**2024/01/12 00:00|Лекция. Логирование**

## Логирование

Что такое логирование? Зачем мы это обсуждаем? Проверим мы это все на калькуляторе.

Для работы с калькулятором нам понадобится библиотека. Она есть в стандартном наборе, поэтому устанавливать не придётся. Сделаем «import logging» (Рис.1).

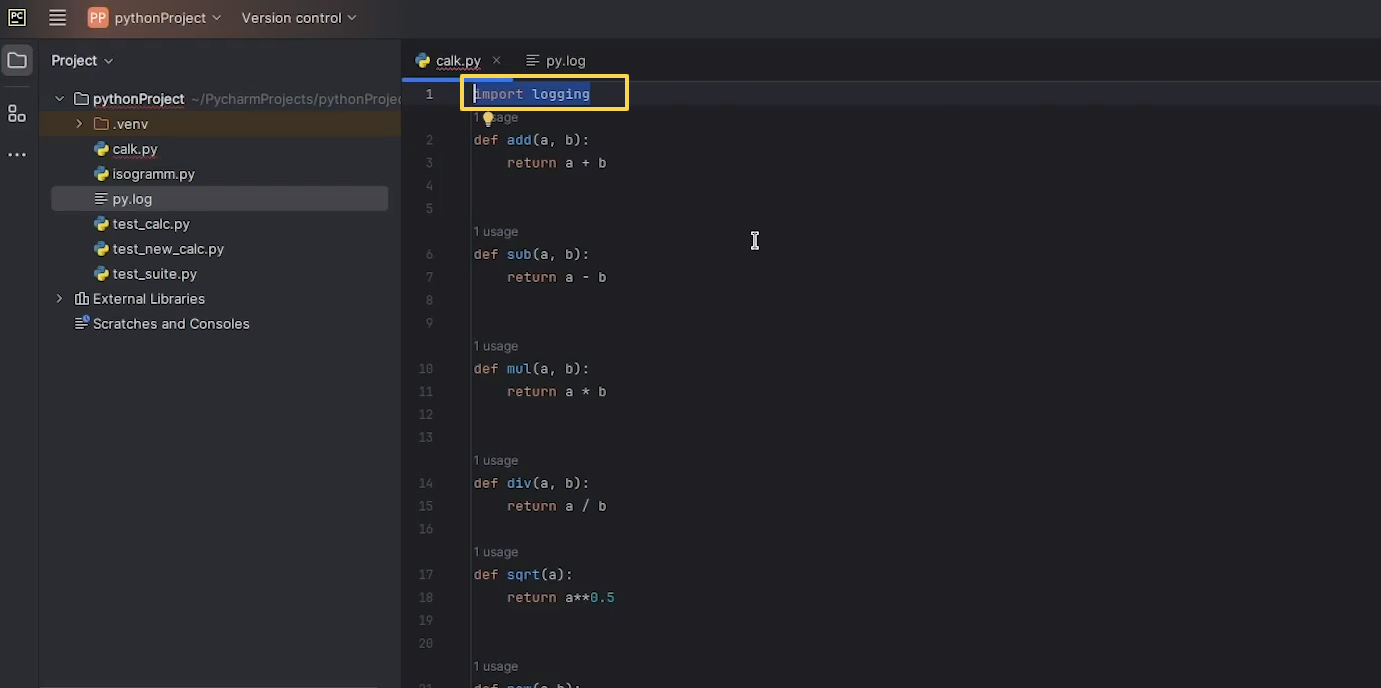


Рис.1

Логирование-это на самом деле очень простой ответ на очень сложную ситуацию. Ситуация следующая: у нас есть пользователь, он пользуется нашим приложением, и что-то у него не работает. Он приходит к нам и говорит, что у него ничего не работает. Что сделает хороший человек среди разработки? Он попросит логин. Именно эту ситуацию мы сегодня с вами будем разруливать на примере одной из функций калькулятора. Логи-это осознанные программистом записи программы, которые хранят информацию о работе программы и записывают там некие шаблоны о том, что происходило с программой. Это позволяет потом легко понять, что случилось с программой, откуда взялась проблема. Логи обычно открывают в случае каких-то проблем. Так что, просто так вы логи вряд ли будете открывать. Поэтому они нужны для проблемных случаев и адаптируются именно для этого.

Как с ними вообще работать? **«Logging» имеет пять уровней сообщений** (Рис.2).В принципе можно добавить и новые, это не проблема. Но есть пять уровней по умолчанию. Скорее всего, вы ими и будете пользоваться. Вам их хватит с запасом:

1. сообщения уровня «loggnig.debug()»;

2. информационные «logging.info()»;

3. сообщения уровня предупреждений «logging.warning()»;

4. сообщения уровня ошибок «logging.error()»;

5. сообщения уровня критических ошибок «logging.critical()», когда все совсем плохо.

Во все можно передавать строки или любые подобные строковые данные.

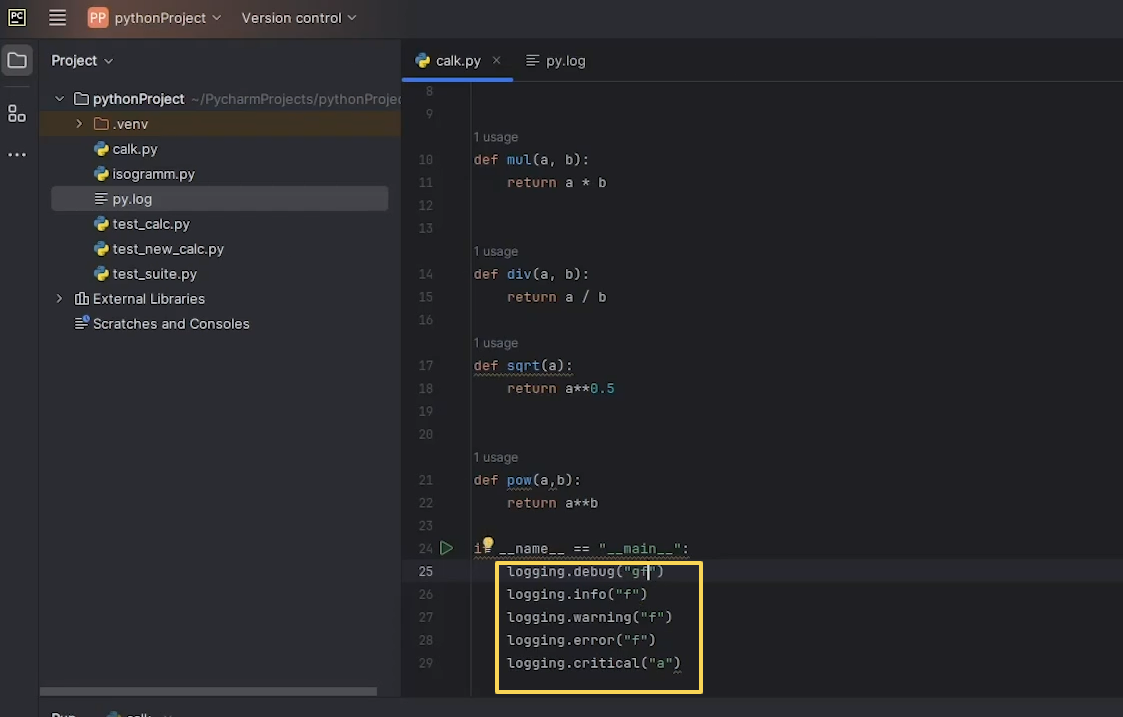


Рис.2

Если мы сейчас просто запустим программу, у нас выведутся только «warning», «error» и «critical» (Рис.3).

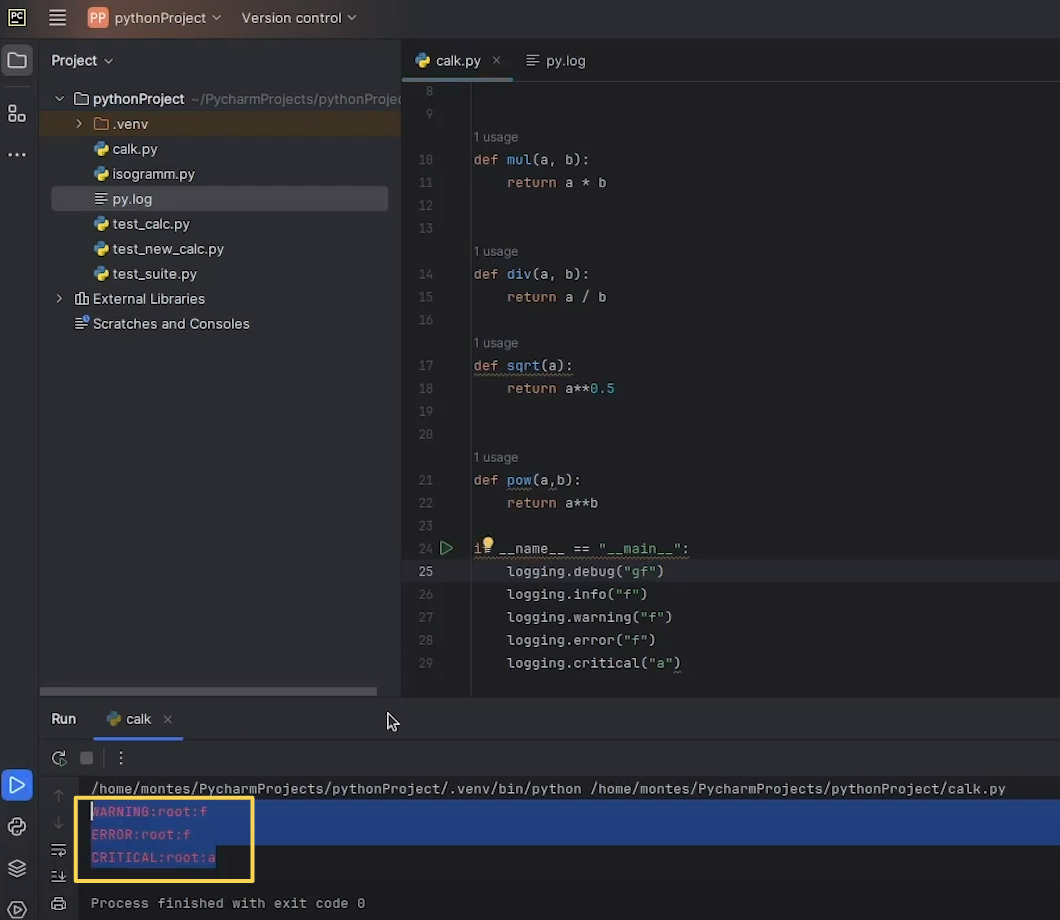


Рис.3

Чтобы точно поверили, напишем «logging.info(“j”)». (Рис.4). И «j» не выводится. Это связано со спецификами.

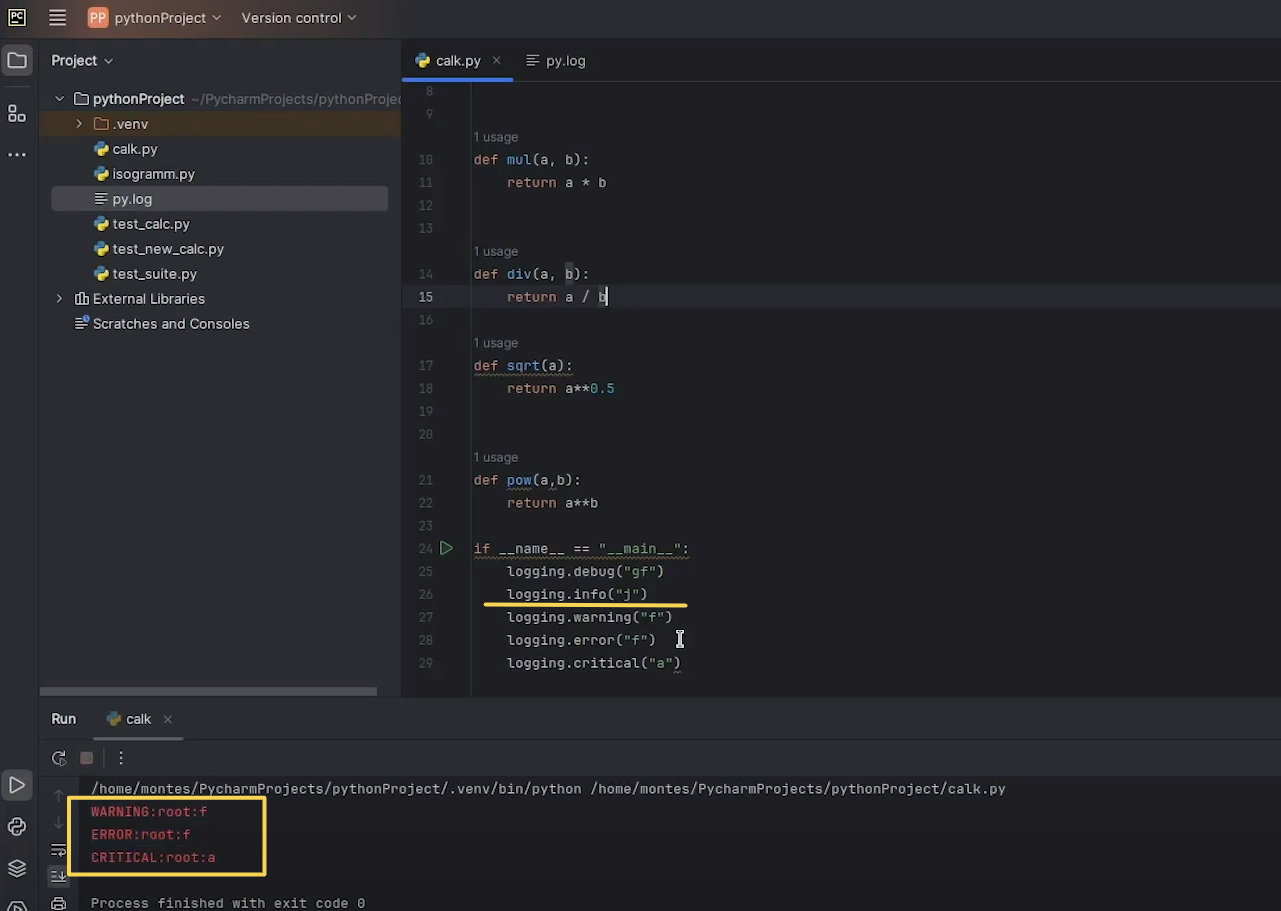


Рис.4

Легко можно настроить, чтобы выводилось все, что вам нужно. Но важно понимать, что сообщения идут по срочности снизу вверх. «Debug»-будничные сообщения о работе, «info» какая-то важная информация о работе, «warning» предупреждение о потенциальных ошибках, «error» конкретные ошибки, «critical» это то, из-за чего могла «лечь» программа.

Чем хороши логи, почему их все используют? Они записываются обычно в файл. И давайте мы с вами это сразу и настроим. Для настройки работы с логами есть специальная функция «basicConfig». И в ней мы указываем несколько параметров. Один из них обязательный «level»-это с какого уровня мы сообщения воспринимаем и записываем в логи. К примеру, с уровня информации. Дальше идет куда мы записываем. Сразу укажем «filemode» запись «write» и куда «filename». Например, мы записываем файлик «py.log» (Рис.5). Логи принято записывать в файлы с расширением «.log». Это просто текстовичок, ничего особенного.

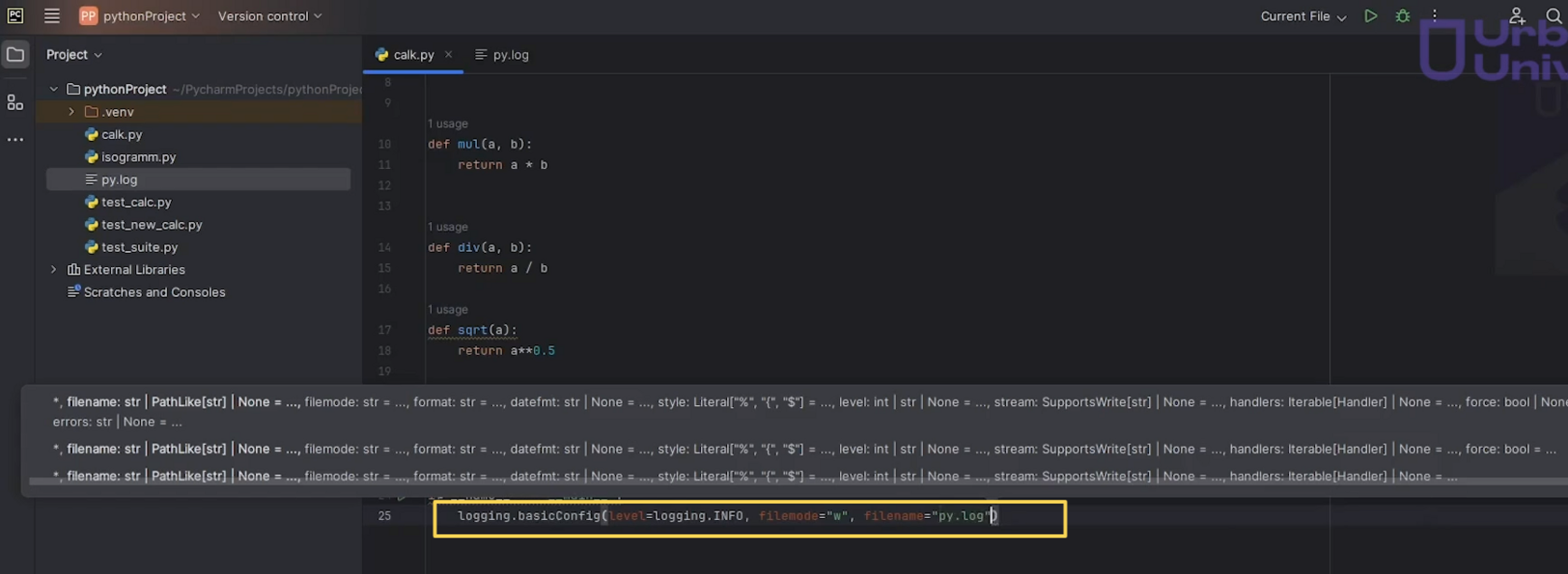


Рис.5

Также ещё один важный элемент это «format». Это на самом деле отдельный элемент, которому можно посвятить несколько занятий. Но не будем сейчас углубляться в него очень надолго. Мы оставим ссылочку на документацию, там можно будет про него прочитать подробно. Нам он нужен сейчас на базовом уровне для того, чтобы мы понимали, для чего он нужен. Он позволяет нам оформить, как именно выводить сообщения из логов. Он использует для этого специальные элементы, к примеру, «asctime». Все эти элементы оформляются при помощи скобочек и сначала знак «%», а в конце буковка «s». И здесь мы, как в строке подставляем эти элементы. Потом ставим разделитель «|». Разделитель можно ставить любой какой вы хотите. Дальше добавляем параметр «levelname». Подобных параметров есть достаточно много, с ними сможете познакомиться в документации, Самые ходовые три параметра мы вам покажем. Добавляем параметр «message» (Рис.6).

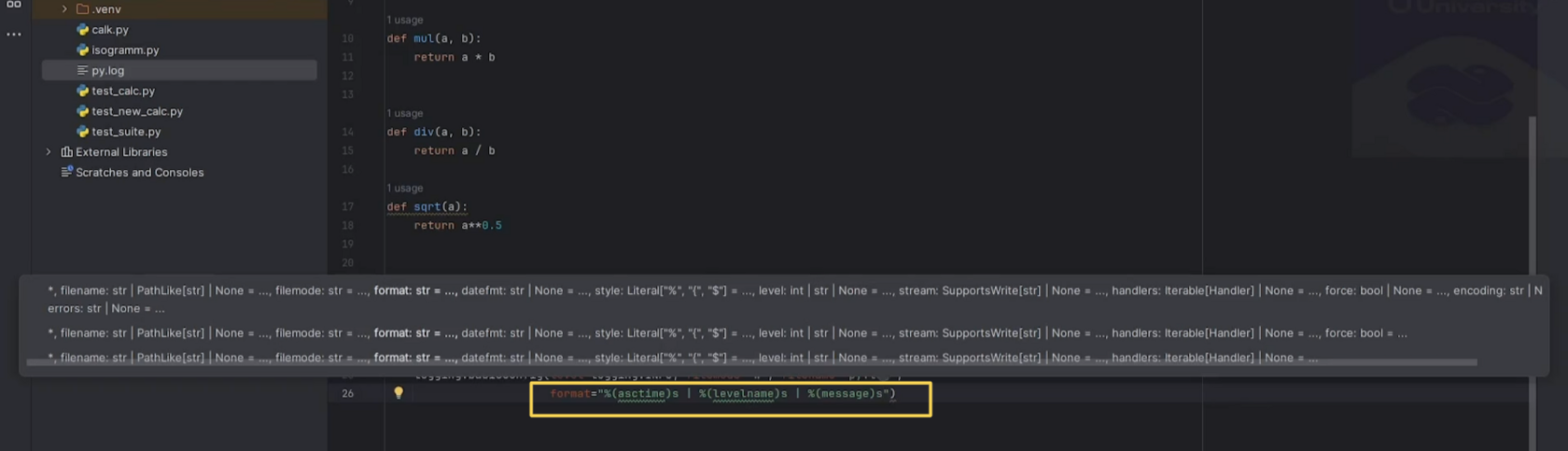


Рис.6

«Аsctime»-время обращения, позволяет понять, когда это произошло. «Levelname»-какого уровня произошло сообщение «info», «debug», «error», «critical» или «warning». А «message»-это то, что мы пишем внутри скобочек функций, то есть, что именно произошло.

Мы с вами разберём все выше написанное на примере функции деления. Какие у нас вообще могут быть случаи поведения с делением?

У нас есть самый стандартный случай, когда мы просто делим «а» на «b», и у нас все получается. И в таком случае мы можем записать себе сообщение «logging.info(f“Succesful divide {a} / {b}”)» (Рис.7). У нас все прекрасно, «а» на «b» разделилось. Соответственно, в логе запишется сообщение об этом.

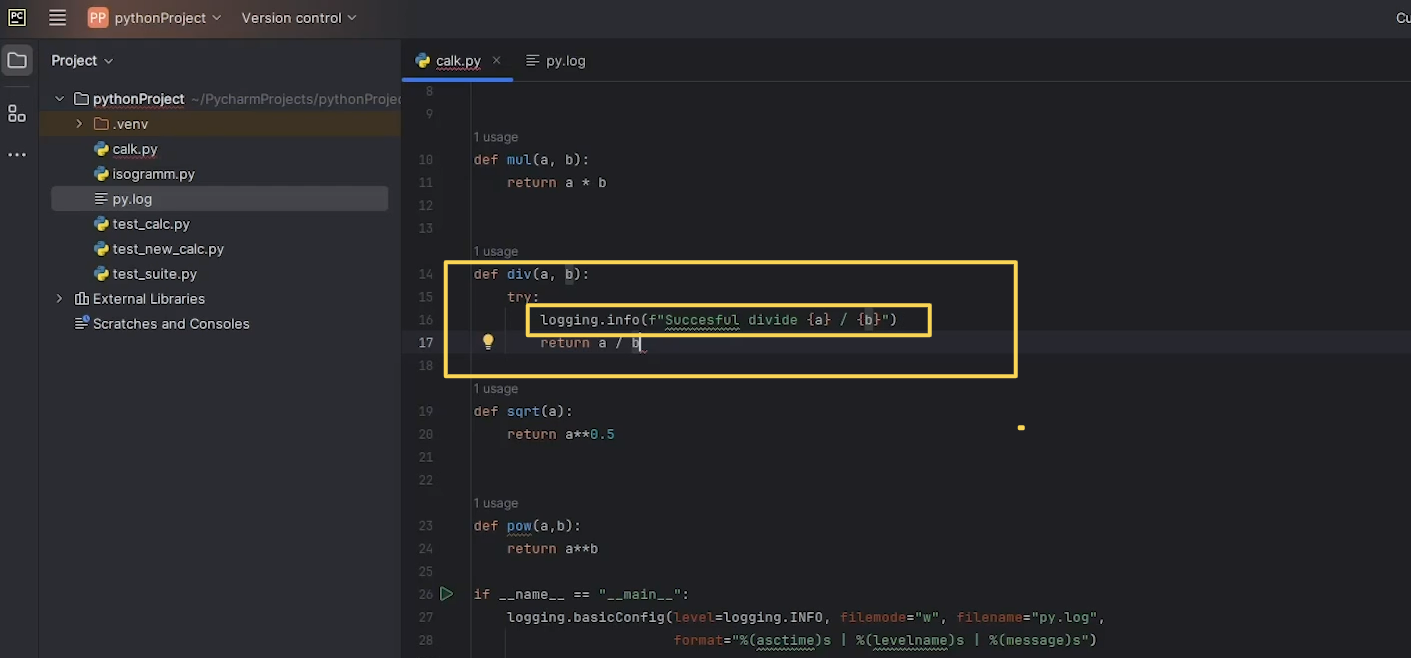


Рис.7

Но может быть случай, когда что-то не сработало. Напомним вам блок «try/except». «Try» попробовать выполнить некий код, «except» в случае получения ошибки мы выполняем другую ветку.

Какая может быть ошибка в делении? Самое очевидное деление на 0 «except ZeroDivisionError». Давайте мы его сохраним как «error» и произведём запись об этом в наш лог «logging.error(“Na nol ne deli!!!”)». Обычно пишется просто код ошибки, но можно написать любое сообщение, которое произошло в случае ошибки. И в этой записи также пишем «exp\_info=True», чтобы передать в логи также сообщение об ошибке, что именно у нас сломалось (Рис.8).

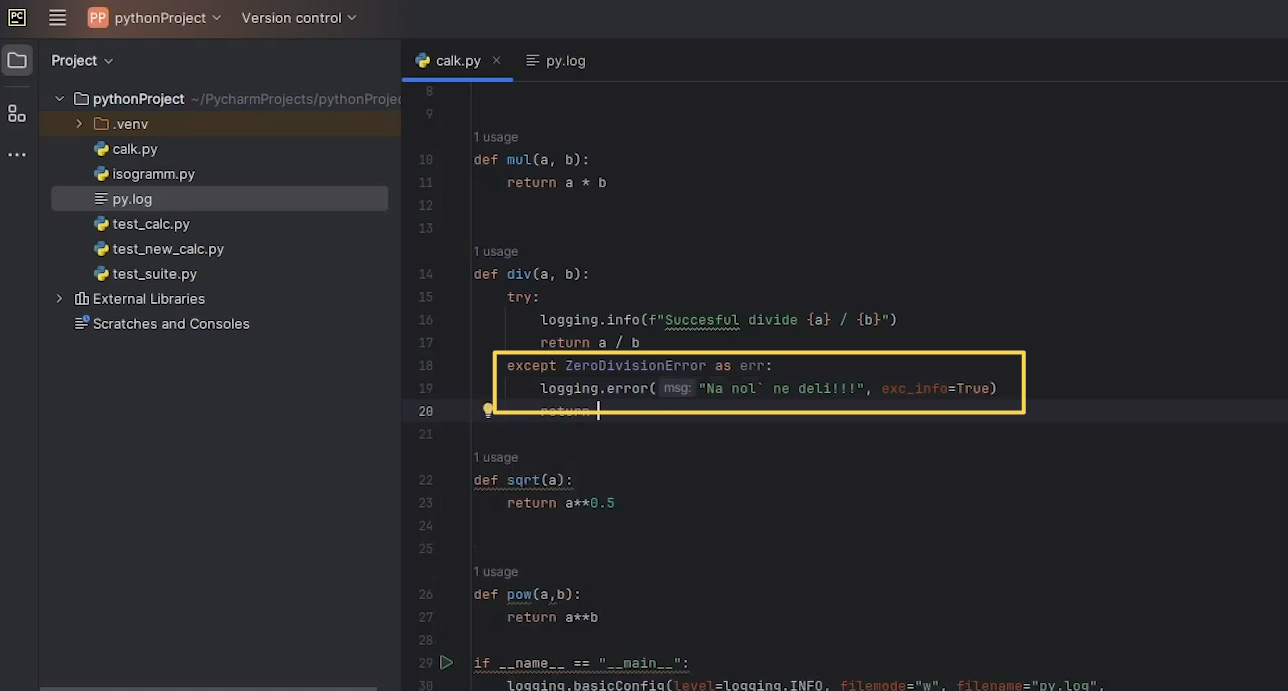


Рис.8

И возвращаем в случае, если делим на 0 какое-нибудь значение по умолчанию, к примеру, 0 или - 1, если вы хотите (Рис.9).

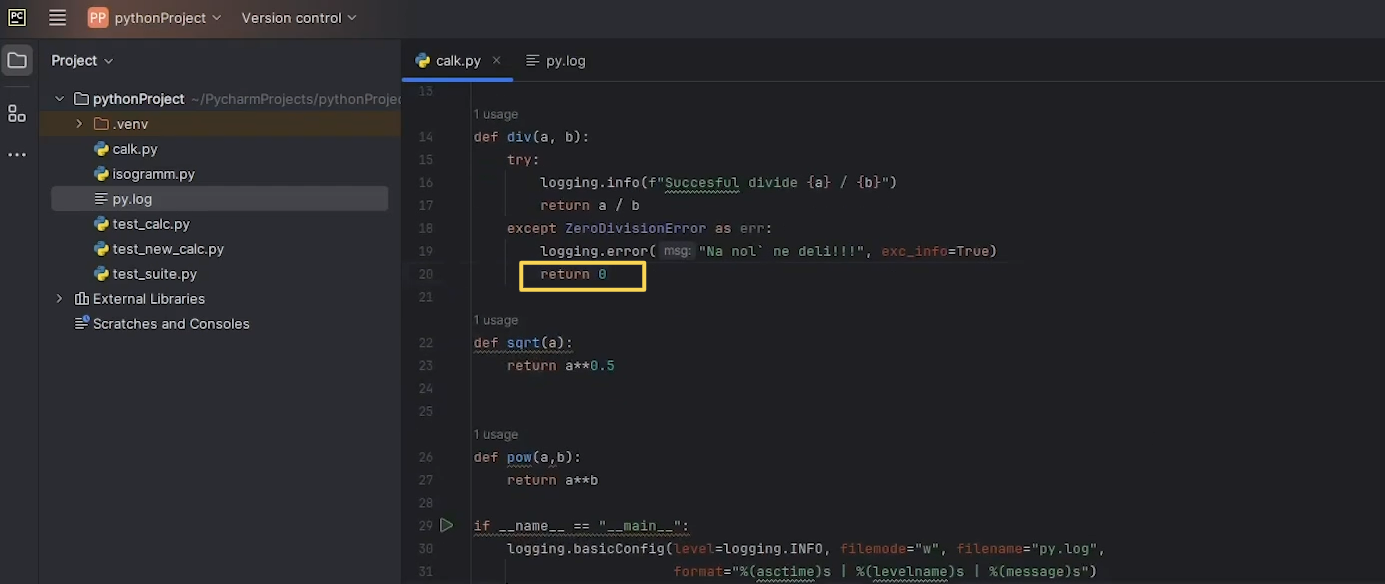


Рис.9

И теперь мы можем запустить нашу функцию: «print(div(3, 4))» и «print(div(3, 0))» (Рис.10).

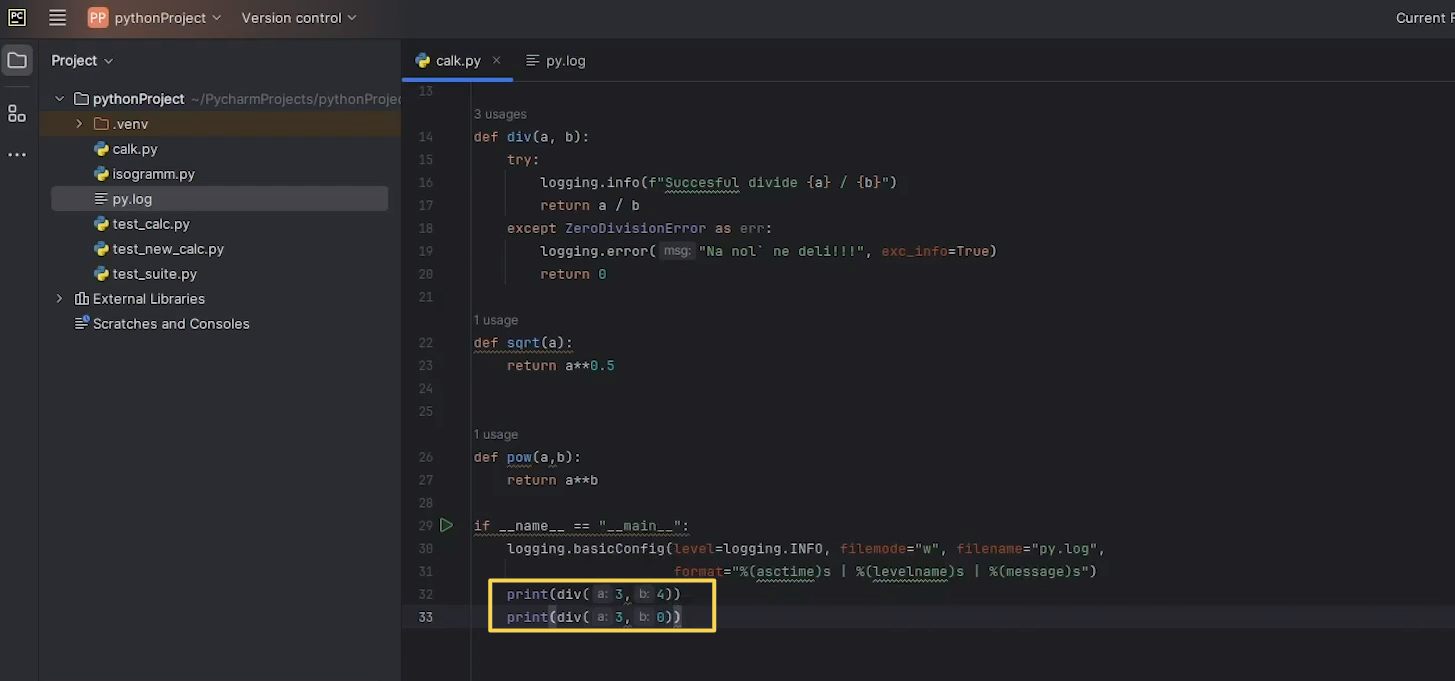


Рис.10

Попробуем запустить (Рис.11). Как видите, 3 на 4 прекрасно разделилось, а 3 на 0 не разделилось, и результат у нас вернулся как 0.

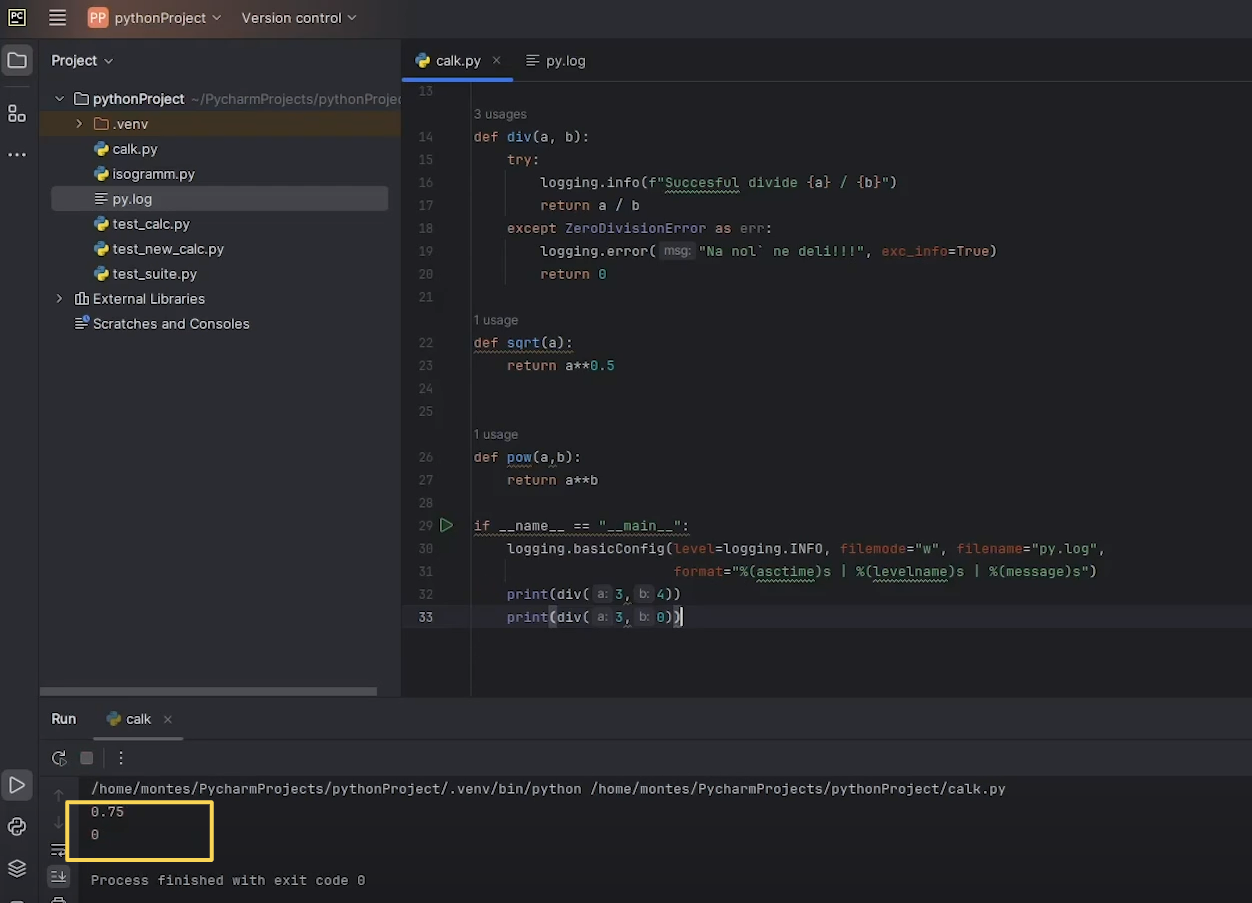


Рис.11

А что произошло в логах? Мы видим следующую ситуацию (Рис.12).

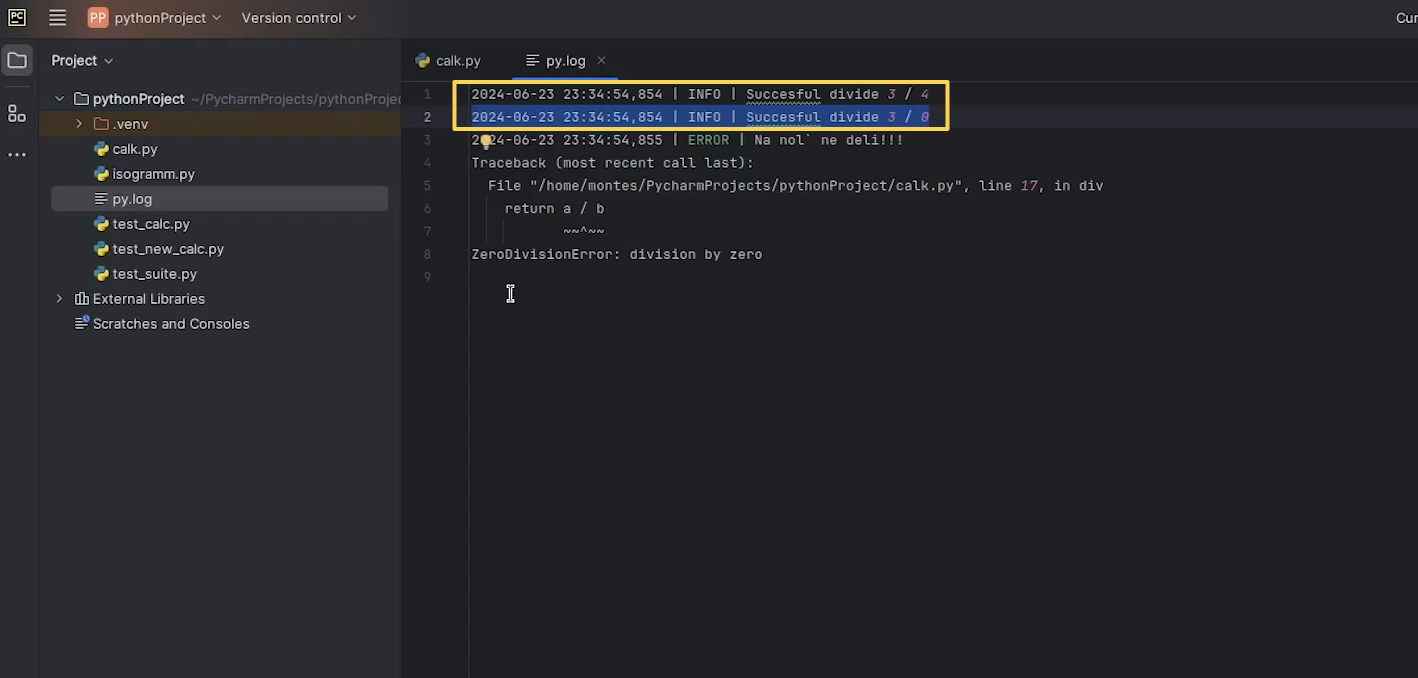


Рис.12

На самом деле это ненормально. Почему это вообще произошло? Давайте смотреть. Мы с вами пишем блок «try», а ошибка у нас происходит только при возвращении значения (Рис.13).

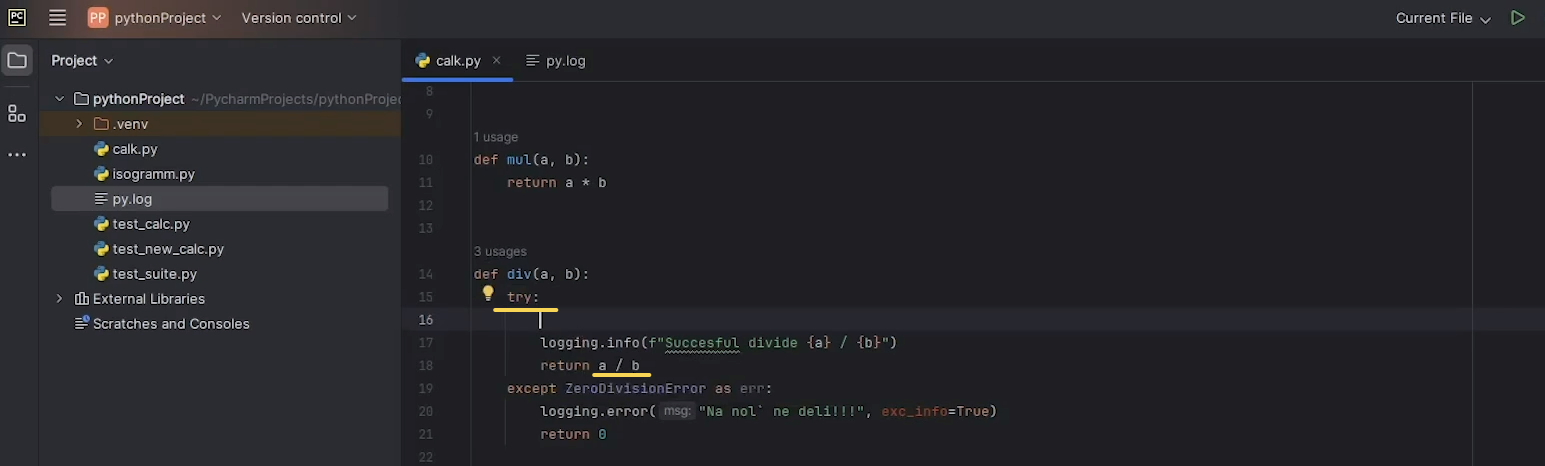


Рис.13

В случае 3 делим на 0 у нас никакой ошибки не было, и блок «try» действительно успевает выполниться. Поэтому мы, для того чтобы проверить, случается ли ошибка, просто попробуем превентивно выполнить действие деления, никуда его не записывая, никуда не сохраняя (Рис.14).

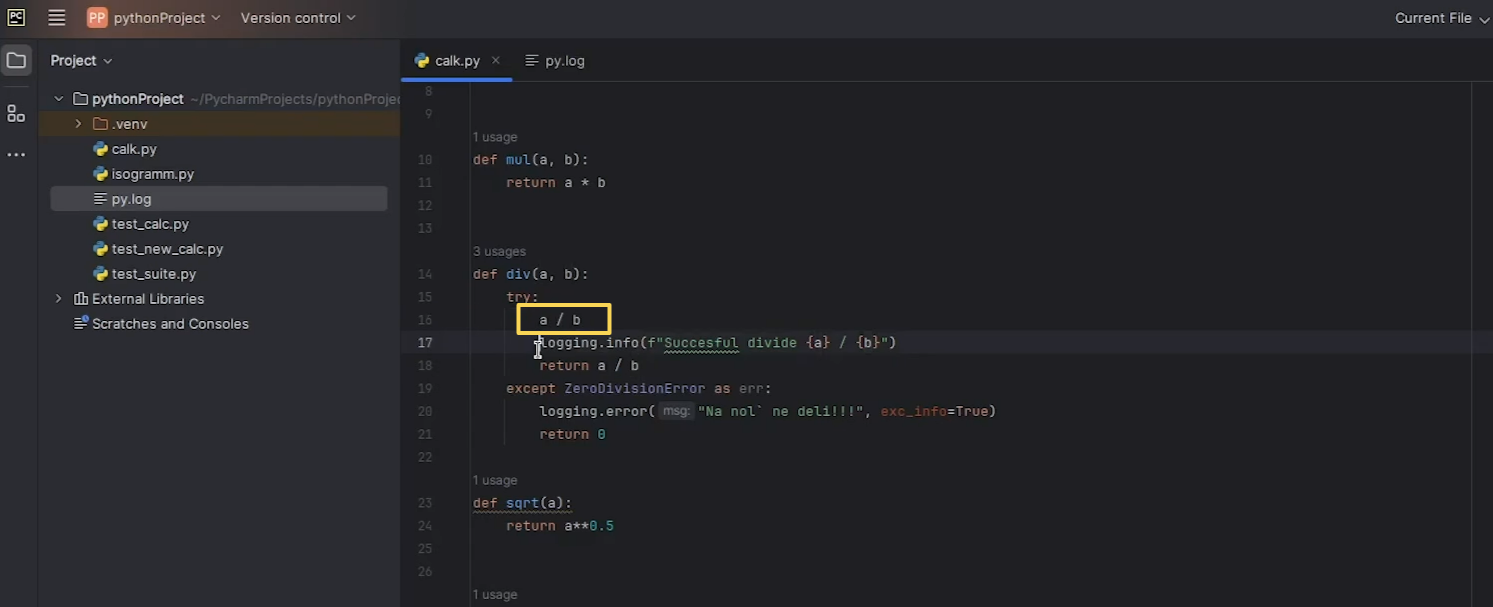


Рис.14

Тогда мы сможем избавиться от этой строчки (Рис.15).

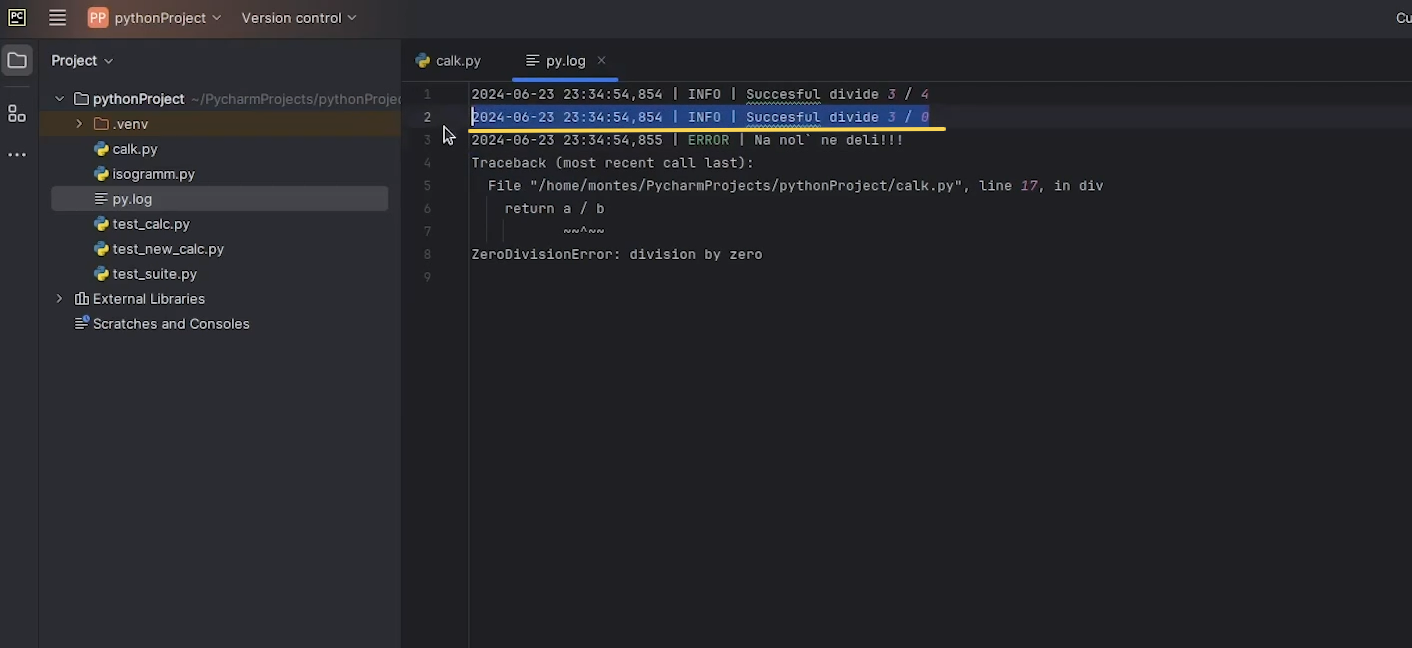


Рис.15

Запустили и видим, что эта строчка пропала (Рис.16).

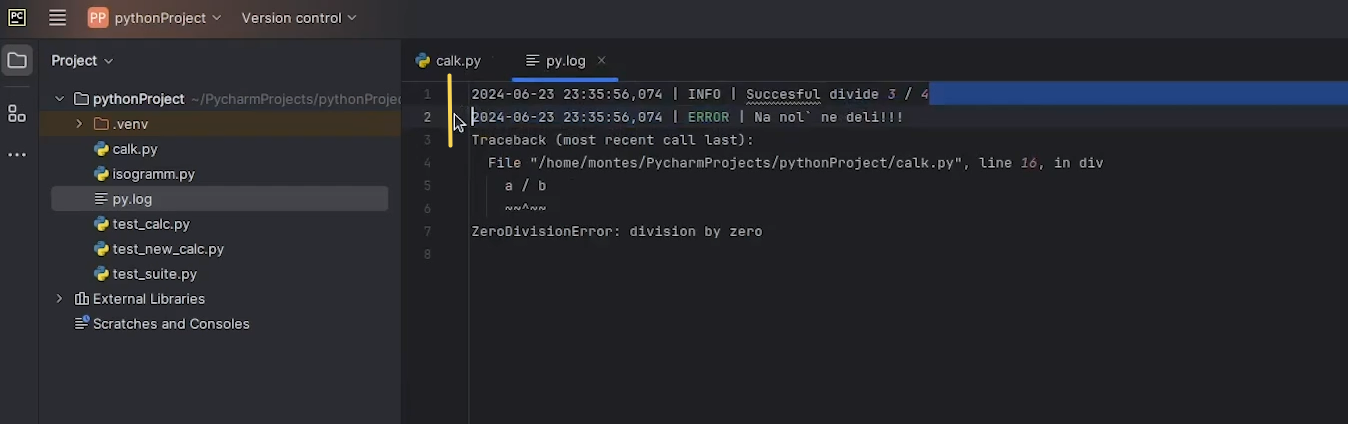


Рис.16

Почему? Потому что у нас вот здесь вот произошла ошибка (Рис.17). Это система, мы можем так писать математические действия, ничего страшного от этого не случится.

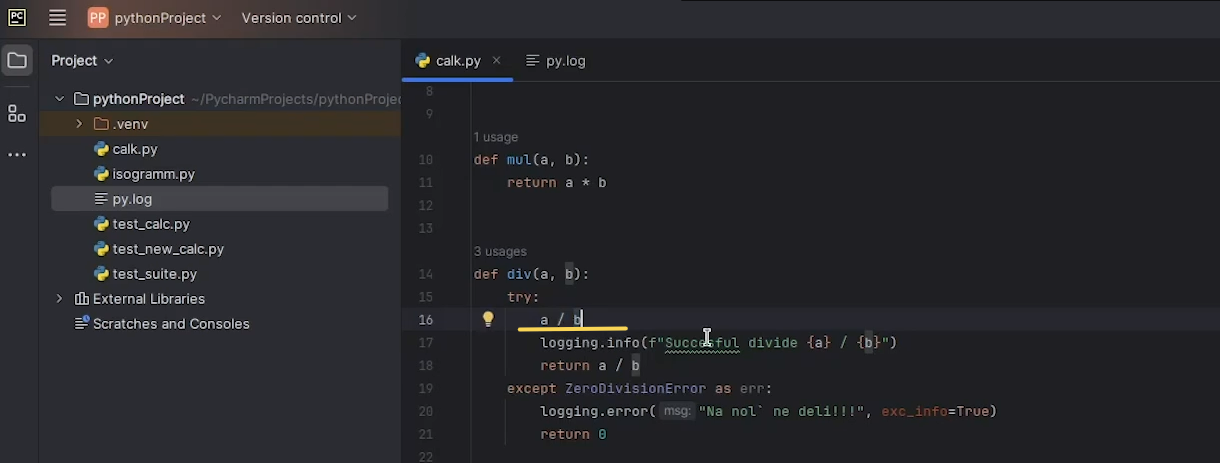


Рис.17

В логах видим, что сначала у нас 3 на 4 прекрасно поделились. Потом мы увидели ошибку «Na nol ne deli!!!». И увидели, что ошибка произошла в делении «a» на «b» по проблеме «ZeroDivisionError: division by zero» (Рис.18).

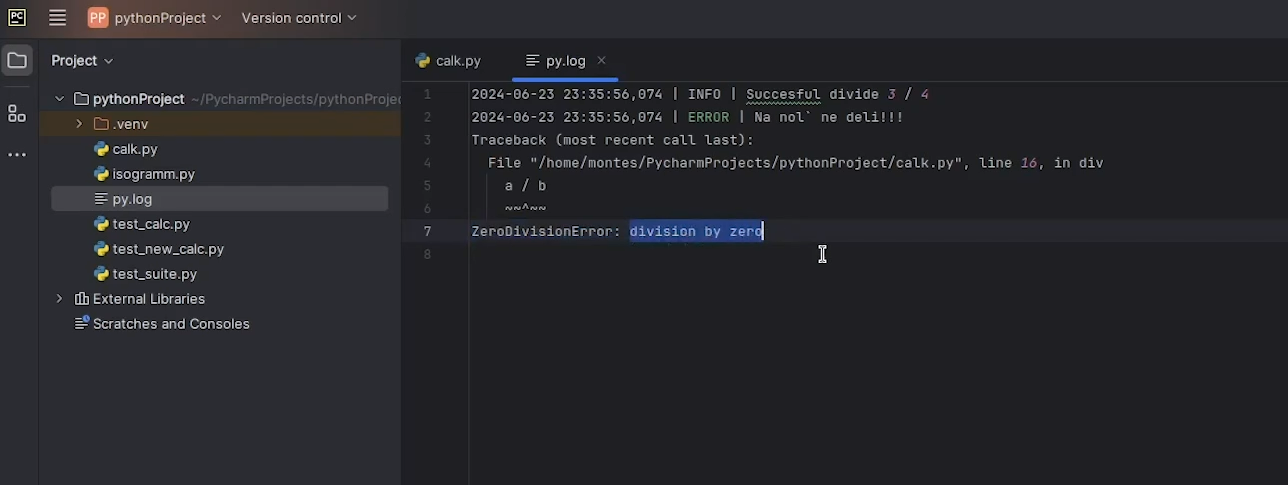


Рис.18

Мы с вами получили в результате систему, которая проверяет работу нашей функции и выводит её состояние.

**Небольшое вам упражнение:**проделайте тоже самое с функцией «sqrt». У неё тоже есть два варианта поведения: классическое и в случае отрицательных чисел. Подумайте о том, как лучше это расписать. Как раз вам в этом плане поможет эта система (Рис.19).

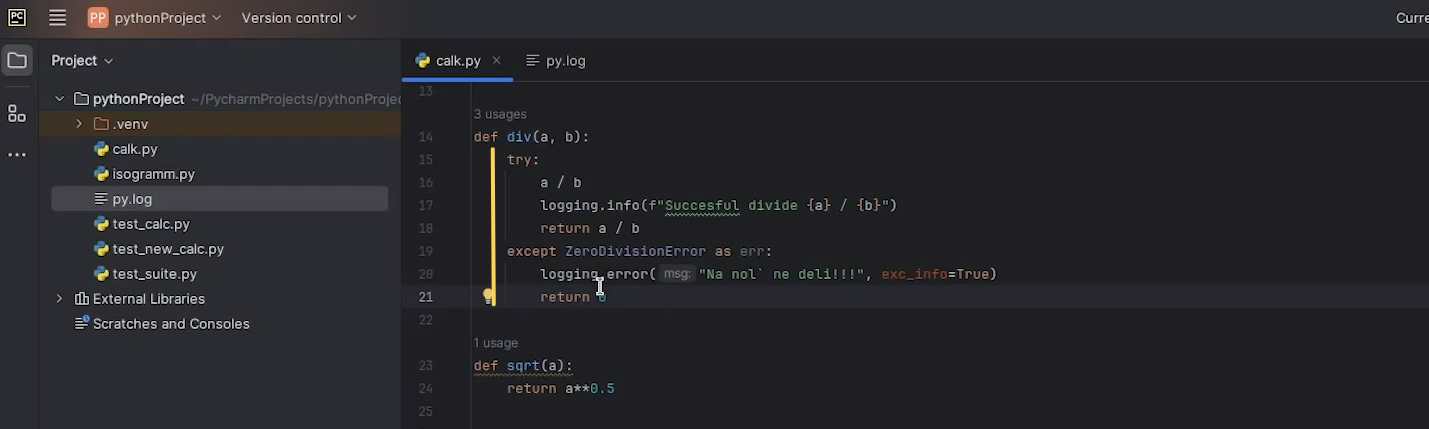


Рис.19

Писать код ошибки необязательно. Можно написать просто «try:», «except:», и название ошибки само передастся в лог. Не беспокойтесь, с ним все будет замечательно.

Если вы знаете, какую конкретно ошибку вы ожидаете, можете написать. В случае, если ошибки не знаете или может прийти любая ошибка, просто «except:».

Все это то и есть наше погружение в логи. Возможно вам, оно покажется достаточно коротким. Оно действительно короткое. Но мы разобрали 80% функционала логирования здесь, и чтобы его освоить, нужно провести много практической работы над своими проектами.

Остались ещё интересные функции, но их использование связано уже с какими-то специфическими ситуациями. И мы вам советуем сначала освоиться с базовыми ситуациями, а уж до сложных ситуаций вы успеете добраться и посмотреть.

Единственное, в чем мы посоветовали бы вам разобраться более детально: в случае, если захотите сделать свой какой-то уникальный фильтр, посмотрите в документации, как устраивается формат. Там есть много различных тегов, которые можно использовать, потому что мы использовали всего три. Но мы постарались сделать такое ёмкое короткое сообщение. Имейте в виду, что логи не должны быть очень длинными. Они должны хорошо читаться.