

Deric Feters

6/8/2024

IT FDN 110 A

Assignment 06

Functions

Introduction

Module 5 most notably introduces working with global vs local variables, classes, functions, parameters and return values. These topics improve the functionality, usability and shareability of our code.

Classes

Classes were utilized in this assignment to organize and modularize functions. Separation of concerns organizes functions into three categories: data layer, processing layer and presentation layer. Data layer handles data-related operations such as data storage and retrieval. Class ProcessFileData represents the data layer and contains the functions for reading and writing data to files. Class IO represents the presentation layer and contains functions for presenting information and collecting user input. In addition, the @staticmethod decorator was used to allow the calling of functions without creating a object. See Figure 1.

```
28 # Processing -----#
29 > class ProcessFileData:
30 >     """This class is a collection of processing functions that work with json files..."""
31 >     1 usage
32 >
33 >     @staticmethod
34 >     def read_data_from_file(file_name: str, students: list):...
35 >
36 >     1 usage
37 >
38 >     @staticmethod
39 >     def write_data_to_file(file_name: str, students: list):...
40 >
41 >
42 >
43 >
44 >
45 >
46 >
47 >
48 >
49 >
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 >
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 1: Assignment06.py Class organization (some code hidden)

Functions

Functions were used for modularity and reusability of code. The main program was greatly simplified by moving logic to functions and then calling those functions from the main program. See Figure 2.

```
187 while (True):
188     # Present the menu of choices
189     IO.output_menu(menu=MENU)
190     # Request menu input from the user
191     menu_choice = IO.input_menu_choice()
192
193     # Input user data
194     if menu_choice == "1": # This will not work if it is an integer!
195         students = IO.input_student_data(students=students)
196         continue
197
198     # Present the current data
199     elif menu_choice == "2":
200         # Process the data to create and display a custom message
201         IO.output_student_courses(students)
202         continue
203
204     # Save the data to a file
205     elif menu_choice == "3":
206         ProcessFileData.write_data_to_file(file_name="FILE_NAME", students)
207         continue
208
209     # Stop the loop
210     elif menu_choice == "4":
211         break # out of the loop
212     else:
213         print("Please only choose option 1, 2, or 3")
214
215 print("Program Ended")
```

Figure 2: Assignment06.py Main Program

Function `IO.output_error_messages` is the best example of code reusability being called from the `read_data_from_file`, `write_data_to_file` and `input_student_data` functions. See Figure 3.

```
40 def read_data_from_file(file_name: str, students: list):
41     """This function reads the data from the Enrollments.json file and returns list students..."""
42
43     try:
44         file = io.TextIOWrapper
45         file = open(file_name, "r")
46         students = json.load(file)
47         file.close()
48     except FileNotFoundError as e:
49         IO.output_error_messages(message="Text file must exist before running this script", error=e)
50     except Exception as e:
51         IO.output_error_messages(message="Error: There was a problem with reading the file.", error=e)
52     finally:
53         if file.closed == False:
54             file.close()
55     return students
56
57 1 usage
58 @staticmethod
59 def write_data_to_file(file_name: str, students: list):
60     """This function writes the data to the Enrollments.json file..."""
61
62     try:
63         file = open(file_name, "w")
64         json.dump(students, file)
65         file.close()
66         print("The following data was saved to file!")
67         for student in students:
68             print(f'Student {student["FirstName"]} '
69                   f'{student["LastName"]} is enrolled in {student["CourseName"]}')
70     except Exception as e:
71         if file.closed == False:
72             file.close()
73
74     IO.output_error_messages(message="Error: There was a problem with writing to the file. \n"
75                               "Please check that the file is not open by another program.", error=e)
```

Figure 3: Assignment06.py IO.output_error_messages

Global vs Local Variables, Parameters, Arguments, Return Values

The use of global and local variables, parameters, arguments and return values has often caused me issues when programming. In this assignment, I was able to cut down global variables and constants down to four. See Figure 4.

```
11 # Define the Data Constants
12 MENU: str = ''
13
14 ---- Course Registration Program ----
15 Select from the following menu:
16 1. Register a Student for a Course.
17 2. Show current data.
18 3. Save data to a file.
19 4. Exit the program.
20 -----
21
22 FILE_NAME: str = "Enrollments.json"
23
24 # Define the Data Variables and constants
25 students: list = [] # a table of student data
26 menu_choice: str = "" # Hold the choice made by the user.
```

Figure 4: Assignment06.py Global Variables

All other variables such as “student_first_name” and “student_last_name” were local variables within functions. See Figure 5 for an example.

```
154 def input_student_data(students: list):
155     """ This function inputs student data from the user
156
157     Change Log: (Who, When, What)
158     DFetters, 05.30.2024, Created function
159
160     """
161
162     try:
163         student_first_name = input("\n Enter the student's first name: ")
164         if not student_first_name.isalpha():
165             raise ValueError("\nThe last name should not contain numbers.")
166         student_last_name = input("\n Enter the student's last name: ")
167         if not student_last_name.isalpha():
168             raise ValueError("\nThe last name should not contain numbers.")
169         course_name = input("\n Please enter the name of the course: ")
170         student_data = {"FirstName": student_first_name,
171                        "LastName": student_last_name,
172                        "CourseName": course_name}
173         # adds new data to students list
174         students.append(student_data)
175         print(f"You have registered {student_first_name} {student_last_name} for {course_name}.")
176     except ValueError as e:
177         IO.output_error_messages(message="", error=e)
178     except Exception as e:
179         IO.output_error_messages(message="Error: There was a problem with your entered data.", error=e)
180     return students
```

Figure 5: Assignment06.py Local Variables

To update the global list “students” with new student data, the value of global list “students” was first passed to the “input_student_data” function as an argument where it was appended with the data input by the user in a local dictionary, named “student_data”. The updated local list “student_data” was returned by the function. The global list “students” value was then set equal local list “students”, updating the global list with the appended data. See figures 5 and 6.

```
196
197 students = IO.input_student_data(students=students)
198 continue
```

Figure 6: Assignment06.py Global List “students” update

GitHub

A GitHub account with a repository for classwork was created. It can be found at:

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

Summary

In Module 6, I learned about working with global vs local variables, classes, functions, parameters and return values. All of these topics improve the functionality, usability and shareability of our code.