

2303A51739

Batch-25

Assignment-4.4

1. Sentiment Classification for Customer Reviews

Scenario:

An e-commerce platform wants to analyze customer reviews and classify

Week2

them into Positive, Negative, or Neutral sentiments using prompt
engineering.

Tasks:

- a) Prepare 6 short customer reviews mapped to sentiment labels.
- b) Design a Zero-shot prompt to classify sentiment.
- c) Design a One-shot prompt with one labeled example.
- d) Design a Few-shot prompt with 3–5 labeled examples.
- e) Compare the outputs and discuss accuracy differences.

The screenshot shows a Microsoft Visual Studio Code interface. On the left is the Explorer sidebar with files like 'ass 4.4.pdf', 'lab 4.3 word.docx', and 'lab 4.3 pdf'. The center has a terminal window titled 'AI Assisted' showing Python code for sentiment analysis. The right side has a 'CHAT' window with a message about generating Python code for customer reviews.

```
24     {
25         "review": "It's okay. Does what it's supposed to do, nothing special.",
26         "sentiment": "Neutral"
27     }
28 }
29 # Print reviews with sentiment labels
30 print(*x * 70)
31 print("Commerce Customer Reviews - Sentiment Analysis")
32 print(*x * 70)
33 print(*x * 70)
34
35 for idx, item in enumerate(reviews, 1):
36     print("\nReview #{}:".format(idx))
37     print("Text: {}".format(item['review']))
38     print("Sentiment: {}".format(item['sentiment']))
39     print(*x * 70)
40
41 print("Summary:")
42 positive_count = sum(1 for item in reviews if item['sentiment'] == "Positive")
43 negative_count = sum(1 for item in reviews if item['sentiment'] == "Negative")
44 neutral_count = sum(1 for item in reviews if item['sentiment'] == "Neutral")
45
46 print("Positive: {} | Negative: {} | Neutral: {}".format(positive_count, negative_count, neutral_count))
```

This screenshot shows the same Visual Studio Code environment. The terminal window now displays a different Python script for sentiment classification using keyword counts. The right side still shows the 'CHAT' window with the message about generating Python code for customer reviews.

```
54     def classify_sentiment(review):
55         """
56             # Classify based on keyword counts
57             if positive_count > negative_count:
58                 return "Positive"
59             elif negative_count > positive_count:
60                 return "Negative"
61             else:
62                 return "Neutral"
63
64
65         # Test the classifier with user input
66         print(*x * 70)
67         print("Customer Review Sentiment Classifier")
68         print(*x * 70)
69
70         user_review = input("\nEnter a customer review: ")
71         sentiment = classify_sentiment(user_review)
72
73         print("User Review: {}".format(user_review))
74         print("Classified Sentiment: {}".format(sentiment))
75
76         # Optional: test classifier on existing reviews
77         print(*x * 70)
78         print("Testing Classifier on Existing Reviews")
79         print(*x * 70)
```

Screenshot of the Visual Studio Code interface showing a Python script named `ass 4.4.py`. The code defines a function `classify_sentiment` that takes a review as input and returns Positive, Negative, or Neutral based on keyword counts. The code includes examples and a main program loop. The terminal shows the script running and printing results. The status bar indicates it's 12:09 PM on 1/29/2026.

```
File Edit Selection View Go Run Terminal Help AI Assisted

EXPLORER AI Assisted add.py ass 4.4.pdf lab 4.4.pdf check_lab_yearay lab 4.3 word.docx lab 4.3 word.pdf lab 4.3.pdf lab 3.4.pdf lab assignment 3.3.pdf lab assignment 1.4.pdf lab assignment 2.3.pdf leap_year.py

ass 4.4.py > ...
139 def classify_sentiment(review):
140     """
141         Classifies sentiment of a review using keyword-based logic.
142         Returns: Positive, Negative, or Neutral
143         """
144     review_lower = review.lower()
145
146     # Define keyword lists
147     positive_keywords = [
148         "amazing", "excellent", "great", "love", "wonderful", "fantastic", "perfect", "welt", "good", "ext", "satisfied",
149         "exceeded", "happy", "impressed", "quality", "value"
150     ]
151
152     negative_keywords = [
153         "terrible", "bad", "worst", "hate", "sour", "damaged", "unhelpful",
154         "not satisfied", "disappointed", "waste", "broken", "awful", "useless",
155         "disappointing", "issue", "problem", "defective", "unhappy"
156     ]
157
158     # Count keyword matches
159     positive_count = sum(1 for keyword in positive_keywords if keyword in review_lower)
160     negative_count = sum(1 for keyword in negative_keywords if keyword in review_lower)
161
162     # Classify based on keyword counts
163     if positive_count > negative_count:
164         return "Positive"
165     elif negative_count > positive_count:
166         return "Negative"
167     else:
168         return "Neutral"
169
170     # Main program
171     print("+" * 20)
172     print("Customer Review Sentiment Classifier")
173     print("+" * 20)
174
175     # Display existing reviews with classifications
176
177 PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
NameError: name 'review' is not defined
NameError: name 'review' is not defined
NameError: name 'review' is not defined
PS C:\Users\parva\OneDrive\Desktop\AI Assisted> & C:/Users/parva/AppData/Local/Programs/Python/Python313/python.exe "c:/users/parva/OneDrive/Desktop/AI Assisted/ass 4.4.py"
PS C:\Users\parva\OneDrive\Desktop\AI Assisted> & C:/Users/parva/AppData/Local/Programs/Python/Python313/python.exe "c:/users/parva/OneDrive/Desktop/AI Assisted/ass 4.4.py"
File "c:/users/parva/OneDrive/Desktop/AI Assisted/ass 4.4.py", line 56
    Classifies sentiment of a review using keyword-based logic.
                                         ^
IndentationError: unexpected indent
PS C:\Users\parva\OneDrive\Desktop\AI Assisted>

28°C Sunny
28°C
12:09
1/29/2026
1:28 PM
In 207. Col 46 Space 4 UFT-8 CR LF { Python 3.13.7

```

Screenshot of the Visual Studio Code interface showing a Python script named `ass 4.4.py`. The code adds user interaction to the sentiment classifier, allowing users to input reviews and see their predicted sentiment. The terminal shows the script running and interacting with the user. The status bar indicates it's 1:30 PM on 1/29/2026.

```
File Edit Selection View Go Run Terminal Help AI Assisted

EXPLORER AI Assisted add.py ass 4.4.pdf lab 4.4.pdf check_lab_yearay lab 4.3 word.docx lab 4.3 word.pdf lab 4.3.pdf lab 3.4.pdf lab assignment 3.3.pdf lab assignment 1.4.pdf lab assignment 2.3.pdf leap_year.py

ass 4.4.py > ...
276 print("Customer Review Sentiment Classifier")
277 print("+" * 20)
278
279 # Display example classifications
280 print("Example Classifications:")
281 print("+" * 20)
282 examples = [
283     "The product is excellent and works perfectly",
284     "The item is okay, not great",
285     "Very disappointed with the quality",
286     "Average experience overall"
287 ]
288
289 for example in examples:
290     print(classify_sentiment(example))
291     print("Predicted: (example)")
292     print("Sentiment: (sentiment)")
293
294 # Display existing reviews with classifications
295 print("+" * 20)
296 print("Existing Reviews Analysis")
297 print("+" * 20)
298
299 for idx, item in enumerate(reviews, 1):
300     predicted_sentiment = classify_sentiment(item["review"])
301     actual_sentiment = item["sentiment"]
302     match = "<" if predicted_sentiment == actual_sentiment else "X"
303
304     print(f"\nReview {idx}: {match}")
305     print(f"Text: {item['review']}")
306     print(f"Predicted: {predicted_sentiment} | Actual: {actual_sentiment}")
307
308 # Get user input and classify
309 print("+" * 20)
310 print("Classify Your Own Review")
311 print("+" * 20)
312
313 while True:
314     user_review = input("\nEnter a customer review (or 'quit' to exit): ")
315
316     if user_review.lower() == "quit":
317         print("Thank you for using the sentiment classifier!")
318         break
319
320     if user_review.strip():
321         sentiment = classify_sentiment(user_review)
322
323 DEBUGGING OUTPUT DEBUG CONSOLE TERMINAL PORTS
Classifies sentiment of a review using keyword-based logic.
IndentationError: unexpected indent
PS C:\Users\parva\OneDrive\Desktop\AI Assisted> 70
70
PS C:\Users\parva\OneDrive\Desktop\AI Assisted>

28°C
1:30
1/29/2026
In 203. Col 46 Space 4 UFT-8 CR LF { Python 3.13.7

```

The screenshot shows a dual-pane interface of VS Code. The left pane displays a Python script named 'ass 4-4.py' with code for sentiment analysis. The right pane shows a Jupyter Notebook cell titled 'SENTIMENT CLASSIFICATION FOR CUSTOMERS'. The notebook cell contains a snippet of Python code for testing a sentiment classifier on multiple reviews and prints the predicted sentiment for each review. A comment in the code indicates it's a short comment comparing zero-shot, one-shot, and few-shot logic approaches.

```

# Detailed Results:
# 000
# 001
# 002
# 003
# 004
# 005
# 006
# 007
# 008
# 009
# 010
# 011
# 012
# 013
# 014
# 015
# 016
# 017
# 018
# 019
# 020
# 021
# 022
# 023
# 024
# 025
# 026
# 027
# 028
# 029
# 030
# 031
# 032
# 033
# 034
# 035
# 036
# 037
# 038
# 039
# 040
# 041
# 042
# 043
# 044
# 045
# 046
# 047
# 048
# 049
# 050
# 051
# 052
# 053
# 054
# 055
# 056
# 057
# 058
# 059
# 060
# 061
# 062
# 063
# 064
# 065
# 066
# 067
# 068
# 069
# 070
# 071
# 072
# 073
# 074
# 075
# 076
# 077
# 078
# 079
# 080
# 081
# 082
# 083
# 084
# 085
# 086
# 087
# 088
# 089
# 090
# 091
# 092
# 093
# 094
# 095
# 096
# 097
# 098
# 099
# 100
# 101
# 102
# 103
# 104
# 105
# 106
# 107
# 108
# 109
# 110
# 111
# 112
# 113
# 114
# 115
# 116
# 117
# 118
# 119
# 120
# 121
# 122
# 123
# 124
# 125
# 126
# 127
# 128
# 129
# 130
# 131
# 132
# 133
# 134
# 135
# 136
# 137
# 138
# 139
# 140
# 141
# 142
# 143
# 144
# 145
# 146
# 147
# 148
# 149
# 150
# 151
# 152
# 153
# 154
# 155
# 156
# 157
# 158
# 159
# 160
# 161
# 162
# 163
# 164
# 165
# 166
# 167
# 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
# 559
# 560
# 561
# 562
# 563
# 564
# 565
# 566
# 567
# 568
# 569
# 570
# 571
# 572
# 573
# 574
# 575
# 576
# 577
# 578
# 579
# 579
# 580
# 581
# 582
# 583
# 584
# 585
# 586
# 587
# 588
# 589
# 589
# 590
# 591
# 592
# 593
# 594
# 595
# 596
# 597
# 598
# 599
# 599
# 600
# 601
# 602
# 603
# 604
# 605
# 606
# 607
# 608
# 609
# 609
# 610
# 611
# 612
# 613
# 614
# 615
# 616
# 617
# 618
# 619
# 619
# 620
# 621
# 622
# 623
# 624
# 625
# 626
# 627
# 628
# 629
# 629
# 630
# 631
# 632
# 633
# 634
# 635
# 636
# 637
# 638
# 639
# 639
# 640
# 641
# 642
# 643
# 644
# 644
# 645
# 646
# 647
# 648
# 649
# 649
# 650
# 651
# 652
# 653
# 654
# 655
# 656
# 657
# 658
# 659
# 659
# 660
# 661
# 662
# 663
# 664
# 665
# 666
# 667
# 668
# 669
# 669
# 670
# 671
# 672
# 673
# 674
# 675
# 676
# 677
# 678
# 679
# 679
# 680
# 681
# 682
# 683
# 684
# 685
# 686
# 687
# 688
# 689
# 689
# 690
# 691
# 692
# 693
# 694
# 695
# 696
# 697
# 698
# 699
# 699
# 700
# 701
# 702
# 703
# 704
# 705
# 706
# 707
# 708
# 709
# 709
# 710
# 711
# 712
# 713
# 714
# 715
# 716
# 717
# 718
# 719
# 719
# 720
# 721
# 722
# 723
# 724
# 725
# 726
# 727
# 728
# 729
# 729
# 730
# 731
# 732
# 733
# 734
# 735
# 736
# 737
# 738
# 739
# 739
# 740
# 741
# 742
# 743
# 744
# 745
# 746
# 747
# 748
# 749
# 749
# 750
# 751
# 752
# 753
# 754
# 755
# 756
# 757
# 758
# 759
# 759
# 760
# 761
# 762
# 763
# 764
# 765
# 766
# 767
# 768
# 769
# 769
# 770
# 771
# 772
# 773
# 774
# 775
# 776
# 777
# 778
# 779
# 779
# 780
# 781
# 782
# 783
# 784
# 785
# 786
# 787
# 788
# 789
# 789
# 790
# 791
# 792
# 793
# 794
# 795
# 796
# 797
# 798
# 799
# 799
# 800
# 801
# 802
# 803
# 804
# 805
# 806
# 807
# 808
# 809
# 809
# 810
# 811
# 812
# 813
# 814
# 815
# 816
# 817
# 818
# 819
# 819
# 820
# 821
# 822
# 823
# 824
# 825
# 826
# 827
# 828
# 829
# 829
# 830
# 831
# 832
# 833
# 834
# 835
# 836
# 837
# 838
# 839
# 839
# 840
# 841
# 842
# 843
# 844
# 844
# 845
# 846
# 847
# 848
# 849
# 849
# 850
# 851
# 852
# 853
# 854
# 855
# 856
# 857
# 858
# 859
# 859
# 860
# 861
# 862
# 863
# 864
# 865
# 866
# 867
# 868
# 869
# 869
# 870
# 871
# 872
# 873
# 874
# 875
# 876
# 877
# 878
# 879
# 879
# 880
# 881
# 882
# 883
# 884
# 885
# 886
# 887
# 888
# 889
# 889
# 890
# 891
# 892
# 893
# 894
# 895
# 896
# 897
# 898
# 899
# 899
# 900
# 901
# 902
# 903
# 904
# 905
# 906
# 907
# 908
# 909
# 909
# 910
# 911
# 912
# 913
# 914
# 915
# 916
# 917
# 918
# 919
# 919
# 920
# 921
# 922
# 923
# 924
# 925
# 926
# 927
# 928
# 929
# 929
# 930
# 931
# 932
# 933
# 934
# 935
# 936
# 937
# 938
# 939
# 939
# 940
# 941
# 942
# 943
# 944
# 944
# 945
# 946
# 947
# 948
# 949
# 949
# 950
# 951
# 952
# 953
# 954
# 955
# 956
# 957
# 958
# 959
# 959
# 960
# 961
# 962
# 963
# 964
# 965
# 966
# 967
# 968
# 969
# 969
# 970
# 971
# 972
# 973
# 974
# 975
# 976
# 977
# 978
# 979
# 979
# 980
# 981
# 982
# 983
# 984
# 985
# 986
# 987
# 988
# 989
# 989
# 990
# 991
# 992
# 993
# 994
# 995
# 996
# 997
# 998
# 999
# 999
# 1000
# 1001
# 1002
# 1003
# 1004
# 1005
# 1006
# 1007
# 1008
# 1009
# 1009
# 1010
# 1011
# 1012
# 1013
# 1014
# 1015
# 1016
# 1017
# 1017
# 1018
# 1019
# 1019
# 1020
# 1021
# 1022
# 1023
# 1024
# 1025
# 1026
# 1027
# 1027
# 1028
# 1029
# 1029
# 1030
# 1031
# 1032
# 1033
# 1034
# 1035
# 1036
# 1037
# 1037
# 1038
# 1039
# 1039
# 1040
# 1041
# 1042
# 1043
# 1044
# 1044
# 1045
# 1046
# 1047
# 1048
# 1049
# 1049
# 1050
# 1051
# 1052
# 1053
# 1054
# 1055
# 1056
# 1057
# 1058
# 1058
# 1059
# 1060
# 1061
# 1062
# 1063
# 1064
# 1065
# 1066
# 1067
# 1067
# 1068
# 1069
# 1069
# 1070
# 1071
# 1072
# 1073
# 1074
# 1075
# 1075
# 1076
# 1077
# 1077
# 1078
# 1079
# 1079
# 1080
# 1081
# 1082
# 1083
# 1084
# 1084
# 1085
# 1086
# 1086
# 1087
# 1088
# 1088
# 1089
# 1090
# 1090
# 1091
# 1092
# 1092
# 1093
# 1094
# 1094
# 1095
# 1096
# 1096
# 1097
# 1098
# 1098
# 1099
# 1099
# 1100
# 1101
# 1101
# 1102
# 1103
# 1103
# 1104
# 1105
# 1105
# 1106
# 1107
# 1107
# 1108
# 1109
# 1109
# 1110
# 1111
# 1111
# 1112
# 1113
# 1113
# 1114
# 1115
# 1115
# 1116
# 1117
# 1117
# 1118
# 1119
# 1119
# 1120
# 1121
# 1121
# 1122
# 1123
# 1123
# 1124
# 1125
# 1125
# 1126
# 1127
# 1127
# 1128
# 1129
# 1129
# 1130
# 1131
# 1131
# 1132
# 1133
# 1133
# 1134
# 1135
# 1135
# 1136
# 1137
# 1137
# 1138
# 1139
# 1139
# 1140
# 1141
# 1141
# 1142
# 1143
# 1143
# 1144
# 1145
# 1145
# 1146
# 1147
# 1147
# 1148
# 1149
# 1149
# 1150
# 1151
# 1151
# 1152
# 1153
# 1153
# 1154
# 1155
# 1155
# 1156
# 1157
# 1157
# 1158
# 1159
# 1159
# 1160
# 1161
# 1161
# 1162
# 1163
# 1163
# 1164
# 1165
# 1165
# 1166
# 1167
# 1167
# 1168
# 1169
# 1169
# 1170
# 1171
# 1171
# 1172
# 1173
# 1173
# 1174
# 1175
# 1175
# 1176
# 1177
# 1177
# 1178
# 1179
# 1179
# 1180
# 1181
# 1181
# 1182
# 1183
# 1183
# 1184
# 1185
# 1185
# 1186
# 1187
# 1187
# 1188
# 1189
# 1189
# 1190
# 1191
# 1191
# 1192
# 1193
# 1193
# 1194
# 1195
# 1195
# 1196
# 1197
# 1197
# 1198
# 1199
# 1199
# 1200
# 1201
# 1201
# 1202
# 1203
# 1203
# 1204
# 1205
# 1205
# 1206
# 1207
# 1207
# 1208
# 1209
# 1209
# 1210
# 1211
# 1211
# 1212
# 1213
# 1213
# 1214
# 1215
# 1215
# 1216
# 1217
# 1217
# 1218
# 1219
# 1219
# 1220
# 1221
# 1221
# 1222
# 1223
# 1223
# 1224
# 1225
# 1225
# 1226
# 1227
# 1227
# 1228
# 1229
# 1229
# 1230
# 1231
# 1231
# 1232
# 1233
# 1233
# 1234
# 1235
# 1235
# 1236
# 1237
# 1237
# 1238
# 1239
# 1239
# 1240
# 1241
# 1241
# 1242
# 1243
# 1243
# 1244
# 1245
# 1245
# 1246
# 1247
# 1247
# 1248
# 1249
# 1249
# 1250
# 1251
# 1251
# 1252
# 1253
# 1253
# 1254
# 1255
# 1255
# 1256
# 1257
# 1257
# 1258
# 1259
# 1259
# 1260
# 1261
# 1261
# 1262
# 1263
# 1263
# 1264
# 1265
# 1265
# 1266
# 1267
# 1267
# 1268
# 1269
# 1269
# 1270
# 1271
# 1271
# 1272
# 1273
# 1273
# 1274
# 1275
# 1275
# 1276
# 1277
# 1277
# 1278
# 1279
# 1279
# 1280
# 1281
# 1281
# 1282
# 1283
# 1283
# 1284
# 1285
# 1285
# 1286
# 1287
# 1287
# 1288
# 1289
# 1289
# 1290
# 1291
# 1291
# 1292
# 1293
# 1293
# 1294
# 1295
# 1295
# 1296
# 1297
# 1297
# 1298
# 1299
# 1299
# 1300
# 1301
# 1301
# 1302
# 1303
# 1303
# 1304
# 1305
# 1305
# 1306
# 1307
# 1307
# 1308
# 1309
# 1309
# 1310
# 1311
# 1311
# 1312
# 1313
# 1313
# 1314
# 1315
# 1315
# 1316
# 1317
# 1317
# 1318
# 1319
# 1319
# 1320
# 1321
# 1321
# 1322
# 1323
# 1323
# 1324
# 1325
# 1325
# 1326
# 1327
# 1327
# 1328
# 1329
# 1329
# 1330
# 1331
# 1331
# 1332
# 1333
# 1333
# 1334
# 1335
# 1335
# 1336
# 1337
# 1337
# 1338
# 1339
# 1339
# 1340
# 1341
# 1341
# 1342
# 1343
# 1343
# 1344
# 1345
# 1345
# 1346
# 1347
# 1347
# 1348
# 1349
# 1349
# 1350
# 1351
# 1351
# 1352
# 1353
# 1353
# 1354
# 1355
# 1355
# 1356
# 1357
# 1357
# 1358
# 1359
# 1359
# 1360
# 1361
# 1361
# 1362
# 1363
# 1363
# 1364
# 1365
# 1365
# 1366
# 1367
# 1367
# 1368
# 1369
# 1369
# 1370
# 1371
# 1371
# 1372
# 1373
# 1373
# 1374
# 1375
# 1375
# 1376
# 1377
# 1377
# 1378
# 1379
# 1379
# 1380
# 1381
# 1381
# 1382
# 1383
# 1383
# 1384
# 1385
# 1385
# 1386
# 1387
# 1387
# 1388
# 1389
# 1389
# 1390
# 1391
# 1391
# 1392
# 1393
# 1393
# 1394
# 1395
# 1395
# 1396
# 1397
# 1397
# 1398
# 1399
# 1399
# 1400
# 1401
# 1401
# 1402
# 1403
# 1403
# 1404
# 1405
# 1405
# 1406
# 1407
# 1407
# 1408
# 1409
# 1409
# 1410
# 1411
# 1411
# 1412
# 1413
# 1413
# 1414
# 1415
# 1415
# 1416
# 1417
# 1417
# 1418
# 1419
# 1419
# 1420
# 1421
# 1421
# 1422
# 1423
# 1423
# 1424
# 1425
# 1425
# 1426
# 1427
# 1427
# 1428
# 1429
# 1429
# 1430
# 1431
# 1431
# 1432
# 1433
# 1433
# 1434
# 1435
# 1435
# 1436
# 1437
# 1437
# 1438
# 1439
# 1439
# 1440
# 1441
# 1441
# 1442
# 1443
# 1443
# 1444
# 1445
# 1445
# 1446
# 1447
# 1447
# 1448
# 1449
# 1449
# 1450
# 1451
# 1451
# 1452
# 1453
# 1453
# 1454
# 1455
# 1455
# 1456
# 1457
# 1457
# 1458
# 1459
# 1459
# 1460
# 1461
# 1461
# 1462
# 1463
# 1463
# 1464
# 1465
# 1465
# 1466
# 1467
# 1467
# 1468
# 1469
# 1469
# 1470
# 1471
# 1471
# 1472
# 1473
# 1473
# 1474
# 1475
# 1475
# 1476
# 1477
# 1477
# 1478
# 1479
# 1479
# 1480
# 1481
# 1481
# 1482
# 1483
# 1483
# 1484
# 1485
# 1485
# 1486
# 1487
# 1487
# 1488
```

File Edit Selection View Go Run Terminal Help

AI Asst

EXPLORER AI ASSISTED add.py ass 4.pdf ass 4.py check_leap_year.py lab 4.3 word.doc lab 4.3 word.pdf lab 4.3 word.py lab 4.3.py lab assignment 3.3.pdf lab assignment 1.4.pdf lab assignment 2.3.pdf leap_year.py

```
ass 4.py >
1 # Simple list of email tuples: (subject, body, priority)
2 emails = [
3     ("Urgent: System Outage - Immediate Action Required", "The main database server is down. All operations are halted.", "High"),
4     ("Q1 Budget Review Meeting - Next Friday at 2 PM", "Please review the attached budget documents.", "Medium"),
5     ("Office Lunch - Catering Menu for Next week", "Please vote on your preferred lunch option.", "Low"),
6     ("Client Presentation Delayed - Decision Needed Today", "Our major client has requested to reschedule the presentation.", "High"),
7     ("Monthly Team Updates - Please Submit by End of Week", "Your monthly progress report by Friday.", "Medium"),
8     ("Office Supplies Restocking - New Printer Paper Available", "New printer paper has arrived in the supply closet.", "Low")
9 ]
10
11 # Print emails with priority
12 print("OFFICE EMAILS")
13 print("-" * 80)
14
15 for idx, (subject, body, priority) in enumerate(emails, 1):
16     print(f"\nEmail {idx}: {subject} [{priority}] Priority")
17     print(f"Subject: {subject}")
18     print(f"Body: {body}")
19
20 # Summary
21 print("\nHigh: " + "=" * 80)
22 print(f"High: {sum(1 for e in emails if e[2] == 'High')} | Medium: {sum(1 for e in emails if e[2] == 'Medium')} | Low: {sum(1 for e in emails if e[2] == 'Low')}"
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

Body: The main database server is down. All operations are halted.
 Email #2 (Medium Priority)
 Subject: Q1 Budget Review Meeting - Next Friday at 2 PM
 Body: Please review the attached budget documents.
 Email #3 (Low Priority)
 Subject: Office Lunch - Catering Menu for Next week
 Body: Please vote on your preferred lunch option.
 Email #4 (High Priority)
 Subject: Critical! Client Presentation Delayed - Decision Needed Today
 Body: Our major client has requested to reschedule the presentation.
 Email #5 (Medium Priority)
 Subject: Monthly Team Updates - Please Submit by End of Week
 Body: Submit your monthly progress report by Friday.
 Email #6 (Low Priority)
 Subject: Office Supplies Restocking - New Printer Paper Available
 Body: New printer paper has arrived in the supply closet.

 High: 2 | Medium: 2 | Low: 2
 PS C:\Users\parva\Desktop\AI Asst>

28°C

File Edit Selection View Go Run Terminal Help

AI Asst

EXPLORER AI ASSISTED add.py ass 4.pdf ass 4.py check_leap_year.py lab 4.3 word.doc lab 4.3 word.pdf lab 4.3 word.py lab 4.3.py lab assignment 3.3.pdf lab assignment 1.4.pdf lab assignment 2.3.pdf leap_year.py

File Edit Selection View Go Run Terminal Help

AI Asst

EXPLORER AI ASSISTED add.py ass 4.pdf ass 4.py check_leap_year.py lab 4.3 word.doc lab 4.3 word.pdf lab 4.3 word.py lab 4.3.py lab assignment 3.3.pdf lab assignment 1.4.pdf lab assignment 2.3.pdf leap_year.py

```
ass 4.py >
1 # Simple priority classifier function
2 def classify_priority(subject, body):
3     """Classify email priority using basic keywords"""
4     text = (subject + " " + body).lower()
5
6     if any(word in text for word in ["urgent", "critical", "immediate", "outage", "emergency"]):
7         return "High"
8     elif any(word in text for word in ["important", "meeting", "review", "deadline", "required"]):
9         return "Medium"
10    else:
11        return "Low"
12
13 # Test classifier on sample emails
14 print("PRIORITY CLASSIFIER TEST")
15 print("-" * 80)
16
17 for idx, (subject, body, actual_priority) in enumerate(emails, 1):
18     predicted = classify_priority(subject, body)
19     match "X" if predicted == actual_priority else "X"
20     print(f"\nEmail {idx}: {subject} [{predicted}] Actual: {actual_priority}")
21     print(f"Subject: {subject}")
22     print(f"Body: {body}")
23
24 # Test with custom email
25 print("\nEnter email subject: ")
26 custom_subject = input("Enter email subject: ")
27 custom_body = input("Enter email body: ")
28 result = classify_priority(custom_subject, custom_body)
29 print(f"Classified Priority: ({result})")
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

Email #1 (Low Priority)
 Subject: Office Lunch - Catering Menu for Next week
 Body: Please vote on your preferred lunch option.
 Email #4 (High Priority)
 Subject: Critical! Client Presentation Delayed - Decision Needed Today
 Body: Our major client has requested to reschedule the presentation.
 Email #5 (Medium Priority)
 Subject: Monthly Team Updates - Please Submit by End of Week
 Body: Submit your monthly progress report by Friday.
 Email #6 (Low Priority)
 Subject: Office Supplies Restocking - New Printer Paper Available
 Body: New printer paper has arrived in the supply closet.

 High: 2 | Medium: 2 | Low: 2
 PS C:\Users\parva\Desktop\AI Asst> & C:\Users\parva\AppData\Local\Programs\Python\Python311\python.exe
 Python 3.11.1 (tags/v3.11.1:cc5e09f, Aug 14 2023, 14:15:13) [MSC v.3994 64 bit (AMD64)] on win32
 Type "help", "copyright", "credits" or "license" for more information.
 Ctrl+C to launch Vt Code Native REPL
 >>> </c:/users/parva/appdata/local/programs/python/python311/python.exe c:/users/parva/Desktop/AI Asst/ass 4.4.py>
 File "ass 4.4.py", line 2

File Edit Selection View Go Run Terminal Help

AI Asst

EXPLORER AI ASSISTED add.py ass 4.pdf ass 4.py check_leap_year.py lab 4.3 word.doc lab 4.3 word.pdf lab 4.3 word.py lab 4.3.py lab assignment 3.3.pdf lab assignment 1.4.pdf lab assignment 2.3.pdf leap_year.py

```

46
47 # Test with custom email
48 print("n + " + BB)
49 custom_subject = input("nenter email subject: ")
50 custom_body = input("nenter email body: ")
51 result = classify_priority(custom_subject, custom_body)
52 print("nclassified priority: ", result)
53
54 # ...existing code...
55
56
57 def classify_priority(subject, body=""):
58     text = (subject + " " + body).lower()
59     if any(k in text for k in ("server down", "urgent", "critical", "immediate", "outage", "emergency")):
60         return "High"
61     elif any(k in text for k in ("meeting", "deadline", "important", "review", "required", "asap")):
62         return "Medium"
63     else:
64         return "Low"
65
66 if __name__ == "__main__":
67     subj = input("Email subject: ").strip()
68     body = input("Email body (optional): ").strip()
69     print("Priority: ", classify_priority(subj, body))
70
71 # ...existing code...

```

Body: Please vote on your preferred lunch option.

Email #4 [High Priority]
Subject: Client Presentation Delayed - Decision Needed Today
Body: Our major client has requested to reschedule the presentation.

Email #5 [Medium Priority]
Subject: Monthly Team Updates - Please Submit by End of Week
Body: Submit your monthly progress report by Friday.

Email #6 [Low Priority]
Subject: Office Supplies Restocking - New Printer Paper Available
Body: New printer paper has arrived in the supply closet.

High: 2 | Medium: 2 | Low: 2

File Explorer: AI Assisted, .add.py, AI ass1.pdf, ass4.pdf, check_leap_year.py, lab_4.3_word.docx, lab_4.3_word.pdf, lab_4.3_word.py, lab_4.3_word1.pdf, lab_4.3_word2.pdf, lab_4.3_word3.pdf, lab_assignment_3.3.pdf, lab_assignment_1.4.pdf, lab_assignment_2.5.pdf, leap_year.py

Terminal: Python 3.13.7 (tags/v3.13.7-rc1/26ecf), Aug 14 2025, 14:15:11 [MSC v.3994 64 bit (AMD64)] on win32

Output: Type 'help', 'copyright', 'credits' or 'license' for more information.
>>> C:\Users\parva\OneDrive\Desktop\AI Assisted & C:/Users/parva/AppData/Local/Programs/Python/Python313/python.exe "C:/Users/parva/Desktop/AI Assisted/ass4.4.py"
File "cstim", line 1
& C:/Users/parva/AppData/Local/Programs/Python/Python313/python.exe "C:/Users/parva/Desktop/AI Assisted/ass4.4.py"

SystemError: invalid syntax

```

67 if __name__ == "__main__":
68     subj = input("Email subject: ").strip()
69     body = input("Email body (optional): ").strip()
70     print("Priority: ", classify_priority(subj, body))
71
72 # ...existing code...
73
74 # ...existing code...
75
76 def classify_priority(text):
77     t = text.lower()
78     if any(k in t for k in ("urgent", "server down", "issue", "critical", "immediate", "outage")):
79         return "High"
80     elif any(k in t for k in ("meeting", "tomorrow", "deadline", "schedule", "reminder")):
81         return "Medium"
82     else:
83         return "Low"
84
85 # Examples:
86 for e in ["Urgent issue", "Meeting tomorrow", "Greetings"]:
87     print(f"(e) -> {classify_priority(e)}")

```

Body: Please vote on your preferred lunch option.

Email #4 [High Priority]
Subject: Critical: Client Presentation Delayed - Decision Needed Today
Body: Our major client has requested to reschedule the presentation.

Email #5 [Medium Priority]
Subject: Monthly Team Updates - Please Submit by End of Week
Body: Submit your monthly progress report by Friday.

Email #6 [Low Priority]
Subject: Office Supplies Restocking - New Printer Paper Available
Body: New printer paper has arrived in the supply closet.

High: 2 | Medium: 2 | Low: 2

File Explorer: AI Assisted, .add.py, AI ass1.pdf, ass4.pdf, check_leap_year.py, lab_4.3_word.docx, lab_4.3_word.pdf, lab_4.3_word.py, lab_4.3_word1.pdf, lab_4.3_word2.pdf, lab_4.3_word3.pdf, lab_assignment_3.3.pdf, lab_assignment_1.4.pdf, lab_assignment_2.5.pdf, leap_year.py

Terminal: Python 3.13.7 (tags/v3.13.7-rc1/26ecf), Aug 14 2025, 14:15:11 [MSC v.3994 64 bit (AMD64)] on win32

Output: Type 'help', 'copyright', 'credits' or 'license' for more information.
>>> C:\Users\parva\OneDrive\Desktop\AI Assisted & C:/Users/parva/AppData/Local/Programs/Python/Python313/python.exe "C:/Users/parva/Desktop/AI Assisted/ass4.4.py"
File "cstim", line 1
& C:/Users/parva/AppData/Local/Programs/Python/Python313/python.exe "C:/Users/parva/Desktop/AI Assisted/ass4.4.py"

SystemError: invalid syntax

3. Student Query Routing System

Scenario:

A university chatbot must route student queries to Admissions, Exams,

Academics, or Placements.

Tasks:

1. Create 6 sample student queries mapped to departments.
 2. Implement Zero-shot intent classification using an LLM.
 3. Improve results using One-shot prompting.
 4. Further refine results using Few-shot prompting.
 5. Analyze how contextual examples affect classification accuracy.

5. Analyze how contextual examples affect classification accuracy.

The screenshot shows a VS Code interface with the following details:

- File Explorer:** Shows files like `ass 4.4.py`, `check leap year.py`, `lab 4.3 word.docx`, and `leap year.py`.
- Terminal:** Displays the following Python code:

```
ass 4.4.py
# Simple list of student queries (query, department)
queries = [
    ("how do I apply for admission?", "Admissions"),
    ("when are the exams scheduled?", "Exams"),
    ("what is my grade point average?", "GPA"),
    ("what are the placement criteria?", "Placements"),
    ("how do I request a transcript?", "Academics"),
    ("what is the application deadline?", "Admissions"),
]
for q, dept in queries:
    print(f"<{dept}>: {q}")


```
- Output:** Shows multiple error messages from the terminal:

```
SyntaxError: invalid syntax
>>> & C:/Users/parva/AppData/Local/Programs/Python/Python313/python.exe "c:/Users/parva/OneDrive/Desktop/AI Assisted/ass 4.4.py"
File "", line 1
  & C:/Users/parva/AppData/Local/Programs/Python/Python313/python.exe "c:/Users/parva/OneDrive/Desktop/AI Assisted/ass 4.4.py"
SyntaxError: invalid syntax
>>> & C:/Users/parva/AppData/Local/Programs/Python/Python313/python.exe "c:/Users/parva/OneDrive/Desktop/AI Assisted/ass 4.4.py"
  & C:/Users/parva/AppData/Local/Programs/Python/Python313/python.exe "c:/Users/parva/OneDrive/Desktop/AI Assisted/ass 4.4.py"
SyntaxError: invalid syntax
>>> & C:/Users/parva/AppData/Local/Programs/Python/Python313/python.exe "c:/Users/parva/OneDrive/Desktop/AI Assisted/ass 4.4.py"
  & C:/Users/parva/AppData/Local/Programs/Python/Python313/python.exe "c:/Users/parva/OneDrive/Desktop/AI Assisted/ass 4.4.py"
SyntaxError: invalid syntax
>>> exit()
```
- Search Bar:** Contains the text "AI Assisted".
- Right Panel:** A "CHAT" window titled "SENTIMENT CLASSIFICATION FOR CUST..." and a "PROBLEMS" panel showing existing code snippets.

The screenshot shows the Visual Studio Code interface with the "AI Assisted" extension open. The main editor window contains Python code for sentiment classification, with several code completion suggestions overlaid. A sidebar on the right displays these suggestions, including examples for admissions, exams, placements, and academic queries. The status bar at the bottom right shows the date and time as 1/29/2026 at 1:50 PM.

```
1 # ...existing code...
2
3 def classify_query(q):
4     t = q.lower()
5
6     if any(k in t for k in ("admission", "apply", "application", "enroll", "fee", "deadline")):
7         return "Admissions"
8
9     elif any(k in t for k in ("exam", "exams", "result", "results", "grade", "schedule")):
10        return "Exams"
11
12    elif any(k in t for k in ("placement", "placements", "internship", "job", "career", "interview")):
13        return "Placements"
14
15    elif any(k in t for k in ("course", "change", "transcript", "academic", "semester", "register")):
16        return "Academic"
17
18    else:
19        return "Academics"
20
21
22 for q in queries:
23     print(classify_query(q))
24
25 # ...existing code...
26
27 def route_query(q):
28     t = q.lower()
29
30     if any(k in t for k in ("admission", "apply", "application", "deadline")):
31         return "Admissions"
32
33     elif any(k in t for k in ("exam", "exams", "schedule", "results", "grade")):
34         return "Exams"
35
36     elif any(k in t for k in ("placement", "placements", "interview", "job", "internship")):
37         return "Placements"
38
39     else:
40         return "Academics"
41
42
43 for q in queries:
44     print(route_query(q))
45
46 # ...existing code...
```

This screenshot shows the same VS Code environment with a different set of AI-assisted code completion suggestions. The main editor window now contains Python code for classifying student queries into departments like Admissions, Exams, Placements, or Academic. The sidebar on the right shows specific examples for each category. The status bar at the bottom right shows the date and time as 1/29/2026 at 1:51 PM.

```
1 # ...existing code...
2
3 def classify_query(q):
4     t = q.lower()
5
6     if any(k in t for k in ("admission", "apply", "application", "enroll", "fee", "deadline")):
7         return "Admissions"
8
9     elif any(k in t for k in ("exam", "exams", "result", "results", "grade", "schedule")):
10        return "Exams"
11
12    elif any(k in t for k in ("placement", "placements", "internship", "job", "career", "interview")):
13        return "Placements"
14
15    elif any(k in t for k in ("course", "change", "transcript", "academic", "semester", "register")):
16        return "Academic"
17
18    else:
19        return "Academics"
20
21
22 if __name__ == "__main__":
23     q = input("Query: ").strip()
24     print("Department:", classify_query(q))
25
26 # ...existing code...
```

```

File Edit Selection View Go Run Terminal Help < > AI Assted
EXPLORER ass 4.4.py
    # ...existing code...
    def classify_query(q):
        t = q.lower()
        if "exam" in t or "semester" in t:
            return "Exams"
        elif "course" in t or "syllabus" in t:
            return "Academics"
        elif "placement" in t or "campus" in t:
            return "Placements"
        elif "admission" in t or "apply" in t:
            return "Admissions"
        else:
            return "General"
    if __name__ == "__main__":
        q = input().strip()
        print("Department:", classify_query(q))
    # ...existing code...

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```

PS C:\Users\parva\OneDrive\Desktop\AI Assted> print("n" + "n" * 80)
Usage to initialize device PIN
PS C:\Users\parva\OneDrive\Desktop\AI Assted> print("High: sum[1 for e in emails if e[2] == 'High']) | Medium: sum[1 for e in emails if e[2] == 'Medium']) | Low: sum[1 for e in emails if e[2] == 'Low'])
>>> exit()
>>> exit()
>>> C:/Users/parva/AppData/Local/Programs/Python/Python313/python.exe "c:/Users/parva/OneDrive/Desktop/AI Assted/ass 4.4.py"
>>> exit()
>>> exit()
>>> C:/Users/parva/AppData/Local/Programs/Python/Python313/python.exe "c:/Users/parva/OneDrive/Desktop/AI Assted/ass 4.4.py"
>>> exit()
>>> exit()

```

OUTLINE TIMELINE 0.0s

Breaking news Supreme Court...

Search

File Edit Selection View Go Run Terminal Help < > AI Assted

EXPLORER ass 4.4.py

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

CHAT Write every small Python code to route student queries.

Examples: "When are semester exams?" -- Exams "Course syllabus details" -- Academics "Campus placement updates" -- Placements "How to apply for admission?" -- Admissions

Use these examples and write minimal if else code. Avoid extra details.

Used 1 reference

```

# ...existing code...
def classify_query(q):
    t = q.lower()
    if "exam" in t or "semester" in t:
        return "Exams"
    elif "course" in t or "syllabus" in t:
        return "Academics"
    elif "placement" in t or "campus" in t:
        return "Placements"
    elif "admission" in t or "apply" in t:
        return "Admissions"
    else:
        return "Academics"

if __name__ == "__main__":
    q = input().strip()
    print("Department:", classify_query(q))
# ...existing code...

```

Used 1 reference

Used 1 reference

EVALUATED TEXT AND IMPLEMENTED FINAL CODE

```

# ...existing code...
tests = [
    ("When are semester exams?", "Exams"),
    ("Course syllabus details", "Academics"),
    ("Campus placement updates", "Placements"),
    ("How to apply for admission?", "Admissions"),
    ("What is the admission process?", "Admissions"),
    ("Are finals scheduled next month?", "Exams"),
    ("Internship opportunities", "Placements"),
    ("Change of course procedure", "Academics"),
]

def zero_shot():
    t = q.lower()
    if "exam" in t or "semester" in t: return "Exams"
    if "course" in t or "syllabus" in t: return "Academics"
    if "placement" in t or "campus" in t: return "Placements"
    if "admission" in t or "apply" in t: return "Admissions"
    else: return "Academics"

one_example = "admission process"
def one_shot(q):
    t = q.lower()
    if one_example in t: return "Admissions"
    return zero_shot(q)

zero_shot = zero_shot()
def one_shot(q):
    t = q.lower()
    if one_example in t: return "Admissions"
    return zero_shot(q)

few_examples = [
    "admission process", "Admissions",
    "How to apply for admission?", "Admissions",
    "What is the admission proc?", "Admissions",
    "Are finals scheduled next month?", "Exams",
    "Internship opportunities", "Placements",
    "Change of course procedure", "Academics"
]
def few_shot():
    t = q.lower()
    for k,v in few_examples.items():
        if k in t: return v
    return zero_shot(q)

def score(fm):
    return sum(1 for q,a in tests if fm(q)==a)/len(tests)

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

>>> exit()
>>> exit()
>>> C:/Users/parva/AppData/Local/Programs/Python/Python313/python.exe "c:/Users/parva/OneDrive/Desktop/AI Assted/ass 4.4.py"
>>> exit()
>>> exit()
>>> C:/Users/parva/AppData/Local/Programs/Python/Python313/python.exe "c:/Users/parva/OneDrive/Desktop/AI Assted/ass 4.4.py"
>>> exit()
>>> exit()
>>> C:/Users/parva/AppData/Local/Programs/Python/Python313/python.exe "c:/Users/parva/OneDrive/Desktop/AI Assted/ass 4.4.py"
>>> exit()
>>> exit()

OUTLINE TIMELINE 1:52 PM 1/29/2026

```

File Edit Selection View Go Run Terminal Help < > AI Assted
EXPLORER ass 4.4.py
    # ...existing code...
    def classify_query(q):
        t = q.lower()
        if "exam" in t or "semester" in t:
            return "Exams"
        elif "course" in t or "syllabus" in t:
            return "Academics"
        elif "placement" in t or "campus" in t:
            return "Placements"
        elif "admission" in t or "apply" in t:
            return "Admissions"
        else:
            return "General"
    if __name__ == "__main__":
        q = input().strip()
        print("Department:", classify_query(q))
    # ...existing code...

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```

PS C:\Users\parva\OneDrive\Desktop\AI Assted> print("n" + "n" * 80)
Usage to initialize device PIN
PS C:\Users\parva\OneDrive\Desktop\AI Assted> print("High: sum[1 for e in emails if e[2] == 'High']) | Medium: sum[1 for e in emails if e[2] == 'Medium']) | Low: sum[1 for e in emails if e[2] == 'Low'])
>>> exit()
>>> exit()
>>> C:/Users/parva/AppData/Local/Programs/Python/Python313/python.exe "c:/Users/parva/OneDrive/Desktop/AI Assted/ass 4.4.py"
>>> exit()
>>> exit()
>>> C:/Users/parva/AppData/Local/Programs/Python/Python313/python.exe "c:/Users/parva/OneDrive/Desktop/AI Assted/ass 4.4.py"
>>> exit()
>>> exit()

```

OUTLINE TIMELINE 0.0s

Air: Moderate Saturday

File Edit Selection View Go Run Terminal Help < > AI Assted

EXPLORER ass 4.4.py

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

CHAT Write every small Python code to route student queries.

Examples: "When are semester exams?" -- Exams "Course syllabus details" -- Academics "Campus placement updates" -- Placements "How to apply for admission?" -- Admissions

Use these examples and write minimal if else code. Avoid extra details.

Used 1 reference

EVALUATED TEXT AND IMPLEMENTED FINAL CODE

```

# ...existing code...
def classify_query(q):
    t = q.lower()
    if "exam" in t or "semester" in t:
        return "Exams"
    elif "course" in t or "syllabus" in t:
        return "Academics"
    elif "placement" in t or "campus" in t:
        return "Placements"
    elif "admission" in t or "apply" in t:
        return "Admissions"
    else:
        return "Academics"

if __name__ == "__main__":
    q = input().strip()
    print("Department:", classify_query(q))
# ...existing code...

```

Used 1 reference

EVALUATED TEXT AND IMPLEMENTED FINAL CODE

```

# ...existing code...
tests = [
    ("When are semester exams?", "Exams"),
    ("Course syllabus details", "Academics"),
    ("Campus placement updates", "Placements"),
    ("How to apply for admission?", "Admissions"),
    ("What is the admission proc?", "Admissions"),
    ("Are finals scheduled next month?", "Exams"),
    ("Internship opportunities", "Placements"),
    ("Change of course procedure", "Academics")
]

def zero_shot():
    t = q.lower()
    if one_example in t: return "Admissions"
    return zero_shot(q)

zero_shot = zero_shot()
def one_shot(q):
    t = q.lower()
    if one_example in t: return "Admissions"
    return zero_shot(q)

few_examples = [
    "admission process", "Admissions",
    "How to apply for admission?", "Admissions",
    "What is the admission proc?", "Admissions",
    "Are finals scheduled next month?", "Exams",
    "Internship opportunities", "Placements",
    "Change of course procedure", "Academics"
]
def few_shot():
    t = q.lower()
    for k,v in few_examples.items():
        if k in t: return v
    return zero_shot(q)

def score(fm):
    return sum(1 for q,a in tests if fm(q)==a)/len(tests)

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

>>> exit()
>>> exit()
>>> C:/Users/parva/AppData/Local/Programs/Python/Python313/python.exe "c:/Users/parva/OneDrive/Desktop/AI Assted/ass 4.4.py"
>>> exit()
>>> exit()
>>> C:/Users/parva/AppData/Local/Programs/Python/Python313/python.exe "c:/Users/parva/OneDrive/Desktop/AI Assted/ass 4.4.py"
>>> exit()
>>> exit()
>>> C:/Users/parva/AppData/Local/Programs/Python/Python313/python.exe "c:/Users/parva/OneDrive/Desktop/AI Assted/ass 4.4.py"
>>> exit()
>>> exit()

OUTLINE TIMELINE 1:54 PM 1/29/2026

4. Chatbot Question Type Detection

Scenario:

A chatbot must identify whether a user query is Informational,

Transactional, Complaint, or Feedback.

Tasks:

1. Prepare 6 chatbot queries mapped to question types.
2. Design prompts for Zero-shot, One-shot, and Few-shot learning.
3. Test all prompts on the same unseen queries.
4. Compare response correctness and ambiguity handling.
5. Document observations.

```
File Edit Selection View Go Run Terminal Help < > AI Assisted ass 4.4.py > ass 4.4.py > # Sample queries and types 1 # What's the status of my order? 2 # Can you update my shipping address? 3 # The product arrived broken and unusable. 4 # How do I use the new dashboard feature? 5 # Thanks for the quick support, great job! ] 6 7 # Zero-shot classification 8 def zero_shot(q): 9     t = q.lower() 10    if any(k in t for k in ("broken", "damaged", "not working", "complain", "issue", "problem")): 11        return "Complaint" 12    if any(k in t for k in ("reset", "change", "cancel", "status", "buy", "buy", "billing")): 13        return "Transactional" 14    if any(k in t for k in ("love", "great", "thanks", "thank", "feedback", "suggest")): 15        return "Feedback" 16    return "Informational" 17 18 # One-shot classification 19 def one_shot(q): 20    if "reset my password" in q.lower(): return "Transactional" 21    return zero_shot(q) 22 23 # Few-shot classification 24 def few_shot(q): 25    examples = { 26        "reset my password": "Transactional", 27        "order arrived damaged": "Complaint", 28        "love the new interface": "Feedback", 29        "user guide": "Informational" 30    } 31    t = q.lower() 32    for k, v in examples.items(): 33        if k in t: return v 34    return zero_shot(q) 35 36 # Run all queries 37 for q in queries: 38    print("Query:", q) 39    print("Zero-shot:", zero_shot(q)) 40    print("One-shot:", one_shot(q)) 41    print("Few-shot:", few_shot(q)) 42 43 # Used 1 reference 44 # No actions performed 45 46 # ...existing code... 47 samples = [ 48    ("How do I reset my password", "Transactional"), 49    ("What are the recent price cuts?", "Informational"), 50    ("My order arrived damaged", "Complaint"), 51    ("I love the new interface", "Feedback"), 52    ("How to change my billing info", "Informational"), 53    ("Where can I find the user manual", "Informational") 54] 55 56 def zero_shot(q): 57    t = q.lower() 58    if any(k in t for k in ("broken", "damaged", "not working", "complain", "issue", "problem")): 59        return "Complaint" 60    if any(k in t for k in ("reset", "change", "cancel", "status", "buy", "buy", "billing")): 61        return "Transactional" 62    if any(k in t for k in ("love", "great", "thanks", "thank", "feedback", "suggest")): 63        return "Feedback" 64    return "Informational"
```

5. Emotion Detection in Text

Scenario:

A mental-health chatbot needs to detect emotions: Happy, Sad, Angry,

Anxious, Neutral.

Tasks:

1. Create labeled emotion samples.
2. Use Zero-shot prompting to identify emotions.
3. Use One-shot prompting with an example.
4. Use Few-shot prompting with multiple emotions.
5. Discuss ambiguity handling across techniques.

```
File Edit Selection View Go Run Terminal Help < > Q AI Coding

EXPLORER ... Welcome assignment 3.py AI lab43.py lab assignment 44.py X
Generate code
Add Context...
1 import pandas as pd
2
3 # Create a DataFrame from the provided data
4 data = [
5     "Text": [
6         "I am very happy today",
7         "I feel lonely and depressed",
8         "This is so frustrating",
9         "I am worried about my future",
10        "Today is just normal",
11        "Feeling excited about results"
12    ],
13    "Emotion": [
14        "Happy",
15        "Sad",
16        "Angry",
17        "Anxious",
18        "Neutral",
19        "Happy"
20    ]
21 }
22
23 df = pd.DataFrame(data)
24
25 # Display the DataFrame
26 print(df)

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

PS D:\AI Coding> & C:/Users/ANJALI/AppData/Local/Programs/Python/Python313/python.exe "d:/AI Coding/lab assignment 44.py"
PS D:\AI Coding> & C:/Users/ANJALI/AppData/Local/Programs/Python/Python313/python.exe "d:/AI Coding/lab assignment 44.py"
Traceback (most recent call last):
  File "d:/AI Coding/lab assignment 44.py", line 1, in <module>
    import pandas as pd
ModuleNotFoundError: No module named 'pandas'.
PS D:\AI Coding> & C:/Users/ANJALI/AppData/Local/Programs/Python/Python313/python.exe "d:/AI Coding/lab assignment 44.py"
Traceback (most recent call last):
  File "d:/AI Coding/lab assignment 44.py", line 1, in <module>
    import pandas as pd
ModuleNotFoundError: No module named 'pandas'.
PS D:\AI Coding>
```

```
File Edit Selection View Go Run Terminal Help < > Q AI Coding

EXPLORER ... Welcome assignment 3.py AI lab43.py lab assignment 44.py X
Generate code
Add Context...
1 import pandas as pd
2
3 # Create a DataFrame from the provided data
4 data = [
5     "Text": [
6         "I am very happy today",
7         "I feel lonely and depressed",
8         "This is so frustrating",
9         "I am worried about my future",
10        "Today is just normal",
11        "Feeling excited about results"
12    ],
13    "Emotion": [
14        "Happy",
15        "Sad",
16        "Angry",
17        "Anxious",
18        "Neutral",
19        "Happy"
20    ]
21 }
22
23 df = pd.DataFrame(data)
24
25 # Display the DataFrame
26 print(df)

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

PS D:\AI Coding> & C:/Users/ANJALI/AppData/Local/Programs/Python/Python313/python.exe "d:/AI Coding/lab assignment 44.py"
PS D:\AI Coding> & C:/Users/ANJALI/AppData/Local/Programs/Python/Python313/python.exe "d:/AI Coding/lab assignment 44.py"
Traceback (most recent call last):
  File "d:/AI Coding/lab assignment 44.py", line 1, in <module>
    import pandas as pd
ModuleNotFoundError: No module named 'pandas'.
PS D:\AI Coding> & C:/Users/ANJALI/AppData/Local/Programs/Python/Python313/python.exe "d:/AI Coding/lab assignment 44.py"
Traceback (most recent call last):
  File "d:/AI Coding/lab assignment 44.py", line 1, in <module>
    import pandas as pd
ModuleNotFoundError: No module named 'pandas'.
PS D:\AI Coding>
```

The screenshot shows the VS Code interface with the 'AI Coding' workspace selected. In the center editor area, the file 'lab assignment 44.py' is open, displaying the following Python code:

```
1 def identify_emotion(text):
2     if "frustrating" in text:
3         return "Frustrated"
4     return "Neutral"
5
6 # Example usage
7 text = "This is so frustrating"
8 emotion = identify_emotion(text)
9 print(f"Emotion: {emotion}")
```

A code completion tooltip is displayed above the word 'identify_emotion'. The tooltip contains the function signature and a snippet of code demonstrating its use. The status bar at the bottom shows the path 'D:\AI Coding'.

The screenshot shows the VS Code interface with the 'AI Coding' workspace selected. In the center editor area, the file 'lab assignment 44.py' is open, displaying the following Python code:

```
1 def classify_emotion(text):
2     emotions = {
3         "happy": ["happy", "joyful", "excited", "pleased"],
4         "sad": ["lonely", "depressed", "sad", "down"],
5         "anxious": ["worried", "anxious", "nervous", "stressed"],
6         "neutral": ["normal", "fine", "okay", "average"],
7         "frustrated": ["frustrating", "annoyed", "irritated"]
8     }
9
10    for emotion, keywords in emotions.items():
11        if any(keyword in text.lower() for keyword in keywords):
12            return emotion
13    return "Unknown"
14
15 # Example usage
16 text = "This is so frustrating"
17 emotion = classify_emotion(text)
18 print(f"Text: \'{text}\'\nEmotion: {emotion}")
```

A code completion tooltip is displayed above the word 'classify_emotion'. The tooltip contains the function signature and a snippet of code demonstrating its use. The status bar at the bottom shows the path 'D:\AI Coding'.

The screenshot shows a code editor interface with a dark theme. On the left is the Explorer sidebar, which lists various files and folders under the 'AI CODING' category, including 'add.py', 'AI lab43.py', 'Assignment1(CP).pdf', 'Assignment 2-4.pdf', 'assignment 3.4', 'Assignment2.pdf', 'assignment3.4.docx', 'factorial.py', 'jobs.py', 'jobs.py', 'lab assignment ...', 'lab assignment3.3.pdf', 'lab1 HCP.pdf', 'mathttHCP.py', 'Untitled20.py', and 'week2 HCP.pdf'. The main workspace displays a Python script named 'lab assignment 44.py'. The script defines a function 'classify_emotion' that takes a string 'text' and returns an emotion based on a dictionary of keywords. A sample usage example is shown at the bottom. Below the code editor is a terminal window showing a stack trace and command-line output indicating a 'ModuleNotFoundError' for the 'pandas' module. The bottom navigation bar includes tabs for PROBLEMS, OUTPUT, DEBUG CONSOLE, TERMINAL (which is selected), and PORTS.

```
def classify_emotion(text):
    emotions = {
        "happy": ["happy", "joyful", "excited", "pleased"],
        "sad": ["lonely", "depressed", "sad", "down"],
        "anxious": ["worried", "anxious", "nervous", "stressed"],
        "neutral": ["normal", "fine", "okay", "average"],
        "frustrated": ["frustrating", "annoyed", "irritated"]
    }

    for emotion, keywords in emotions.items():
        if any(keyword in text.lower() for keyword in keywords):
            return emotion
    return "Unknown"

# Example usage
text = "This is so frustrating"
emotion = classify_emotion(text)
print(f"Text: '{text}'\nEmotion: {emotion}")
```

```
Traceback (most recent call last):
  File "d:\AI Coding\lab assignment 44.py", line 1, in <module>
    import pandas as pd
ModuleNotFoundError: No module named 'pandas'
PS D:\AI Coding> & C:/Users/ANJALI/AppData/Local/Programs/Python/Python313/python.exe "d:/AI Coding/lab assignment 44.py"
Emotion: Anxious
PS D:\AI Coding> & C:/Users/ANJALI/AppData/Local/Programs/Python/Python313/python.exe "d:/AI Coding/lab assignment 44.py"
Emotion: Frustrated
PS D:\AI Coding> & C:/Users/ANJALI/AppData/Local/Programs/Python/Python313/python.exe "d:/AI Coding/lab assignment 44.py"
Text: "This is so frustrating"
Emotion: frustrated
PS D:\AI Coding>
```