test.py:

```python
import logging
import student
logging.basicConfig(filename='test.log',level=logging.DEBUG)
logging.debug('debug message from test module')
```

student.log:
INFO:root:info message from student module

In the above application the configurations performed in test module won't be reflected,b'z root logger is already configured in student module.

Need of Our own customized logger:

The problems with root logger are:

1. Once we set basic configuration then that configuration is final and we cannot change

2. It will always work for only one handler at a time, either console or file, but not both simultaneously

3. It is not possible to configure logger with different configurations at different levels

4. We cannot specify multiple log files for multiple modules/classes/methods.

To overcome these problems we should go for our own customized loggers

Advanced logging Module Features: Logger:

Logger is more advanced than basic logging.
It is highly recommended to use and it provides several extra features.

Steps for Advanced Logging:

1. Creation of Logger object and set log level

```
logger = logging.getLogger('demologger')
logger.setLevel(logging.INFO)
```

2. Creation of Handler object and set log level
There are several types of Handlers like StreamHandler, FileHandler etc

```
consoleHandler = logging.StreamHandler()
consoleHandler.setLevel(logging.INFO)
```

Note: If we use StreamHandler then log messages will be printed to console