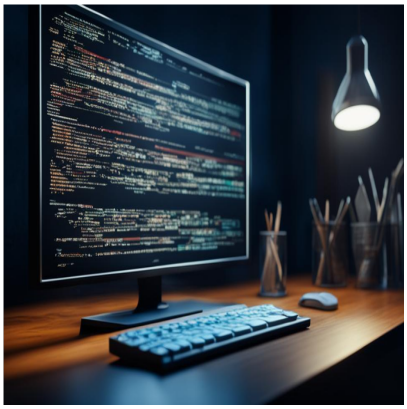


Логирование в Python: полное руководство по освоению модулей и обработке исключений

### Описание

Журналы – это записи, сделанные приложением в процессе его работы. Они полезны для отладки и отслеживания ошибок. Python позволяет легко создавать и записывать журналы. Эта статья представляет собой руководство по созданию журналов в ваших приложениях.

## Что такое протоколирование в Python?

Логирование в Python – это практика генерации логов для приложений, написанных на Python. Хотя для этого существуют внешние пакеты, проще всего использовать встроенный модуль логирования. Этот модуль невероятно прост в использовании и при этом полезен. Поэтому в этой статье мы будем использовать этот модуль для генерации логов Python.

## Преимущества создания журналов

Это помогает при отладке. Когда пользователи сообщают об ошибке, разработчики могут легко проанализировать журналы, генерируемые системой, чтобы получить более подробную информацию о том, что могло пойти не так.

Он позволяет собирать данные об использовании вашего приложения. Это позволит вам создавать аналитику, например, отслеживать, сколько уникальных IP-адресов посещают ваш веб-сервер Python или какие конечные точки API являются

наиболее популярными.

Собранные данные также показывают, как используются ваши ресурсы, и позволяют планировать потребности в инфраструктуре.

Вы также можете автоматизировать запись при сбоях в работе приложений, чтобы быстро выявить проблемы и ограничить масштабы сбоя.

# Ведение журнала по сравнению с печатью

Функция печати – это простой способ создания журналов для ваших приложений. Как правило, большинство людей отлаживают таким образом свои приложения на производстве. Однако в производстве печать не является идеальным вариантом. Это происходит по двум причинам, которые перечислены ниже:

- Печать записывает данные на стандартный вывод. Чаще всего это терминал. Однако журналы, записанные в терминал, не хранятся постоянно. Однако в производстве вам понадобится записывать журналы на диск, чтобы в будущем их можно было просматривать для аналитики и отладки.
- При записи журналов с помощью модуля протоколирования можно задать уровень приоритета для этих журналов. Однако при печати эти уровни доступны не сразу.

# Модуль протоколирования Python

## Импорт

Модуль Python Logging Module является частью стандартной библиотеки. Поэтому для его использования не нужно устанавливать дополнительные инструменты. Вместо этого вы должны импортировать его, прежде чем сможете использовать. Вот как выглядит оператор импорта.

```
import logging
```

## Уровни ведения журнала

Прежде чем приступить к созданию журналов, необходимо рассказать об уровнях регистрации. При создании журналов существуют различные уровни приоритета для журналов. Когда журналы будут записаны, им будут присвоены уровни

приоритета. Различные уровни, расположенные от самого низкого до самого высокого приоритета, выглядят следующим образом:

1. Отладка (уровень приоритета: 10)
2. Информация (Уровень приоритета: 20)
3. Предупреждение (уровень приоритета: 30)
4. Ошибка (уровень приоритета: 40)
5. Критический (уровень приоритета: 50)

В дополнение к пяти вариантам есть возможность не устанавливать уровень приоритета. Как и ожидалось, журнал без установленного приоритета будет иметь самый низкий рейтинг с уровнем приоритета 0. По умолчанию в журнал заносятся только журналы с уровнем предупреждения или выше.

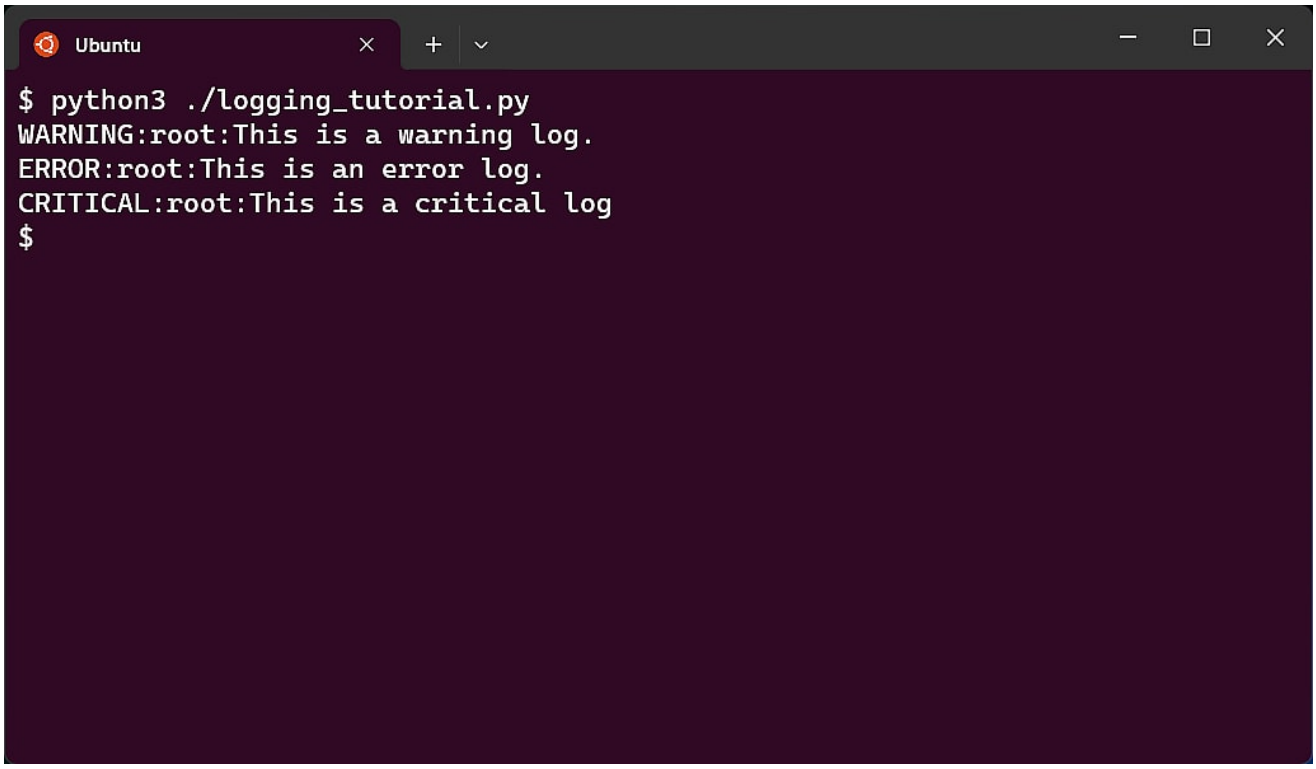
## Создание журналов

Генерировать журналы очень просто. Для каждого уровня приоритета существует соответствующая функция, которую можно вызвать из модуля протоколирования. Эти функции работают одинаково, то есть вы предоставляете строковый аргумент. Эта строка будет использоваться в качестве сообщения журнала. Вот пример, в котором мы вызываем функции журнала для каждого уровня приоритета журнала.

```
logging.debug('This is a debug log.')
logging.info('This is an information log.')
logging.warning('This is a warning log.')
logging.error('This is an error log.')
logging.critical('This is a critical log')
```

Результат будет следующим:

```
WARNING:root:This is a warning log.
ERROR:root:This is an error log.
CRITICAL:root:This is a critical log
```




Из этого можно сделать два вывода:

1. Как уже говорилось, в журнал записываются только журналы WARNING и выше. Это поведение можно настраивать, как мы увидим в следующем разделе.
2. Во-вторых, лог – это сообщение, которое вы указали в качестве аргумента, плюс текст, сообщающий об уровне приоритета.

Используя приведенный выше код, вы можете записывать журналы в терминал. Но это мало чем отличается от использования операторов печати. В последующих разделах будут рассмотрены возможности настройки, которые, помимо прочего, позволяют записывать журналы в файлы.

## Параметры конфигурации

В этом разделе мы рассмотрим настройку модуля протоколирования таким образом, чтобы его функции работали по-другому. Для этого вы вызываете функцию `basicConfig`. Эта функция использует несколько аргументов-ключей (**kwargs) для настройки поведения.

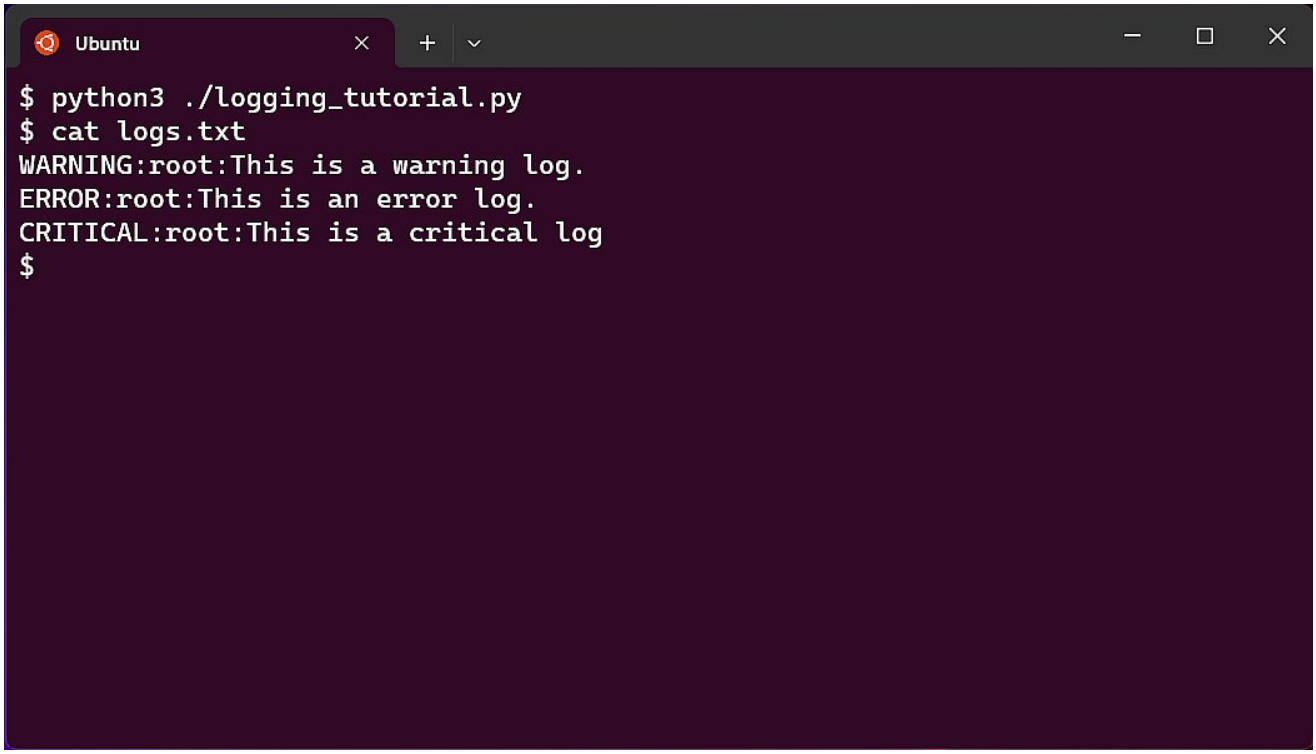
### Указание файла журналов

По умолчанию журналы записываются в терминал; однако если вы хотите записывать журналы в файл, вы можете указать его с помощью ключевого аргумента filename. Помимо имени файла, вы можете указать режим, в котором файл должен быть открыт, с помощью аргумента `filemode`. Вот пример:

```
import logging

logging.basicConfig(filename='logs.txt', filemode='w')
```

В приведенном выше примере мы указали место, куда должны быть записаны журналы, как logs.txt. Кроме того, мы указали режим файла как режим записи. По умолчанию используется режим добавления. Если вы добавите несколько примеров журналов и запустите скрипт, вы увидите, что журналы записываются в указанный файл.



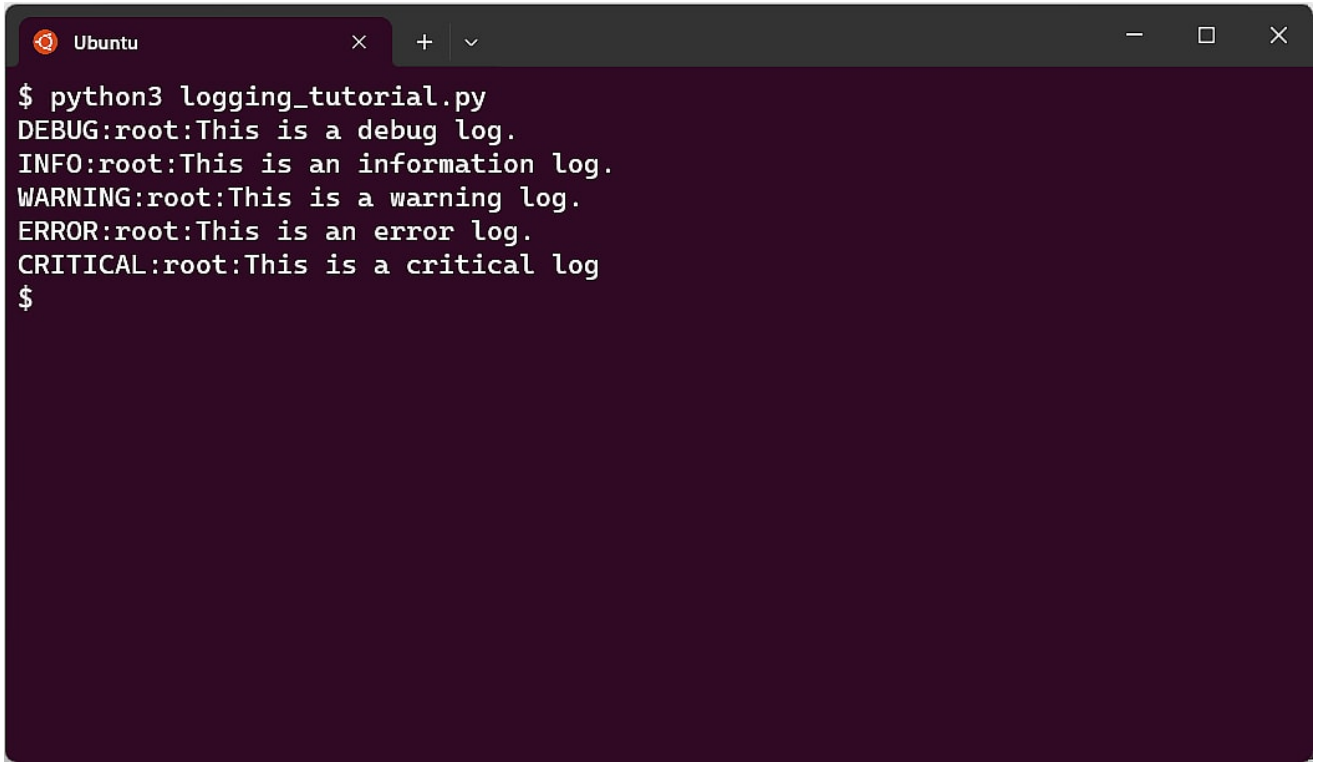

### Настройка приоритета регистрации в журнале

Как уже говорилось, в журнал записываются только журналы уровня WARNING и выше. В этом разделе мы изменим это поведение, чтобы регистрировать все журналы, начиная с уровня DEBUG. Чтобы изменить минимальный уровень журнала, который будет отображаться, вы передаете аргумент с ключевым словом level в функцию `basicConfig`. Значение аргумента представляет собой перечисление

различных уровней протоколирования. Различные значения: DEBUG, INFO, WARNING, ERROR и CRITICAL. Итак, чтобы изменить уровень журнала на DEBUG, мы вызываем функцию `basicConfig` следующим образом:

```
logging.basicConfig(level=logging.DEBUG)
```

Если использовать предыдущий код для регистрации различных значений, то получится что-то вроде этого.



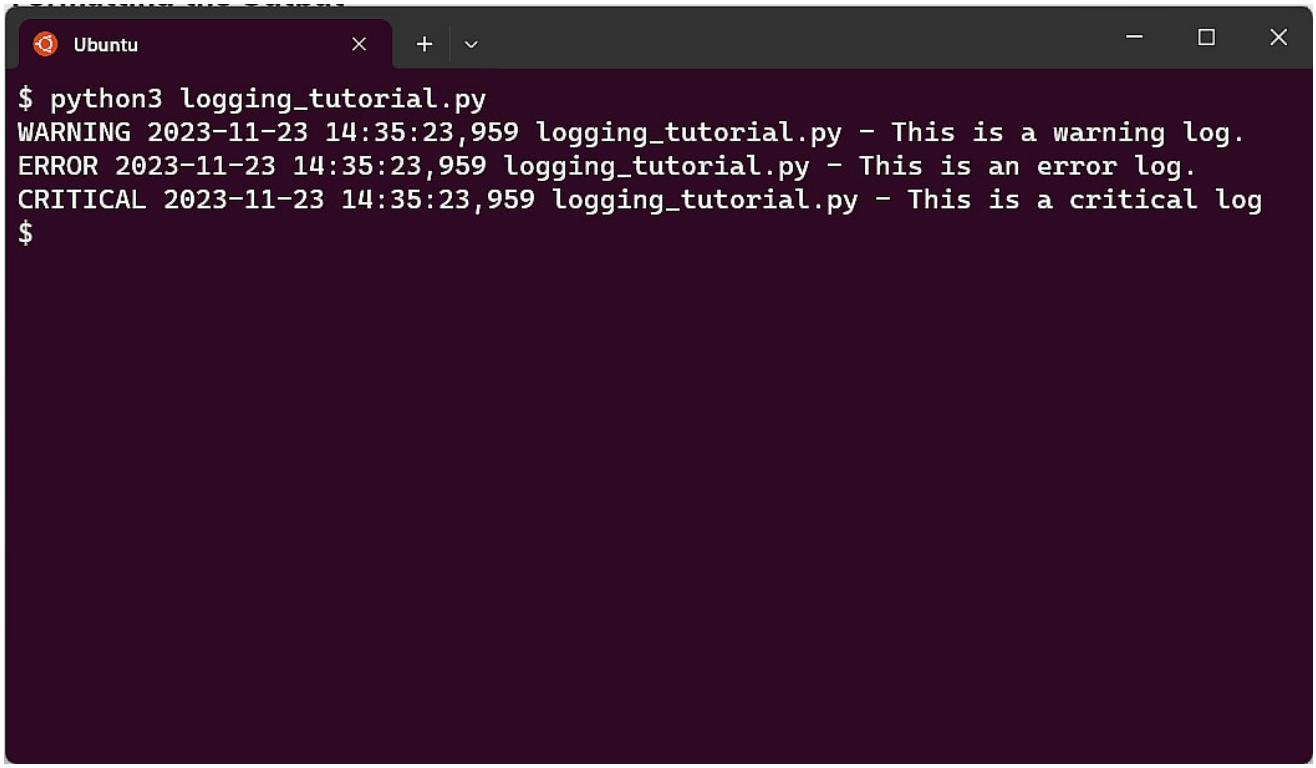

**Форматирование вывода**

Основной журнал содержит уровень и сообщение журнала. Однако вы можете изменить это поведение, чтобы отобразить больше контекста вокруг журнала. Например, вы можете отобразить время создания журнала, файл, сгенерировавший журнал, уровень приоритета журнала и сообщение. Для этого нужно указать формат в качестве аргумента ключевого слова, например:

```
logging.basicConfig(format='%(levelname)s %(asctime)s %(filename)s - %(message
```

Значением аргумента format является шаблон строки, содержащий несколько заполнителей. Например, `%(??? ??????)s`. Эти заполнители будут заменены реальными данными при записи журнала. Полный список заполнителей доступен здесь. Выполнив приведенный выше код, вы должны получить следующий

результат.



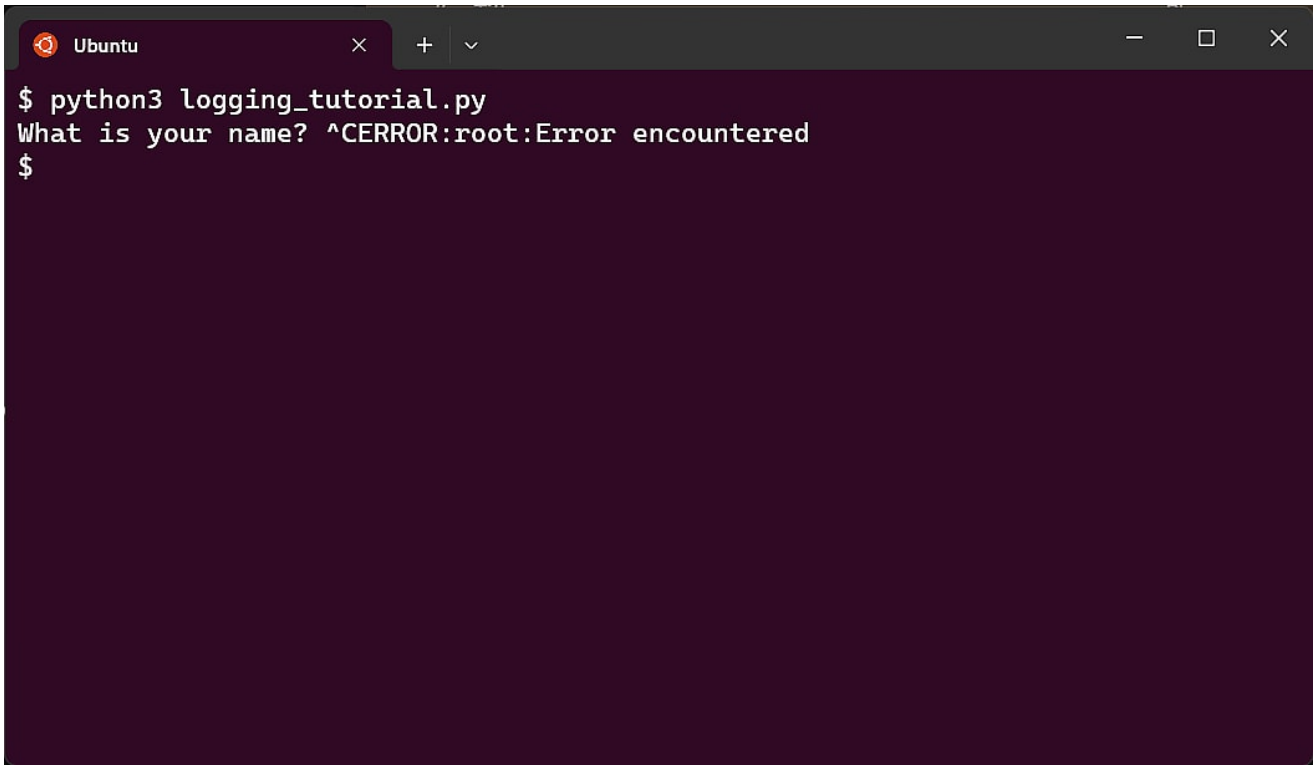

# Журналы при обработке исключений

Одно из лучших мест для использования журналов – это запись ошибок, возникающих внутри блоков try/except. Блоки Try/Except – это конструкции в Python, которые предотвращают аварийное завершение программы при возникновении ошибки. Вот подробное руководство по Try/Except в Python. Чтобы интегрировать протоколирование в обработку исключений, достаточно добавить вызов протоколирования как часть блока except. Например:

```
import logging

try:
    name = input("What is your name? ")
except:
    logging.error('Error encountered')
```

Эта простая программа ожидает ввода от пользователя. Однако если пользователь отменит ввод с помощью Ctrl + C, программа выбросит исключение. Это будет обработано кодом в блоке except. Этот код просто фиксирует, что произошла ошибка. Вот что происходит при запуске программы:




## Заключение

В этой статье мы рассказали о журналах Python, о том, что это такое, почему они полезны и как их генерировать. Далее мы рассмотрели, как настроить стандартное поведение ведения журналов, чтобы выводить больше журналов, записывать журналы в файл и форматировать их определенным образом.

**Дата Создания**
17.04.2024