Python Logging

Group 2

Group Members:

JINWOOK KIM
BEN NGUYEN
FAISAL ALDILAIJAN
SEAN WANG
AHMED AL NABIL

What is a logger?

A logger is an object that helps you log different events systematically. These events can have different levels such as warning, severe, debug, etc. They can also be written to different destinations like files and standard output.

Why use Loggers?

Having log files makes it easier to go back and track the events that lead to a certain error. It is different from printing out statements in the sense that you can specify what level of severity this statement has.

For example, a user requesting a page from flask could have a lower severity level and that could be logged on the server somewhere. However, a 500 (Internal Server Error) could have a higher severity error and it could notify the server admins through email instantly.

Uses

- Keeping track of all the errors that all user received.
- Keeping track of all the server errors.
- Keeping track of business related events.

GCP Example

```
starting build "fd9bf734-32ff-48b1-8977-46819ff2e2c0"
FETCHSOURCE
Fetching storage object:
gs://staging.collegedb-163518.appspot.com/us.gcr.io/collegedb-163518/appengine/default.20170417t184451:latest#1492472792564999
Operation completed over 1 objects/23.6 MiB.
BUILD
Already have image (with digest): gcr.io/cloud-builders/docker
Sending build context to Docker daemon 557.1 kB
Sending build context to Docker daemon 1.114 MB
..
Step 1 : FROM gcr.io/google_appengine/python
latest: Pulling from google_appengine/python
Status: Downloaded newer image for gcr.io/google_appengine/python:latest
8f4058290e98: Pushed
8ba3ad868e1e: Mounted from google-appengine/python
00516b5821a1: Mounted from google-appengine/python
9f7a6903c659: Mounted from google-appengine/openjdk
8d994814627c: Mounted from google-appengine/openjdk
b91b006ae7bc: Pushed
2574035bc614: Mounted from google-appengine/openjdk
latest: digest: sha256:3624c0e4c50e0fc81aa9ca3c61ec49ae31cd8aedc0420ba6e46861a83049472f size: 3253
DONE
```

AWS example

```
2017-04-17 21:02:00 UTC::@:[3362]:LOG: checkpoint starting: time
2017-04-17 21:02:00 UTC::@:[3362]:LOG: checkpoint complete: wrote 1 buffers (0.0%); 0 transaction log file(s) added, 0 removed, 1 recycled;
write=0.000 s, sync=0.002 s, total=0.015 s; sync files=1, longest=0.002 s, average=0.002 s; distance=16384 kB, estimate=16385 kB
2017-04-17 21:07:00 UTC::@:[3362]:LOG: checkpoint starting: time
2017-04-17 21:07:02 UTC::@:[3362]:LOG: checkpoint complete: wrote 20 buffers (0.1%); 0 transaction log file(s) added, 0 removed, 1 recycled;
write=1.917 s, sync=0.011 s, total=1.941 s; sync files=17, longest=0.011 s, average=0.000 s; distance=16481 kB, estimate=16481 kB
2017-04-17 21:09:40 UTC:nat-128-62-59-84.public.utexas.edu(22103):master@collegedb:[31336]:LOG: could not receive data from client: Connection reset
by peer
2017-04-17 21:09:40 UTC:nat-128-62-59-84.public.utexas.edu(21724):master@collegedb:[31335]:LOG: could not receive data from client: Connection reset
by peer
2017-04-17 21:09:40 UTC:nat-128-62-59-84.public.utexas.edu(24960):master@collegedb:[31338]:LOG: could not receive data from client: Connection reset
by peer
2017-04-17 21:09:40 UTC:nat-128-62-59-84.public.utexas.edu(24960):master@collegedb:[31338]:LOG: unexpected EOF on client connection with an open
transaction
2017-04-17 21:47:00 UTC::@:[3362]:LOG: checkpoint complete: wrote 3 buffers (0.0%); 0 transaction log file(s) added, 0 removed, 1 recycled;
write=0.201 s, sync=0.007 s, total=0.222 s; sync files=3, longest=0.007 s, average=0.002 s; distance=16393 kB, estimate=16423 kB
2017-04-17 21:47:41 UTC:nat-128-62-59-84.public.utexas.edu(22834):master@collegedb:[4816]:LOG: could not receive data from client: Connection reset
by peer
2017-04-17 21:47:41 UTC:nat-128-62-59-84.public.utexas.edu(22834):master@collegedb:[4816]:LOG: unexpected EOF on client connection with an open
transaction
```

Heroku Example

```
:10:05.831818+00:00 heroku[router]: at=info method=GET path="/js/index.js" host=freefoodfinders.seanywang.com
request_id=b06c2ac2-f228-4f21-af51-d1f3daf959e3 fwd="128.62.37.177" dyno=web.1 connect=1ms service=7ms status=304 bytes=237
protocol=http
2017-04-20T17:10:05.760174+00:00 heroku[router]: at=info method=GET path="/js/index.js" host=freefoodfinders.seanywang.com
request_id=6175e766-4bd3-4b75-a0a0-67259311304f fwd="128.62.37.177" dyno=web.1 connect=0ms service=16ms status=304 bytes=237
protocol=http
2017-04-20T17:10:05.811338+00:00 heroku[router]: at=info method=GET path="/css/style.css" host=freefoodfinders.seanywang.com
request_id=b8ce0879-f3b0-450e-98c4-c20c817dec1d fwd="128.62.37.177" dyno=web.1 connect=1ms service=2ms status=304 bytes=237
protocol=http
2017-04-20T17:10:05.820381+00:00 heroku[router]: at=info method=GET path="/js/handlebars-v4.0.5.js"
host=freefoodfinders.seanywang.com request_id=438a2507-f5a5-40f6-9a96-8f59518a2ebe fwd="128.62.37.177" dyno=web.1 connect=1ms
service=2ms status=304 bytes=239 protocol=http
2017-04-20T17:10:05.793477+00:00 heroku[router]: at=info method=GET path="/js/moment.js" host=freefoodfinders.seanywang.com
request_id=5ede68cc-d13a-4dba-a852-f195ed93b8d1 fwd="128.62.37.177" dyno=web.1 connect=15ms service=17ms status=304 bytes=239
protocol=http
```

Logging levels in logging.py

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.

From : https://docs.python.org/3/howto/logging.html#logging-basic-tutorial

Logging to File

```
import logging
logging.basicConfig(filename='info.log')
```

Logging Levels Example

https://github.com/aldilaff/cs373-idb-extra-credit/blob/master/logging_levels_examples.py

Logging from different modules

https://github.com/aldilaff/cs373-idb-extra-credit/blob/master/module_1.py

https://github.com/aldilaff/cs373-idb-extra-credit/blob/master/module_2.py

Handlers

“Handlers send the log records (created by loggers) to the appropriate destination.” [1]

Different type of handlers:

- StreamHandler: send log to streams e.g. stdout, sterror, etc.
- FileHandler: send log to disk file
- RotatingFileHandler: writes to file, but if a maxByte and backupCount are specified and are non-zero, it opens up a new log file when the current file size reaches maxByte.
- More examples: https://docs.python.org/3/library/logging.handlers.html

1: https://docs.python.org/2/howto/logging.html

Handler example

https://github.com/aldilaff/cs373-idb-extra-credit/blob/master/logging_handlers_examples.py

Flask Logging Example

https://gist.github.com/ibeex/3257877

Github

https://github.com/aldilaff/cs373-idb-extra-credit