

# ASSIGNMENT-6.5

NAME: ARCHITHA

ROLLNO: 2306A91001

BATCH: 30

TASK-1:

Prompt:

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

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

CODE:

```
1 # Program to check voting eligibility based on age and citizenship
2
3 age = int(input("Enter your age: "))
4 citizen = input("Are you a citizen? (yes/no): ").lower()
5
6 # Check eligibility conditions
7 if age >= 18 and citizen == "yes":
8     print("You are eligible to vote.")
9 elif age < 18 and citizen == "yes":
10    print("You are not eligible to vote due to age.")
11 elif age >= 18 and citizen != "yes":
12    print("You are not eligible to vote due to citizenship.")
13 else:
14    print("You are not eligible to vote.")
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
(base) bhuvanviskamalla@Bhuvans-MacBook-Air AI % /usr/local/bin/python3 "/Users/bhuvanviskamalla/Desktop/AI/ai(6.5).py"
Enter your age: 20
Are you a citizen? (yes/no): yes
You are eligible to vote.
```

## OBSERVATION:

- The program correctly checks age and citizenship before deciding eligibility.
- All possible cases are covered with clear conditional branches. • Output messages are descriptive and user-friendly.
- Runs efficiently in constant time  $O(1)$ .
- Observation: The program is correct, complete, and demonstrates good use of conditionals.

## TASK-2:

### Prompt:

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

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

## CODE:

```
22
23 # Program to count vowels and consonants in a string
24
25 text = input("Enter a string: ")
26
27 vowels = "aeiouAEIOU"
28 vowel_count = 0
29 consonant_count = 0
30
31 for ch in text:
32     if ch.isalpha():
33         if ch in vowels:
34             vowel_count += 1
35         else:
36             consonant_count += 1
37
38 print("Vowels:", vowel_count)
39 print("Consonants:", consonant_count)
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
(base) bhuvanviskamalla@Bhuvans-MacBook-Air AI % /usr/local/bin/python3 "/Users/bhuvanviskamalla/Desktop/AI/ai(6.5).py"
Enter a string: Hello World
Vowels: 3
Consonants: 7
(base) bhuvanviskamalla@Bhuvans-MacBook-Air AI %
```

## OBSERVATION:

- .The function accurately distinguishes vowels and consonants.. Non-alphabetic characters are ignored using `isalpha()`.
- .Output matches expected results (e.g., “Hello, World!” → 3 vowels, 7 consonants).
- .Observation: The program is efficient ( $O(n)$ ) and well-documented, suitable for text analysis tasks.

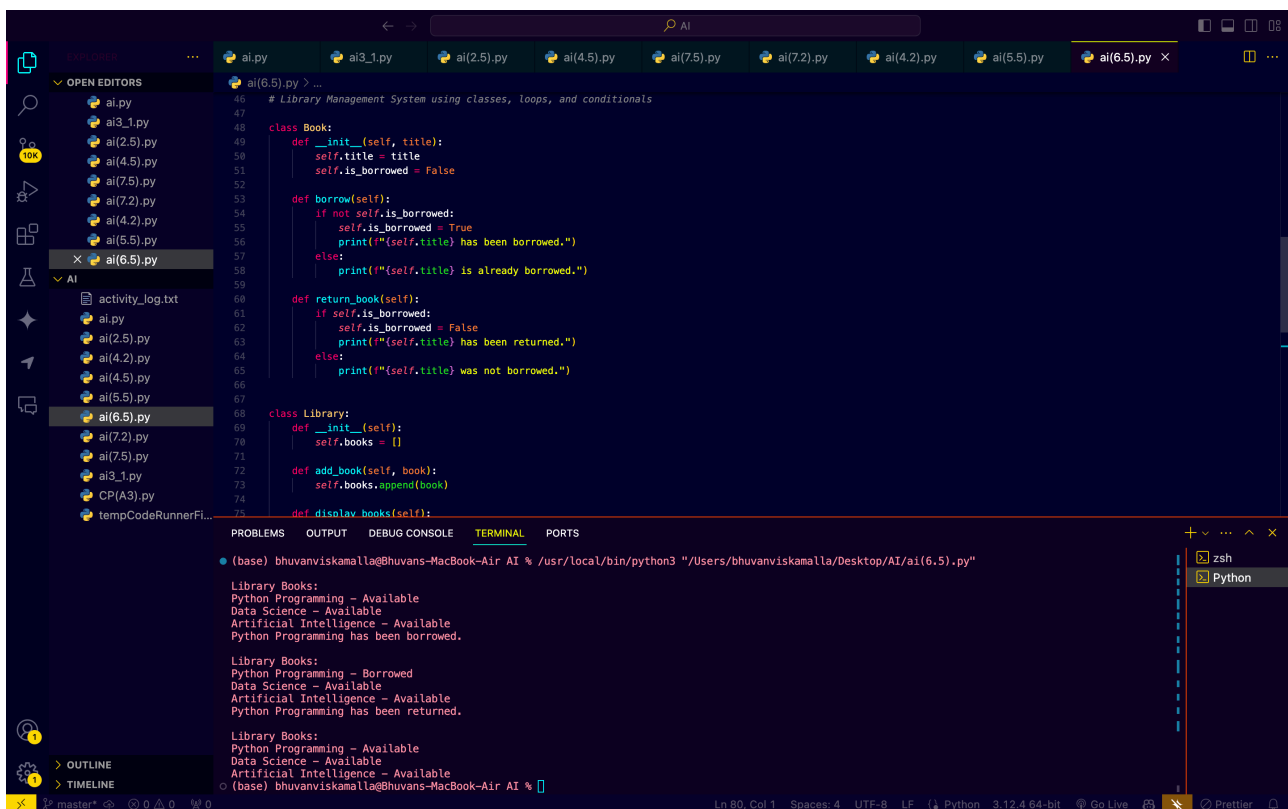
## TASK-3:

Prompt:

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

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

CODE:



```
ai(6.5).py > ...
# Library Management System using classes, loops, and conditionals

class Book:
    def __init__(self, title):
        self.title = title
        self.is_borrowed = False

    def borrow(self):
        if not self.is_borrowed:
            self.is_borrowed = True
            print(f"{self.title} has been borrowed.")
        else:
            print(f"{self.title} is already borrowed.")

    def return_book(self):
        if self.is_borrowed:
            self.is_borrowed = False
            print(f"{self.title} has been returned.")
        else:
            print(f"{self.title} was not borrowed.")

class Library:
    def __init__(self):
        self.books = []

    def add_book(self, book):
        self.books.append(book)

    def display_books(self):
        print("Library Books:")
        for book in self.books:
            print(f"{book.title} - {'Available' if not book.is_borrowed else 'Borrowed'}")
```

Terminal Output:

```
(base) bhuvanviskamalla@Bhuvans-MacBook-Air AI % /usr/local/bin/python3 "/Users/bhuvanviskamalla/Desktop/AI/ai(6.5).py"

Library Books:
Python Programming - Available
Data Science - Available
Artificial Intelligence - Available
Python Programming has been borrowed.

Library Books:
Python Programming - Borrowed
Data Science - Available
Artificial Intelligence - Available
Python Programming has been returned.

Library Books:
Python Programming - Available
Data Science - Available
Artificial Intelligence - Available
Python Programming has been returned.
```

OBSERVATION:

- Uses object-oriented programming with Book and Library classes. Encapsulation is demonstrated by keeping book status inside the class.

- Borrow/return logic prevents invalid operations.
- Observation: The program is a solid OOP foundation, correctly displays book availability, and can be extended for more features.

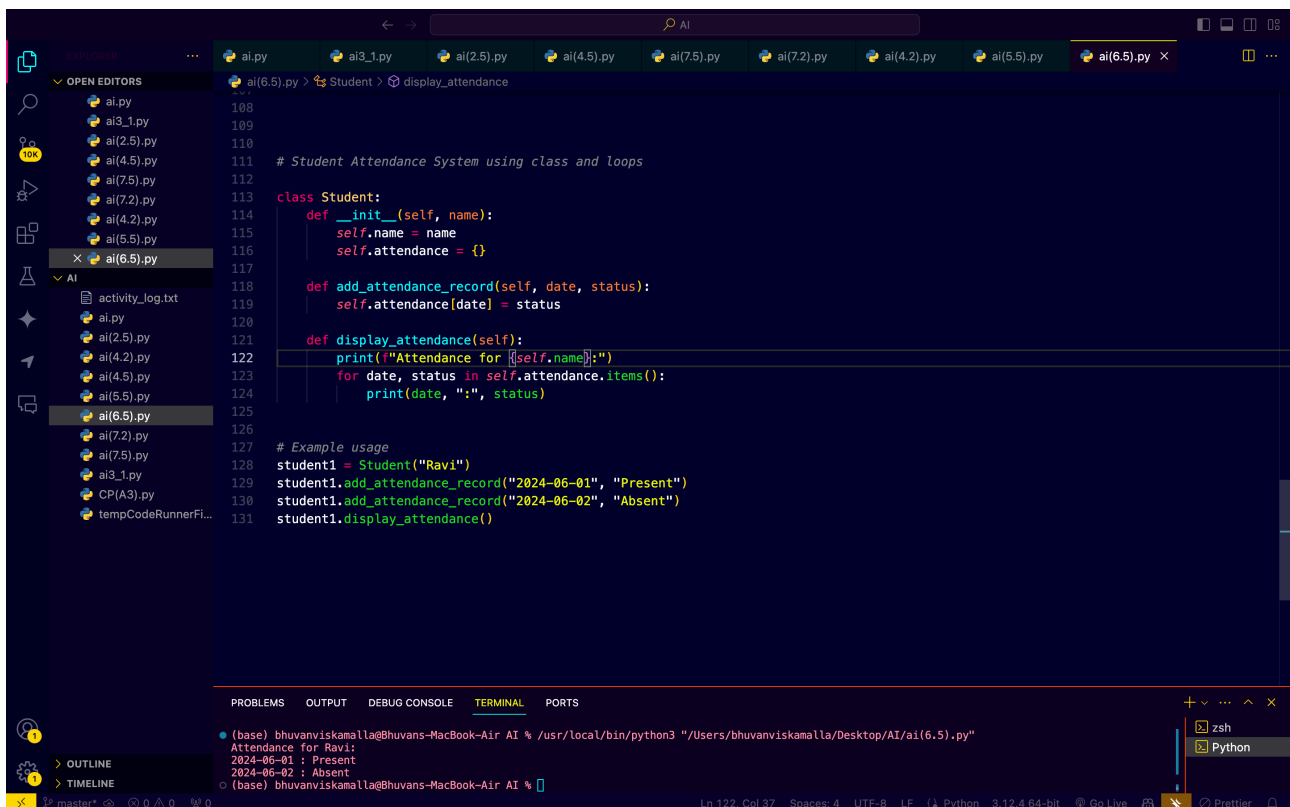
## TASK-4:

### Prompt:

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

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

### CODE:



```

108
109
110
111 # Student Attendance System using class and loops
112
113 class Student:
114     def __init__(self, name):
115         self.name = name
116         self.attendance = {}
117
118     def add_attendance_record(self, date, status):
119         self.attendance[date] = status
120
121     def display_attendance(self):
122         print(f"Attendance for {self.name}:")
123         for date, status in self.attendance.items():
124             print(date, ":", status)
125
126 # Example usage
127 student1 = Student("Ravi")
128 student1.add_attendance_record("2024-06-01", "Present")
129 student1.add_attendance_record("2024-06-02", "Absent")
130 student1.display_attendance()
131

```

Terminal Output:

```

(base) bhuvanviskamalla@bhuvals-MacBook-Air AI % /usr/local/bin/python3 "/Users/bhuvanviskamalla/Desktop/AI/ai(6.5).py"
Attendance for Ravi:
2024-06-01 : Present
2024-06-02 : Absent

```

## OBSERVATION:

- Each student object maintains attendance records in a dictionary.
  - `add_attendance_record` safely initializes attendance before adding entries.· Output correctly shows attendance for each student.
  - Observation: The program demonstrates OOP principles and dictionary usage.
- Minor caution: `set_attendance` may fail if attendance is still `None`.

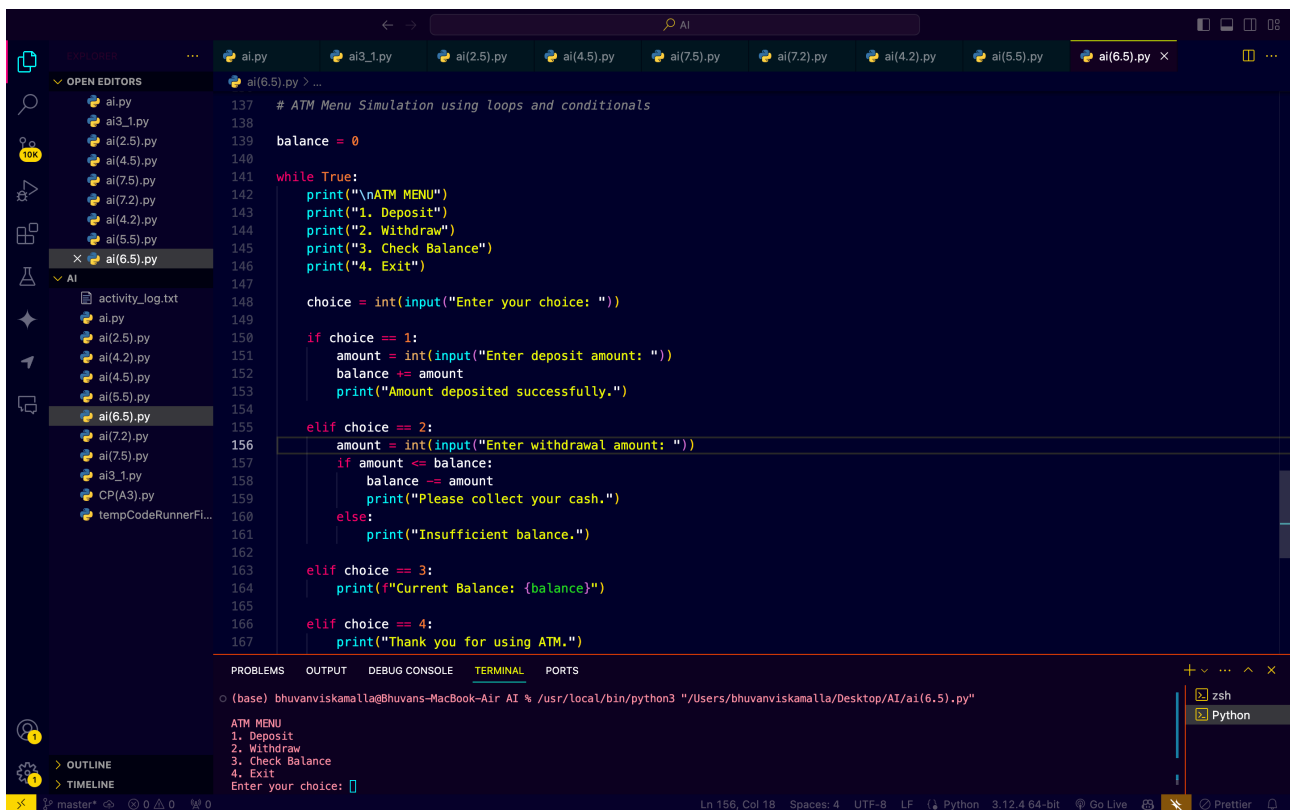
## TASK-5:

### Prompt:

“Generate a Python program using loops and conditionals to simulate an ATM menu.”

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

## CODE:



```
137 # ATM Menu Simulation using loops and conditionals
138
139 balance = 0
140
141 while True:
142     print("\nATM MENU")
143     print("1. Deposit")
144     print("2. Withdraw")
145     print("3. Check Balance")
146     print("4. Exit")
147
148     choice = int(input("Enter your choice: "))
149
150     if choice == 1:
151         amount = int(input("Enter deposit amount: "))
152         balance += amount
153         print("Amount deposited successfully.")
154
155     elif choice == 2:
156         amount = int(input("Enter withdrawal amount: "))
157         if amount <= balance:
158             balance -= amount
159             print("Please collect your cash.")
160         else:
161             print("Insufficient balance.")
162
163     elif choice == 3:
164         print(f"Current Balance: {balance}")
165
166     elif choice == 4:
167         print("Thank you for using ATM.")
168
169
170
171
172
173
174
175
176
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
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
```

```
(base) bhuvanviskamalla@bhuvans-MacBook-Air AI % /usr/local/bin/python3 "/Users/bhuvanviskamalla/Desktop/AI/ai(6.5).py"
ATM MENU
1. Deposit
2. Withdraw
3. Check Balance
4. Exit
Enter your choice: 1
```

## OBSERVATION:

- Implements deposit, withdraw, and balance check methods.
  - Menu-driven interface allows user interaction.
- Deposit/withdraw logic is correct; balance display needs a small fix (use f-

string).

.Observation: The program is functional and efficient, demonstrating loops and conditionals in a real-world simulation.