

Exercise 7 for Students of Computer Science

Leon Tabak
l.tabak@ieee.org

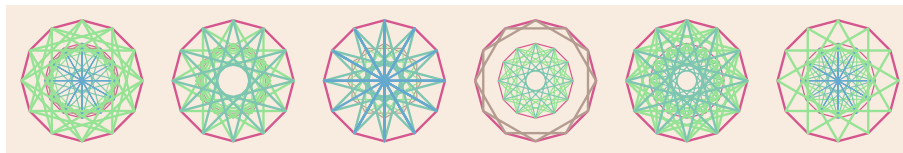
11 August 2021

©Copyright 2021 Leon Hannah Tabak

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.



```
1 # try-logging.py
2 # Leon Tabak
3 # l.tabak@ieee.org
4 # 11 August 2021
5
6 # This program is an example and an exercise with which to introduce
7 # students to Python's logging facility.
8
9 import logging
10
11 def fun():
```

```

12     logging.debug( 'I am fun. I will call my friend more fun.' )
13     logging.info( 'We are having fun.' )
14     more_fun()
15 # end of fun()
16
17 def more_fun():
18     logging.debug( 'I am more fun. I will call my friend most fun.' )
19     logging.info( 'We are having more_fun.' )
20     most_fun()
21 # end of more_fun()
22
23 def most_fun():
24     logging.debug( 'I am most fun. You will not find a better friend!' )
25     logging.info( 'We are having the most_fun.' )
26 # end of most_fun()
27
28 def try_logging():
29     # Call logging.basicConfig() just once.
30     # Call it before calling any other logging functions.
31
32     # Specify what kind of messages to print
33     # by assigning a value to the level parameter.
34     # The choices are:
35     #     * logging.DEBUG
36     #     * logging.INFO
37     #     * logging.WARNING
38     #     * logging.ERROR
39     #     * logging.CRITICAL
40     # Specify the format of the message by assigning
41     # a format string to the format parameter.
42     # This format string can contain a reference to:
43     #     * funcName (Name of function that logged the message.)
44     #     * level (This means DEBUG, INFO, WARNING, etc.)
45     #     * lineno (Number of the line in the program.)
46     #     * message (The content of the log entry.)
47     # These variables (funcName, level, and so on) are enclosed
48     # in parentheses. A percent sign precedes the left parenthesis.
49     # A formatting code ('s' for string, 'd' for decimal integer)
50     # follows the right parenthesis.
51     # The formatting code can optionally include a number.
52     # For example, '16s' means allow 16 characters for a string
53     # and '8d' means allow 8 digits for an integer.
54     formatString = ( '%(levelname)s message in %(funcName)s() ' +
55                     'on line# %(lineno)d: \n\t%(message)s\n' )
56     logging.basicConfig( level = logging.DEBUG,
57                          format = formatString )
58
59     # TO-DO: Try setting the level in the call to
60     # logging.basicConfig() to something other than
61     # logging.DEBUG
62
63     # TO-DO: Try constructing a different formatString
64     # to assign to the format parameter in the call to
65     # logging.basicConfig(). For example, you might choose

```

```

66 # to leave out the level or put the line number first
67 # or fix the width of each part of the log entry.
68 # Fixing the width might mean specifying 8 characters
69 # for the level or 5 digits for the line number.
70
71 # TO-DO: Learn how to use Python's assert statement
72 # by searching on the Web for explanations and examples.
73 # Explain how you might use assert statements with
74 # logging. Could you use assert statements in place
75 # of logging? What do the experts on the Web say?
76
77 # TO-DO: Learn more by reading the documentation
78 # that you will find here:
79 #   https://docs.python.org/3/library/logging.html
80
81 # TO-DO: Add logging to a program that you wrote
82 # as part of another exercise.
83
84 # TO-DO: Learn more by reading the documentation
85 # that you will find here:
86 #   https://docs.python.org/3/library/logging.html
87
88
89 logging.debug( 'Here is help for debugging.' )
90 logging.info( 'Here is some helpful information.' )
91 logging.warning( 'I will give you just a warning this time.' )
92 logging.error( 'I cannot let this one go. You have made an error.' )
93 logging.critical( 'Now you have done it! Your error is critical.' )
94
95     fun()
96 # end of try_logging()
97
98 if __name__ == '__main__':
99     try_logging()

```