

# Lab Assignment- 3.1

AI Assisted Coding  
M Geethika – 2303A52276

## Question1: Zero-Shot Prompting (Palindrome Number Program)

**Prompt:** generate a Python code, function that checks whether a given number is a palindrome. And input is given by the user

The screenshot shows the BLACKBOX AI assistant interface. In the center, there's a code editor window titled "Lab3.1\_2256.py" containing Python code for checking if a number is a palindrome. Below the code editor is a terminal window showing the execution of the script and its output for two different inputs: 1331 and 12321.

```
#Question 1: generate a Python code, function that checks whether a given number is a palindrome.
#Input is given by the user
def is_palindrome(n):
    num_str = str(n)
    return num_str == num_str[::-1]
if __name__ == "__main__":
    user_input = input("Enter a number: ")
    try:
        number = int(user_input)
        if is_palindrome(number):
            print(f"{number} is a palindrome.")
        else:
            print(f"{number} is not a palindrome.")
    except ValueError:
        print("Invalid input. Please enter a valid integer.")
```

Terminal output:

```
PS C:\Users\kotas\OneDrive\AI assist> & C:/Users/kotas/AppData/Local/Microsoft/WindowsApps/python3.11.exe "c:/Users/kotas/OneDrive/AI assist/Lab3.1_2256.py"
Enter a number: 1331
1331 is not a palindrome.
PS C:\Users\kotas\OneDrive\AI assist> & C:/Users/kotas/AppData/Local/Microsoft/WindowsApps/python3.11.exe "c:/Users/kotas/OneDrive/AI assist/Lab3.1_2256.py"
Enter a number: 12321
12321 is a palindrome.
PS C:\Users\kotas\OneDrive\AI assist>
```

- It generates correct logic but does not handle negative numbers explicitly.\

## Question 2: One-Shot Prompting (Factorial Calculation)

**Prompt:** generate a python code, to get the factorial number of a given number.  
for example, i/p: 5, o/p:120

The screenshot shows the BLACKBOX AI assistant interface. In the center, there's a code editor window titled "Lab3.1\_2256.py" containing Python code for calculating the factorial of a given number. Below the code editor is a terminal window showing the execution of the script and its output for two different inputs: 345 and 48328.

```
#Question 2: generate a python code, to get the factorial number of a given number.
# for example i/p: 5, o/p:120
def factorial(n):
    if n < 0:
        return "Invalid input"
    result = 1
    for i in range(1, n + 1):
        result *= i
    return result
```

Terminal output:

```
PS C:\Users\kotas\OneDrive\AI assist> & C:/Users/kotas/AppData/Local/Microsoft/WindowsApps/python3.11.exe "c:/Users/kotas/OneDrive/AI assist/Lab3.1_2256.py"
Enter a number: 345
345 is not a palindrome.
Enter a number: 48328
48328
```

- When compared to zero- short prompting this prompt improves correctness and handles edge cases like zero and negative values.

### Question 3: Few-Shot Prompting (Armstrong Number Check)

**Prompt:** Generate a Python code to check whether a number is an Armstrong number.

Examples: i/p: 153, o/p: Armstrong Number

i/p: 370, o/p: Armstrong Number

i/p: 123, o/p: Not an Armstrong Number

The screenshot shows the BLACKBOX AI assist interface. On the left, there's a sidebar with icons for Task, Examples, and Evaluating Testing Strategy. The main area displays a Python script named Lab3.1\_2256.py. The code checks if a number is an Armstrong number by calculating the sum of its digits raised to the power of the number of digits. It also includes a function to calculate factorial. The terminal below shows the script running and outputting results for 153, 370, and 123. A message at the bottom indicates an unexpected error due to maximum recursion depth exceeded.

```
28 #Question-3: Generate a Python code to check whether a number is an Armstrong number.
29 #Examples: i/p: 153, o/p: Armstrong Number
30 #           i/p: 370, o/p: Armstrong Number
31 #           i/p: 123, o/p: Not an Armstrong Number
32 number = int(input("Enter a number: "))
33 print(factorial(number))
34
35 def is_armstrong(num):
36     digits = str(num)
37     power = len(digits)
38     total = 0
39
40     for d in digits:
41         total += int(d) ** power
42
43     return total == num
44 number = int(input("Enter a number: "))
45
46 if is_armstrong(number):
47     print("Armstrong Number")
48 else:
49     print("Not an Armstrong Number")
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
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
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
7010
7011
7012
7013
7014
7015
7016
7017
7018
7019
7020
7021
7022
7023
7024
7025
7026
7027
7028
7029
70210
70211
70212
70213
70214
70215
70216
70217
70218
70219
70220
70221
70222
70223
70224
70225
70226
70227
70228
70229
70230
70231
70232
70233
70234
70235
70236
70237
70238
70239
70240
70241
70242
70243
70244
70245
70246
70247
70248
70249
70250
70251
70252
70253
70254
70255
70256
70257
70258
70259
702510
702511
702512
702513
702514
702515
702516
702517
702518
702519
702520
702521
702522
702523
702524
702525
702526
702527
702528
702529
702530
702531
702532
702533
702534
702535
702536
702537
702538
702539
702540
702541
702542
702543
702544
702545
702546
702547
702548
702549
702550
702551
702552
702553
702554
702555
702556
702557
702558
702559
702560
702561
702562
702563
702564
702565
702566
702567
702568
702569
702570
702571
702572
702573
702574
702575
702576
702577
702578
702579
702580
702581
702582
702583
702584
702585
702586
702587
702588
702589
7025810
7025811
7025812
7025813
7025814
7025815
7025816
7025817
7025818
7025819
7025820
7025821
7025822
7025823
7025824
7025825
7025826
7025827
7025828
7025829
7025830
7025831
7025832
7025833
7025834
7025835
7025836
7025837
7025838
7025839
70258310
70258311
70258312
70258313
70258314
70258315
70258316
70258317
70258318
70258319
70258320
70258321
70258322
70258323
70258324
70258325
70258326
70258327
70258328
70258329
70258330
70258331
70258332
70258333
70258334
70258335
70258336
70258337
70258338
70258339
70258340
70258341
70258342
70258343
70258344
70258345
70258346
70258347
70258348
70258349
70258350
70258351
70258352
70258353
70258354
70258355
70258356
70258357
70258358
70258359
70258360
70258361
70258362
70258363
70258364
70258365
70258366
70258367
70258368
70258369
70258370
70258371
70258372
70258373
70258374
70258375
70258376
70258377
70258378
70258379
70258380
70258381
70258382
70258383
70258384
70258385
70258386
70258387
70258388
70258389
70258390
70258391
70258392
70258393
70258394
70258395
70258396
70258397
70258398
70258399
702583100
702583101
702583102
702583103
702583104
702583105
702583106
702583107
702583108
702583109
702583110
702583111
702583112
702583113
702583114
702583115
702583116
702583117
702583118
702583119
702583120
702583121
702583122
702583123
702583124
702583125
702583126
702583127
702583128
702583129
702583130
702583131
702583132
702583133
702583134
702583135
702583136
702583137
702583138
702583139
702583140
702583141
702583142
702583143
702583144
702583145
702583146
702583147
702583148
702583149
702583150
702583151
702583152
702583153
702583154
702583155
702583156
702583157
702583158
702583159
702583160
702583161
702583162
702583163
702583164
702583165
702583166
702583167
702583168
702583169
702583170
702583171
702583172
702583173
702583174
702583175
702583176
702583177
702583178
702583179
702583180
702583181
702583182
702583183
702583184
702583185
702583186
702583187
702583188
702583189
702583190
702583191
702583192
702583193
702583194
702583195
702583196
702583197
702583198
702583199
702583200
702583201
702583202
702583203
702583204
702583205
702583206
702583207
702583208
702583209
702583210
702583211
702583212
702583213
702583214
702583215
702583216
702583217
702583218
702583219
702583220
702583221
702583222
702583223
702583224
702583225
702583226
702583227
702583228
702583229
702583230
702583231
702583232
702583233
702583234
702583235
702583236
702583237
702583238
702583239
702583240
702583241
702583242
702583243
702583244
702583245
702583246
702583247
702583248
702583249
702583250
702583251
702583252
702583253
702583254
702583255
702583256
702583257
702583258
702583259
702583260
702583261
702583262
702583263
702583264
702583265
702583266
702583267
702583268
702583269
702583270
702583271
702583272
702583273
702583274
702583275
702583276
702583277
702583278
702583279
702583280
702583281
702583282
702583283
702583284
702583285
702583286
702583287
702583288
702583289
702583290
702583291
702583292
702583293
702583294
702583295
702583296
702583297
702583298
702583299
702583300
702583301
702583302
702583303
702583304
702583305
702583306
702583307
702583308
702583309
702583310
702583311
702583312
702583313
702583314
702583315
702583316
702583317
702583318
702583319
702583320
702583321
702583322
702583323
702583324
702583325
702583326
702583327
702583328
702583329
702583330
702583331
702583332
702583333
702583334
702583335
702583336
702583337
702583338
702583339
702583340
702583341
702583342
702583343
702583344
702583345
702583346
702583347
702583348
702583349
702583350
702583351
702583352
702583353
702583354
702583355
702583356
702583357
702583358
702583359
702583360
702583361
702583362
702583363
702583364
702583365
702583366
702583367
702583368
702583369
702583370
702583371
702583372
702583373
702583374
702583375
702583376
702583377
702583378
702583379
702583380
702583381
702583382
702583383
702583384
702583385
702583386
702583387
702583388
702583389
702583390
702583391
702583392
702583393
702583394
702583395
702583396
702583397
702583398
702583399
702583400
702583401
702583402
702583403
702583404
702583405
702583406
702583407
702583408
702583409
702583410
702583411
702583412
702583413
702583414
702583415
702583416
702583417
702583418
702583419
702583420
702583421
702583422
702583423
702583424
702583425
702583426
702583427
702583428
702583429
702583430
702583431
702583432
702583433
702583434
702583435
702583436
702583437
702583438
702583439
702583440
702583441
702583442
702583443
702583444
702583445
702583446
702583447
702583448
702583449
702583450
702583451
702583452
702583453
702583454
702583455
702583456
702583457
702583458
702583459
702583460
702583461
702583462
702583463
702583464
702583465
702583466
702583467
702583468
702583469
702583470
702583471
702583472
702583473
702583474
702583475
702583476
702583477
702583478
702583479
702583480
702583481
702583482
702583483
702583484
702583485
702583486
702583487
702583488
702583489
702583490
702583491
702583492
702583493
702583494
702583495
702583496
702583497
702583498
702583499
702583500
702583501
702583502
702583503
702583504
702583505
702583506
702583507
702583508
702583509
702583510
702583511
702583512
702583513
702583514
702583515
702583516
702583517
702583518
702583519
702583520
702583521
702583522
702583523
702583524
702583525
702583526
702583527
702583528
702583529
702583530
702583531
702583532
702583533
702583534
702583535
702583536
702583537
702583538
702583539
702583540
702583541
702583542
702583543
702583544
702583545
702583546
702583547
702583548
702583549
702583550
702583551
702583552
702583553
702583554
702583555
702583556
702583557
702583558
702583559
702583560
702583561
702583562
702583563
702583564
702583565
702583566
702583567
702583568
702583569
702583570
702583571
702583572
702583573
702583574
702583575
702583576
702583577
702583578
702583579
702583580
702583581
702583582
702583583
702583584
702583585
702583586
702583587
702583588
702583589
702583590
702583591
702583592
702583593
702583594
702583595
702583596
702583597
702583598
702583599
702583600
702583601
702583602
702583603
702583604
702583605
702583606
702583607
702583608
702583609
702583610
702583611
702583612
702583613
702583614
702583615
702583616
702583617
702583618
702583619
702583620
702583621
702583622
702583623
702583624
702583625
702583626
702583627
702583628
702583629
702583630
702583631
702583632
702583633
702583634
702583635
702583636
702583637
702583638
702583639
702583640
702583641
702583642
702583643
702583644
702583645
702583646
702583647
702583648
702583649
702583650
702583651
702583652
702583653
702583654
702583655
702583656
702583657
702583658
702583659
702583660
702583661
702583662
702583663
702583664
702583665
702583666
702583667
702583668
702583669
702583670
702583671
702583672
702583673
702583674
702583675
702583676
702583677
702583678
702583679
702583680
702583681
702583682
702583683
702583684
702583685
702583686
702583687
702583688
702583689
702583690
702583691
702583692
702583693
702583694
7
```

## Question 5: Zero-Shot Prompting (Perfect Number Check)

**Prompt:** Generate a Python function to check whether a given number is a perfect number.

The screenshot shows the Microsoft Visual Studio Code interface. In the center, there is a code editor window titled "Lab3.1\_2256.py" with the following Python code:

```
# Question-5: Generate a Python function to check whether a given number is a perfect number.
def is_perfect(num):
    if num <= 0:
        return False

    total = 0
    for i in range(1, num):
        if num % i == 0:
            total += i

    return total == num

number = int(input("Enter a number: "))

if is_perfect(number):
    print("Perfect Number")
else:
    print("Not a Perfect Number")
```

Below the code editor, the terminal window shows the execution of the script and its output:

```
PS C:\Users\kotas\OneDrive\AI assist> python Lab3.1_2256.py
Enter a number: 543
Not an Armstrong Number
Enter a number: 243
243 is not a palindrome.
Enter a number: 12
479001600
Enter a number: 535
Not an Armstrong Number
Enter a number: 53
Prime Number
Enter a number: 343
Not a Perfect Number
```

On the left side of the interface, there is a sidebar titled "TASK" which contains the prompt: "generate a Python code, function that ...". Below this, under "Evaluating Testing Strategy", there is a list of bullet points and some additional text. A tooltip box is overlaid on the sidebar, stating: "An unexpected error occurred: maximum recursion depth exceeded give me the correct code".

This screenshot is identical to the one above, showing the Microsoft Visual Studio Code interface with the same code editor and terminal content. The only difference is the background color of the interface, which is dark mode.

- Zero-shot prompting works but is less optimized due to unnecessary full-range iteration.

## Question 6: Few-Shot Prompting (Even or Odd Classification with Validation)

**Prompt:** Generate a Python program to check whether a number is even or odd with input validation.

### Examples:

I/p: 8, O/p: Even

I/p: 15, O/p: Odd

I/p: 0, O/p: Even

The screenshot shows the AI assist interface in Visual Studio Code. The code editor displays a Python file named Lab3.1\_2256.py with the following content:

```
#Question-6: Generate a Python program to check whether a number is even or odd with input validation.
#Examples:
#I/p: 8, O/p: Even
#I/p: 15, O/p: Odd
#I/p: 0, O/p: Even
number = int(input("Enter a number: "))
if is_perfect(number):
    print("Perfect Number")
else:
    print("Not a Perfect Number")
def even_or_odd(value):
    if not value.lstrip('-').isdigit():
        return "Invalid input"
    num = int(value)
    return "Even" if num % 2 == 0 else "Odd"
user_input = input("Enter a number: ")
print(even_or_odd(user_input))
```

The terminal below shows the execution of the script and some user interactions:

```
PS C:\Users\kotas\OneDrive\AI assist> & C:/Users/kotas/AppData/Local/Microsoft/WindowsApps/python3.11.exe "c:/Users/kotas/OneDrive/AI assist/Lab3.1_2256.py"
25852016738884976640000
Enter a number: 234
Not an Armstrong Number
Enter a number: 23
Prime Number
Enter a number: 234
Not a Perfect Number
Enter a number: 23
Odd
```

- Few-shot prompting significantly improves the quality of AI-generated code by providing clear input–output examples.
- The generated program handles input validation effectively, correctly classifies even and odd numbers, and manages negative and non-integer inputs more reliably compared to zero-shot prompting.