

Renato Felicio
November 17, 2024
IT FDN 110 A
Assignment 06

Creating Python Scripts – Functions

Introduction

This Assignment 06 consists in creating a Python program that uses constants, variables, and print statements to display a message about a student's registration for a Python course. Building on the knowledge from Assignment 05, this assignment introduces new concepts, such as use of use of functions, classes, and using the separation of concerns pattern.

Preparation for this assignment

To prepare for this assignment, I reviewed the "Module 06 Notes" (Reference 1), completed the three lab examples, and watched both the Module 06 videos available on Washington University's Canvas platform (Figure 1) and the recommended external videos in References 2 through 3. Through these materials, I learnt how to work with functions, including their parameters and arguments, overloaded functions, and return values. I also gained an understanding of how to work with classes and apply the Separation of Concerns (SoC) pattern. I learned that functions and classes are fundamental building blocks of the SoC principle in Python programming.

I also learned the common practice of including a header at the beginning of a class or function, known as a "docstring" in Python. Including additional notes within the docstring can be helpful for developers, providing clear explanations and context for the code.

Mod06 Videos











Module	Name	Link
6	Demo01 - Using Functions	https://www.youtube.com/watch?v=8tZdqlArsbc  
6	Demo02 - Using Arguments	https://www.youtube.com/watch?v=a6dmUlaNB0Q  
6	Demo02 - Using Returns	https://www.youtube.com/watch?v=tTPdCTUsDb8  
6	Demo03 - Using Classes	https://www.youtube.com/watch?v=TAD_BczzOIQ  
6	Demo04- Separation Of Concerns	https://www.youtube.com/watch?v=fapZdUP-vdw  

Figure 1 – Mod06-Videos

Python Scripting

I began by reviewing the provided Assignment06-Starter.py file and use it as the starting point for my script. The objective was to implement the use of function classes and organize the script according to the SoC pattern. The script should allow the user to register multiple students, display their inputs, and save the data back into a '.json' file containing all registered students. Additionally, structured error handling should be implemented when reading the file into dictionary rows, when the user enters first and last names, and when writing the dictionary rows back into the file.

The functions should be grouped into two classes: FileProcessor and IO The FileProcessor class contains functions responsible for reading data from and writing data to a .json file. The IO class includes functions that manage user interactions, such as collecting menu options, gathering input data, displaying messages, and printing error notifications.

Two global variables are defined: menu_choice (a string) to store the user's selected option, and students (a list) to hold the information of all registered students. These variables are initialized with an empty string and an empty list, respectively.

I began by organizing the script into major sections: the header, imports, variable and constant definitions, class and function definitions, and the main body. The functions were grouped into classes based on their roles, aligning with the data processing and presentation layers of concern.

```
# Header
# Import
# Global Data
# Data Layer
    Definition of data constants
    Definition of variable

# Class Definition

    Processing data layer
        Class FileProcessor created
            Function read_data_from_file created
            Function write_data_to_file created

    Presentation data layer
        Class IO created
            Function output_error_messages created
            Function input_menu_choice
            Function output_student_courses
            input_student_data

# Main body of the script
```

The script is displayed in separated figures, split across Figure 2 to Figure 5 according its correspondent sections.

Header, import, constants and variables script parts are presented in the Figure 2.

```
1  # ----- #
2  # Title: Assignment06
3  # Desc: This assignment demonstrates using functions
4  # with structured error handling
5  # Change Log: (Who, When, What)
6  #   Renato Felicio, 11/16/2024, Created Script
7  #   <Your Name Here>, <Date>, <Activity>
8  # ----- #
9
10 # Import section
11 import json
12 from typing import TextIO
13
14 # Global Data Layer
15
16 # Define the Data Constants
17 MENU: str = '''
18 ---- Course Registration Program ----
19 Select from the following menu:
20 1. Register a Student for a Course.
21 2. Show current data.
22 3. Save data to a file.
23 4. Exit the program.
24 -----
25 '''
26
27 # Define the Data Constants
28 FILE_NAME: str = "Enrollments.json" # Constant holds the name of the file with students data
29
30 # Define the Data Variables and constants
31 students: list = [] # This variable holds the information of all registered students.
32 menu_choice: str = '' # It holds the user choice.
33
```

Figure 2 – Python Script Header Import and Variables

Processing data layer is presented in Figure 3.

```
34 # Class definition
35
36 # Processing Data Layer
37 class FileProcessor:
38     """
39     A collection of processing layer functions that work with json files
40
41     ChangeLog: (Who, When, What)
42     Renato Felicio, 11/16/2024, Created Class
43     """
44
45     @staticmethod
46     def read_data_from_file(file_name: str, student_data: list):
47         """
48         This function reads data from a json file into a list of dictionary rows
49
50         Notes:
51         - Data sent to the student_data parameter will be overwritten.
52
53         ChangeLog: (Who, When, What)
54         Renato Felicio, 11/16/2024, Created function
55
56         :param file_name: string with the name of the file we are reading
57         :param student_data: list of dictionary rows we are adding data to
58         :return: list of dictionary rows filled with data
59         """
60
61         try:
62             file: TextIO = open(file_name, "r") # Open the JSON file for reading
63             student_data: list = json.load(file) # File data is loaded into a table
64             # Now 'student_data' contains the parsed JSON data as a Python list of dictionaries
65             file.close()
66         except FileNotFoundError as e: # Handles error in case there is no initial file
67             IO.output_error_messages("Data file must exist before running this script!")
68             file = open(FILE_NAME, "w") # Creates an empty initial file, in case of file not found
69             IO.output_error_messages("Empty file was created!\n")
70         except Exception as e:
71             IO.output_error_messages("Error: There was a problem with reading the file.", e)
72         finally:
73             if file.closed == False:
74                 file.close()
75         return student_data
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
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
```

Figure 3 – Python Script Processing Data Layer

Presentation data layer and main body are presented in Figure 4 and Figure 5, respectively.

```

140 # Presentation Data Layer
141
142 class IO: 13 usages
143     """A collection of presentation layer functions that manage user input and output
144
145     Changelog: (Who, When, What)
146     Renato Felicio,11/16/2024,Created Class
147
148     """
149
150     @staticmethod 8 usages
151     def output_error_messages(message: str, error: Exception = None):
152         """ This function displays a custom error messages to the user
153
154         Changelog: (Who, When, What)
155         Renato Felicio,11/16/2024,Created function
156
157         """
158         _return: None
159
160         print(message, end="\n\n")
161         if error is not None:
162             print("--- Technical Error Message -- ")
163             print(error, error.__doc__, type(error), sep='\n')
164
165     @staticmethod 1 usage
166     def output_menu(menu: str):
167         """ This function displays the menu of choices to the user
168
169         Changelog: (Who, When, What)
170         Renato Felicio,11/16/2024,Created function
171
172         """
173         _return: None
174
175         print(menu)
176
177     @staticmethod 1 usage
178     def input_menu_choice():
179         """ This function gets a menu choice from the user
180
181         Changelog: (Who, When, What)
182         Renato Felicio,11/16/2024,Created function
183
184         """
185         _return: string with the users choice
186
187         choice="0"
188
189         try:
190             choice: str = input("What would you like to do: ")
191             if choice not in ("1", "2", "3", "4"): # Note these are strings
192                 raise Exception("Please, choose only 1, 2, 3, or 4")
193         except Exception as e:
194             IO.output_error_messages(e.__str__()) # Not passing e to avoid the technical message
195
196         return choice
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

```

Figure 4 – Python Script Presentation Data Layer

```

196
197 # Start of the main body of the script
198
199 # Read data from a file
200 students:list = FileProcessor.read_data_from_file(file_name=FILE_NAME, student_data=students)
201
202 while (True): # Loops through the menu of options
203     # Present the menu of choices
204     IO.output_menu(MENU)
205     menu_choice=IO.input_menu_choice()
206
207     # Input user data
208     if menu_choice == "1": # This will not work if it is an integer!
209         students=IO.input_student_data(students)
210         continue
211
212     # Present the current data
213     elif menu_choice == "2":
214
215         # Process the data to create and display a custom message
216         IO.output_student_courses(students)
217         continue
218
219     # Save the data to a file and present to user
220     elif menu_choice == "3":
221         FileProcessor.write_data_to_file(FILE_NAME,students)
222         IO.output_student_courses(students)
223         continue
224
225     # Stop the loop
226     elif menu_choice == "4":
227         break # out of the loop
228     else:
229         print("Please only choose option 1, 2, or 3")
230
231 print("Program Ended")
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

```

Figure 5 – Python Script Main Body

Python Script Testing

I executed the Python script using PyCharm (see Figure 7) and also tested it in the Windows Command Prompt (see Figure 8), verifying that the script performed as expected in both environments. Additionally, I confirmed that the Enrollments.json file was updated correctly and contained the expected output (see Figure 9). Following the assignment instructions, I used the initial Enrollments.json file provided in the module_06.zip file. The original content of this file is shown in Figure 6.

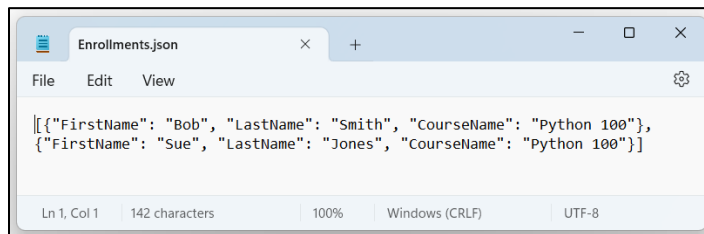


Figure 6 – Initial Enrollment File Python

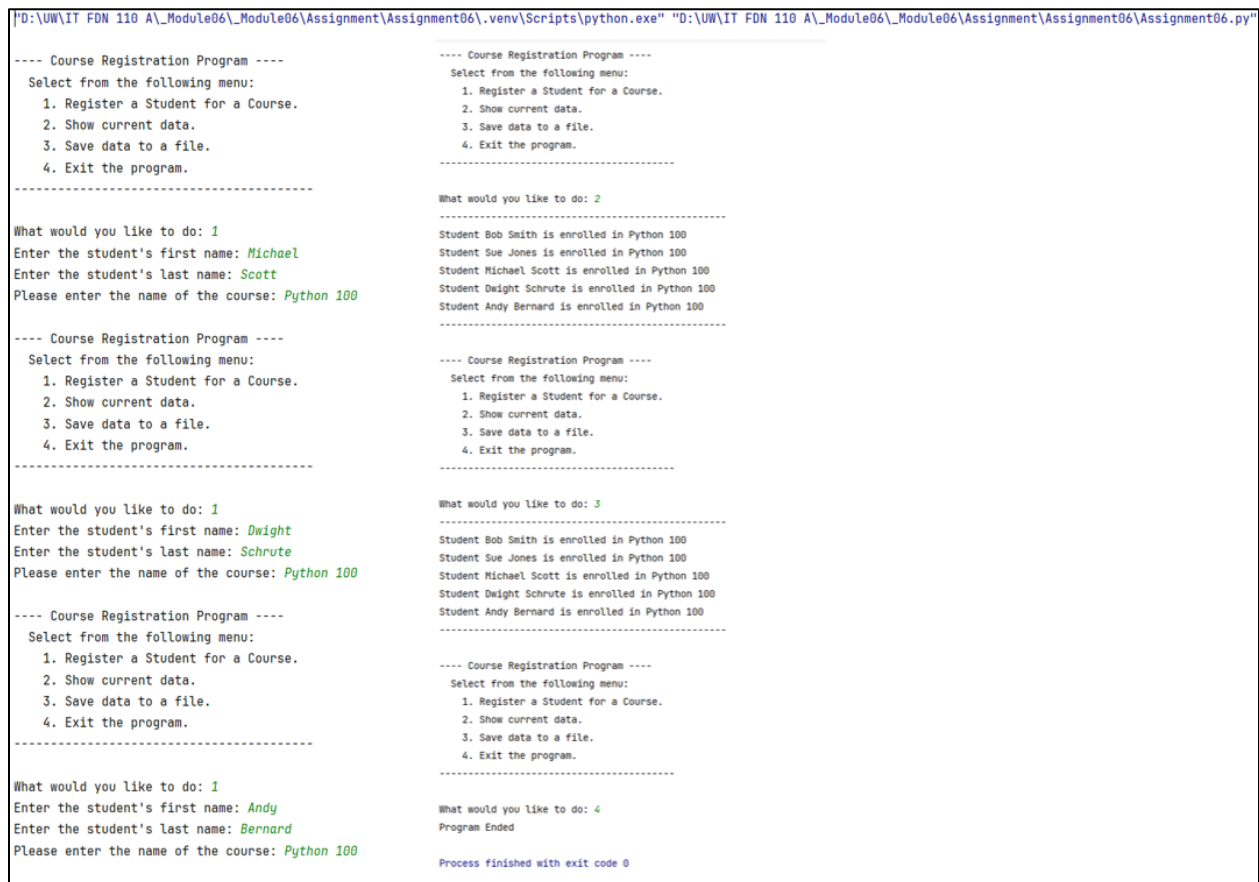


Figure 7 – Python Script PyCharm Run

```

D:\UW\IT FDN 110 A\_Module06\_Module06\Assignment\Assignment06>python Assignment06.py

---- Course Registration Program ----
Select from the following menu:
1. Register a Student for a Course.
2. Show current data.
3. Save data to a file.
4. Exit the program.
-----

What would you like to do: 1
Enter the student's first name: Michael
Enter the student's last name: Scott
Please enter the name of the course: Python 100

---- Course Registration Program ----
Select from the following menu:
1. Register a Student for a Course.
2. Show current data.
3. Save data to a file.
4. Exit the program.
-----

What would you like to do: 1
Enter the student's first name: Dwight
Enter the student's last name: Schrute
Please enter the name of the course: Python 100

---- Course Registration Program ----
Select from the following menu:
1. Register a Student for a Course.
2. Show current data.
3. Save data to a file.
4. Exit the program.
-----

What would you like to do: 1
Enter the student's first name: Andy
Enter the student's last name: Bernard
Please enter the name of the course: Python 100

---- Course Registration Program ----
Select from the following menu:
1. Register a Student for a Course.
2. Show current data.
3. Save data to a file.
4. Exit the program.
-----

What would you like to do: 2
-----
Student Bob Smith is enrolled in Python 100
Student Sue Jones is enrolled in Python 100
Student Michael Scott is enrolled in Python 100
Student Dwight Schrute is enrolled in Python 100
Student Andy Bernard is enrolled in Python 100
-----

---- Course Registration Program ----
Select from the following menu:
1. Register a Student for a Course.
2. Show current data.
3. Save data to a file.
4. Exit the program.
-----

What would you like to do: 3
-----
Student Bob Smith is enrolled in Python 100
Student Sue Jones is enrolled in Python 100
Student Michael Scott is enrolled in Python 100
Student Dwight Schrute is enrolled in Python 100
Student Andy Bernard is enrolled in Python 100
-----

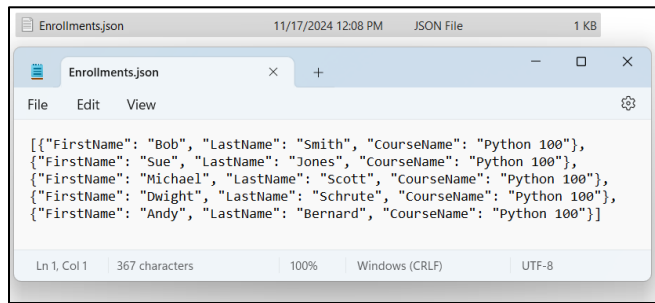
---- Course Registration Program ----
Select from the following menu:
1. Register a Student for a Course.
2. Show current data.
3. Save data to a file.
4. Exit the program.
-----

What would you like to do: 4
Program Ended

```

Figure 8 – Python Script Windows Command Prompt Run

Figure 9 shows that the Enrollments.json file was updated correctly with the three new student data.



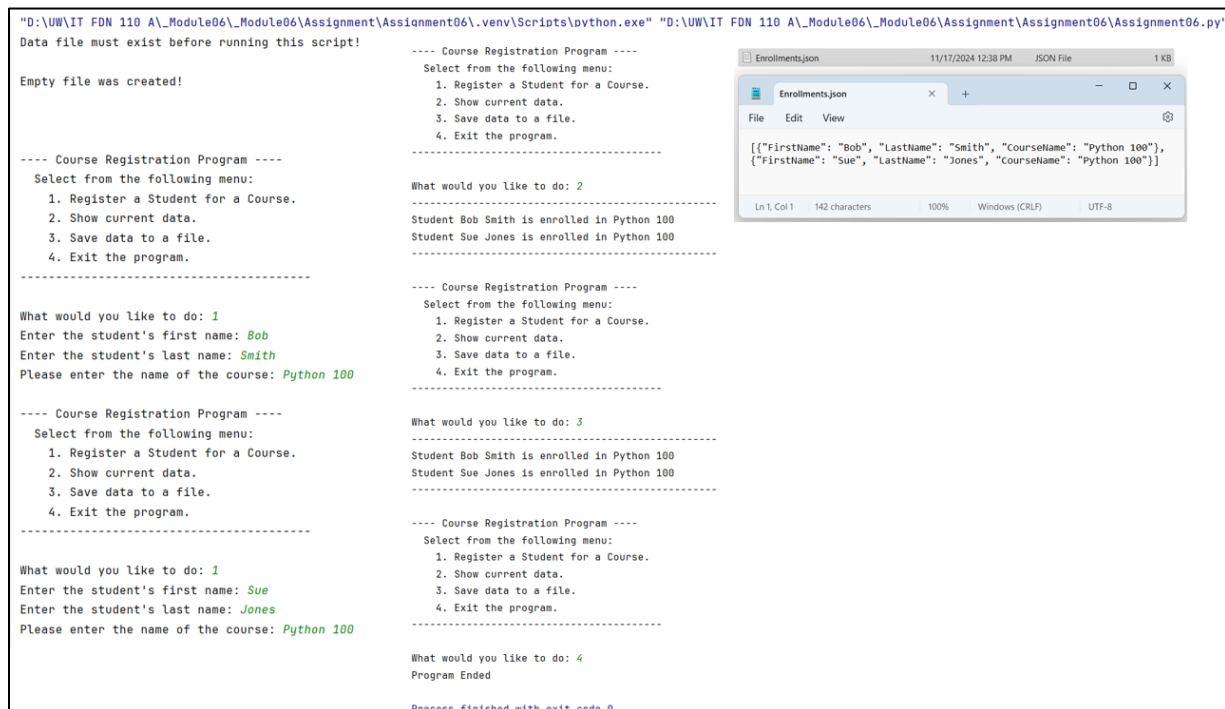
```
[{"FirstName": "Bob", "LastName": "Smith", "CourseName": "Python 100"}, {"FirstName": "Sue", "LastName": "Jones", "CourseName": "Python 100"}, {"FirstName": "Michael", "LastName": "Scott", "CourseName": "Python 100"}, {"FirstName": "Dwight", "LastName": "Schrute", "CourseName": "Python 100"}, {"FirstName": "Andy", "LastName": "Bernard", "CourseName": "Python 100"}]
```

Figure 9 – Output JSON File Content

Python Script Error Handling Test

After testing the script with valid user inputs using a pre-existing Enrollments.json file, I evaluated its error-handling capabilities for scenarios such as a "file not found" exception and invalid first and last name inputs. All tests were conducted using PyCharm.

The Figure 10 below shows the FileNotFoundError handling and the Enrolments.json being updated correctly after menu choice 3 is selected.



```
"D:\UW\IT FDN 110 A\Module06\Module06\Assignment\Assignment06\.venv\Scripts\python.exe" "D:\UW\IT FDN 110 A\Module06\Module06\Assignment\Assignment06\Assignment06.py"
Data file must exist before running this script!

Empty file was created!

---- Course Registration Program ----
Select from the following menu:
1. Register a Student for a Course.
2. Show current data.
3. Save data to a file.
4. Exit the program.

What would you like to do: 1
Enter the student's first name: Bob
Enter the student's last name: Smith
Please enter the name of the course: Python 100

---- Course Registration Program ----
Select from the following menu:
1. Register a Student for a Course.
2. Show current data.
3. Save data to a file.
4. Exit the program.

What would you like to do: 3
Student Bob Smith is enrolled in Python 100
Student Sue Jones is enrolled in Python 100

---- Course Registration Program ----
Select from the following menu:
1. Register a Student for a Course.
2. Show current data.
3. Save data to a file.
4. Exit the program.

What would you like to do: 1
Enter the student's first name: Sue
Enter the student's last name: Jones
Please enter the name of the course: Python 100

---- Course Registration Program ----
Select from the following menu:
1. Register a Student for a Course.
2. Show current data.
3. Save data to a file.
4. Exit the program.

What would you like to do: 4
Program Ended

Process finished with exit code 0
```

Figure 10 – FileNotFoundError Handling

The second error handling test involved checking the condition where the user enters non-alphabetic characters for the first or last names (see Figure 11). The Enrollments.json file was also correctly updated

```
"D:\UW\IT FDN 110 A\Module06\Module06\Assignment\Assignment06\.env\Scripts\python.exe" "D:\UW\IT FDN 110 A\Module06\Module06\Assignment\Assignment06\Assignment06.py"
---- Course Registration Program ----
Select from the following menu:
1. Register a Student for a Course.
2. Show current data.
3. Save data to a file.
4. Exit the program.
-----
What would you like to do: 1
Enter the student's first name: 1

-- Technical Error Message --
The last name should not contain numbers.
Inappropriate argument value (of correct type).
<class 'ValueError'>

---- Course Registration Program ----
Select from the following menu:
1. Register a Student for a Course.
2. Show current data.
3. Save data to a file.
4. Exit the program.
-----
What would you like to do: 1
Enter the student's first name: Michael
Enter the student's last name: Scott
Please enter the name of the course: Python 100

---- Course Registration Program ----
Select from the following menu:
1. Register a Student for a Course.
2. Show current data.
3. Save data to a file.
4. Exit the program.
-----
What would you like to do: 1
Enter the student's first name: Dwight
Enter the student's last name: 2

-- Technical Error Message --
The last name should not contain numbers.
Inappropriate argument value (of correct type).
<class 'ValueError'>

---- Course Registration Program ----
Select from the following menu:
1. Register a Student for a Course.
2. Show current data.
3. Save data to a file.
4. Exit the program.
-----
What would you like to do: 1
Enter the student's first name: Dwight
Enter the student's last name: Schrute
Please enter the name of the course: Python 100

---- Course Registration Program ----
Select from the following menu:
1. Register a Student for a Course.
2. Show current data.
3. Save data to a file.
4. Exit the program.
-----
What would you like to do: 2
Student Bob Smith is enrolled in Python 100
Student Sue Jones is enrolled in Python 100
Student Michael Scott is enrolled in Python 100
Student Dwight Schrute is enrolled in Python 100
-----

---- Course Registration Program ----
Select from the following menu:
1. Register a Student for a Course.
2. Show current data.
3. Save data to a file.
4. Exit the program.
-----
What would you like to do: 3
Student Bob Smith is enrolled in Python 100
Student Sue Jones is enrolled in Python 100
Student Michael Scott is enrolled in Python 100
Student Dwight Schrute is enrolled in Python 100
-----

---- Course Registration Program ----
Select from the following menu:
1. Register a Student for a Course.
2. Show current data.
3. Save data to a file.
4. Exit the program.
-----
What would you like to do: 4
Program Ended

Process finished with exit code 0
```

Figure 11 – Non-Alphabetic Characters Error Handling

GitHub

Script and documentation for this assignment is available in my GitHub site:

<https://github.com/rfnaval/IntroToProg-Python-Mod06.git>

Summary

This assignment built upon the previous one, providing an opportunity to learn and practice key Python concepts such as functions, function parameters, return values, and classes. Functions and classes are essential building blocks of the Separation of Concerns (SoC) principle in Python programming. By applying the SoC design principle, I was able to create a more encapsulated and organized script, significantly reducing the complexity of the main body of the code while improving its readability.

References

1. Module 06 - Functions, Randal Root, January 02, 2024.
2. External site: [Let's Learn Python - Basics #6 of 8 - Functions](#), Anchor Rainbow.
3. External site: [Python Functions || Python Tutorial || Learn Python Programming](#), Socratica.