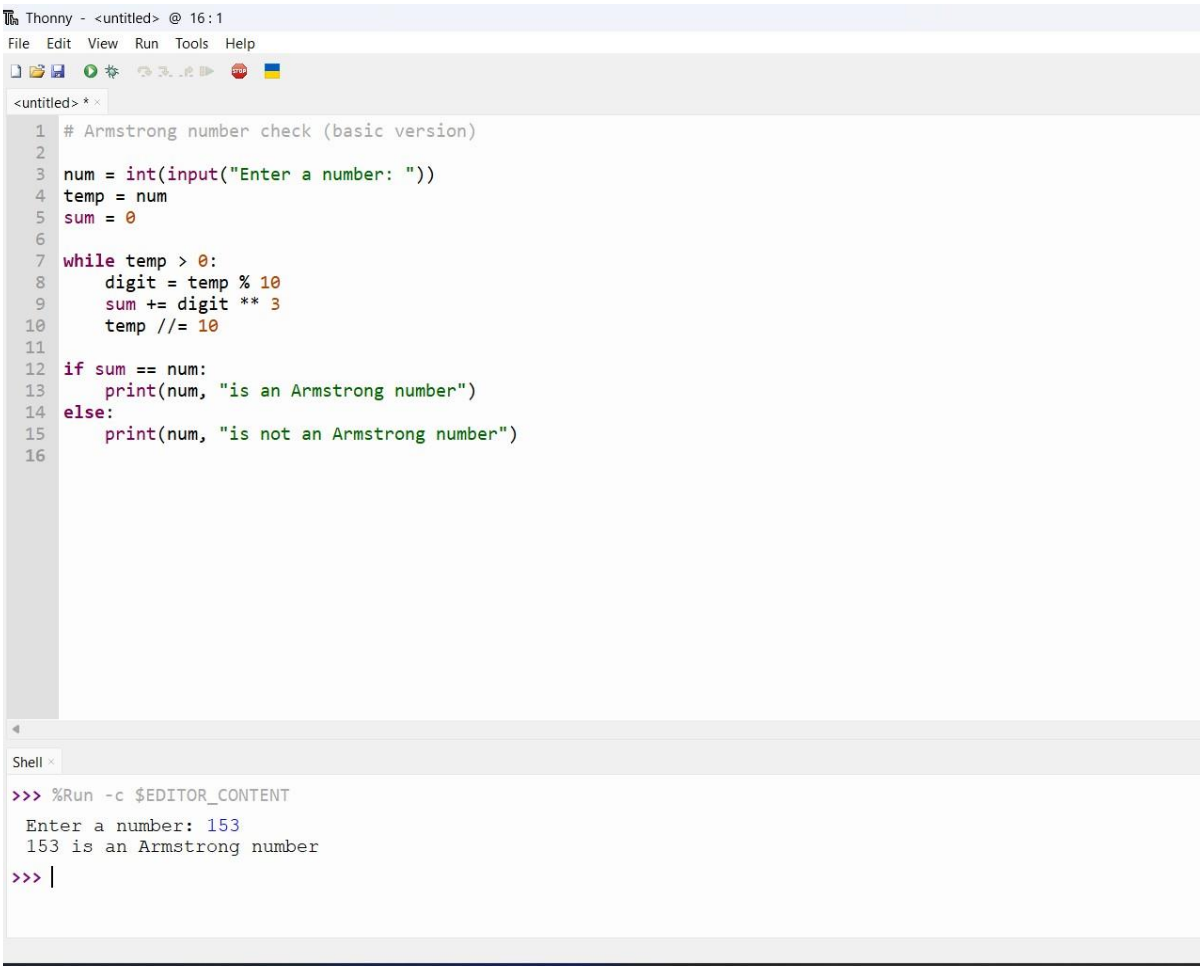


```python
# Basic calculator using functions
def add(a, b):
    return a + b
def subtract(a, b):
    return a - b
def multiply(a, b):
    return a * b
def divide(a, b):
    if b == 0:
        return "Division by zero is not allowed"
    return a / b
# Example usage
x = int(input("Enter first number: "))
y = int(input("Enter second number: "))
print("Addition:", add(x, y))
print("Subtraction:", subtract(x, y))
print("Multiplication:", multiply(x, y))
print("Division:", divide(x, y))
```

```
>>> %Run -c $EDITOR_CONTENT

 Enter first number: 5
 Enter second number: 18
 Addition: 23
 Subtraction: -13
 Multiplication: 90
 Division: 0.2777777777777778
```

Task4:ArmstrongNumberOptimization

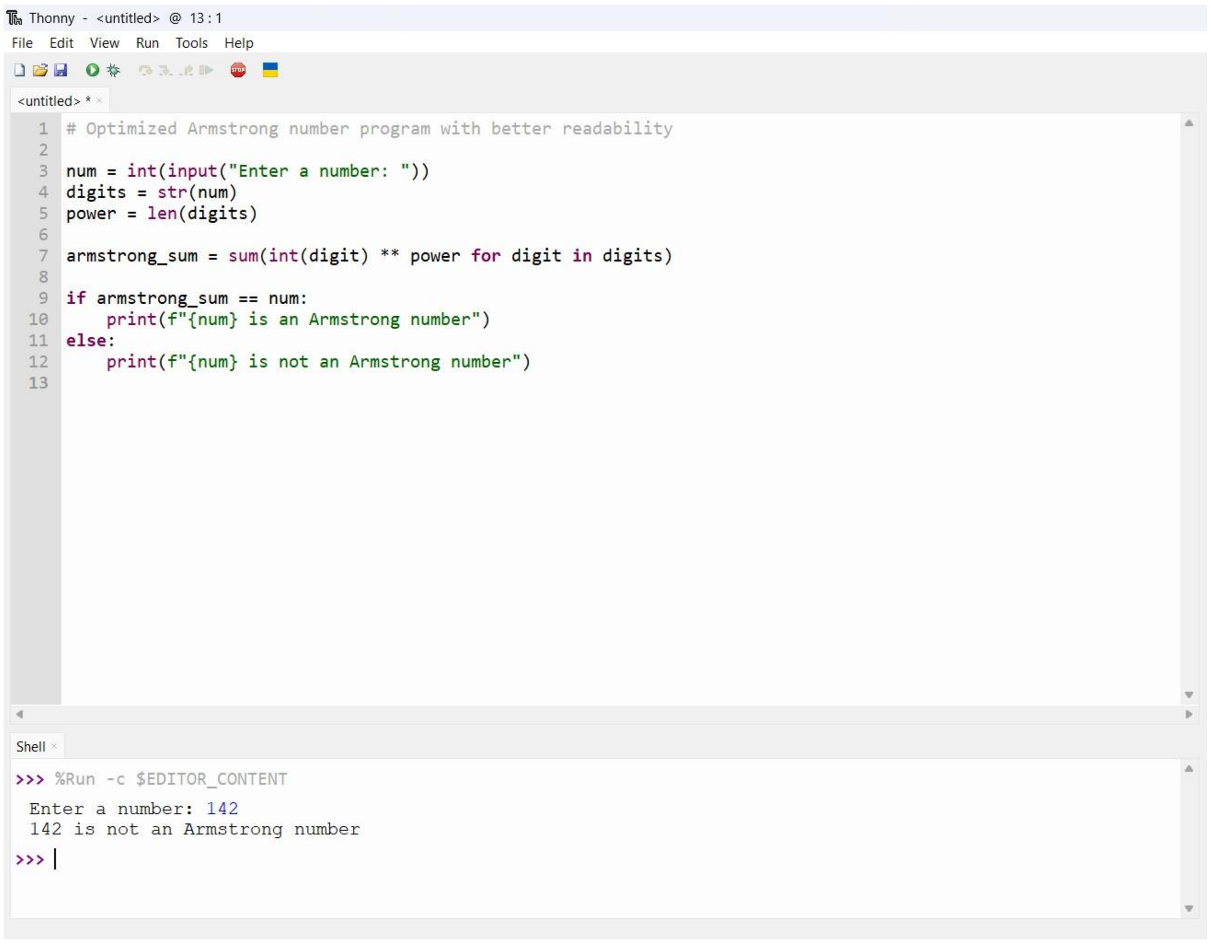
◆ **Version1:Gemini-GeneratedArmstrongProgram**



```python
# Armstrong number check (basic version)

num = int(input("Enter a number: "))
temp = num
sum = 0

while temp > 0:
    digit = temp % 10
    sum += digit ** 3
    temp //= 10

if sum == num:
    print(num, "is an Armstrong number")
else:
    print(num, "is not an Armstrong number")

```

```
>>> %Run -c $EDITOR_CONTENT

 Enter a number: 153
 153 is an Armstrong number

>>> 
```

Version2:OptimizedUsingCursorAI



```python
# Optimized Armstrong number program with better readability

num = int(input("Enter a number: "))
digits = str(num)
power = len(digits)

armstrong_sum = sum(int(digit) ** power for digit in digits)

if armstrong_sum == num:
    print(f"{num} is an Armstrong number")
else:
    print(f"{num} is not an Armstrong number")

```

```
>>> %Run -c $EDITOR_CONTENT

 Enter a number: 142
 142 is not an Armstrong number

>>> 
```

SummaryofImprovements:

- **Reducedlinesofcode**
- **Removedunnecessaryvariables**
- **ImprovedreadabilityusingPythonbuilt-infunctions**
- **WorksforArmstrongnumbersofanylength**