41-logging

April 29, 2016

1 Logging

- Often two types of logging are performed
- during development, may want verbose logging to help debug system
- during production, want to log “important” events, like web hits, major failures, services performed, accounting data
- want one system to handle both needs
- can send logger output to files and streams
- Python logging similiar to Java log4j

```
In [1]: import logging

        # Can instantiate any number of named loggers, and set their log level
        log = logging.getLogger('test')

        def testlog():
            log.critical('critical')
            log.error('error')
            log.warning('warning')
            log.info('info')
            log.debug('debug')

        # the same name will get the same logger
        log2 = logging.getLogger('test')
        log is log2
```

```
Out[1]: True
```

```
In [2]: # only critical events will be logged
        log.setLevel(level = logging.CRITICAL)
        testlog()
```

```
CRITICAL:test:critical
```

```
In [3]: # warning events and everything above
        log.setLevel(level = logging.WARNING)
        testlog()
```

```
CRITICAL:test:critical
ERROR:test:error
WARNING:test:warning
```

```
In [4]: # everything will be logged
        log.setLevel(level = logging.DEBUG)
        testlog()
```

```
CRITICAL:test:critical
ERROR:test:error
WARNING:test:warning
INFO:test:info
DEBUG:test:debug
```

2 Notice the args to debug, info, etc

Logger.debug(msg, *args, **kwargs)

```
In [ ]: # What is the critical difference between these two calls??

        log.debug('debugging system #%d version #%d status=%s' % (34, 104, 'alpha'))
        log.debug('debugging system #%d version #%d status=%s', 34, 104, 'alpha')
```

```
In [ ]: # first statement generates a ton of garbage
```

```
In [ ]: # can get alot of stack info

        def foo():
            print(log.findCaller(stack_info=True))

        foo()
```

```
In [ ]:
```