

ASSIGNMENT - 6.5

Name : J. Sai Harini

Ht No : 2303A52426

Batch : 35

Experiment 6: AI-Based Code Completion: Working with suggestions for classes, loops, conditionals Week3 - Friday

LO1. Use AI-based code completion tools to generate Python code involving classes, loops, and conditionals.

LO2. Interpret and explain AI-generated code line-by-line.

LO3. Identify errors, inefficiencies, or logical flaws in AI-suggested implementations.

LO4. Optimize AI-generated code for better readability and performance.

LO5. Demonstrate ethical and responsible use of AI tools in coding tasks

Task Description #1 (AI-Based Code Completion for Conditional

Eligibility Check)

Task: Use an AI tool to generate eligibility logic.

Prompt:

“Generate Python code to check voting eligibility based on age and citizenship.”

Expected Output:

- AI-generated conditional logic.
 - Correct eligibility decisions.
 - Explanation of conditions.

The screenshot shows the Microsoft Visual Studio Code interface with the following details:

- File Explorer:** Shows files like `main.py`, `check_voting_eligibility.py`, `task_1_analysis.py`, `task_2.py`, and `task_3.py`.
- Code Editor:** Displays Python code for a voting eligibility check. The code defines a function `check_voting_eligibility` that takes age and citizenship status as inputs and returns a result and reason. It then prints analysis for each case in `test_cases_1`.
- Terminal:** Shows command-line output for the `task_1_analysis` script, which includes conditions checked, logic clarity, and error messages.
- Output:** Shows logs for tasks 1, 2, and 3.
- Search:** A search bar at the bottom.
- Taskbar:** Shows icons for various applications like File Explorer, Task View, and Start.
- Status Bar:** Shows system information like battery level, network, and date.

Task Description #2(AI-Based Code Completion for Loop-Based

String Processing)

Task: Use an AI tool to process strings using loops.

Prompt:

“Generate Python code to count vowels and consonants in a string using a loop.”

Expected Output:

- AI-generated string processing logic.
- Correct counts.

- Output verification

The screenshot displays two separate instances of Microsoft Visual Studio Code running on a Windows operating system. Both instances have the same file open: `count_vowels_and_consonants.py`. The code itself is as follows:

```
95
96 # Task 2: Test Cases
97 print("x" * 70)
98 print("TASK 2: LOOP-BASED STRING PROCESSING (VOWELS & CONSONANTS)")
99 print("x" * 70)
100
101 test_strings = [
102     "Hello World",
103     "Python Programming",
104     "aeiou",
105     "bcdg",
106     "123 ABC xyz!",
107     ""
108 ]
109
110 for test_str in test_strings:
111     result = count_vowels_and_consonants(test_str)
112     print(f"String: '{test_str}'")
113     print(f" Vowels: {result['vowels']}, Consonants: {result['consonants']}, ")
114     print(f" Other: {result['other']}")
115     print()
116
```

The interface includes a sidebar with 'RECENTS' showing 'Raised Exceptions', 'Uncaught Exceptions', and 'User Uncaught Exceptions'. Below the code editor are tabs for 'OUTPUT', 'DEBUG CONSOLE', 'TERMINAL', 'PORTS', and 'SWIFTER'. The right panel contains a 'Task 2' section with a summary and several checkmarks indicating completed tasks such as 'Vowel & Consonant Counting' and 'Loop-based string processing using for loop'. A 'Summary of the Assignment Solution' section is also present.

The bottom of the screen shows the Windows taskbar with icons for File Explorer, Task View, and various open browser tabs. The system tray indicates a battery level of 24%, a date of 23-01-2026, and a network connection status.

Task Description #3 (AI-Assisted Code Completion Reflection)

Task)

Task: Use an AI tool to generate a complete program using classes,

loops, and conditionals.

Prompt:

“Generate a Python program for a library management system using classes, loops, and conditional statements.”

Expected Output:

- Complete AI-generated program.
- Review of AI suggestions quality.
- Short reflection on AI-assisted coding experience.

Task Description #4 (AI-Assisted Code Completion for Class-Based Attendance System)

Task: Use an AI tool to generate an attendance management class.

Prompt: “Generate a Python class to mark and display student attendance using loops.”

Expected Output:

- AI-generated attendance logic.
- Correct display of attendance.
- Test cases.

File Edit Selection View Go Run Terminal Help ← → ⌘ A.M.C

RUN AND DEBUG

No customise Run and Debug
Create a launch.json file.

Debug using a terminal command or in an interactive chat.

Run and Debug

TASK 3: AI-ASSISTED CODE COMPLETION - LIBRARY MANAGEMENT SYSTEM

```

124 #
125 # TASK 3: AI-ASSISTED CODE COMPLETION - LIBRARY MANAGEMENT SYSTEM
126 #
127 class Book:
128     """Represents a book in the library."""
129
130     def __init__(self, book_id, title, author, available=True):
131         self.book_id = book_id
132         self.title = title
133         self.author = author
134         self.available = available
135
136     def __str__(self):
137         status = "Available" if self.available else "Checked Out"
138         return f"ID: {self.book_id} | {self.title} by {self.author} [{status}]"
139
140
141 class Library:
142     """Manages a collection of books."""
143
144     def __init__(self):
145         self.books = []

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL WORKS PYTHON

TASK 3: LIBRARY MANAGEMENT SYSTEM

```

... Adding Books ...
✓ Added: 'Python Crash Course' (ID: 1)
✓ Added: 'Clean Code' (ID: 2)
✓ Added: 'Design Patterns' (ID: 3)
✓ Added: 'The Pragmatic Programmer' (ID: 4)

... ALL Books ...

```

WATCHPOINTS

- Added Exceptions
- Thought Exceptions
- User Uncaught Exceptions

24°C Sunny

File Edit Selection View Go Run Terminal Help ← → ⌘ A.M.C

RUN AND DEBUG

No customise Run and Debug
Create a launch.json file.

Debug using a terminal command or in an interactive chat.

Run and Debug

TASK 3: AI-ASSISTED CODE COMPLETION - LIBRARY MANAGEMENT SYSTEM

```

146 class Library:
147
148     def add_book(self, book_id, title, author):
149         """Add a new book to the library."""
150         for book in self.books:
151             if book.book_id == book_id:
152                 print(f"Error: Book ID (book_id) already exists!")
153                 return False
154
155         new_book = Book(book_id, title, author)
156         self.books.append(new_book)
157         print(f"\u2708 Added: '{title}' (ID: {book_id})")
158         return True
159
160     def remove_book(self, book_id):
161         """Remove a book from the library."""
162         for i, book in enumerate(self.books):
163             if book.book_id == book_id:
164                 if book.available:
165                     self.books.pop(i)
166                     print(f"\u2708 Removed: '{(book.title)}'")
167                     return True

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL WORKS PYTHON DATA

TASK 3: LIBRARY MANAGEMENT SYSTEM

```

... Adding Books ...
✓ Added: 'Python Crash Course' (ID: 1)
✓ Added: 'Clean Code' (ID: 2)
✓ Added: 'Design Patterns' (ID: 3)
✓ Added: 'The Pragmatic Programmer' (ID: 4)

... ALL Books ...

```

WATCHPOINTS

- Added Exceptions
- Thought Exceptions
- User Uncaught Exceptions

24°C Sunny

The screenshot shows a code editor with a Python script named `library.py`. The script defines a `Library` class with methods for checking out and returning books. A task summary on the right indicates the completion of a task involving conditional logic and string processing.

```
class Library:
    def checkout_book(self, book_id):
        """Check out a book from the library."""
        for book in self.books:
            if book.book_id == book_id:
                if book.available:
                    book.available = False
                    print(f"/ Checked out: '{book.title}'")
                    return True
                else:
                    print(f"Error: '{book.title}' is not available!")
                    return False

    def return_book(self, book_id):
        """Return a book to the library."""
        for book in self.books:
            if book.book_id == book_id:
                if not book.available:
                    book.available = True
                    print(f"> Returned: '{book.title}'")
                    return True
                else:
                    print(f"Error: Book ID ({book_id}) not found!")
                    return False
```

Task 1: Library Management System

- Adding Books
- ✓ Added: 'Python Crash Course' (ID: 1)
- ✓ Added: 'Clean Code' (ID: 2)
- ✓ Added: 'Design Patterns' (ID: 3)
- ✓ Added: 'The Pragmatic Programmer' (ID: 4)

Task 2: Library Management System

- Loop-based string processing using list comprehension
- Character classification (lowercase, uppercase, whitespace)
- Test cases with various string types
- Character-by-character analysis

Task 3: Library Management System

- Descriptive error message for invalid input
- Agent: Auto - 99%

The screenshot shows a Windows desktop environment with several open windows. The main focus is a code editor window titled 'chatbot_intent_classification.py' which contains Python code for a library management system. The code includes functions for adding books and printing book details. Below the code editor is a terminal window showing the output of running the script. To the right of the terminal is a 'PyCharm' interface with tabs for 'Python Data' and 'PowerShell'. At the bottom of the screen, there's a taskbar with icons for various applications like File Explorer, Edge browser, and file explorers for 'D:\', 'C:\', and 'Python'. The system tray shows battery status at 24% and a date/time of 23-01-2026.

```
145 class Library:
146     def list_all_books(self):
147         """List all books in the library.
148         If no books:
149             print("The library is empty.")
150         else:
151             print(f"All Books in Library ({len(self.books)}) total:")
152             for book in self.books:
153                 print(f"- {book}")
154         return self.books
155
156
157 # Task 3: Test Library System
158 print("-" * 70)
159 print("TASK 3: LIBRARY MANAGEMENT SYSTEM")
160 print("-" * 70)
161
162 library = Library()
163
164 # Add books
165 print("\n---- Adding Books ----")
166 library.add_book(1, "Python Crash Course", "Eric Matthes")
167 library.add_book(2, "Clean Code", "Robert C. Martin")
168 library.add_book(3, "Design Patterns", "Erich Gamma")
169 library.add_book(4, "The Pragmatic Programmer", "Andrew Hunt & David Thomas")
170
171 results = library.list_all_books()
172
173 print(results)
174
175
176 # Task 3: LIBRARY MANAGEMENT SYSTEM
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
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
389
390
391
392
393
394
395
396
397
398
399
399
400
401
402
403
404
405
406
407
408
409
409
410
411
412
413
414
415
416
417
418
419
419
420
421
422
423
424
425
426
427
427
428
429
429
430
431
432
433
433
434
435
435
436
436
437
437
438
438
439
439
440
440
441
441
442
442
443
443
444
444
445
445
446
446
447
447
448
448
449
449
450
450
451
451
452
452
453
453
454
454
455
455
456
456
457
457
458
458
459
459
460
460
461
461
462
462
463
463
464
464
465
465
466
466
467
467
468
468
469
469
470
470
471
471
472
472
473
473
474
474
475
475
476
476
477
477
478
478
479
479
480
480
481
481
482
482
483
483
484
484
485
485
486
486
487
487
488
488
489
489
490
490
491
491
492
492
493
493
494
494
495
495
496
496
497
497
498
498
499
499
500
500
501
501
502
502
503
503
504
504
505
505
506
506
507
507
508
508
509
509
510
510
511
511
512
512
513
513
514
514
515
515
516
516
517
517
518
518
519
519
520
520
521
521
522
522
523
523
524
524
525
525
526
526
527
527
528
528
529
529
530
530
531
531
532
532
533
533
534
534
535
535
536
536
537
537
538
538
539
539
540
540
541
541
542
542
543
543
544
544
545
545
546
546
547
547
548
548
549
549
550
550
551
551
552
552
553
553
554
554
555
555
556
556
557
557
558
558
559
559
560
560
561
561
562
562
563
563
564
564
565
565
566
566
567
567
568
568
569
569
570
570
571
571
572
572
573
573
574
574
575
575
576
576
577
577
578
578
579
579
580
580
581
581
582
582
583
583
584
584
585
585
586
586
587
587
588
588
589
589
590
590
591
591
592
592
593
593
594
594
595
595
596
596
597
597
598
598
599
599
600
600
601
601
602
602
603
603
604
604
605
605
606
606
607
607
608
608
609
609
610
610
611
611
612
612
613
613
614
614
615
615
616
616
617
617
618
618
619
619
620
620
621
621
622
622
623
623
624
624
625
625
626
626
627
627
628
628
629
629
630
630
631
631
632
632
633
633
634
634
635
635
636
636
637
637
638
638
639
639
640
640
641
641
642
642
643
643
644
644
645
645
646
646
647
647
648
648
649
649
650
650
651
651
652
652
653
653
654
654
655
655
656
656
657
657
658
658
659
659
660
660
661
661
662
662
663
663
664
664
665
665
666
666
667
667
668
668
669
669
670
670
671
671
672
672
673
673
674
674
675
675
676
676
677
677
678
678
679
679
680
680
681
681
682
682
683
683
684
684
685
685
686
686
687
687
688
688
689
689
690
690
691
691
692
692
693
693
694
694
695
695
696
696
697
697
698
698
699
699
700
700
701
701
702
702
703
703
704
704
705
705
706
706
707
707
708
708
709
709
710
710
711
711
712
712
713
713
714
714
715
715
716
716
717
717
718
718
719
719
720
720
721
721
722
722
723
723
724
724
725
725
726
726
727
727
728
728
729
729
730
730
731
731
732
732
733
733
734
734
735
735
736
736
737
737
738
738
739
739
740
740
741
741
742
742
743
743
744
744
745
745
746
746
747
747
748
748
749
749
750
750
751
751
752
752
753
753
754
754
755
755
756
756
757
757
758
758
759
759
760
760
761
761
762
762
763
763
764
764
765
765
766
766
767
767
768
768
769
769
770
770
771
771
772
772
773
773
774
774
775
775
776
776
777
777
778
778
779
779
780
780
781
781
782
782
783
783
784
784
785
785
786
786
787
787
788
788
789
789
790
790
791
791
792
792
793
793
794
794
795
795
796
796
797
797
798
798
799
799
800
800
801
801
802
802
803
803
804
804
805
805
806
806
807
807
808
808
809
809
810
810
811
811
812
812
813
813
814
814
815
815
816
816
817
817
818
818
819
819
820
820
821
821
822
822
823
823
824
824
825
825
826
826
827
827
828
828
829
829
830
830
831
831
832
832
833
833
834
834
835
835
836
836
837
837
838
838
839
839
840
840
841
841
842
842
843
843
844
844
845
845
846
846
847
847
848
848
849
849
850
850
851
851
852
852
853
853
854
854
855
855
856
856
857
857
858
858
859
859
860
860
861
861
862
862
863
863
864
864
865
865
866
866
867
867
868
868
869
869
870
870
871
871
872
872
873
873
874
874
875
875
876
876
877
877
878
878
879
879
880
880
881
881
882
882
883
883
884
884
885
885
886
886
887
887
888
888
889
889
890
890
891
891
892
892
893
893
894
894
895
895
896
896
897
897
898
898
899
899
900
900
901
901
902
902
903
903
904
904
905
905
906
906
907
907
908
908
909
909
910
910
911
911
912
912
913
913
914
914
915
915
916
916
917
917
918
918
919
919
920
920
921
921
922
922
923
923
924
924
925
925
926
926
927
927
928
928
929
929
930
930
931
931
932
932
933
933
934
934
935
935
936
936
937
937
938
938
939
939
940
940
941
941
942
942
943
943
944
944
945
945
946
946
947
947
948
948
949
949
950
950
951
951
952
952
953
953
954
954
955
955
956
956
957
957
958
958
959
959
960
960
961
961
962
962
963
963
964
964
965
965
966
966
967
967
968
968
969
969
970
970
971
971
972
972
973
973
974
974
975
975
976
976
977
977
978
978
979
979
980
980
981
981
982
982
983
983
984
984
985
985
986
986
987
987
988
988
989
989
990
990
991
991
992
992
993
993
994
994
995
995
996
996
997
997
998
998
999
999
1000
1000
1001
1001
1002
1002
1003
1003
1004
1004
1005
1005
1006
1006
1007
1007
1008
1008
1009
1009
1010
1010
1011
1011
1012
1012
1013
1013
1014
1014
1015
1015
1016
1016
1017
1017
1018
1018
1019
1019
1020
1020
1021
1021
1022
1022
1023
1023
1024
1024
1025
1025
1026
1026
1027
1027
1028
1028
1029
1029
1030
1030
1031
1031
1032
1032
1033
1033
1034
1034
1035
1035
1036
1036
1037
1037
1038
1038
1039
1039
1040
1040
1041
1041
1042
1042
1043
1043
1044
1044
1045
1045
1046
1046
1047
1047
1048
1048
1049
1049
1050
1050
1051
1051
1052
1052
1053
1053
1054
1054
1055
1055
1056
1056
1057
1057
1058
1058
1059
1059
1060
1060
1061
1061
1062
1062
1063
1063
1064
1064
1065
1065
1066
1066
1067
1067
1068
1068
1069
1069
1070
1070
1071
1071
1072
1072
1073
1073
1074
1074
1075
1075
1076
1076
1077
1077
1078
1078
1079
1079
1080
1080
1081
1081
1082
1082
1083
1083
1084
1084
1085
1085
1086
1086
1087
1087
1088
1088
1089
1089
1090
1090
1091
1091
1092
1092
1093
1093
1094
1094
1095
1095
1096
1096
1097
1097
1098
1098
1099
1099
1100
1100
1101
1101
1102
1102
1103
1103
1104
1104
1105
1105
1106
1106
1107
1107
1108
1108
1109
1109
1110
1110
1111
1111
1112
1112
1113
1113
1114
1114
1115
1115
1116
1116
1117
1117
1118
1118
1119
1119
1120
1120
1121
1121
1122
1122
1123
1123
1124
1124
1125
1125
1126
1126
1127
1127
1128
1128
1129
1129
1130
1130
1131
1131
1132
1132
1133
1133
1134
1134
1135
1135
1136
1136
1137
1137
1138
1138
1139
1139
1140
1140
1141
1141
1142
1142
1143
1143
1144
1144
1145
1145
1146
1146
1147
1147
1148
1148
1149
1149
1150
1150
1151
1151
1152
1152
1153
1153
1154
1154
1155
1155
1156
1156
1157
1157
1158
1158
1159
1159
1160
1160
1161
1161
1162
1162
1163
1163
1164
1164
1165
1165
1166
1166
1167
1167
1168
1168
1169
1169
1170
1170
1171
1171
1172
1172
1173
1173
1174
1174
1175
1175
1176
1176
1177
1177
1178
1178
1179
1179
1180
1180
1181
1181
1182
1182
1183
1183
1184
1184
1185
1185
1186
1186
1187
1187
1188
1188
1189
1189
1190
1190
1191
1191
1192
1192
1193
1193
1194
1194
1195
1195
1196
1196
1197
1197
1198
1198
1199
1199
1200
1200
1201
1201
1202
1202
1203
1203
1204
1204
1205
1205
1206
1206
1207
1207
1208
1208
1209
1209
1210
1210
1211
1211
1212
1212
1213
1213
1214
1214
1215
1215
1216
1216
1217
1217
1218
1218
1219
1219
1220
1220
1221
1221
1222
1222
1223
1223
1224
1224
1225
1225
1226
1226
1227
1227
1228
1228
1229
1229
1230
1230
1231
1231
1232
1232
1233
1233
1234
1234
1235
1235
1236
1236
1237
1237
1238
1238
1239
1239
1240
1240
1241
1241
1242
1242
1243
1243
1244
1244
1245
1245
1246
1246
1247
1247
1248
1248
1249
1249
1250
1250
1251
1251
1252
1252
1253
1253
1254
1254
1255
1255
1256
1256
1257
1257
1258
1258
1259
1259
1260
1260
1261
1261
1262
1262
1263
1263
1264
1264
1265
1265
1266
1266
1267
1267
1268
1268
1269
1269
1270
1270
1271
1271
1272
1272
1273
1273
1274
1274
1275
1275
1276
1276
1277
1277
1278
1278
1279
1279
1280
1280
1281
1281
1282
1282
1283
1283
1284
1284
1285
1285
1286
1286
1287
1287
1288
1288
1289
1289
1290
1290
1291
1291
1292
1292
1293
1293
1294
1294
1295
1295
1296
1296
1297
1297
1298
1298
1299
1299
1300
1300
1301
1301
1302
1302
1303
1303
1304
1304
1305
1305
1306
1306
1307
1307
1308
1308
1309
1309
1310
1310
1311
1311
1312
1312
1313
1313
1314
1314
1315
1315
1316
1316
1317
1317
1318
1318
1319
1319
1320
1320
1321
1321
1322
1322
1323
1323
1324
1324
1325
1325
1326
1326
1327
1327
1328
1328
1329
1329
1330
1330
1331
1331
1332
1332
1333
1333
1334
1334
1335
1335
1336
1336
1337
1337
1338
1338
1339
1339
1340
1340
1341
1341
1342
1342
1343
1343
1344
1344
1345
1345
1346
1346
1347
1347
1348
1348
1349
1349
1350
1350
1351
1351
1352
1352
1353
1353
1354
1354
1355
1355
1356
1356
1357
1357
1358
1358
1359
1359
1360
1360
1361
1361
1362
1362
1363
1363
1364
1364
1365
1365
1366
1366
1367
1367
1368
1368
1369
1369
1370
1370
1371
1371
1372
1372
1373
1373
1374
1374
1375
1375
1376
1376
1377
1377
1378
1378
1379
1379
1380
1380
1381
1381
1382
1382
1383
1383
1384
1384
1385
1385
1386
1386
1387
1387
1388
1388
1389
1389
1390
1390
1391
1391
1392
1392
1393
1393
1394
1394
1395
1395
1396
1396
1397
1397
1398
1398
1399
1399
1400
1400
1401
1401
1402
1402
1403
1403
1404
1404
1405
1405
1406
1406
1407
1407
1408
1408
1409
1409
1410
1410
1411
1411
1412
1412
1413
1413
1414
1414
1415
1415
1416
1416
1417
1417
1418
1418
1419
1419
1420
1420
1421
1421
1422
1422
1423
1423
1424
1424
1425
1425
1426
1426
1427
1427
1428
1428
1429
1429
1430
1430
1431
1431
1432
1432
1433
1433
1434
1434
1435
1435
1436
1436
1437
1437
1438
1438
1439
1439
1440
1440
1441
1441
1442
1442
1443
1443
1444
1444
1445
1445
1446
1446
1447
1447
1448
1448
1449
1449
1450
1450
1451
1451
1452
1452
1453
1453
1454
1454
1455
1455
1456
1456
1457
1457
1458
1458
1459
1459
1460
1460
1461
1461
1462
1462
1463
1463
1464
1464
1465
1465
1466
1466
1467
1467
1468
1468
1469
1469
1470
1470
1471
1471
1472
1472
1473
1473
1474
1474
1475
1475
1476
1476
1477
1477
1478
1478
1479
1479
1480
1480
1481
1481
1482
1482
1483
1483
1484
1484
1485
1485
1486
1486
1487
1487
1488
1488
1489
1489
1490
1490
1491
1491
1492
1492
1493
1493
1494
1494
1495
1495
1496
1496
1497
1497
1498
1498
1499
1499
1500
1500
1501
1501
1502
1502
1503
1503
1504
1504
1505
1505
1506
1506
1507
1507
1508
1508
1509
1509
1510
1510
1511
1511
1512
1512
1513
1513
1514
1514
1515
1515
1516
1516
1517
1517
1518
1518
1519
1519
1520
1520
1521
1521
1522
1522
1523
1523
1524
1524
1525
1525
1526
1526
1527
1527
1528
1528
1529
1529
1530
1530
1531
1531
1532
1532
1533
1533
1534
1534
1535
1535
1536
1536
1537
1537
1538
1538
1539
1539
1540
1540
1541
1541
1542
1542
1543
1543
1544
1544
1545
1545
1546
1546
1547
1547
1548
1548
1549
1549
1550
1550
1551
1551
1552
1552
1553
1553
1554
1554
1555
1555
1556
1556
1557
1557
1558
1558
1559
1559
1560
1560
1561
1561
1562
1562
1563
1563
1564
1564
1565
1565
1566
1566
1567
1567
1568
1568
1569
1569
1570
1570
1571
1571
1572
1572
1573
1573
1574
1574
1575
1575
1576
1576
1577
1577
1578
1578
1579
1579
1580
1580
1581
1581
1582
1582
1583
1583
1584
1584
1585
1585
1586
1586
1587
1587
1588
1588
1589
1589
1590
1590
1591
1591
1592
1592
1593
1593
1594
1594
1595
1595
1596
1596
1597
1597
1598
1598
1599
1599
1600
1600
1601
1601
1602
1602
1603
1603
1604
1604
1605
1605
1606
1606
1607
1607
1608
1608
1609
1609
1610
1610
1611
1611
1612
1612
1613
1613
1614
1614
1615
1615
1616
1616
1617
1617
1618
1618
1619
1619
1620
1620
1621
1621
1622
1622
1623
1623
1624
1624
1625
1625
1626
1626
1627
162
```

File Edit Selection View Go Run Terminal Help ← →

Run and Debug

No customise Run and Debug
create a livereload file.

Debug using a terminal command or in an interactive shell.

```

244 # Add books
245 print("\n---- Adding Books ---")
246 library.add_book(1, "Python Crash Course", "Eric Matthes")
247 library.add_book(2, "Clean Code", "Robert C. Martin")
248 library.add_book(3, "Design Patterns", "Gang of Four")
249 library.add_book(4, "The Pragmatic Programmer", "David Thomas")
250
251 # List all books
252 print("\n---- All Books ---")
253 library.list_all_books()
254
255 # Checkout books
256 print("\n---- Checkout Operations ---")
257 library.checkout_book(1)
258 library.checkout_book(2)
259
260 # List available books
261 print("\n---- Available Books ---")
262 library.list_available_books()
263
264
265 # Search for books

```

RESULTS OUTPUT DEBUG CONSOLE TERMINAL PORTS JUPYTER

TASK 3: LIBRARY MANAGEMENT SYSTEM

```

---- Adding Books ---
✓ Added: 'Python Crash Course' (ID: 1)
✓ Added: 'Clean Code' (ID: 2)
✓ Added: 'Design Patterns' (ID: 3)
✓ Added: 'The Pragmatic Programmer' (ID: 4)

---- All Books ---

```

RECOMMENDATIONS

- Related Exceptions
- Uncaught Exceptions
- User Uncaught Exceptions

24°C Sunny

File Edit Selection View Go Run Terminal Help ← →

Run and Debug

No customise Run and Debug
create a livereload file.

Debug using a terminal command or in an interactive shell.

```

265 # Search for books
266 print("\n---- Search Operations ---")
267 search_results = library.search_book("Python")
268 print(f"Search for 'Python': {len(search_results)} result(s)")
269 for book in search_results:
270     print(f" - {book}")

272 # Return books
273 print("\n---- Return Operations ---")
274 library.return_book(1)
275
276 # List available books again
277 print("\n---- Available Books (After Return) ---")
278 library.list_available_books()
279
280 print("\nTask 3 Analysis:")
281 print("- Demonstrates OOP with Book and Library classes")
282 print("- Uses loops for iteration and conditionals for validation")
283 print("- Implements CRUD operations (Create, Read, Update, Delete)")
284 print("- Error handling for edge cases (duplicate IDs, not found, etc.)\n")
285

```

RESULTS OUTPUT DEBUG CONSOLE TERMINAL PORTS JUPYTER

TASK 3: LIBRARY MANAGEMENT SYSTEM

```

---- Adding Books ---
✓ Added: 'Python Crash Course' (ID: 1)
✓ Added: 'Clean Code' (ID: 2)
✓ Added: 'Design Patterns' (ID: 3)
✓ Added: 'The Pragmatic Programmer' (ID: 4)

---- All Books ---

```

RECOMMENDATIONS

- Related Exceptions
- Uncaught Exceptions
- User Uncaught Exceptions

24°C Sunny

Task Description #4 (AI-Assisted Code Completion for Class-Based Attendance System)

Task: Use an AI tool to generate an attendance management class.

Prompt: “Generate a Python class to mark and display student attendance using loops.”

Expected Output:

- AI-generated attendance logic.
 - Correct display of attendance.
 - Test cases.

The screenshot shows a code editor interface with two identical windows side-by-side. Both windows display Python code for managing student attendance. The code includes methods for adding students, marking attendance as present or absent, and calculating attendance percentages. The right window includes an AI tool sidebar titled "AI-BASED CODE COMPLETION" which provides feedback and suggestions for the code.

```
286 # -----
287 # TASK A: AI-ASSISTED CODE COMPLETION - ATTENDANCE SYSTEM
288
289 class StudentAttendance:
290     """Manages student attendance tracking."""
291
292     def __init__(self):
293         self.attendance_records = {} # {student_name: [True/False, ...]}
294
295     def add_student(self, student_name):
296         """Add a new student to the attendance system."""
297         if student_name not in self.attendance_records:
298             self.attendance_records[student_name] = []
299             print(f"✓ Added student: '{student_name}'")
300         else:
301             print(f"Student '{student_name}' already exists!")
302
303     def mark_attendance(self, student_name, present):
304         """Mark a student as present (True) or absent (False)."""
305         if student_name not in self.attendance_records:
306             print("Error: Student '{student_name}' not found!")
307             return False
308
309         self.attendance_records[student_name].append(present)
310         status = "Present" if present else "Absent"
311
312     def get_attendance_percentage(self, student_name):
313         """Calculate attendance percentage for a student."""
314         if student_name not in self.attendance_records:
315             print("Error: Student '{student_name}' not found!")
316             return 0
317
318         records = self.attendance_records[student_name]
319         if len(records) == 0:
320             return 0
321
322         present_count = sum(1 for record in records if record)
323         return (present_count / len(records)) * 100
324
325     def display_attendance_report(self):
326         """Display attendance report for all students."""
327         if not self.attendance_records:
328             print("No attendance records found.")
329             return
330
331         print("\n" + "=" * 60)
332
333         print("PERFORMANCE REPORT")
334
335         print("\n" + "=" * 60)
```


The screenshot shows the Microsoft Visual Studio Code interface with several tabs open:

- File
- Edit
- Selection
- View
- Go
- Run
- Terminal
- Help

Current file: `attendance_mark_classification.py`

```
def mark_attendance(attendance, students):
    # Mark attendance for multiple classes
    print("\n--- Marking Attendance (Class 1) ---")
    attendance.mark_attendance("Alice Johnson", True)
    attendance.mark_attendance("Bob Smith", True)
    attendance.mark_attendance("Carol White", False)
    attendance.mark_attendance("David Brown", True)

    print("\n--- Marking Attendance (Class 2) ---")
    attendance.mark_attendance("Alice Johnson", True)
    attendance.mark_attendance("Bob Smith", False)
    attendance.mark_attendance("Carol White", True)
    attendance.mark_attendance("David Brown", True)

    print("\n--- Marking Attendance (Class 3) ---")
    attendance.mark_attendance("Alice Johnson", True)
    attendance.mark_attendance("Bob Smith", True)
    attendance.mark_attendance("Carol White", False)
    attendance.mark_attendance("David Brown", False)

# Display full report
attendance.display_attendance_report()

# Get students above 80% threshold
print("\n--- Students with 80%+ Attendance ---")
qualified = attendance.get_students_above_threshold(80)
```

Bottom left pane: Breakpoints

- Raised Exceptions
- Uncaught Exceptions
- User Uncaught Exceptions

Bottom right pane: Taskbar

- Python Crash Course (ID: 1)
- Clean Code (ID: 2)
- Design Patterns (ID: 3)
- The Pragmatic Programmer (ID: 4)

Bottom center: Search bar

Bottom right: System tray icons

```

attendance.mark_attendance("Bob Smith", False)
attendance.mark_attendance("Carol White", True)
attendance.mark_attendance("David Brown", True)

print("\n--- Marking Attendance (Class 3) ---")
attendance.mark_attendance("Alice Johnson", True)
attendance.mark_attendance("Bob Smith", True)
attendance.mark_attendance("Carol White", False)
attendance.mark_attendance("David Brown", False)

# Display full report
attendance.display_attendance_report()

# Get students above 80% threshold
print("\n--- Students with 80%+ Attendance ---")
qualified = attendance.get_students_above_threshold(80)
for student_name, percentage in qualified:
    print(f" {student_name}: {percentage:.1f}%")

print("\nTask 4 Analysis:")
print("- Uses dictionaries to store student records")
print("- Loops iterate through attendance records for calculations")
print("- Conditionals check thresholds and validate data")
print("- Provides percentage-based statistics and filtering\n")

# Task 4: STUDENT ATTENDANCE SYSTEM
# Adding Students
# Added student: Alice Johnson
# Added student: Bob Smith
# Added student: Carol White
# Added student: David Brown

# Marking Attendance (Class 1)
# Marked Alice Johnson as Present
# Marked Bob Smith as Present
# Marked Carol White as Absent
# Marked David Brown as Present

# Marking Attendance (Class 2)
# Marked Alice Johnson as Present
# Marked Bob Smith as Absent
# Marked Carol White as Present
# Marked David Brown as Present

# Marking Attendance (Class 3)
# Marked Alice Johnson as Present
# Marked Bob Smith as Present
# Marked Carol White as Absent
# Marked David Brown as Absent

# ATTENDANCE REPORT
# Alice Johnson | Present: 3/3 | Attendance: 100.0%
# Bob Smith | Present: 2/3 | Attendance: 66.67%
# Carol White | Present: 2/3 | Attendance: 66.67%
# David Brown | Present: 2/3 | Attendance: 66.67%

# Students with 80% Attendance
# Alice Johnson: 100.0%

```

Task 4: STUDENT ATTENDANCE SYSTEM

- Adding Students
- Added student: Alice Johnson
- Added student: Bob Smith
- Added student: Carol White
- Added student: David Brown

Task 5: VOTING ELIGIBILITY CHECK

- All-generated conditional logic checking age (18-60) and citizenship.
- Multiple test cases with explanations.
- Clear error messages for each invalidity scenario.

Task 6: VOWEL & CONSONANT COUNTING

- Loop-based string processing using (for loops)
- Character classification (vowel, consonant, non-alphabetic).
- Test cases with various string types.
- Character-by-character analysis.

Task 7: LIBRARY MANAGEMENT SYSTEM

Task Description #5 (AI-Based Code Completion for Conditional Menu Navigation)

Task: Use an AI tool to complete a navigation menu.

Prompt: “Generate a Python program using loops and conditionals

to simulate an ATM menu."

Expected Output:

- AI-generated menu logic.
- Correct option handling.
- Output verification.

The screenshot shows a code editor window with Python code for an ATM simulator. The code includes methods for verifying a PIN, checking the balance, withdrawing money, and depositing money. The code is annotated with AI-generated comments and tasks. The interface includes a sidebar for breakpoints and a status bar at the bottom.

```
413 # =====
414 # TASK 5: AI-BASED CODE COMPLETION - ATM MENU NAVIGATION
415 #
416 class ATMSimulator:
417     """Simulates a basic ATM machine."""
418
419     def __init__(self, initial_balance=1000):
420         self.balance = initial_balance
421         self.transaction_history = []
422         self.pin = "1234" # Default PIN for testing
423
424     def verify_pin(self, pin):
425         """Verify the entered PIN."""
426         if pin == self.pin:
427             print("✓ PIN Verified successfully!")
428             return True
429         else:
430             print("✗ Incorrect PIN!")
431             return False
432
433     def check_balance(self):
434         """Display current balance."""
435         print("----- Account Balance -----")
436         print(f"Your current balance: ${self.balance:.2f}")
437         self.transaction_history.append(("Balance Check", 0, self.balance))
438         return self.balance
439
440     def withdraw(self, amount):
441         """Withdraw money from the account."""
442         if amount < 0:
443             print("✗ Withdrawal amount must be positive!")
444
445         return False
446         elif amount > self.balance:
447             print("✗ Insufficient funds! Available: ${self.balance:.2f}")
448             return False
449         else:
450             self.balance -= amount
451             self.transaction_history.append(("Withdrawal", -amount, self.balance))
452             print("✓ Successfully withdrawn: ${amount:.2f}")
453             print(f" Remaining balance: ${self.balance:.2f}")
454             return True
455
456     def deposit(self, amount):
457         """Deposit money into the account."""
458         if amount < 0:
459             print("✗ Deposit amount must be positive!")
460             return False
461         else:
462             self.balance += amount
463             self.transaction_history.append(("Deposit", amount, self.balance))
464             print("✓ Successfully deposited: ${amount:.2f}")
465             print(f" New balance: ${self.balance:.2f}")
466             return True
467
468     def transfer(self, amount, recipient):
469         """Transfer money to another account."""
470         if amount < 0:
471             print("✗ Transfer amount must be positive!")
472             return False
473         elif amount > self.balance:
474             print("✗ Insufficient funds! Available: ${self.balance:.2f}")
475             return False
476         else:
477             self.balance -= amount
478             self.transaction_history.append(("Transfer", -amount, self.balance))
479             recipient.deposit(amount)
480             print("✓ Transfer successful to account: ${recipient.pin}")
481             print(f" Remaining balance: ${self.balance:.2f}")
482             return True
```

This screenshot is identical to the one above, showing the same AI-generated Python code for an ATM simulator. The code includes methods for verifying a PIN, checking the balance, withdrawing money, and depositing money. The code is annotated with AI-generated comments and tasks. The interface includes a sidebar for breakpoints and a status bar at the bottom.

```
413 # =====
414 # TASK 5: AI-BASED CODE COMPLETION - ATM MENU NAVIGATION
415 #
416 class ATMSimulator:
417     """Simulates a basic ATM machine."""
418
419     def __init__(self, initial_balance=1000):
420         self.balance = initial_balance
421         self.transaction_history = []
422         self.pin = "1234" # Default PIN for testing
423
424     def verify_pin(self, pin):
425         """Verify the entered PIN."""
426         if pin == self.pin:
427             print("✓ PIN Verified successfully!")
428             return True
429         else:
430             print("✗ Incorrect PIN!")
431             return False
432
433     def check_balance(self):
434         """Display current balance."""
435         print("----- Account Balance -----")
436         print(f"Your current balance: ${self.balance:.2f}")
437         self.transaction_history.append(("Balance Check", 0, self.balance))
438         return self.balance
439
440     def withdraw(self, amount):
441         """Withdraw money from the account."""
442         if amount < 0:
443             print("✗ Withdrawal amount must be positive!")
444
445         return False
446         elif amount > self.balance:
447             print("✗ Insufficient funds! Available: ${self.balance:.2f}")
448             return False
449         else:
450             self.balance -= amount
451             self.transaction_history.append(("Withdrawal", -amount, self.balance))
452             print("✓ Successfully withdrawn: ${amount:.2f}")
453             print(f" Remaining balance: ${self.balance:.2f}")
454             return True
455
456     def deposit(self, amount):
457         """Deposit money into the account."""
458         if amount < 0:
459             print("✗ Deposit amount must be positive!")
460             return False
461         else:
462             self.balance += amount
463             self.transaction_history.append(("Deposit", amount, self.balance))
464             print("✓ Successfully deposited: ${amount:.2f}")
465             print(f" New balance: ${self.balance:.2f}")
466             return True
467
468     def transfer(self, amount, recipient):
469         """Transfer money to another account."""
470         if amount < 0:
471             print("✗ Transfer amount must be positive!")
472             return False
473         elif amount > self.balance:
474             print("✗ Insufficient funds! Available: ${self.balance:.2f}")
475             return False
476         else:
477             self.balance -= amount
478             self.transaction_history.append(("Transfer", -amount, self.balance))
479             recipient.deposit(amount)
480             print("✓ Transfer successful to account: ${recipient.pin}")
481             print(f" Remaining balance: ${self.balance:.2f}")
482             return True
```

```
416     class ATMsimulator:
417         def transfer(self, amount, recipient):
418             if amount < 0:
419                 return False
420             else:
421                 self.balance -= amount
422                 self.transaction_history.append(
423                     ("Transfer to " + recipient, -amount, self.balance))
424             print(f"\n✓ Successfully transferred: ${amount:.2f} to {recipient}")
425             print(f"\nRemaining balance: ${self.balance:.2f}")
426             return True
427
428     def view_transaction_history(self):
429         """Display transaction history."""
430         if not self.transaction_history:
431             print("No transactions yet.")
432             return
433
434         print("\n--- Transaction History ---")
435         for i, (transaction_type, amount, balance_after) in enumerate(
436             self.transaction_history, 1
437         ):
438             print(f"({i}). {transaction_type}: Amount: ${abs(amount):>8.2f} | "
439                  f"Balance: ${balance_after:>8.2f}")
440
441     def change_pin(self, old_pin, new_pin):
442         """Change the account PIN."""
443         if old_pin != self.pin:
444             print("X Current PIN is incorrect!")
445             return False
446         elif len(new_pin) < 4:
447             print("X New PIN must be at least 4 digits!")
448             return False
449         else:
450             self.pin = new_pin
451             print(f"\n✓ PIN changed successfully!")
452             return True
453
454     def run_atm_menu(self):
455         """Run the interactive ATM menu."""
456         print("\n" + "=" * 70)
457         print("WELCOME TO ATM SIMULATOR")
458         print("=" * 70)
459
460         # PIN verification
461         attempts = 0
462         max_attempts = 3
463
464         while attempts < max_attempts:
465             pin_input = input("\nEnter your PIN: ")
466             if self.verify_pin(pin_input):
467                 break
468             else:
469                 attempts += 1
470                 remaining = max_attempts - attempts
471                 if remaining > 0:
472                     print(f"Attempts remaining: {remaining}")
473                 else:
474                     print("X Card blocked after 3 failed attempts!")
475                     return
476
477         # Main menu loop
```

```
416     class ATMsimulator:
417         def change_pin(self, old_pin, new_pin):
418             if old_pin != self.pin:
419                 self.pin = new_pin
420                 print(f"\n✓ PIN changed successfully!")
421                 return True
422             else:
423                 self.pin = new_pin
424                 print(f"\n✓ PIN changed successfully!")
425                 return True
426
427         def run_atm_menu(self):
428             """Run the interactive ATM menu."""
429             print("\n" + "=" * 70)
430             print("WELCOME TO ATM SIMULATOR")
431             print("=" * 70)
432
433             # PIN verification
434             attempts = 0
435             max_attempts = 3
436
437             while attempts < max_attempts:
438                 pin_input = input("\nEnter your PIN: ")
439                 if self.verify_pin(pin_input):
440                     break
441                 else:
442                     attempts += 1
443                     remaining = max_attempts - attempts
444                     if remaining > 0:
445                         print(f"Attempts remaining: {remaining}")
446                     else:
447                         print("X Card blocked after 3 failed attempts!")
448                         return
449
450             # Main menu loop
```

The screenshot shows a Microsoft Visual Studio Code (VS Code) interface with the following details:

- File Bar:** File, Edit, Selection, View, Go, Run, Terminal, Help.
- Search Bar:** Q. ABC
- Left Sidebar:** RUN AND DEBUG (highlighted), RECENT FILES (chatbot_invert_classification.pyw, chatbot_invert_classification.py, delivery_complete.txt, delivery_summary.txt, completion.log), and a section for REPORTS (Raised Exceptions, Uncaught Exceptions, User Uncaught Exceptions).
- Code Editor:** The main area contains Python code for an ATM simulator. The code defines a class `ATMSimulator` with a `run_atm_menu` method. It prints a menu with options 1 through 6, followed by an exit option. It handles user input for each choice, including balance checks, withdrawal processing (with error handling for invalid amounts), and deposit processing.
- Right Sidebar:** Includes sections for RECENT FILES, RECENT FOLDERS, and RECENT WORKSPACES. A message at the top right says "I'll help you complete Experiment 5 on AI-Based Code Completion. Let me check the current state of the unmanaged file, then create a comprehensive solution-including all 5 tasks." Below it are sections for Task 1: Voting Eligibility Check, Task 2: Vocal & Consonant Counting, and Task 3: Library Management System.
- Status Bar:** Shows the file path (un473.pyw), line number (47), column (79), space count (8), and file type (Python). It also shows the status bar with icons for battery, signal, and time (23:01 2020 11:33).

The screenshot shows the Microsoft Visual Studio Code interface with the following details:

- File Explorer:** Shows a tree view with files like `shatil_held_classification.ipynb`, `shatil_inert_classifications.ipynb`, `00_DELIVERY_COMPLETE.txt`, `DELIVERY_SUMMARY.txt`, and `COMPLETION.CHE`.
- Code Editor:** Displays Python code for an ATM Simulator. The code handles various menu options including balance inquiry, transfer, transaction history, pin change, and card ejection.
- Output:** Shows the status "Task 5: Test ATM System (Non-interactive demonstration) completed".
- Terminal:** Shows the command "python atm.py" and its output.
- Status Bar:** Provides information about the file (`413.py`), line count (792), and other system details.

The screenshot shows a Microsoft Visual Studio Code (VS Code) window with the following details:

- File Explorer:** Shows a folder structure for 'chatter_invert_classification' containing files like 'chatter_invert_classification.py', 'chatter_invert_classification.ipynb', and 'COMPLETION LOG'.
- Code Editor:** Displays the 'chatter_invert_classification.py' file with the following content:

```
def __init__(self, initial_balance=2500):
    self.balance = initial_balance

    def withdraw(amount):
        if amount > self.balance:
            print("Insufficient funds")
            return False
        else:
            self.balance -= amount
            print(f"Withdrawal successful. New balance: {self.balance}")
            return True

    def deposit(amount):
        self.balance += amount
        print(f"Deposited {amount}. New balance: {self.balance}")

    def transfer(amount, recipient):
        if amount > self.balance:
            print("Insufficient funds")
            return False
        else:
            self.balance -= amount
            recipient.deposit(amount)
            print(f"Transfer to {recipient.name} successful. Your new balance: {self.balance}")
            return True
```
- Output Panel:** Shows the results of the 'analyze' command, including:
 - Task 1: Writing Eligibility Check
 - All generated conditional logic checking age (≥ 18) and citizenship.
 - Multiple test cases with explanations.
 - Clear error messages for each invalidity scenario.
 - Task 2: Vowel & Consonant Counting
 - String-based processing using for loops.
 - Character classification (vowel, consonant, non-alphabetic).
 - Test cases with various string types.
 - Character-by-character analysis.
 - Task 3: Library Management System
 - Object-based processing using for loops.
 - Character classification (vowel, consonant, non-alphabetic).
 - Test cases with various string types.
 - Character-by-character analysis.
- Bottom Status Bar:** Shows file paths, line numbers (14-79), column numbers (1-100), and other status information.

The screenshot shows a Python script named `atm.py` running in a terminal window of an IDE. The code simulates an ATM's menu system, including balance checking, withdrawal, deposit, transfer, and transaction history viewing. It also includes a section for Task 5 analysis. The right side of the interface displays a comprehensive solution for Experiment 5, which includes generated conditional logic, loops, and inheritance, along with multiple class representations and clear error messages for each ineligibility scenario. Below the main code area, there are sections for Breakpoints, Run and Debug, and a summary of the assignment solution.

```
# Simulate operations
print("\n1. Checking balance...")
atm.check_balance()

print("\n2. Withdrawing $200...")
atm.withdraw(200)

print("\n3. Depositing $500...")
atm.deposit(500)

print("\n4. Transferring $150 to John...")
atm.transfer(150, "John")

print("\n5. Withdrawing $400...")
atm.withdraw(400)

print("\n6. Attempting invalid withdrawal...")
atm.withdraw(5000)

print("\n7. Viewing transaction history...")
atm.view_transaction_history()

print(f"\nFinal balance: ${atm.balance:.2f}")

# Task 5 Analysis:
print("- Uses while loops for menu navigation")
print("- Conditional statements handle menu options and validation")
print("- Error handling for invalid inputs and edge cases")
print("- Transaction history provides audit trail")
print("- PIN verification provides security layer")
```

The screenshot shows a Windows desktop environment with a code editor window open. The title bar of the window reads "File Edit Selection View Go Run Terminal Help". The main pane displays a terminal session titled "TASK 5: ATM MENU SIMULATION (Non-Interactive Demonstration)". The session logs the following transactions:

```

... Simulating ATM Operations ...
Starting Balance: $2500.00
1. Checking balance...
Account Balance: $2500.00
Your current balance: $2500.00
2. Withdrawing $200...
✓ Successfully withdrawn: $200.00
Remaining balance: $2300.00
3. Depositing $50...
✓ Successfully deposited: $50.00
New balance: $2350.00
4. Transferring $150 to John...
✓ Successfully transferred: $150.00 to John
Remaining balance: $2200.00
5. Withdrawing $400...
✓ Successfully withdrawn: $400.00
Remaining balance: $2250.00
6. Attempting invalid withdrawal...
✗ Insufficient funds! Available: $2250.00
7. Viewing transaction history...
Transaction History ...
1. Balance Check | Amount: $ 0.00 | Balance: $2500.00
2. Withdrawal | Amount: $ 200.00 | Balance: $2300.00
3. Deposit | Amount: $ 50.00 | Balance: $2350.00
4. Transfer to John | Amount: $ 150.00 | Balance: $2200.00
5. Withdrawal | Amount: $ 400.00 | Balance: $2250.00
Final balance: $2250.00

```

The sidebar on the right contains several sections:

- AI-BASED CODE COMPLETION**: A note from AI stating it will help complete Experiment 6.
- Task 1: Voting Eligibility Check**: Includes bullet points about AI-generated logic for age and citizenship checks.
- Task 2: Vowel & Consonant Counting**: Includes bullet points about loop-based string processing and character classification.
- Task 3: Library Management System**: A small preview window showing a search interface.

Summary of the Assignment Solution:

Task 1: Voting Eligibility Check

- AI-generated conditional logic checking age (≥ 18) and citizenship
- Multiple test cases with explanations
- Clear error messages for each ineligibility scenario

Task 2: Vowel & Consonant Counting

- Loop-based string processing using for loops
- Character classification (vowel, consonant, non-alphabetic)
- Test cases with various string types
- Character-by-character analysis

Task 3: Library Management System

- Complete OOP implementation with Book and Library classes
- Features: add, remove, search, checkout, return books
- Demonstrates CRUD operations and error handling
- Real-world example of classes + loops + conditionals

Task 4: Student Attendance System

- StudentAttendance class for tracking attendance
- Methods to add students, mark attendance, calculate percentages
- Attendance report with statistics
- Filtering students by attendance threshold (80%+)

Task 5: ATM Menu Simulation

- ATMSimulator class with full banking operations
- Operations: balance check, withdraw, deposit, transfer
- PIN verification and transaction history
- Non-interactive demonstration with realistic scenarios

Plus:

- Comprehensive documentation and explanations for each task
- Test cases demonstrating correctness
- Analysis of each solution's key features

- **Reflection on AI-Assisted Coding** covering strengths, improvements, ethical considerations, and best practices

The file is ready to run and demonstrates all the learning objectives (LO1-LO5) with proper code interpretation, error identification, optimization, and ethical considerations.