Let's Debug Django like pro

Backend Engineer @Continual Engine

Open source contributor | Mentor | Building tech communities ()

AGENDA

- Debugging Techniques
- What is Logging ?
- Print Vs Logger
- Deep diving into Logger
- Configuring with Django settings.py
- Developing a visualisation using logger data
- Conclusion

Let's Begin!

DEBUGGING TECHNIQUES

Identify -> Analyze -> Remove

- Using output statements
- Using Flags
- Comments
- Brute Force
- Using Debugging Tools

LOGGING AND LOGGER

- **Logging** is basically tracking events once a code/software runs. It is very essential for debugging your software
- Maintaining a log file helps is fixing crashes faster
- **Logger** is a built-in python module that allows debugging and recording status messages

PRINT VS LOGGER

- Print statements works for small scale
- With Print statements it's difficult to control the log information one want to capture
- Cannot map the severity level with print statements
- Cannot map the timestamp with the message emitted

Deep Dive into Logger (04 Pillars)

LOGGER

- A logger is the entry point into the logging system.
- A logger is configured to have a log level

HANDLER

- Handler defines the destination of the log record
- For example – To a file, Console

FILTER

- Filters provide a finer grained facility for determining which log records to output

FORMATTER

- Specifies layer/entities of record in the output

MORE ABOUT LOGGER LEVELS

- **DEBUG** – Information used for debugging purposes
- **INFO** – General message
- **WARNING** – Information used for Minor fault occurrence
- **ERROR** – Information used for Major problem occurrence
- **CRITICAL** – Information used for Critical issue occurrence

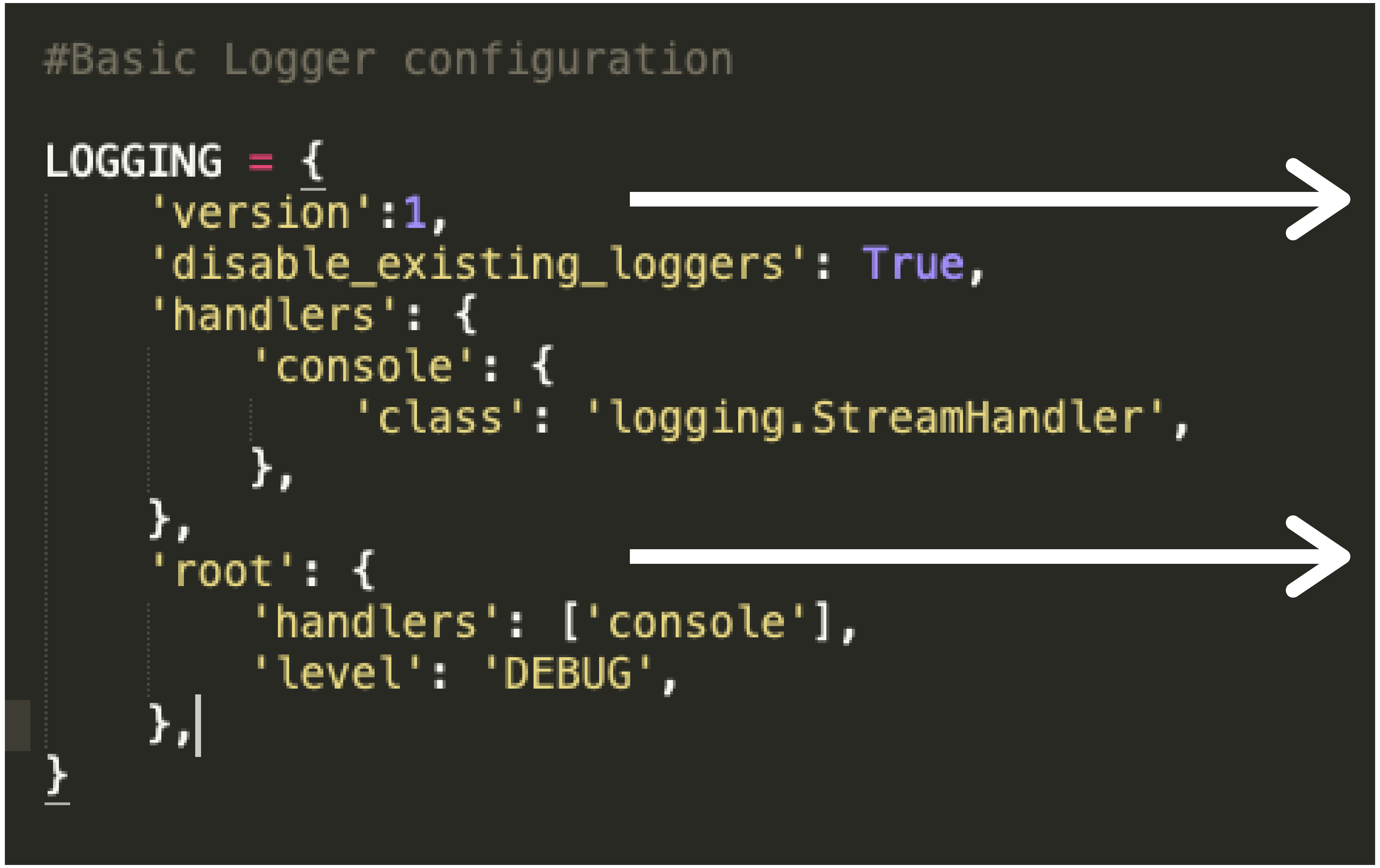
EXAMPLE USAGE

- **DEBUG** – logger.debug()
- **INFO** – logger.info()
- **WARNING** – logger.warning()
- **ERROR** – logger.error()
- **CRITICAL** – logger.critical()

BUILDING LOGGER IN DJANGO

STEP BY STEP

BASIC CONFIGURATION
(SETTINGS.PY)

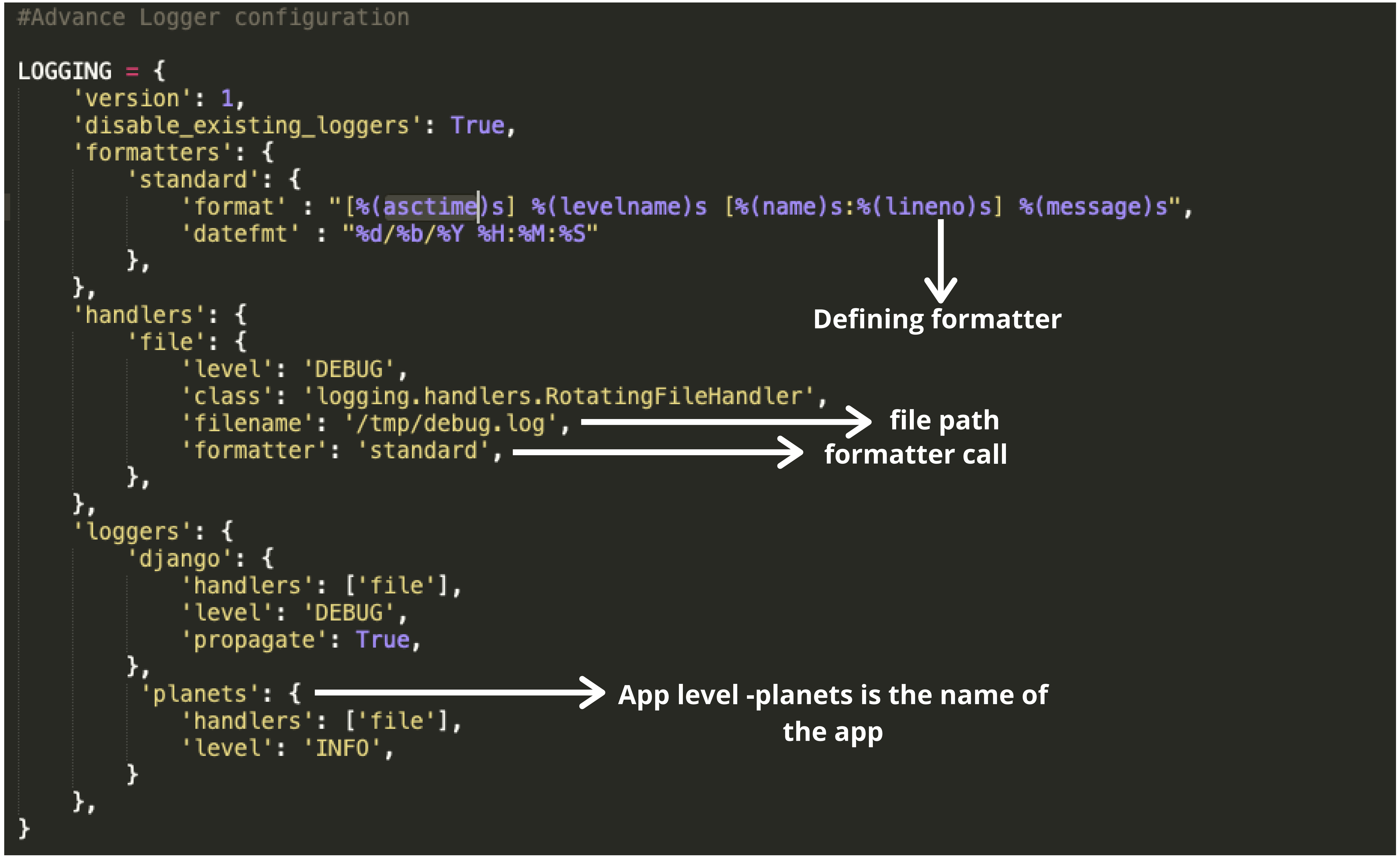


```
#Basic Logger configuration

LOGGING = {
    'version':1,
    'disable_existing_loggers': True,
    'handlers': {
        'console': {
            'class': 'logging.StreamHandler',
        },
    },
    'root': {
        'handlers': ['console'],
        'level': 'DEBUG',
    },
}
```

integer-> representing the schema version

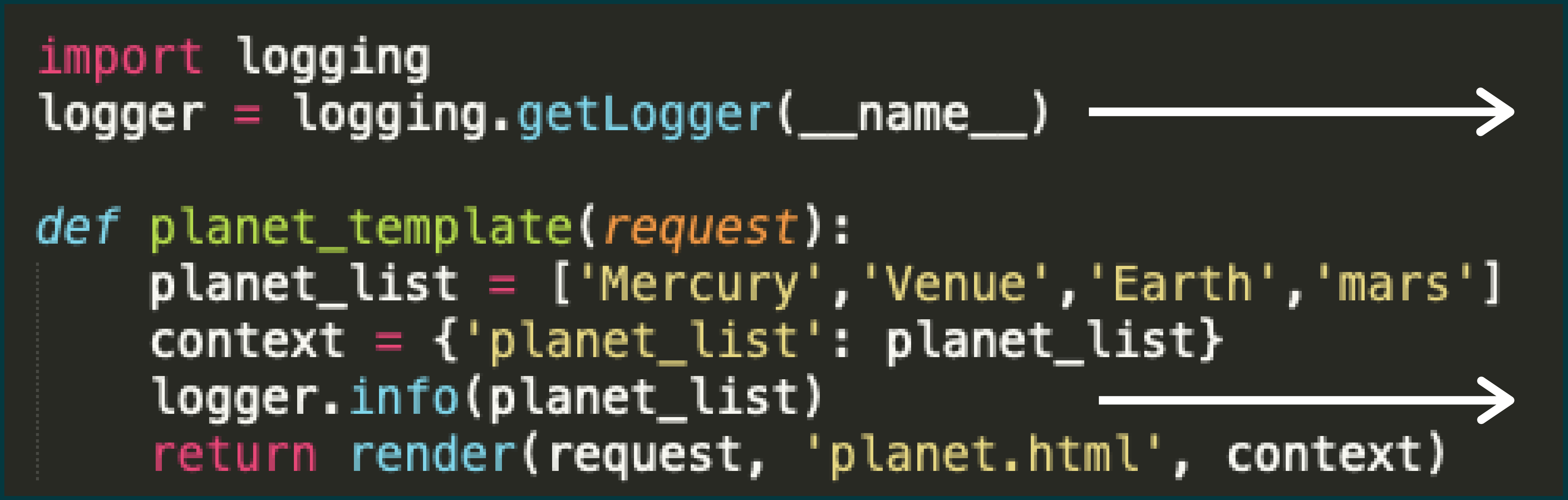
configures root/parent logger to log all the messages in console level with level DEBUG or higher

ADVANCE CONFIGURATION



```
#Advance Logger configuration

LOGGING = {
    'version': 1,
    'disable_existing_loggers': True,
    'formatters': {
        'standard': {
            'format' : "[%(asctime)s] %(levelname)s [%(name)s:%(lineno)s] %(message)s",
            'datefmt' : "%d/%b/%Y %H:%M:%S"
        },
    },
    'handlers': {
        'file': {
            'level': 'DEBUG',
            'class': 'logging.handlers.RotatingFileHandler',
            'filename': '/tmp/debug.log',
            'formatter': 'standard',
        },
    },
    'loggers': {
        'django': {
            'handlers': ['file'],
            'level': 'DEBUG',
            'propagate': True,
        },
         'planets': {
            'handlers': ['file'],
            'level': 'INFO',
        }
    },
}
```

Output of logger



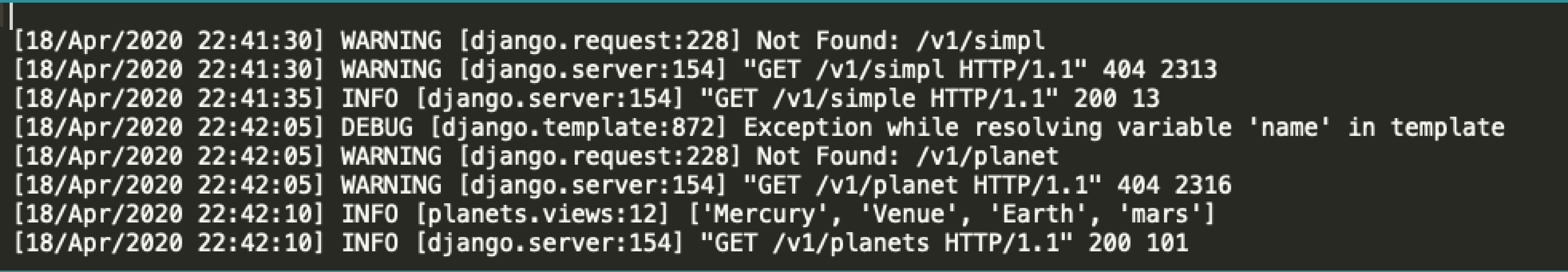
```python
import logging
logger = logging.getLogger(__name__)

def planet_template(request):
    planet_list = ['Mercury','Venue','Earth','mars']
    context = {'planet_list': planet_list}
    logger.info(planet_list)
    return render(request, 'planet.html', context)
```

Instance of a logger with default __name__

creating a log

```
[18/Apr/2020 21:58:55] INFO [planets.views:19]
['Mercury', 'Venue', 'Earth', 'mars']
```

LOGGER FILE PREVIEW



```
[18/Apr/2020 22:41:30] WARNING [django.request:228] Not Found: /v1/simpl
[18/Apr/2020 22:41:30] WARNING [django.server:154] "GET /v1/simpl HTTP/1.1" 404 2313
[18/Apr/2020 22:41:35] INFO [django.server:154] "GET /v1/simple HTTP/1.1" 200 13
[18/Apr/2020 22:42:05] DEBUG [django.template:872] Exception while resolving variable 'name' in template
[18/Apr/2020 22:42:05] WARNING [django.request:228] Not Found: /v1/planet
[18/Apr/2020 22:42:05] WARNING [django.server:154] "GET /v1/planet HTTP/1.1" 404 2316
[18/Apr/2020 22:42:10] INFO [planets.views:12] ['Mercury', 'Venue', 'Earth', 'mars']
[18/Apr/2020 22:42:10] INFO [django.server:154] "GET /v1/planets HTTP/1.1" 200 101
```

DEVELOPING INSIGHTS FROM LOGGER FILE

GETTING OUR DATA READY

STORE DATA IN DESIRED FORMAT

tail -f /tmp/debug.log >out.txt

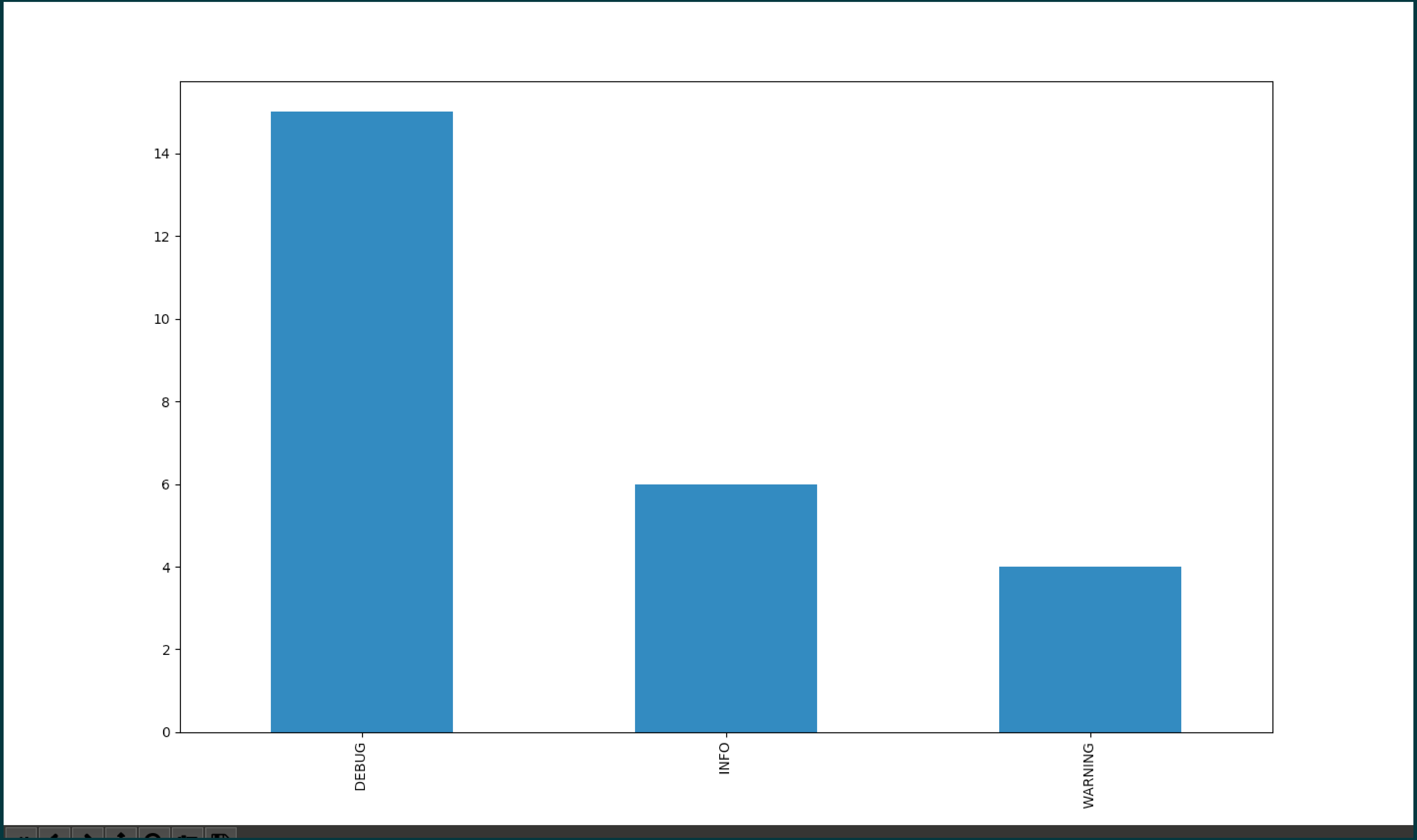
CONVERSION

To CSV/TSV etc

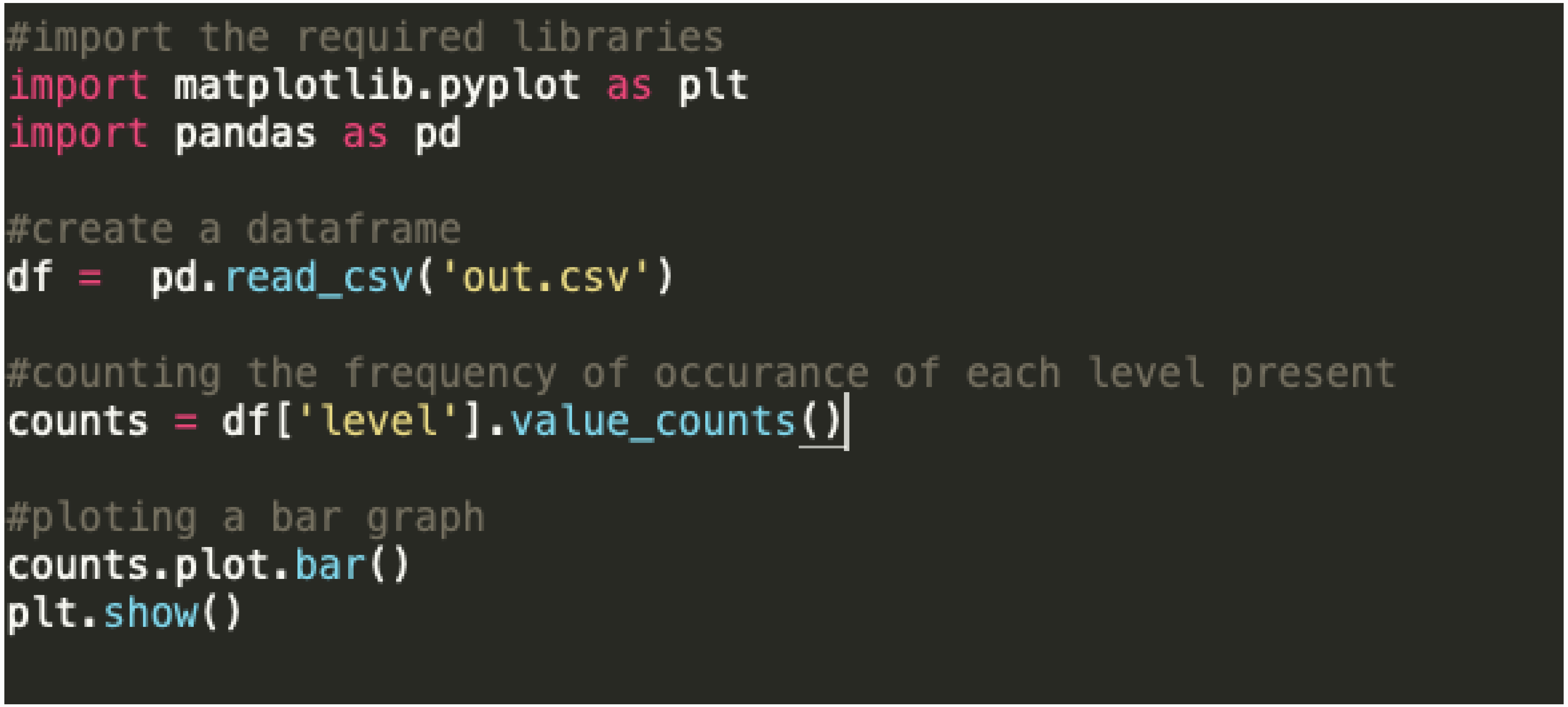
CHOOSE THE DATA POINTS

BUILD (^_^) INSIGHTFUL VISULIZATION

BAR GRAPH - LOGGING LEVEL VS FREQUENCY



DIVING INTO CODE



```
#import the required libraries
import matplotlib.pyplot as plt
import pandas as pd

#create a dataframe
df =  pd.read_csv('out.csv')

#counting the frequency of occurance of each level present
counts = df['level'].value_counts()

#ploting a bar graph
counts.plot.bar()
plt.show()
```

THAT'S NOT ALL!

THERE ARE ENDLESS POSSIBILITIES ^@,@^

THANK YOU SO MUCH :)

@SAYABANIK