Python Logging Basics

by
Ananth Bulusu

What is logging?

- Logging is a means of tracking events that happen when some software runs.
- The software's developer adds logging calls to their code to indicate that certain events have occurred
- The logging functions are named after the level or severity of the events they are used to track. They are
  - debug(), info(), warning(), error() and critical()
- For logging the events in a program, *logging* module should be imported in python

Why to use logging

- In a general/basic python code, whenever a issue/error/warning/info needs to be displayed, it can be displayed in the console using print like below example

```
In [1]: try:
            print(str1)
        except Exception as exp:
            print('Issue occured in the code - ' + str(exp))

        Issue occured in the code - name 'str1' is not defined
```

- But when the code is executing in a project, the above approach will not help due to different reasons like environment restrictions, presence of multiple modules etc.,. In this case, logging will be used

Standard logging levels and their applicability

Level	When it's used
DEBUG	Detailed information, typically of interest only when diagnosing problems.
INFO	Confirmation that things are working as expected.
WARNING	An indication that something unexpected happened, or indicative of some problem in the near future (e.g. 'disk space low'). The software is still working as expected.
ERROR	Due to a more serious problem, the software has not been able to perform some function.
CRITICAL	A serious error, indicating that the program itself may be unable to continue running.

The default level is WARNING, which means that only events of this level and above (below in above table) will be tracked, unless the logging package is configured to do otherwise.

Logging Example

- The logging events that are tracked can be handled in different ways. The simplest way of handling tracked events is to print them to the console. Another common way is to write them to a disk file
- Below is the sample Logging program to display the logging message in console

```
import logging
logging.warning('And this, too')
logging.error('And non-ASCII stuff, too, like Øresund and Malmö')
```

```
WARNING:root:And this, too
ERROR:root:And non-ASCII stuff, too, like Øresund and Malmö
```

- The same logging message can also be saved in a file and this message can also be formatted as per the need

```
import logging as log

log.basicConfig(level=log.DEBUG,filemode='w',filename='file.log',format='%(asctime)s - %(name)s - %(levelname)s - %(message)s')
log.debug('This message should go to the log file')
log.info('So should this')
log.warning('And this, too')
log.error('And non-ASCII stuff, too, like Øresund and Malmö')
```

In the above example, the logging data will be saved in **file.log** file. *mode* accepts values of 'a'/'w' which represents the append or overwrite. Also format represents the data needs to be displayed in the log file. The output file will have as below

```
1  2022-09-16 14:12:58,890 - root - DEBUG - This message should go to the log file
2  2022-09-16 14:12:58,891 - root - INFO - So should this
3  2022-09-16 14:12:58,891 - root - WARNING - And this, too
4  2022-09-16 14:12:58,891 - root - ERROR - And non-ASCII stuff, too, like Øresund and Malmö
```

Integrating Error handling and Logging

- For logging to be more effective, the code with the error handling needs to be integrated with the event logging.
- For successful integration between the error handling and Logging, execution info(exe_info) needs to be used. Below is one example for the same

```
import logging as log

log.basicConfig(level=log.DEBUG,filemode='w',filename='file.log',format='%(asctime)s - %(name)s - %(levelname)s - %(message)s')
try:
    print(str1)
except Exception as exp:
    log.error('Issue occured in the code - ' + str(exp),exc_info=True)
```

The output file will be as below

```
2022-09-16 14:45:23,910 - root - ERROR - Issue occured in the code - name 'str1' is not defined
Traceback (most recent call last):
  File "<ipython-input-1-8573468a9a8d>", line 5, in <module>
    print(str1)
NameError: name 'str1' is not defined
```