

## AI ASSISTED CODING

### ASSIGNMENT-6.3

Name: B.Bhargava Chary

H.No: 2303A51747

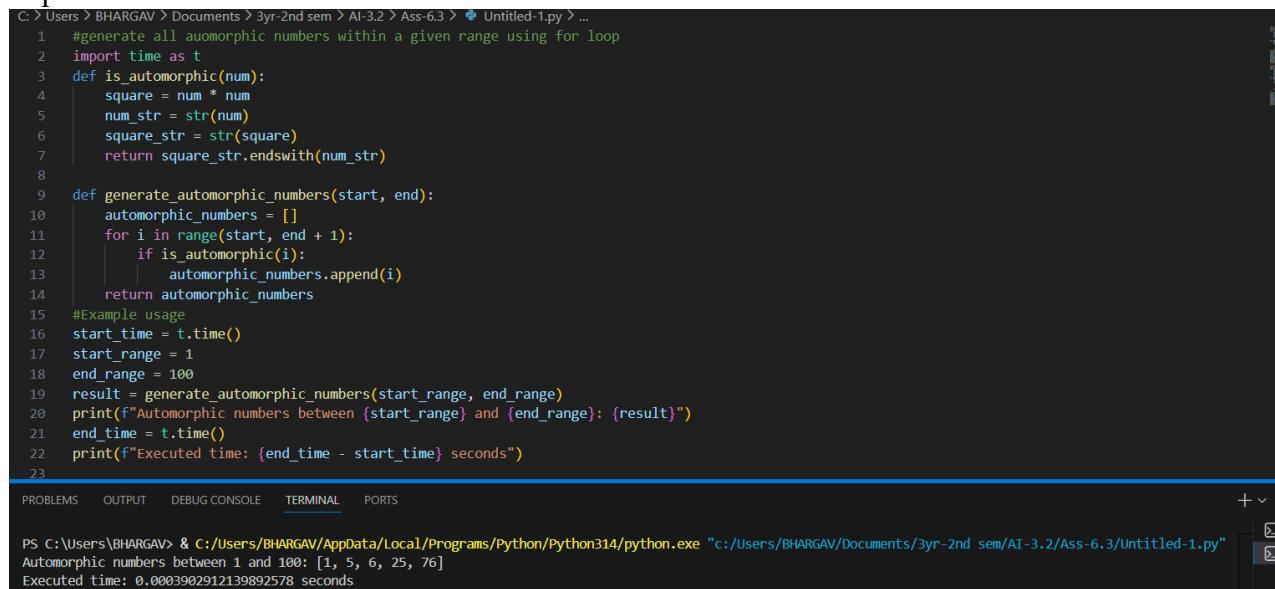
Batch-24

#### Task Description #1 (Loops – Automorphic Numbers in a Range)

- Task: Prompt AI to generate a function that displays all Automorphic numbers between 1 and 1000 using a for loop.
- Instructions:
  - o Get AI-generated code to list Automorphic numbers using a for loop.
  - o Analyze the correctness and efficiency of the generated logic.
  - o Ask AI to regenerate using a while loop and compare both implementations.

Expected Output #1:

- Correct implementation that lists Automorphic numbers using both loop types, with explanation.



```
C:\> Users > BHARGAV > Documents > 3yr-2nd sem > AI-3.2 > Ass-6.3 > Untitled-1.py > ...
1  #generate all automorphic numbers within a given range using for loop
2  import time as t
3  def is_automorphic(num):
4      square = num * num
5      num_str = str(num)
6      square_str = str(square)
7      return square_str.endswith(num_str)
8
9  def generate_automorphic_numbers(start, end):
10     automorphic_numbers = []
11     for i in range(start, end + 1):
12         if is_automorphic(i):
13             automorphic_numbers.append(i)
14     return automorphic_numbers
15
16 #Example usage
17 start_time = t.time()
18 start_range = 1
19 end_range = 100
20 result = generate_automorphic_numbers(start_range, end_range)
21 print(f"Automorphic numbers between {start_range} and {end_range}: {result}")
22 end_time = t.time()
23 print(f"Executed time: {end_time - start_time} seconds")
24
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS +

```
PS C:\Users\BHARGAV> & C:/Users/BHARGAV/AppData/Local/Programs/Python/Python314/python.exe "c:/Users/BHARGAV/Documents/3yr-2nd sem/AI-3.2/Ass-6.3/Untitled-1.py"
Automorphic numbers between 1 and 100: [1, 5, 6, 25, 76]
Executed time: 0.0003902912139892578 seconds
```

```

C:\> Users > BHARGAV > Documents > 3yr-2nd sem > AI-3.2 > Ass-6.3 > Untitled-1.py > ...
24     #Generate all automorphic numbers within a given range using while loop
25     import time as t
26     def is_automorphic(num):
27         square = num * num
28         num_str = str(num)
29         square_str = str(square)
30         return square_str.endswith(num_str)
31     def generate_automorphic_numbers(start, end):
32         automorphic_numbers = []
33         i = start
34         while i <= end:
35             if is_automorphic(i):
36                 automorphic_numbers.append(i)
37             i += 1
38         return automorphic_numbers
39
40 #Example usage
41 start_time = t.time()
42 start_range = 1
43 end_range = 100
44 result = generate_automorphic_numbers(start_range, end_range)
45 print(f"Automorphic numbers between {start_range} and {end_range}: {result}")
46 end_time = t.time()
47 print(f"Executed time: {end_time - start_time} seconds")
48

```

PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS

Automorphic numbers between 1 and 100: [1, 5, 6, 25, 76]  
 Executed time: 9.131431579589844e-05 seconds  
 PS C:\Users\BHARGAV>

1. They use very little memory since only a small list of automorphic numbers is stored.
2. A for loop is faster because it is optimized inside Python.
3. A for loop makes the code cleaner and easier to understand because no manual counter is needed.
4. A for loop is safer and less error-prone since it avoids skipping numbers or running into infinite loops.

### Task Description #2 (Conditional Statements – Online Shopping Feedback Classification)

- Task: Ask AI to write nested if-elif-else conditions to classify online shopping feedback as Positive, Neutral, or Negative based on a numerical rating (1–5).
- Instructions:
  - o Generate initial code using nested if-elif-else.
  - o Analyze correctness and readability.
  - o Ask AI to rewrite using dictionarybased or match-case structure.

Expected Output #2:

- Feedback classification function with explanation and an alternative approach.

```

C:\> Users > BHARGAV > Documents > 3yr-2nd sem > AI-3.2 > Ass-6.3 > Untitled-1.py > ...
49  #generate a nested if-elif-else to classify shopping feedback as positive negative or neutral based
50  #On rating(1-5)
51  def classify_feedback(rating):
52      if rating >= 4:
53          return "Positive"
54      elif rating == 3:
55          return "Neutral"
56      else:
57          return "Negative"
58  #Example usage
59  ratings = [5, 4, 3, 2, 1]
60  for rating in ratings:
61      classification = classify_feedback(rating)
62      print(f"Rating: {rating} - Feedback: {classification}")
63  #Rewrite the above code using dictionary mapping
64  def classify_feedback(rating):
65      feedback_mapping = {
66          5: "Positive",
67          4: "Positive",
68          3: "Neutral",
69          2: "Negative",
70          1: "Negative"
71      }
72      return feedback_mapping.get(rating, "Invalid rating")
73  #Example usage
74  ratings = [5, 4, 3, 2, 1]
75  for rating in ratings:
76      classification = classify_feedback(rating)
77      print(f"Rating: {rating} - Feedback: {classification}")
78

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```

Executed time: 0.00015163421630859375 seconds
Rating: 5 - Feedback: Positive
Rating: 4 - Feedback: Positive
Rating: 3 - Feedback: Neutral
Rating: 2 - Feedback: Negative
Rating: 1 - Feedback: Negative
Rating: 5 - Feedback: Positive
Rating: 4 - Feedback: Positive
Rating: 3 - Feedback: Neutral
Rating: 2 - Feedback: Negative
Rating: 1 - Feedback: Negative
PS C:\Users\BHARGAV>

```

1. A dictionary makes things clear because all rating-to-feedback pairs are visible in one place.
2. It is easy to maintain since you can add or change ratings without changing the main logic.
3. It handles invalid ratings more safely by letting you check or give a default message.
4. It follows Python's standard and clean way of mapping values, so the code looks natural.
5. It stays clean and readable even if the number of rating categories increases.

### Task 3: Statistical\_operations

Define a function named statistical\_operations(tuple\_num) that performs the following statistical operations on a tuple of numbers:

- Minimum, Maximum
- Mean, Median, Mode
- Variance, Standard Deviation

While writing the function, observe the code suggestions provided by GitHub Copilot. Make decisions to accept, reject, or modify the suggestions based on their relevance and correctness

```
C:\> Users > BHARGAV > Documents > 3yr-2nd sem > AI-3.2 > Ass-6.3 > Untitled-1.py > ...
79     #generate a function named statistical_operations that takes list of numbers as input and calculate
80     #minimum,maximum,mean,median,mode,standard deviation,variance of the list
81     import statistics as stats
82     def statistical_operations(numbers):
83         minimum = min(numbers)
84         maximum = max(numbers)
85         mean = stats.mean(numbers)
86         median = stats.median(numbers)
87         mode = stats.mode(numbers)
88         std_dev = stats.stdev(numbers)
89         variance = stats.variance(numbers)
90
91         return {
92             "Minimum": minimum,
93             "Maximum": maximum,
94             "Mean": mean,
95             "Median": median,
96             "Mode": mode,
97             "Standard Deviation": std_dev,
98             "Variance": variance
99         }
100    #Example usage
101    data = [10,20,20,30,40,50]
102    results = statistical_operations(data)
103    for key, value in results.items():
104        print(f'{key}: {value}')
105

PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS

Minimum: 10
Maximum: 50
Mean: 28.33333333333332
Median: 25.0
Mode: 20
Standard Deviation: 14.719601443879744
Variance: 216.66666666666666
PS C:\Users\BHARGAV> []
```

## Task 4: Teacher Profile

- Prompt: Create a class Teacher with attributes teacher\_id, name, subject, and experience.  
Add a method to display teacher details.
- Expected Output: Class with initializer, method, and object creation.

```
108 #Create class teacher with attributes teacher_id, name, subject,
109 # and experience.Add a method to display teacher details.
110 class Teacher:
111     def __init__(self, teacher_id, name, subject, experience):
112         self.teacher_id = teacher_id
113         self.name = name
114         self.subject = subject
115         self.experience = experience
116
117     def display_details(self):
118         print(f"Teacher ID: {self.teacher_id}")
119         print(f"Name: {self.name}")
120         print(f"Subject: {self.subject}")
121         print(f"Experience: {self.experience} years")
122
123 #Example usage
124 teacher = Teacher(1, "John Doe", "Mathematics", 10)
125 teacher.display_details()
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

PS C:\Users\BHARGAV> & C:/Users/BHARGAV/AppData/Local/Programs/Python/Python314/python.exe "c:/Users/BHARGAV/Documents/3yr-2nd sem/AI-3.2/Ass-6.3/Untitled-1.py"

Teacher ID: 1  
Name: John Doe  
Subject: Mathematics  
Experience: 10 years

PS C:\Users\BHARGAV>

## Task #5 – Zero-Shot Prompting with Conditional Validation

Use zero-shot prompting to instruct an AI tool to generate a function that validates an Indian mobile number.

### Requirements

- The function must ensure the mobile number:
  - Starts with 6, 7, 8, or 9
  - Contains exactly 10 digits

```
126 #Generate a python code that validates an indian mobile number.starts with 6,7,9 or 9
127 import re
128 def validate_mobile_number(mobile_number):
129     pattern = r'^[6-9]\d{9}$'
130     if re.match(pattern, mobile_number):
131         return True
132     else:
133         return False
134 #Example usage
135 mobile_number= input("Enter an Indian mobile number: ")
136 if validate_mobile_number(mobile_number):
137     print("Valid mobile number.")
138 else:
139     print("Invalid mobile number.")

PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS
```

PS C:\Users\BHARGAV> & C:/Users/BHARGAV/AppData/Local/Programs/Python/Python314/python.exe "c:/Users/BHARGAV/Documents/3yr-2nd sem/AI-3.6.3/Untitled-1.py"
Enter an Indian mobile number: 8520883781
Valid mobile number.
PS C:\Users\BHARGAV> & C:/Users/BHARGAV/AppData/Local/Programs/Python/Python314/python.exe "c:/Users/BHARGAV/Documents/3yr-2nd sem/AI-3.6.3/Untitled-1.py"
Enter an Indian mobile number: 7993555427
Valid mobile number.
PS C:\Users\BHARGAV>

mucxz66y76ge2

q

## Task Description #6 (Loops – Armstrong Numbers in a Range)

Task: Write a function using AI that finds all Armstrong numbers in a user- specified range (e.g., 1 to 1000).

Instructions:

- Use a for loop and digit power logic.
- Validate correctness by checking known Armstrong numbers (153, 370, etc.).
- Ask AI to regenerate an optimized version (using list comprehensions).

Expected Output #7:

- Python program listing Armstrong numbers in the range.
- Optimized version with explanation.

```

142     #Write a python code that finds all armstrong numbers in a user-specified range(e.g., 1 to 1000).using for loop
143
144     def is_armstrong(num):
145         num_str = str(num)
146         num_digits = len(num_str)
147         armstrong_sum = sum(int(digit) ** num_digits for digit in num_str)
148         return armstrong_sum == num
149
150     def generate_armstrong_numbers(start, end):
151         armstrong_numbers = []
152         for i in range(start, end + 1):
153             if is_armstrong(i):
154                 armstrong_numbers.append(i)
155
156     start_range = 1

```

PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS

PS C:\Users\BHARGAV> & C:/Users/BHARGAV/AppData/Local/Programs/Python/Python314/python.exe "c:/Users/BHARGAV/Documents/3yr-2nd sem/AI-3.2/Ass-6.3/Untitled-1.py"  
c:/Users/BHARGAV/Documents/3yr-2nd sem/AI-3.2/Ass-6.3/Untitled-1.py:130: SyntaxWarning: "\d" is an invalid escape sequence. Such sequences will not work in the future. Did you mean "\\d"? A raw string is also an option.  
pattern = r'^[6-9]\d{9}\$'  
Armstrong numbers between 1 and 1000: [1, 2, 3, 4, 5, 6, 7, 8, 9, 153, 370, 371, 407]  
PS C:\Users\BHARGAV>

1. It does the same work in fewer lines, so there is less overhead and faster execution.
2. It calculates the digit powers directly instead of using extra variables, reducing unnecessary steps.
3. List comprehension is faster than manually appending values in a loop.
4. It avoids storing extra temporary values, so memory usage stays low.
5. The code is cleaner and easier to read while giving the same correct result.

### Task Description #7 (Loops – Happy Numbers in a Range)

Task: Generate a function using AI that displays all Happy Numbers within a user-specified range (e.g., 1 to 500).

Instructions:

- Implement the logic using a loop: repeatedly replace a number with the sum of the squares of its digits until the result is either 1 (Happy Number) or enters a cycle (Not Happy).
- Validate correctness by checking known Happy Numbers (e.g., 1, 7, 10, 13, 19, 23, 28...).
- Ask AI to regenerate an optimized version (e.g., by using a set to detect cycles instead of infinite loops).

Expected Output #8:

- Python program that prints all Happy Numbers within a range.
- Optimized version using cycle detection with explanation.

```

160 #generate a function that displays all happy within a user-specified range(e.g., 1 to 500).regenerate the code using while loop
161 def is_happy(num):
162     seen = set()
163     while num != 1 and num not in seen:
164         seen.add(num)
165         num = sum(int(digit) ** 2 for digit in str(num))
166     return num == 1
167 def generate_happy_numbers(start, end):
168     happy_numbers = []
169     i = start
170     while i <= end:
171         if is_happy(i):
172             happy_numbers.append(i)
173         i += 1
174     return happy_numbers
175 start_range = 1
176 end_range = 500
177 result = generate_happy_numbers(start_range, end_range)
178 print(f"Happy numbers between {start_range} and {end_range}: {result}")

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

PS C:\Users\BHARGAV & C:/Users/BHARGAV/AppData/Local/Programs/Python/Python314/python.exe "c:/Users/BHARGAV/Documents/3yr-2nd sem/AI-3.2/Ass-6.3/Untitled-1.py" c:/Users/BHARGAV/Documents/3yr-2nd sem/AI-3.2/Ass-6.3/Untitled-1.py:130: SyntaxWarning: "d" is an invalid escape sequence. Such sequences will not work in the future. Did you mean "\d"? A raw string is also an option.

pattern = r'^[6-9]d{9}\$'

Happy numbers between 1 and 500: [1, 7, 10, 13, 19, 23, 28, 31, 32, 44, 49, 68, 70, 79, 82, 86, 91, 94, 97, 100, 103, 109, 129, 130, 133, 139, 167, 176, 188, 190, 192, 193, 203, 208, 219, 226, 230, 236, 239, 262, 263, 280, 291, 293, 301, 302, 310, 313, 319, 320, 326, 329, 331, 338, 356, 362, 365, 367, 368, 376, 379, 383, 386, 391, 392, 397, 404, 409, 440, 446, 464, 469, 478, 487, 490, 496]

PS C:\Users\BHARGAV>

1. It saves results of numbers already checked, so the same calculations are not done again and again.
2. This avoids repeated loops for the same digit sequences, which makes it much faster for large ranges.
3. Using a set still prevents infinite loops, but caching adds an extra speed boost.

### Task Description #8 (Loops – Strong Numbers in a Range)

Task: Generate a function using AI that displays all Strong Numbers (sum of factorial of digits equals the number, e.g.,  $145 = 1! + 4! + 5!$ ) within a given range.

Instructions:

- Use loops to extract digits and calculate factorials.
- Validate with examples (1, 2, 145).
- Ask AI to regenerate an optimized version (precompute digit factorials).

Expected Output #9:

- Python program that lists Strong Numbers.
- Optimized version with explanation.

```

180     #generate a function displays all strong numbers(sum of factorial of digits of digiyts equals the number,
181     #e.g.,145=1+4!+5!)within a given range.with 1,2,145
182     import math;
183     def is_strong_number(num):
184         sum_of_factorials=sum(math.factorial(int(digit))for digit in str(num))
185         return sum_of_factorials==num
186     def generate_strong_numbers(start,end):
187         strong_numbers=[]
188         for i in range(start,end+1):
189             if is_strong_number(i):
190                 strong_numbers.append(i)
191         return strong_numbers
192     start_range=1
193     end_range=1000
194     result=generate_strong_numbers(start_range,end_range)
195     print(f"Strong numbers between {start_range} and {end_range}: {result}")

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

PS C:\Users\BHARGAV> & C:/Users/BHARGAV/AppData/Local/Programs/Python/Python314/python.exe "c:/Users/BHARGAV/Documents/3yr-2nd sem/AI-3.2/Ass-6.3/Untitled-1.py"  
c:/Users/BHARGAV/Documents/3yr-2nd sem/AI-3.2/Ass-6.3/Untitled-1.py:130: SyntaxWarning: "\d" is an invalid escape sequence. Such sequences will not work in the future. Did you mean "\\\d"? A raw string is also an option.  
pattern = r'^[6-9]\d{9}\$'  
Strong numbers between 1 and 1000: [1, 2, 145]  
PS C:\Users\BHARGAV>

the

```

PS D:\AI> & "C:/Users/sande/miniconda3/sr univ/python.exe" d:/AI/Assignment-6.3.py  

d:/AI/Assignment-6.3.py:106: SyntaxWarning: invalid escape sequence '\d'  

    pattern = r'^[6-9]\d{9}$'  

Strong numbers between 1 and 500: [1, 2, 145]

```

## Time Complexity Comparison:

For larger ranges, this makes a significant performance difference since you're trading a tiny bit of extra memory (10 dictionary entries) for substantial computation savings across the entire loop.

## Task #9 – Few-Shot Prompting for Nested Dictionary Extraction

### Objective

Use few-shot prompting (2–3 examples) to instruct the AI to create a function that parses a nested dictionary representing student information.

### Requirements

- The function should extract and return:

o Full Name o

Branch o

SGPA

### Expected Output

A reusable Python function that correctly navigates and extracts values from nested dictionaries based on the provided examples

```

190
197 #generate a python code that parses a nested dictionary representing student information\
198 #the dictionary contains full name branch SGPA
199 def parse_student_info(student_info):
200     for student_id, details in student_info.items():
201         full_name = details.get("full_name", "N/A")
202         branch = details.get("branch", "N/A")
203         sgpa = details.get("SGPA", "N/A")
204         print(f"Student ID: {student_id}")
205         print(f"Full Name: {full_name}")
206         print(f"Branch: {branch}")
207         print(f"SGPA: {sgpa}")
208         print("-" * 20)
209     #Example usage
210 students = {
211     "001": {"full_name": "Alice Smith", "branch": "Computer Science", "SGPA": 8.5},
212     "002": {"full_name": "Bob Johnson", "branch": "Mechanical Engineering", "SGPA": 7.8},
213     "003": {"full_name": "Charlie Brown", "branch": "Electrical Engineering", "SGPA": 9.2}
214 }
215 parse_student_info(students)

```

PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS

```

Full Name: Alice Smith
Branch: Computer Science
SGPA: 8.5
-----
Student ID: 002
Full Name: Bob Johnson
Branch: Mechanical Engineering
SGPA: 7.8
-----
Student ID: 003
Full Name: Charlie Brown
Branch: Electrical Engineering
SGPA: 9.2

```

## Task Description #10 (Loops – Perfect Numbers in a Range)

Task: Generate a function using AI that displays all Perfect Numbers within a user-specified range (e.g., 1 to 1000).

Instructions:

- A Perfect Number is a positive integer equal to the sum of its proper divisors (excluding itself).
  - Example:  $6 = 1 + 2 + 3$ ,  $28 = 1 + 2 + 4 + 7 + 14$ .
- Use a for loop to find divisors of each number in the range.
- Validate correctness with known Perfect Numbers (6, 28, 496...).
- Ask AI to regenerate an optimized version (using divisor check only up to root n

C: > Users > BHARGAV > Documents > 3yr-2nd sem > AI-3.2 > Ass-6.3 > Untitled-1.py > ...

```
217 #generate a python code displays all perfect numbers within a user-specified range(e.g., 1 to 1000).using for loop
218 def is_perfect(num):
219     if num < 2:
220         return False
221     divisors_sum = sum(i for i in range(1, num) if num % i == 0)
222     return divisors_sum == num
223 def generate_perfect_numbers(start, end):
224     perfect_numbers = []
225     for i in range(start, end + 1):
226         if is_perfect(i):
227             perfect_numbers.append(i)
228     return perfect_numbers
229 start_range = 1
230 end_range = 1000
231 result = generate_perfect_numbers(start_range, end_range)
232 print(f"Perfect numbers between {start_range} and {end_range}: {result}")
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

PS C:\Users\BHARGAV> & C:/Users/BHARGAV/AppData/Local/Programs/Python/Python314/python.exe "c:/Users/BHARGAV/Documents/3yr-2nd sem/AI-3.2/Ass-6.3/Untitled-1.py"
c:\Users\BHARGAV\Documents\3yr-2nd sem\AI-3.2\Ass-6.3\Untitled-1.py:130: SyntaxWarning: "\d" is an invalid escape sequence. Such sequences will not work in the future. Did you mean "\\\d"? A raw string is also an option.
 pattern = r'^[6-9]\d{9}\$'
Perfect numbers between 1 and 1000: [6, 28, 496]
PS C:\Users\BHARGAV>