Python Module to find a log pattern

Objective:

Our objective is to create a log file and count the occurrence of different logging levels.

Steps:

- Importing logging module
- Create and Configure logger
- Adding logs
- Count the occurrence of log levels

Logging:

Logging means tracking events when software runs, purpose of a logging is record programs and problems

Logging module:

The logging module in Python is a ready-to-use and powerful module that is designed to meet the needs of beginners as well as enterprise teams. It is used by most of the third-party Python libraries, so you can integrate your log messages with the ones from those libraries to produce a homogeneous log for your application.

Adding logging to your Python program is as easy as this:

```
import logging
```

With the logging module imported, you can use something called a “logger” to log messages that you want to see

Levels in logging:

By default, there are 5 standard levels indicating the severity of events. Each has a corresponding method that can be used to log events at that level of severity. The defined levels, in order of increasing severity, are the following:

- DEBUG
- INFO
- WARNING
- ERROR
- CRITICAL

- **Debug :** These are used to give Detailed information, typically of interest only when diagnosing problems.(numerical value=10)
- **Info :** These are used to confirm that things are working as expected.(numerical value=20)
- **Warning :** These are used an indication that something unexpected happened, or is indicative of some problem in the near future.(numerical value=30)
- **Error :** This tells that due to a more serious problem, the software has not been able to perform some function.(numerical value=40)
- **Critical :** This tells serious error, indicating that the program itself may be unable to continue running.(numerical value=50)

Creating and configure logging:

1. Create and configure the logger. It can have several parameters. But importantly, pass the name of the file in which you want to record the events.
2. Here the format of the logger can also be set. By default, the file works in **append** mode but we can change that to write mode if required.
3. Also, the level of the logger can be set which acts as the threshold for tracking based on the numeric values assigned to each level.
   There are several attributes which can be passed as parameters.
4. The list of all those parameters is given in Python Library. The user can choose the required attribute according to the requirement.

After creating the logger add the logs using the log levels.

```
import logging

#now we will Create and configure logger
logging.basicConfig(filename="log_test.log",
                    level=logging.DEBUG,
                    format='%(name)s|%(levelname)s |%(message)s|%(asctime)s',
                    filemode='w',datefmt="%d-%m-%y %H-%M-%S")
def log_test():
    logging.debug("this is a debug message")
    logging.info("this is a information message")
    logging.warning("this is a warning message")
    logging.error("this is a indication of error")
    logging.critical("something critical happened")
log_test()
```

Output log file:

log_test.log

```
root|DEBUG |this is a debug message|23-03-22 12-06-16
root|INFO |this is a information message|23-03-22 12-06-16
root|WARNING |this is a warning message|23-03-22 12-06-16
root|ERROR |this is a indication of error|23-03-22 12-06-16
root|CRITICAL |something critical happened|23-03-22 12-06-16
```

Counting the occurrence of a log level:

```python
import os
#reading the log file
file=open("log_test.log","r")
data=file.read()
#counting the occurence
def occurence(a,b):
    occurence_a=data.count(a)
    print("no of occurenceof" ,a,occurence_a)
    occurence_b=data.count(b)
    print("no of occurenceof ",b,occurence_b)
occurence("ERROR","WARNING")

#printing the required log level
file=open("log_test.log","r")
levelname="INFO"
for line in file:
    if levelname in line:
        print(line)
```

Output:

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL

Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS C:\Users\91984\Desktop\python> & C:/python/python.exe c:/Users/91984/Desktop/python/log_test.py
no of occurenceof ERROR 1
no of occurenceof  WARNING 1
root|INFO |this is a information message|23-03-22 13-36-04
```

Advanced level of logging:

The logging library takes a modular approach and offers several categories of components: loggers, handlers, filters, and formatters.

- Loggers expose the interface that application code directly uses.
- Handlers send the log records (created by loggers) to the appropriate destination.
- Filters provide a finer grained facility for determining which log records to output.
- Formatters specify the layout of log records in the final output.

Log event information is passed between loggers, handlers, filters and formatters in a LogRecord instance.

Logging is performed by calling methods on instances of the Logger.

Conclusion:

A test log file is created and occurrence of a required log level is counted successfully.