

AI Assisted Coding

Assignment – 9.5

Ht.no : 2303A51663

Batch : 23

Problem 1: String Utilities Function

Consider the following Python function: def

reverse_string(text):

return text[::-1]

Task:

1. Write documentation in:

o (a) Docstring o (b) Inline

comments o (c) Google-style

documentation

2. Compare the three documentation styles.

3. Recommend the most suitable style for a utility-based string

library.

```
Fridaypy > DocExample.py > ...
1  # (a) Docstring
2  def reverse_string(text):
3      """
4      This function takes a string as input and returns the reversed version of that string.
5
6      Parameters:
7      text (str): The string to be reversed.
8
9      Returns:
10     str: The reversed version of the input string.
11     """
12     return text[::-1]
13
14 # (b) Inline comments
15 def reverse_string(text):
16     # This function takes a string as input and returns the reversed version of that string.
17
18     # The input parameter 'text' is expected to be a string.
19
20     # The function uses slicing to reverse the string. The syntax text[::-1] creates a new string that is a reversed version of 'text'.
21
22     return text[::-1]
23
24 # (c) Google-style documentation
25 def reverse_string(text):
26     """
27     Reverses the input string.
28
29     Args:
30         text (str): The string to be reversed.
31
32     Returns:
33         str: The reversed version of the input string.
34     """
35     return text[::-1]
```

```

❖ PS C:\Users\Ganne\OneDrive\Desktop\Ai_Assisted_Coding\Friday.py> python -m pydoc DocExample
Help on module DocExample:

NAME
    DocExample

DESCRIPTION
    # Problem 1: String Utilities Function
    # Consider the following Python function:
    # def reverse_string(text):
    #     return text[::-1]
    # Task:

NAME
    DocExample

DESCRIPTION
    # Problem 1: String Utilities Function
    # Consider the following Python function:
    # def reverse_string(text):
    #     return text[::-1]
    # Task:
    # Consider the following Python function:
    # def reverse_string(text):
    #     return text[::-1]
    # Task:
    # Task:
    # 1. Write documentation in:
    -- More --

```

Problem 2: Password Strength Checker

Consider the function: def

check_strength(password):

return len(password) >= 8

Task:

1. Document the function using docstring, inline comments, and Google style.
2. Compare documentation styles for security-related code.
3. Recommend the most appropriate style.

```
Friday.py > DocExample.py -
41 # (a) Docstring
42 def check_strength(password):
43     """
44     This function checks the strength of a password by verifying if it is at least 8 characters long.
45
46     Parameters:
47     password (str): The password to be checked.
48
49     Returns:
50     bool: True if the password is strong (at least 8 characters), False otherwise.
51     """
52     return len(password) >= 8
53 # (b) inline comments
54 def check_strength(password):
55     """ This function checks the strength of a password by verifying if it is at least 8 characters long.
56
57     # The input parameter 'password' is expected to be a string.
58
59     # The function returns True if the length of the password is greater than or equal to 8, indicating that it is strong. Otherwise, it returns False.
60
61     return len(password) >= 8:
62     """
63 # (c) Google-style documentation
64 def check_strength(password):
65     """
66     Checks the strength of a password.
67
68     Args:
69         password (str): The password to be checked.
70
71     Returns:
72         bool: True if the password is strong (at least 8 characters), False otherwise.
73     """
74     return len(password) >= 8
```

PROBLEMS DEBUG CONSOLE OUTPUT TERMINAL PORTS

PS C:\Users\Ganne\OneDrive\Desktop\Ai_Assisted_Coding\Friday.py> python -m pydoc DocExample

Help on module DocExample:

NAME

DocExample

DESCRIPTION

(a) Docstring

def reverse_string(text):

-- More --

Problem 3: Math Utilities Module

Task:

1. Create a module `math_utils.py` with functions:
 - o `square(n)` o `cube(n)`
 - o `factorial(n)`
2. Generate docstrings automatically using AI tools.
3. Export documentation as an HTML file.

```

Friday.py > math_util.py >...
1 def square(n) :
2     """Returns the square of a number.
3     demonstrates how to use docstrings in Python.
4     Parameters:
5     n (int): The number to be squared.
6     Returns:int: The square of n.
7     """
8     return n * n
9 def cube(n) :
10    """Returns the cube of a number.
11    demonstrates how to use docstrings in Python.
12    Parameters:
13    n (int): The number to be cubed.
14    Returns:int: The cube of n.
15    """
16    return n * n * n
17 def factorial(n) :
18    """Returns the factorial of a number.
19    demonstrates how to use docstrings in Python.
20    Parameters:
21    n (int): The number to calculate the factorial of.
22    Returns:int: The factorial of n.
23    """
24    if n == 0: # check if n is 0 and return 1 if it is because factorial of 0 is 1
25        return 1 # Factorial of 0 is defined to be 1
26    else:
27        return n * factorial(n - 1) # Recursive call to calculate factorial of n
28 print(square.__doc__)
29 print(cube.__doc__)
30 print(factorial.__doc__)
31
32

```

```

PS C:\Users\Ganne\OneDrive\Desktop\AI_Assisted_Coding> cd Friday.py
PS C:\Users\Ganne\OneDrive\Desktop\AI_Assisted_Coding\Friday.py> python -m pydoc Math_util
No Python documentation found for 'Math_util'.
Use help() to get the interactive help utility.
Use help(str) for help on the str class.
PS C:\Users\Ganne\OneDrive\Desktop\AI_Assisted_Coding\Friday.py> python -m pydoc math_util
Help on module math_util:

```

NAME

math_util

DESCRIPTION

```

# def square(n) :
#     """Returns the square of a number.
#     demonstrates how to use docstrings in Python.
#     Parameters:
#     n (int): The number to be squared.
#     Returns:int: The square of n.
#     """
#     return n * n
# def cube(n) :
#     """Returns the cube of a number.
#     demonstrates how to use docstrings in Python.
#     Parameters:
#     n (int): The number to be cubed.
#     Returns:int: The cube of n.
#     """
#     return n * n * n
# def factorial(n) :
#     """Returns the factorial of a number.
#     demonstrates how to use docstrings in Python.
#     Parameters:
#     n (int): The number to calculate the factorial of.
#     Returns:int: The factorial of n.
#     """
#     if n == 0: # check if n is 0 and return 1 if it is because factorial of 0 is 1
#         return 1 # Factorial of 0 is defined to be 1
#     else:
#         return n * factorial(n - 1) # Recursive call to calculate factorial of n
# print(square.__doc__)

```

Problem 4: Attendance Management Module

Task:

1. Create a module attendance.py with functions:
 - o mark_present(student)
 - o mark_absent(student)
 - o get_attendance(student)
2. Add proper docstrings.
3. Generate and view documentation in terminal and browse

```
Friday.py > math_util.py > get_attendance
"""
41 attendance = {}
42 def mark_present(student):
43     """
44     Marks a student as present in the attendance record.
45     Parameters:
46     student (str): The name of the student to be marked as present.
47     """
48     attendance[student] = 'Present'
49 def mark_absent(student):
50     """
51     Marks a student as absent in the attendance record.
52     Parameters:
53     student (str): The name of the student to be marked as absent.
54     """
55     attendance[student] = 'Absent'
56 def get_attendance(student):
57     """
58     Returns the attendance status of a student.
59     Parameters:
60     student (str): The name of the student whose attendance is to be retrieved.
61     Returns:
62     str: The attendance status of the student.
63     """
64     return attendance.get(student, 'Not Found')
```

```
PS C:\Users\Ganne\OneDrive\Desktop\AI_Assisted_Coding\Friday.py> python -m pydoc -w math_util
write math_util.html
PS C:\Users\Ganne\OneDrive\Desktop\AI_Assisted_Coding\Friday.py> & "C:\Program Files\Python312\python.exe" "C:\Users\Ganne\OneDrive\Desktop\AI_Assisted_Coding\Friday.py\math_util.py" > "C:\Users\Ganne\OneDrive\Desktop\AI_Assisted_Coding\Friday.py\math_util.html"
KeyboardInterrupt
PS C:\Users\Ganne\OneDrive\Desktop\AI_Assisted_Coding\Friday.py> python -m pydoc -w math_util
write math_util.html
PS C:\Users\Ganne\OneDrive\Desktop\AI_Assisted_Coding\Friday.py> & "C:\Program Files\Python312\python.exe" "C:\Users\Ganne\OneDrive\Desktop\AI_Assisted_Coding\Friday.py\math_util.py" > "C:\Users\Ganne\OneDrive\Desktop\AI_Assisted_Coding\Friday.py\math_util.html"
PS C:\Users\Ganne\OneDrive\Desktop\AI_Assisted_Coding\Friday.py> & "C:\Program Files\Python312\python.exe" "C:\Users\Ganne\OneDrive\Desktop\AI_Assisted_Coding\Friday.py\math_util.py"
PS C:\Users\Ganne\OneDrive\Desktop\AI_Assisted_Coding\Friday.py> python -m pydoc -p 1234
Server ready at http://localhost:1234/
Server commands: [b]rowser, [q]uit
servers: b
servers: [q]
```

math_util

```

# def square(n) :
#     """Returns the square of a number.
#     demonstrates how to use docstrings in Python.
#     Parameters:
#     n (int): The number to be squared.
#     Returns:int: The square of n.
#     """
#     return n * n
# def cube(n) :
#     """Returns the cube of a number.
#     demonstrates how to use docstrings in Python.
#     Parameters:
#     n (int): The number to be cubed.
#     Returns:int: The cube of n.
#     """
#     return n * n * n
# def factorial(n) :
#     """Returns the factorial of a number.
#     demonstrates how to use docstrings in Python.
#     Parameters:
#     n (int): The number to calculate the factorial of.
#     Returns:int: The factorial of n.
#     """
#     if n == 0: # check if n is 0 and return 1 if it is because factorial of 0 is 1
#         return 1 # factorial of 0 is defined to be 1
#     else:
#         return n * factorial(n - 1) # Recursive call to calculate factorial of n
# print(square.__doc__)
# print(cube.__doc__)
# print(factorial.__doc__)

```

Functions

```

def get_attendance(student):
    """Returns the attendance status of a student.
    Parameters:
    student (str): The name of the student whose attendance is to be retrieved.
    Returns:
    str: The attendance status of the student.

def mark_absent(student):
    """Marks a student as absent in the attendance record.
    Parameters:
    student (str): The name of the student to be marked as absent.

def mark_present(student):
    """Marks a student as present in the attendance record.
    Parameters:
    student (str): The name of the student to be marked as present.

```

Data

```

attendance = {}

```

Problem 5: File Handling Function

Consider the function: def

read_file(filename):

with open(filename, 'r') as f:

return f.read()

Task:

1. Write documentation using all three formats.
2. Identify which style best explains exception handling.
3. Justify your recommendation.

```

89 #DocString style:
90 def read_file(filename):
91     """
92     Reads the content of a file and returns it as a string.
93
94     Parameters:
95     filename (str): The name of the file to be read.
96
97     Returns:
98     str: The content of the file.
99
100     Raises:
101     FileNotFoundError: If the specified file does not exist.
102     IOError: If an I/O error occurs while reading the file.
103     """
104     try:
105         with open(filename, 'r') as f:
106             return f.read()
107     except FileNotFoundError:
108         print(f"Error: The file '{filename}' was not found.")
109         raise
110     except IOError as e:
111         print(f"An I/O error occurred: {e}")
112         raise
113 # Google style Docstring:
114 def read_file(filename):
115     """
116     Reads the content of a file and returns it as a string.
117
118     Args:
119     | filename (str): The name of the file to be read.
120
121     Returns:
122     | str: The content of the file.
123     Raises:
124     | FileNotFoundError: If the specified file does not exist.
125     | IOError: If an I/O error occurs while reading the file.
126     """
127     try:
128         with open(filename, 'r') as f:
129             return f.read()
130     except FileNotFoundError:
131         print(f"Error: The file '{filename}' was not found.")
132         raise
133     except IOError as e:

```

```

Python 3.7.4 Shell
File Edit Shell Debug Console Window Help
Python: Friday X
Friday > # math_utils.py ...
111 except IOError as e:
112     print("An I/O error occurred: {e}")
113     raise
114 # Python style Docstring:
115 def read_file(filename):
116     """
117     Reads the content of a file and returns it as a string.
118
119     (param filename): The name of the file to be read.
120     type filename: str
121     return: The content of the file.
122     rtype: str
123     raises FileNotFoundError: If the specified file does not exist.
124     raises IOError: If an I/O error occurs while reading the file.
125     """
126     try:
127         with open(filename, 'r') as f:
128             return f.read()
129     except FileNotFoundError:
130         print("Error: The file '{filename}' was not found.")
131         raise
132     except IOError as e:
133         print("An I/O error occurred: {e}")
134         raise
135 # Recommendations:
136 # The Google style Docstring tool contains exception handling because it clearly separates the description of the function, its parameters, return value, and exceptions in a structured
137 # format. This makes it easier for developers to quickly understand the function's behavior and the potential errors that may arise, enhancing readability and maintainability of the code.
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
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
1001
1002
1003
1004
1005
1006
1007
1008
1009
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1020
1021
1022
1023
1024
1025
1026
1027
1028
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038
1039
1040
1041
1042
1043
1044
1045
1046
1047
1048
1049
1050
1051
1052
1053
1
```