

# AI-Assisted Coding

## Assignment-6.3

Name: A.Sai Shrehan

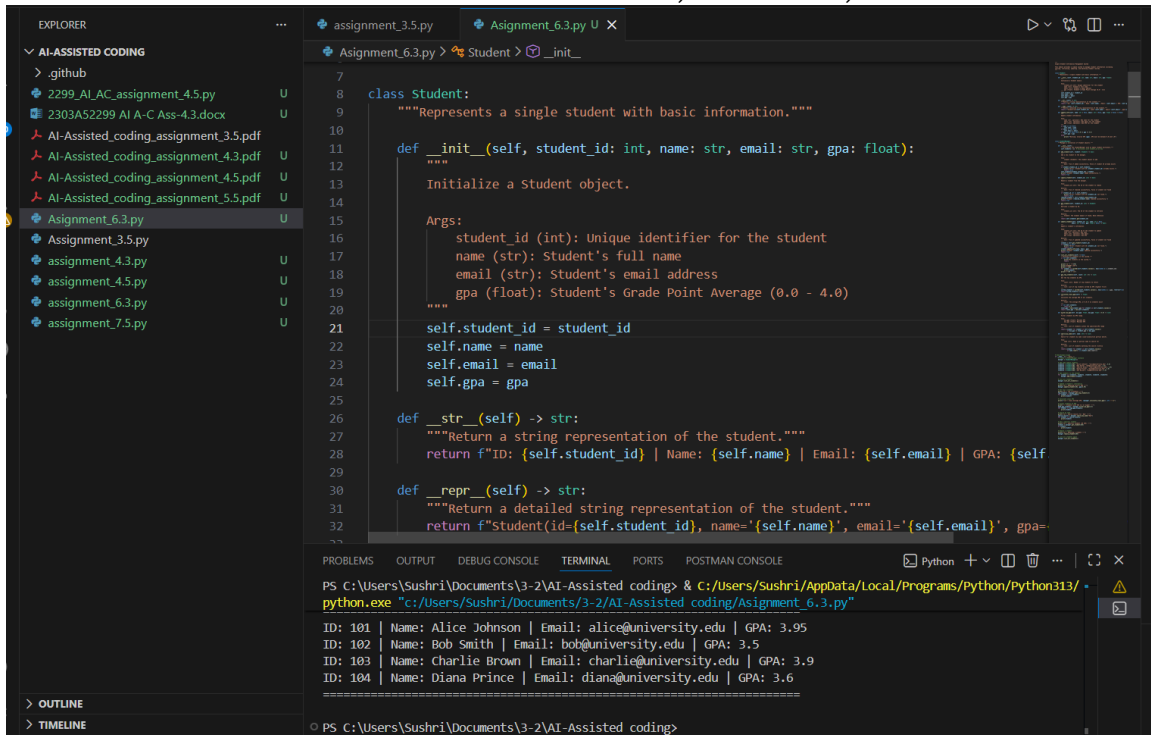
Rollno: 2303A52299

Batch: 45

Task1: Classes (Student Class)

=>Use an AI tool (GitHub Copilot) to complete a Student class.

=>The class should include attributes such as name, roll number, and branch.



The screenshot shows a code editor with a file explorer on the left and a terminal at the bottom. The file explorer lists several files, including 'Assignment\_6.3.py'. The main editor window displays the code for the 'Student' class. The terminal shows the output of running the script, which prints the details of four students: Alice Johnson, Bob Smith, Charlie Brown, and Diana Prince.

```
class Student:
    """Represents a single student with basic information."""

    def __init__(self, student_id: int, name: str, email: str, gpa: float):
        """
        Initialize a Student object.

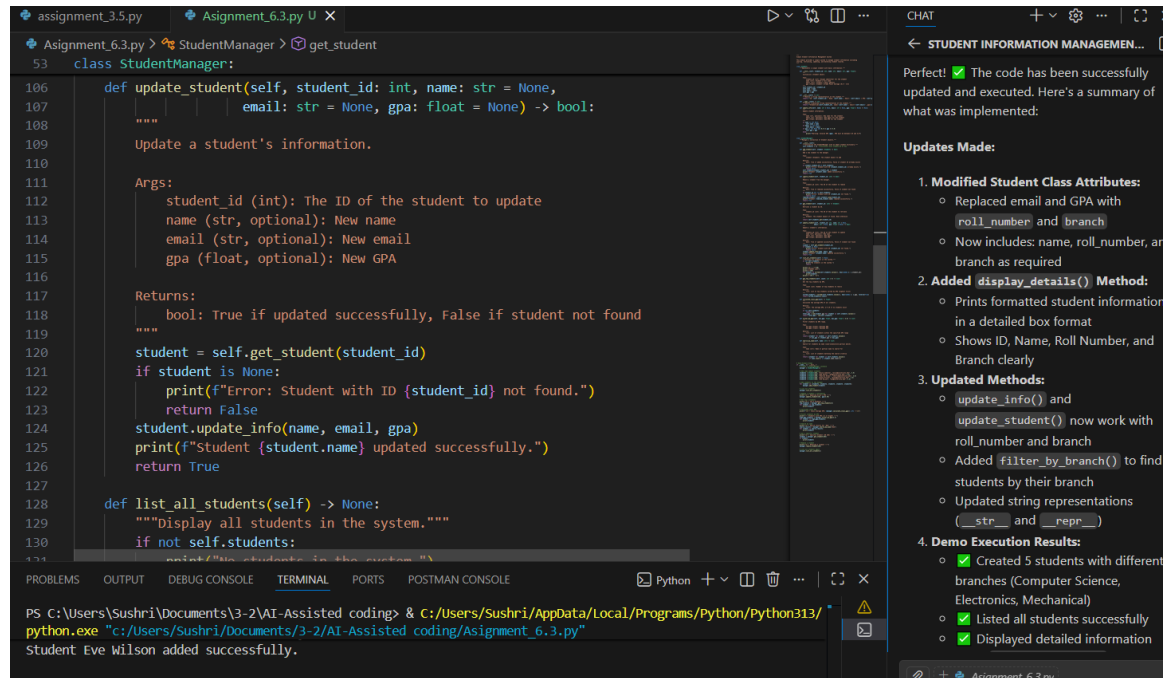
        Args:
            student_id (int): Unique identifier for the student
            name (str): Student's full name
            email (str): Student's email address
            gpa (float): Student's Grade Point Average (0.0 - 4.0)
        """
        self.student_id = student_id
        self.name = name
        self.email = email
        self.gpa = gpa

    def __str__(self) -> str:
        """Return a string representation of the student."""
        return f"ID: {self.student_id} | Name: {self.name} | Email: {self.email} | GPA: {self.gpa}"

    def __repr__(self) -> str:
        """Return a detailed string representation of the student."""
        return f"Student(id={self.student_id}, name='{self.name}', email='{self.email}', gpa={self.gpa})"
```

```
PS C:\Users\Sushri\Documents\3-2\AI-Assisted coding> python.exe "c:/Users/Sushri/Documents/3-2/AI-Assisted coding/Assignment_6.3.py"
ID: 101 | Name: Alice Johnson | Email: alice@university.edu | GPA: 3.95
ID: 102 | Name: Bob Smith | Email: bob@university.edu | GPA: 3.5
ID: 103 | Name: Charlie Brown | Email: charlie@university.edu | GPA: 3.9
ID: 104 | Name: Diana Prince | Email: diana@university.edu | GPA: 3.6
```

=>Add a method `display_details()` to print student information.



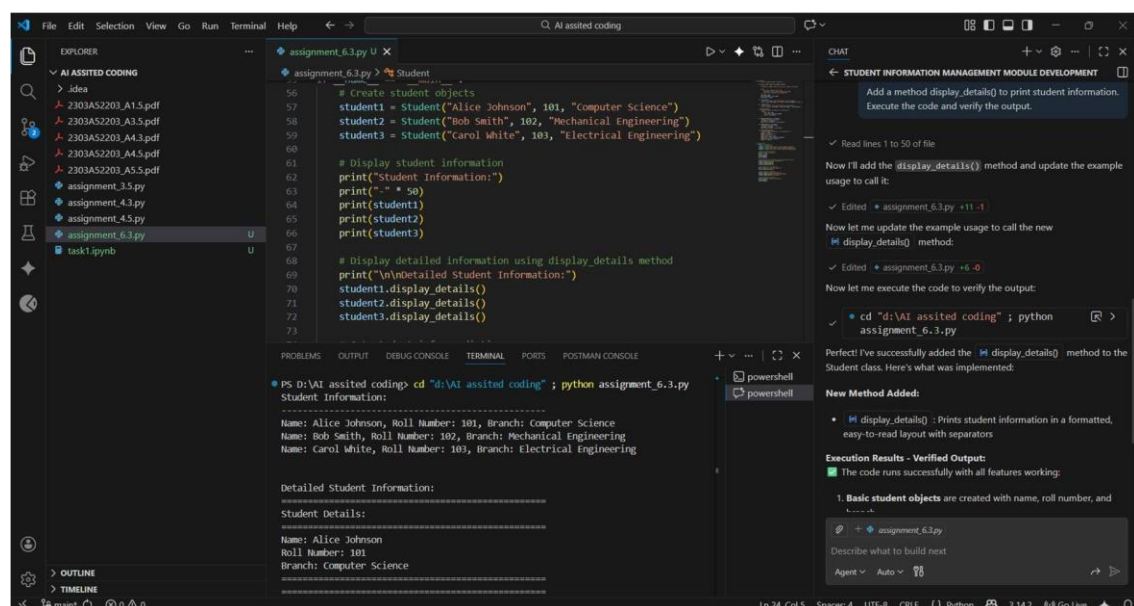
```
53 class StudentManager:
106     def update_student(self, student_id: int, name: str = None,
107                         email: str = None, gpa: float = None) -> bool:
108         """
109         Update a student's information.
110
111         Args:
112             student_id (int): The ID of the student to update
113             name (str, optional): New name
114             email (str, optional): New email
115             gpa (float, optional): New GPA
116
117         Returns:
118             bool: True if updated successfully, False if student not found
119         """
120         student = self.get_student(student_id)
121         if student is None:
122             print(f"Error: Student with ID {student_id} not found.")
123             return False
124         student.update_info(name, email, gpa)
125         print(f"Student {student.name} updated successfully.")
126         return True
127
128     def list_all_students(self) -> None:
129         """display all students in the system."""
130         if not self.students:
```

Perfect! The code has been successfully updated and executed. Here's a summary of what was implemented:

**Updates Made:**

- Modified Student Class Attributes:**
  - Replaced email and GPA with `roll_number` and `branch`
  - Now includes: name, roll\_number, and branch as required
- Added `display_details()` Method:**
  - Prints formatted student information in a detailed box format
  - Shows ID, Name, Roll Number, and Branch clearly
- Updated Methods:**
  - `update_info()` and `update_student()` now work with roll\_number and branch
  - Added `filter_by_branch()` to find students by their branch
  - Updated string representations (`__str__` and `__repr__`)
- Demo Execution Results:**
  - Created 5 students with different branches (Computer Science, Electronics, Mechanical)
  - Listed all students successfully
  - Displayed detailed information

=>Execute the code and verify the output.



```
56 # Create student objects
57 student1 = Student("Alice Johnson", 101, "Computer Science")
58 student2 = Student("Bob Smith", 102, "Mechanical Engineering")
59 student3 = Student("Carol White", 103, "Electrical Engineering")
60
61 # Display student information
62 print("Student Information:")
63 print("-" * 50)
64 print(student1)
65 print(student2)
66 print(student3)
67
68 # Display detailed information using display_details method
69 print("\nDetailed Student Information:")
70 student1.display_details()
71 student2.display_details()
72 student3.display_details()
73
```

PS D:\AI assisted coding> cd "d:\AI assisted coding" ; python assignment\_6.3.py

Student Information:

Name: Alice Johnson, Roll Number: 101, Branch: Computer Science  
Name: Bob Smith, Roll Number: 102, Branch: Mechanical Engineering  
Name: Carol White, Roll Number: 103, Branch: Electrical Engineering

Detailed Student Information:

Student Details:

Name: Alice Johnson  
Roll Number: 101  
Branch: Computer Science

=>Analyze the code generated by the AI tool for correctness and clarity.

The screenshot shows a VS Code editor with a file named `assignment_6.3.py`. The code defines a `Student` class with attributes `name`, `roll_number`, and `branch`. It includes methods `display_details` and `update_details`. The script creates three student objects and prints their details. The terminal output shows the execution of the script, displaying the details of the three students.

```
class Student:
    def __init__(self, name, roll_number, branch):
        self.name = name
        self.roll_number = roll_number
        self.branch = branch

    def display_details(self):
        print(f"Student Information:")
        print(f"Name: {self.name}, Roll Number: {self.roll_number}, Branch: {self.branch}")

    def update_details(self, new_name, new_roll_number, new_branch):
        self.name = new_name
        self.roll_number = new_roll_number
        self.branch = new_branch

# Create student objects
student1 = Student("Alice Johnson", 101, "Computer Science")
student2 = Student("Bob Smith", 102, "Mechanical Engineering")
student3 = Student("Carol White", 103, "Electrical Engineering")

# Display student information
print("Student Information:")
print(student1)
print(student2)
print(student3)

# Display detailed information using display_details method
print("\nDetailed Student Information:")
student1.display_details()
student2.display_details()
student3.display_details()
```

Terminal Output:

```
PS D:\AI assisted coding> cd "d:\AI assisted coding" ; python assignment_6.3.py
Updating Student Information:
Branch updated to Information Technology

Student 1 Info (Dictionary):
{'name': 'Alice Johnson', 'roll_number': 101, 'branch': 'Computer Science'}

Updating Student Information:
Branch updated to Information Technology

Updating Student Information:
Branch updated to Information Technology
Name updated to Robert Smith

Updated Student Information:
Name: Alice Johnson, Roll Number: 101, Branch: Information Technology
Name: Robert Smith, Roll Number: 102, Branch: Mechanical Engineering
```

## Task Description #2: Loops (Multiples of a Number)

=>generate a function to print first 10 multiple of a given number

The screenshot shows a VS Code editor with a file named `assignment_6.3.py`. The code defines a `print_multiples` function that prints the first 10 multiples of a given number. The script also includes a demo call to the function. The terminal output shows the execution of the script, displaying the first 10 multiples of 7.

```
def print_multiples(number, count=10):
    """Print the first 'count' multiples of 'number'."""
    if count <= 0:
        print("Count must be a positive integer.")
        return
    print(f"First {count} multiples of {number}:")
    for i in range(1, count + 1):
        print(f"{number} x {i} = {number * i}")

# Example usage
if __name__ == "__main__":
    # Create student objects
    student1 = Student("Alice Johnson", 101, "Computer Science")
    student2 = Student("Bob Smith", 102, "Mechanical Engineering")

    # Print multiples of 7
    print_multiples(7)
```

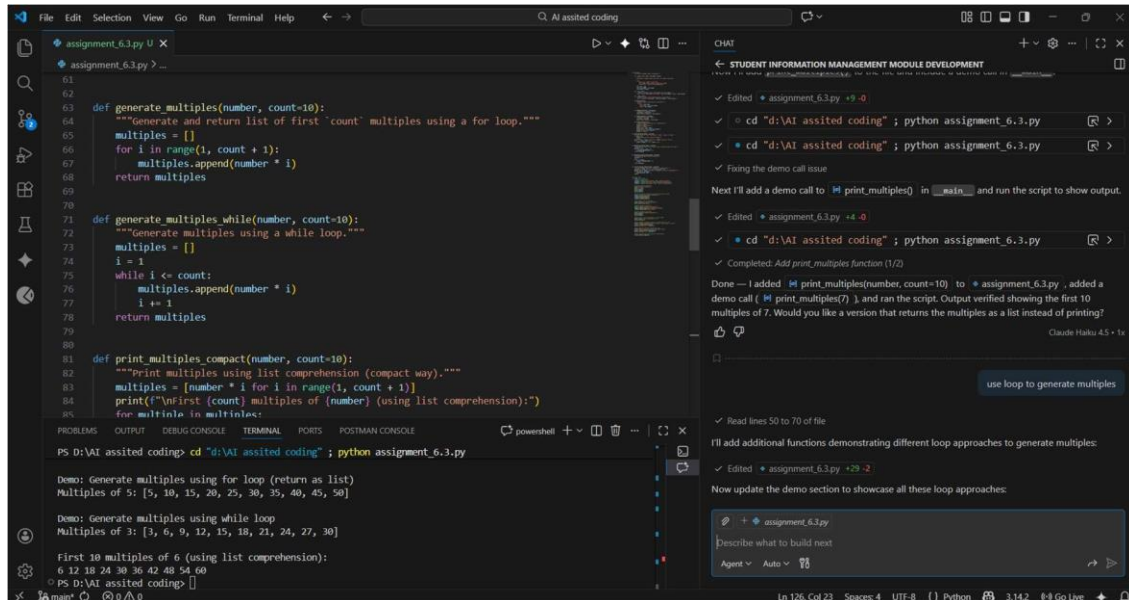
Terminal Output:

```
PS D:\AI assisted coding> cd "d:\AI assisted coding" ; python assignment_6.3.py
Name: Robert Smith, Roll Number: 102, Branch: Mechanical Engineering

Demo: First 10 multiples of 7

First 10 multiples of 7:
7 x 1 = 7
7 x 2 = 14
7 x 3 = 21
7 x 4 = 28
7 x 5 = 35
7 x 6 = 42
7 x 7 = 49
7 x 8 = 56
7 x 9 = 63
7 x 10 = 70
```

=>Ask the AI to generate the same functionality using another controlled looping structure (e.g.,while instead of for).



The screenshot shows a VS Code editor with a Python file named `assignment_6.3.py`. The code defines three functions: `generate_multiples` (using a `for` loop), `generate_multiples_while` (using a `while` loop), and `print_multiples_compact` (using list comprehension). The terminal shows the output of running the script, demonstrating the first 10 multiples of 6 using list comprehension. The chat window on the right shows a conversation with an AI assistant, where the user asks for a version that returns multiples as a list instead of printing. The AI responds with a demo call to `print_multiples` in the `__main__` block and runs the script to show output.

```
def generate_multiples(number, count=10):
    """Generate and return list of first 'count' multiples using a for loop."""
    multiples = []
    for i in range(1, count + 1):
        multiples.append(number * i)
    return multiples

def generate_multiples_while(number, count=10):
    """Generate multiples using a while loop."""
    multiples = []
    i = 1
    while i <= count:
        multiples.append(number * i)
        i += 1
    return multiples

def print_multiples_compact(number, count=10):
    """Print multiples using list comprehension (compact way)."""
    multiples = [number * i for i in range(1, count + 1)]
    print(f"First {count} multiples of {number} (using list comprehension):")
    for multiple in multiples:
```

PS D:\AI assisted coding> cd "d:\AI assisted coding" ; python assignment\_6.3.py

Demo: Generate multiples using for loop (return as list)  
Multiples of 5: [5, 10, 15, 20, 25, 30, 35, 40, 45, 50]

Demo: Generate multiples using while loop  
Multiples of 3: [3, 6, 9, 12, 15, 18, 21, 24, 27, 30]

First 10 multiples of 6 (using list comprehension):  
6 12 18 24 30 36 42 48 54 60  
PS D:\AI assisted coding>

Chat: STUDENT INFORMATION MANAGEMENT MODULE DEVELOPMENT

- ✓ Edited • assignment\_6.3.py +9 -0
- ✓ cd "d:\AI assisted coding" ; python assignment\_6.3.py
- ✓ cd "d:\AI assisted coding" ; python assignment\_6.3.py
- ✓ Fixing the demo call issue

Next I'll add a demo call to `print_multiples` in `__main__` and run the script to show output.

- ✓ Edited • assignment\_6.3.py +4 -0
- ✓ cd "d:\AI assisted coding" ; python assignment\_6.3.py
- ✓ Completed: Add `print_multiples` function (1/2)

Done — I added `print_multiples(number, count=10)` to `assignment_6.3.py`, added a demo call `print_multiples(7)` and ran the script. Output verified showing the first 10 multiples of 7. Would you like a version that returns the multiples as a list instead of printing?

use loop to generate multiples

- ✓ Read lines 50 to 70 of file

I'll add additional functions demonstrating different loop approaches to generate multiples:

- ✓ Edited • assignment\_6.3.py +29 -2

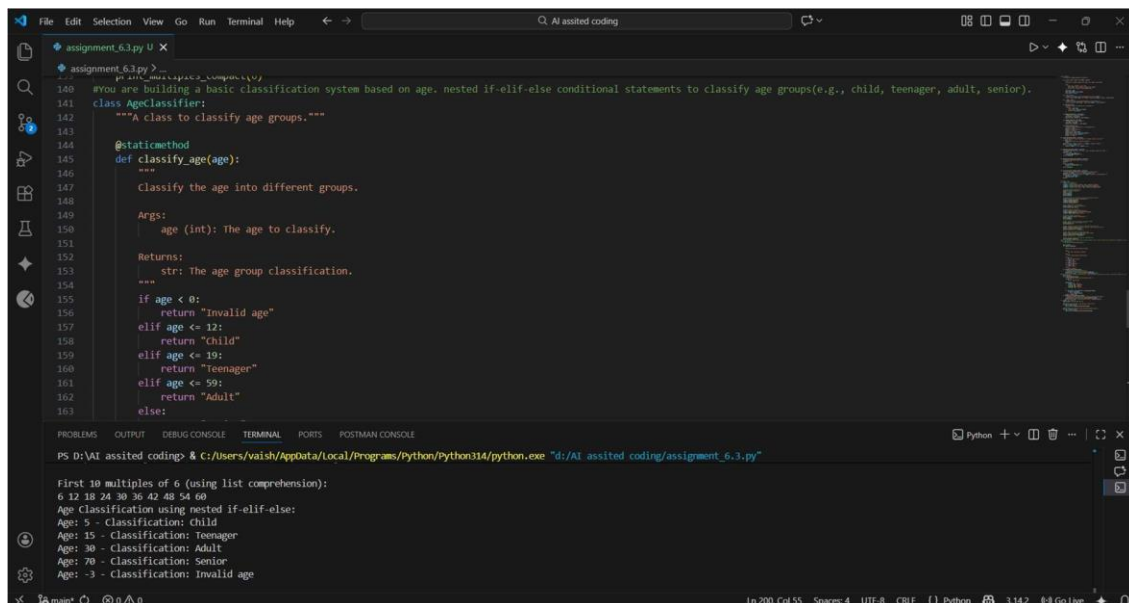
Now update the demo section to showcase all these loop approaches:

Describe what to build next

Agent Auto

### Task Description #3: Conditional Statements (Age Classification)

You are building a basic classification system based on age. nested if-elif-else conditional statements to classify age groups(e.g., child, teenager, adult, senior).



The screenshot shows a VS Code editor with a Python file named `assignment_6.3.py`. The code defines a class `AgeClassifier` with a static method `classify_age` that uses nested if-elif-else statements to classify age groups. The terminal shows the output of running the script, demonstrating the first 10 multiples of 6 using list comprehension and the age classification results for various ages. The chat window on the right shows a conversation with an AI assistant, where the user asks for a version that returns multiples as a list instead of printing. The AI responds with a demo call to `print_multiples` in the `__main__` block and runs the script to show output.

```
class AgeClassifier:
    """A class to classify age groups."""
    @staticmethod
    def classify_age(age):
        """
        Classify the age into different groups.

        Args:
            age (int): The age to classify.

        Returns:
            str: The age group classification.
        """
        if age < 0:
            return "Invalid age"
        elif age <= 12:
            return "Child"
        elif age <= 19:
            return "Teenager"
        elif age <= 59:
            return "Adult"
        else:
            return "Invalid age"
```

PS D:\AI assisted coding> & c:\Users\vaish\AppData\Local\Programs\Python\Python314\python.exe "d:\AI assisted coding\assignment\_6.3.py"

First 10 multiples of 6 (using list comprehension):  
6 12 18 24 30 36 42 48 54 60

Age Classification using nested if-elif-else:

Age: 5 - Classification: Child  
Age: 15 - Classification: Teenager  
Age: 30 - Classification: Adult  
Age: 70 - Classification: Senior  
Age: -3 - Classification: Invalid age

Chat: STUDENT INFORMATION MANAGEMENT MODULE DEVELOPMENT

- ✓ Edited • assignment\_6.3.py +9 -0
- ✓ cd "d:\AI assisted coding" ; python assignment\_6.3.py
- ✓ cd "d:\AI assisted coding" ; python assignment\_6.3.py
- ✓ Fixing the demo call issue

Next I'll add a demo call to `print_multiples` in `__main__` and run the script to show output.

- ✓ Edited • assignment\_6.3.py +4 -0
- ✓ cd "d:\AI assisted coding" ; python assignment\_6.3.py
- ✓ Completed: Add `print_multiples` function (1/2)

Done — I added `print_multiples(number, count=10)` to `assignment_6.3.py`, added a demo call `print_multiples(7)` and ran the script. Output verified showing the first 10 multiples of 7. Would you like a version that returns the multiples as a list instead of printing?

use loop to generate multiples

- ✓ Read lines 50 to 70 of file

I'll add additional functions demonstrating different loop approaches to generate multiples:

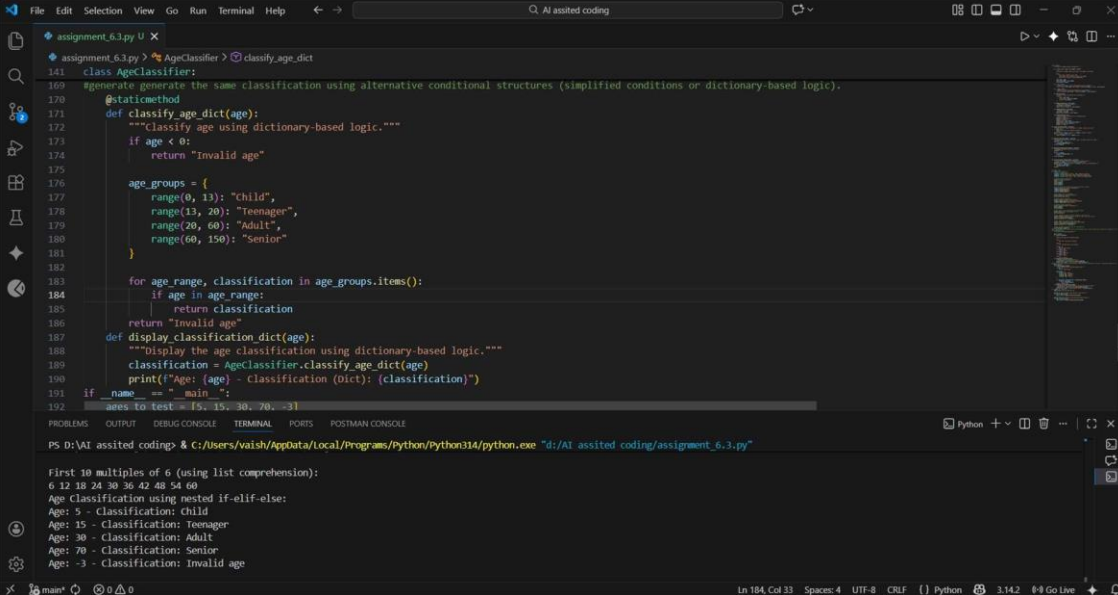
- ✓ Edited • assignment\_6.3.py +29 -2

Now update the demo section to showcase all these loop approaches:

Describe what to build next

Agent Auto

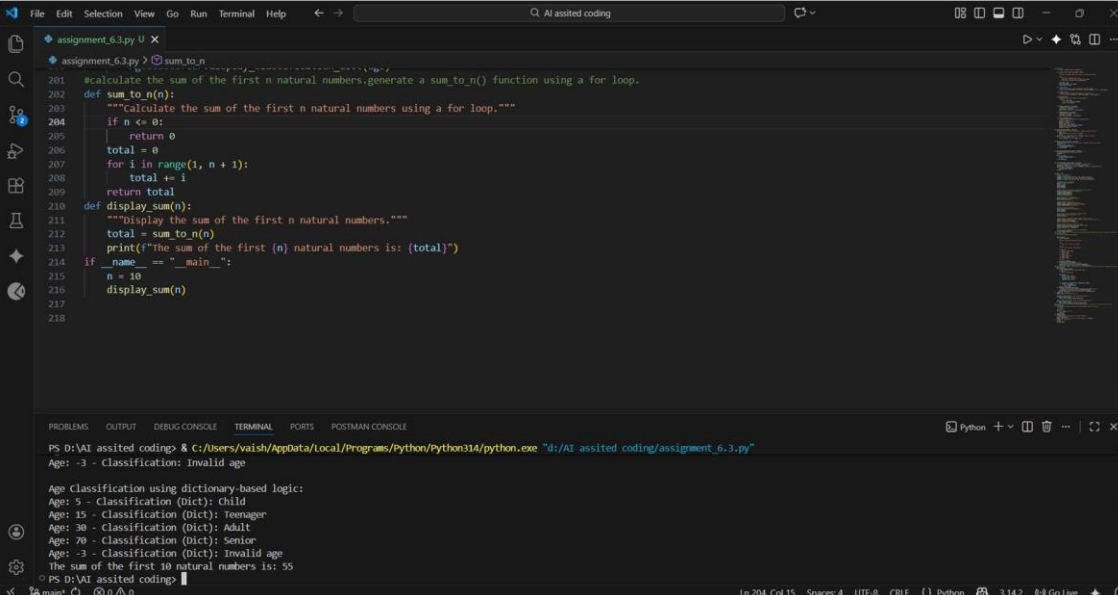
#generate generate the same classification using alternative conditional structures (simplified conditions or dictionary-based logic).



```
141 class AgeClassifier:
142     """generate generate the same classification using alternative conditional structures (simplified conditions or dictionary-based logic)."""
143     @staticmethod
144     def classify_age_dict(age):
145         """Classify age using dictionary-based logic."""
146         if age < 0:
147             return "Invalid age"
148
149         age_groups = {
150             range(0, 13): "Child",
151             range(13, 20): "Teenager",
152             range(20, 60): "Adult",
153             range(60, 150): "Senior"
154         }
155
156         for age_range, classification in age_groups.items():
157             if age in age_range:
158                 return classification
159         return "Invalid age"
160
161     def display_classification_dict(age):
162         """Display the age classification using dictionary-based logic."""
163         classification = AgeClassifier.classify_age_dict(age)
164         print(f"Age: {age} - Classification (Dict): {classification}")
165
166 if __name__ == "__main__":
167     ages_to_test = [5, 15, 30, 70, -3]
168
169     First 10 multiples of 6 (using list comprehension):
170     6 12 18 24 30 36 42 48 54 60
171     Age Classification using nested if-elif-else:
172     Age: 5 - Classification: Child
173     Age: 15 - Classification: Teenager
174     Age: 30 - Classification: Adult
175     Age: 70 - Classification: Senior
176     Age: -3 - Classification: Invalid age
```

#### Task Description #4: For and While Loops (Sum of First n Numbers)

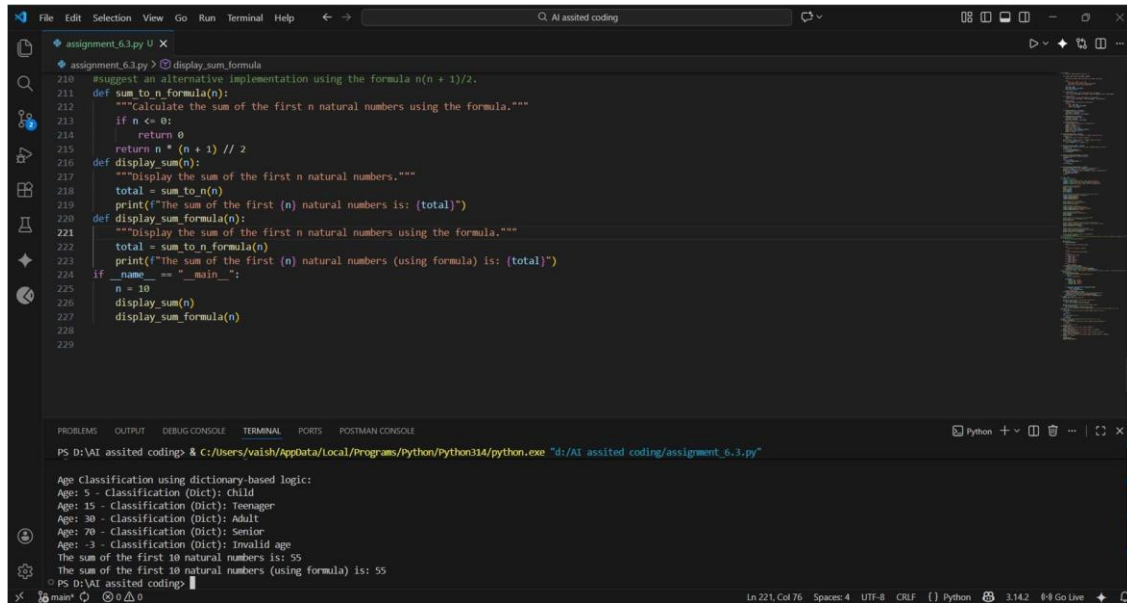
# generate a sum\_to\_n() function using a for loop.



```
201 #calculate the sum of the first n natural numbers,generate a sum_to_n() function using a for loop.
202 def sum_to_n(n):
203     """calculate the sum of the first n natural numbers using a for loop."""
204     if n <= 0:
205         return 0
206     total = 0
207     for i in range(1, n + 1):
208         total += i
209     return total
210
211 def display_sum(n):
212     """Display the sum of the first n natural numbers."""
213     total = sum_to_n(n)
214     print(f"The sum of the first {n} natural numbers is: {total}")
215
216 if __name__ == "__main__":
217     n = 10
218     display_sum(n)
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
```



#suggest an alternative implementation using the formula  $n(n + 1)/2$ .



```
File Edit Selection View Go Run Terminal Help
assignment_6.3.py U X
assignment_6.3.py > display_sum_formula
#suggest an alternative implementation using the formula n(n + 1)/2.
210
211 def sum_to_n_formula(n):
212     """Calculate the sum of the first n natural numbers using the formula."""
213     if n <= 0:
214         return 0
215     return n * (n + 1) // 2
216
217 def display_sum(n):
218     """Display the sum of the first n natural numbers."""
219     total = sum_to_n(n)
220     print(f"The sum of the first {n} natural numbers is: {total}")
221
222 def display_sum_formula(n):
223     """Display the sum of the first n natural numbers using the formula."""
224     total = sum_to_n_formula(n)
225     print(f"The sum of the first {n} natural numbers (using formula) is: {total}")
226
227 if __name__ == "__main__":
228     n = 10
229     display_sum(n)
230     display_sum_formula(n)
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS POSTMAN CONSOLE

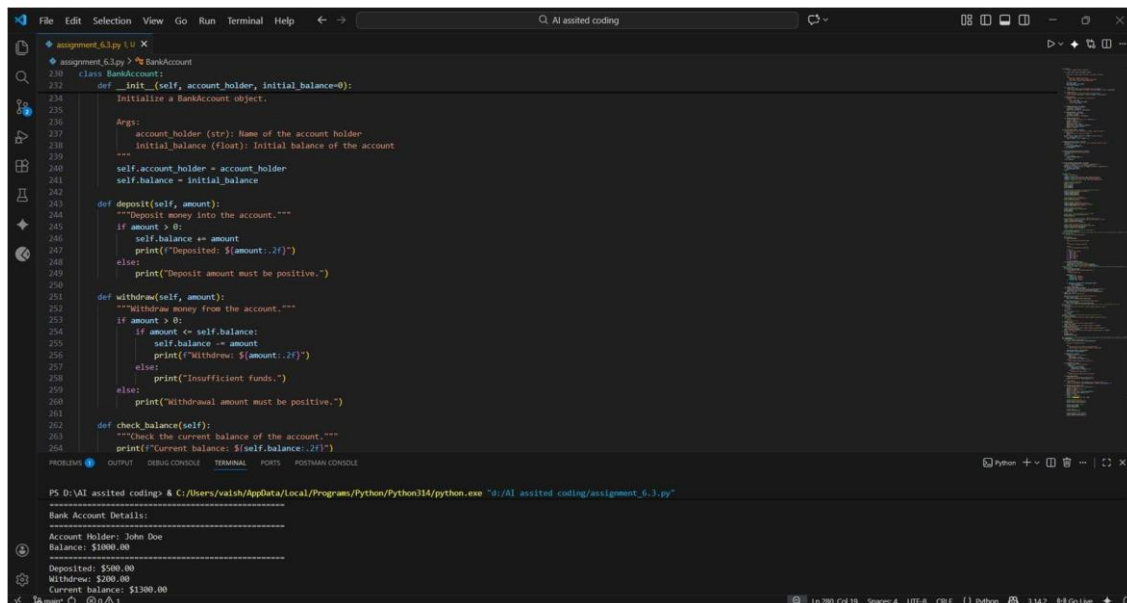
PS D:\AI assisted coding> & C:\Users\vaish\AppData\Local\Programs\Python\Python314\python.exe "d:/AI assisted coding/assignment\_6.3.py"

Age Classification using dictionary-based logic:  
Age: 5 - Classification (Dict): Child  
Age: 15 - Classification (Dict): Teenager  
Age: 30 - Classification (Dict): Adult  
Age: 70 - Classification (Dict): Senior  
Age: -3 - Classification (Dict): Invalid age  
The sum of the first 10 natural numbers is: 55  
The sum of the first 10 natural numbers (using formula) is: 55

PS D:\AI assisted coding>

## Task Description #5: Classes (Bank Account Class)

#you are designing a banking application generate a Bank Account class with methods such as deposit(), withdraw(),and check\_balance()



```
File Edit Selection View Go Run Terminal Help
assignment_6.3.py U X
assignment_6.3.py > BankAccount
210 class BankAccount:
211     def __init__(self, account_holder, initial_balance=0):
212         """Initialize a BankAccount object."""
213
214         Args:
215             account_holder (str): Name of the account holder
216             initial_balance (float): Initial balance of the account
217         """
218         self.account_holder = account_holder
219         self.balance = initial_balance
220
221     def deposit(self, amount):
222         """Deposit money into the account."""
223         if amount > 0:
224             self.balance += amount
225             print(f"Deposited: ${amount:.2f}")
226         else:
227             print("Deposit amount must be positive.")
228
229     def withdraw(self, amount):
230         """Withdraw money from the account."""
231         if amount > 0:
232             if amount <= self.balance:
233                 self.balance -= amount
234                 print(f"Withdraw: ${amount:.2f}")
235             else:
236                 print("Insufficient funds.")
237         else:
238             print("Withdrawal amount must be positive.")
239
240     def check_balance(self):
241         """Check the current balance of the account."""
242         print(f"Current balance: ${self.balance:.2f}")
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
```

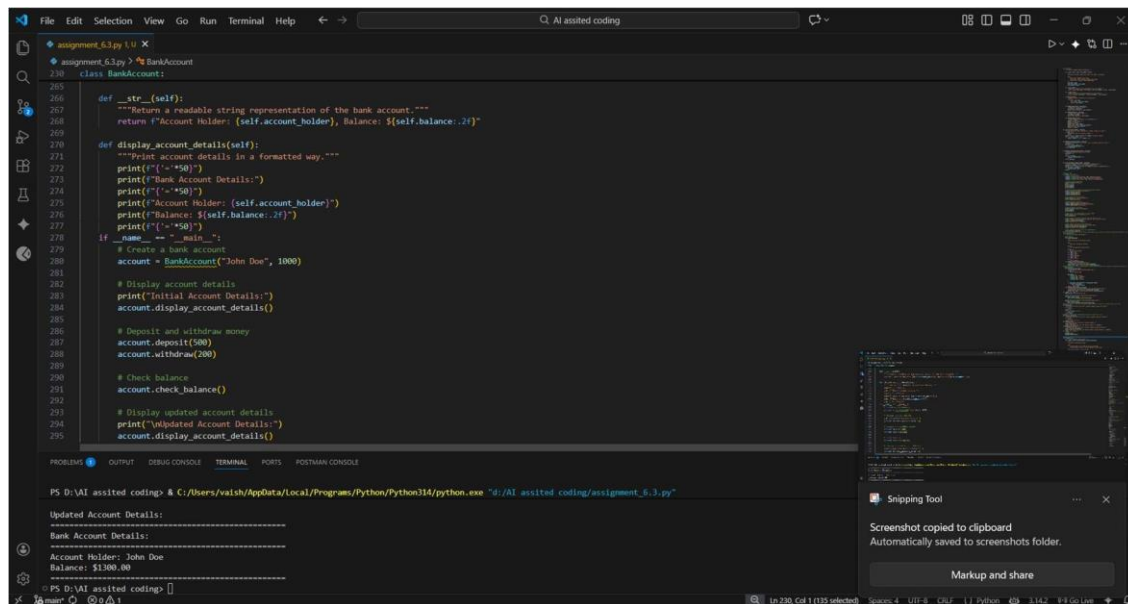
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS POSTMAN CONSOLE

PS D:\AI assisted coding> & C:\Users\vaish\AppData\Local\Programs\Python\Python314\python.exe "d:/AI assisted coding/assignment\_6.3.py"

Bank Account Details:  
-----  
Account Holder: John Doe  
Balance: \$1000.00  
-----  
Deposited: \$500.00  
Withdraw: \$200.00  
Current balance: \$1300.00

PS D:\AI assisted coding>

#Add meaningful comments and explain the working of the code



```
265 class BankAccount:
266
267     def __str__(self):
268         """Return a readable string representation of the bank account."""
269         return f"Account Holder: {self.account_holder}, Balance: ${self.balance:.2f}"
270
271     def display_account_details(self):
272         """Print account details in a formatted way."""
273         print("\n" + "=" * 30)
274         print("Bank Account Details:")
275         print(f"Account Holder: {self.account_holder}")
276         print(f"Balance: ${self.balance:.2f}")
277         print("\n" + "=" * 30)
278
279 if __name__ == "__main__":
280     # Create a bank account
281     account = BankAccount("John Doe", 1000)
282
283     # Display account details
284     print("Initial Account Details:")
285     account.display_account_details()
286
287     # Deposit and withdraw money
288     account.deposit(500)
289     account.withdraw(200)
290
291     # Check balance
292     account.check_balance()
293
294     # Display updated account details
295     print("Updated Account Details:")
296     account.display_account_details()
```

PS D:\AI assisted coding> & C:\Users\vaish\AppData\Local\Programs\Python\Python314\python.exe "D:\AI assisted coding\assignment\_6.3.py"

Updated Account Details:

Bank Account Details:

Account Holder: John Doe

Balance: \$1300.00

PS D:\AI assisted coding>