Logger

Данный модуль (как следует из названия) позволяет создавать нам логи. Лог программы - это фактически ваша карта к пониманию того, что происходит во время запуска вашей программы. Представьте себе простую ситуацию - есть программа (сложная, состоящая из многочисленных файлов, использующая многочисленные фреймворки и т.д.), и жизнеспособность этой программы - ваша первостепенная задача как ее создателя. Вы должны иметь возможность в любой момент времени посмотреть как она работала 5 минут назад, вчера, месяц назад. С какими данными она работала, какой результат выдавала. Возможно пользователи, которые используют вашу программу, используют ее неправильно. И вам нужно все это узнать.

Есть способ, который вы уже знаете - print. Это самая простая команда, которая выводит результат куда-либо (в Maya это Script Editor). Команда хороша, когда в маленькой программе нужно распечатать пару строк информации. Однако, если говорить о профессиональном программировании, на уровне компании этот подход уже не сработает. Представьте, как один из пользователей сообщил вам, чтобы ваш скрипт делает не то, что нужно. Вы все бросаете и идете к этому человеку, в другой кабинет, просите его показать что он делает, смотрите на свои же принты в его Script Editor. В общем делаете лишние телодвижения когда этого можно избежать.

Логгер позволяет записывать информацию в отдельный файл (лог). Этот лог может содержать любую информацию, например имя файла, имя функции, номер строки в коде, какие-то второстепенные данные и все то, что позволит вам прочитать и понять, как отработала ваша программа.

Минимальный сэтап логгера. В данном случае сообщение выводится в ScriptEditor.

```
import logging

logging.basicConfig(level=logging.DEBUG) # set logger message level
logging.debug('This will get logged')
```

Однако логгинг в консоль - не всегда полезно. Более полезно будет создать отдельную папку с файлами logs, куда мы будем записывать всю полезную информацию.

```
import logging
loggingFile = '/some/file/path/test.log'
logging.basicConfig(filename = loggingFile,
                    filemode = 'w',
                    level = logging.INFO,
                    format='[%(module)s.%(funcName)s.%(lineno)d]
%(levelname)s:%(message)s')

logging.info('Some message')
```

Самым интересным аргументом здесь является format, в котором мы можем указать, какую инфорацию мы хотим вывести в лог. Подробнее об этом можно почитать здесь https://docs.python.org/2/library/logging.html#logrecord-attributes.

В логгере есть несколько уровней сообщений - notset, debug, info, warning, error, critical, exception. С помощью приоритета (Level) мы можем установить - сообщения какого плана будут печататься в лог файл. Причем будут печататься сообщения выбранного уровня и всех уровней выше. Например, debug имеет наименьший приоритет (notset это вообще не приоритет - это относится к более сложным логам, когда один логгер может унаследовать другой), если мы установим этот уровень - то распечатаются сообщения всех уровней следующих после debug (включая debug) - info, warning, error, critical, exception. Если мы установим error - то распечатаются сообщения уровня error, critical, exception.

Так получилось что стандартный вывод информации Python Logging в Maya работает иначе, чем просто в Python проекте.

Представим себе обычный питоновский проект с простым логгером, который выводит информацию в файл.

(см на следующей странице)

```
import logging
import os


dir_ = os.path.dirname(__file__)
log_file = os.path.join(dir_, "logs", "maya.log")

logging.basicConfig(filename=log_file,
                    filemode='w',
                    level = logging.ERROR,
                    format='[%(module)s.%(funcName)s.%(lineno)d]
%(levelname)s:%(message)s'
                    )


def foo_A():
    logging.error("Hello")

def main():
    logging.debug("HEHEHE")
    logging.error("HOHOHO")
    logging.warning("CCCCCC")

    foo_A()


if __name__ == "__main__":
    main()
```

Этот вариант для Майки нерабочий. Например, если мы захотим чтобы майка вывела всю информацию в файл - у нас ничего не выйдет, вместо файла майка выведет всю инфу в ScriptEditor, а файла как не было так и не стало.

Это потому, что майка использует свой собственный logger, который работает только со scriptEditor. Для того чтобы наш логгер работал в майке как положено, мы должны явно создать его и прикрутить ему FileHandler.

(см ниже)

```
import logging
import os
import datetime

now = datetime.datetime.now()
year = '{:02d}'.format(now.year)
month = '{:02d}'.format(now.month)
day = '{:02d}'.format(now.day)
hour = '{:02d}'.format(now.hour)
minute = '{:02d}'.format(now.minute)
second = '{:02d}'.format(now.second)


dir_ = os.path.dirname(__file__)
log_file = os.path.join(dir_, "logs",
"{0}_{1}_{2}_{3}_{4}_{5}_maya.log".format(year, month, day, hour, minute, second))

logger = logging.getLogger("MayaLogs")
logger.setLevel(logging.DEBUG)

# clean logger from trash
for hnd in logger.handlers:
    print "Removing {} handler".format(hnd)
    logger.removeHandler(hnd)

file_handler = logging.FileHandler(log_file, mode='w')
file_handler_format = logging.Formatter('[%(module)s.%(funcName)s.%(lineno)d]
%(levelname)s:%(message)s')
file_handler.setFormatter(file_handler_format)

logger.addHandler(file_handler)



def main():

    logger.debug("hello world")
    logger.info("hello world")
    logger.warning("hello world")
    logger.error("hello world")
    logger.critical("hello world")

    file_handler.close()
    logger.removeHandler(file_handler)
```