**2023/11/21 00:00|Лекция.Ошибки**

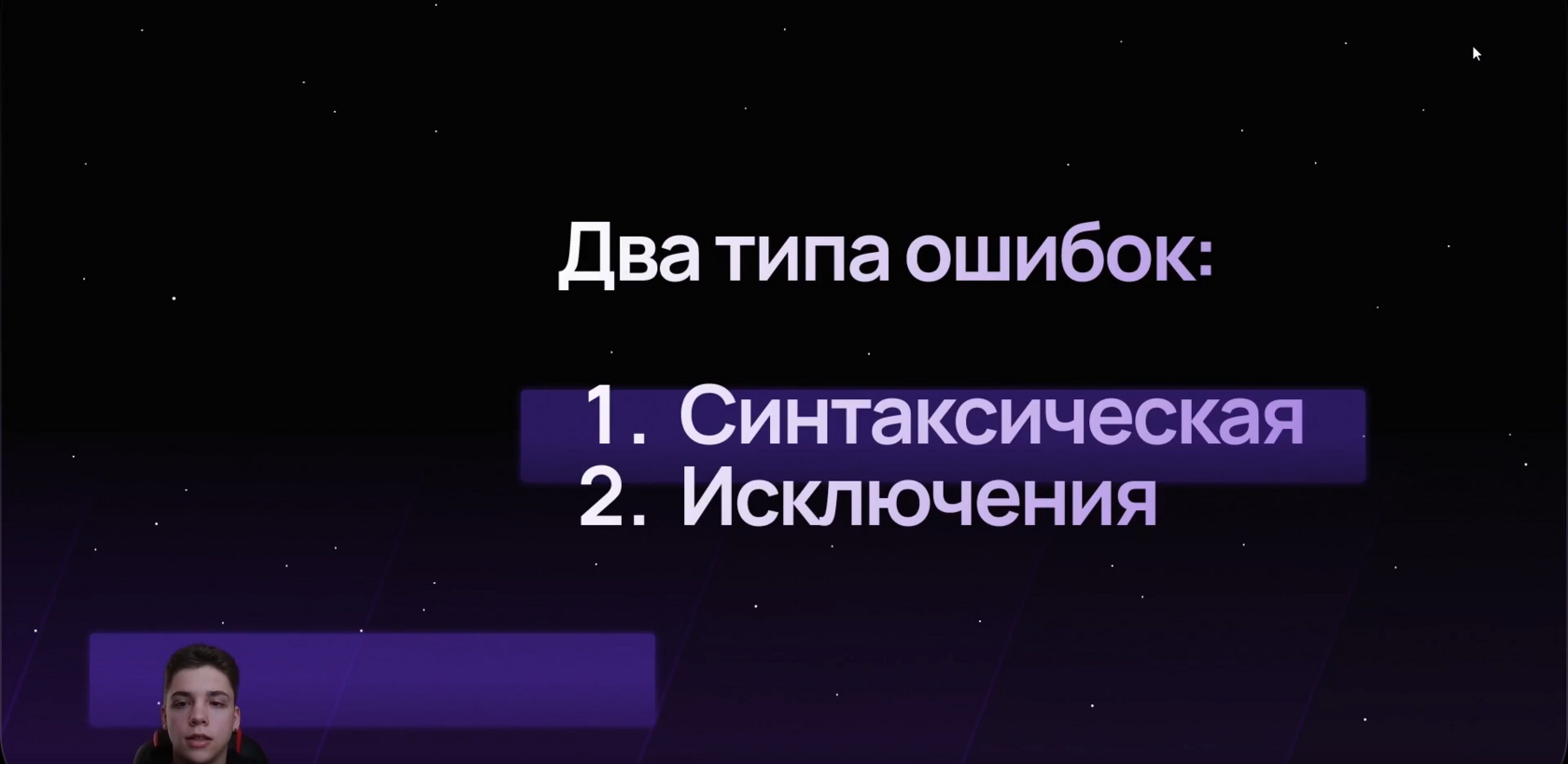
## Ошибки

Всем привет, наши дорогие начинающие программисты с вами на связи, Даниил, и, во-первых, я хотел бы вас сегодня поздравить с тем, что вы пришли на наш новый модуль обучения под названием «исключения», а во-вторых, сегодня хотелось бы с вами поговорить об ошибках, потому что исключения сами по себе — это и есть ошибки. Просто с помощью мощного инструмента мы сможем убирать эти ошибки с кода, а точнее, их предусматривать.

**Ну а что же такое ошибка в целом ошибка?** Это тот случай, когда программа идёт как-то не так, не так, как мы планировали что-то у нас, пойдёт не по плану, какая-то часть кода не выполнится. Или вообще у нас произойдёт то, с чем компьютер, в принципе, не сможет справиться.

Тогда у нас вылезет окошко с красным текстом, что у нас возникла ошибка и программа остановится. Вот именно в этом модуле. Мы и будем учиться отлавливать такие моменты и исправлять их, чтобы программа дальше продолжала работать, потому что, если мы загрузим приложение на сервер и произойдёт какая-то ошибка, приложение не будет останавливать свою работу.

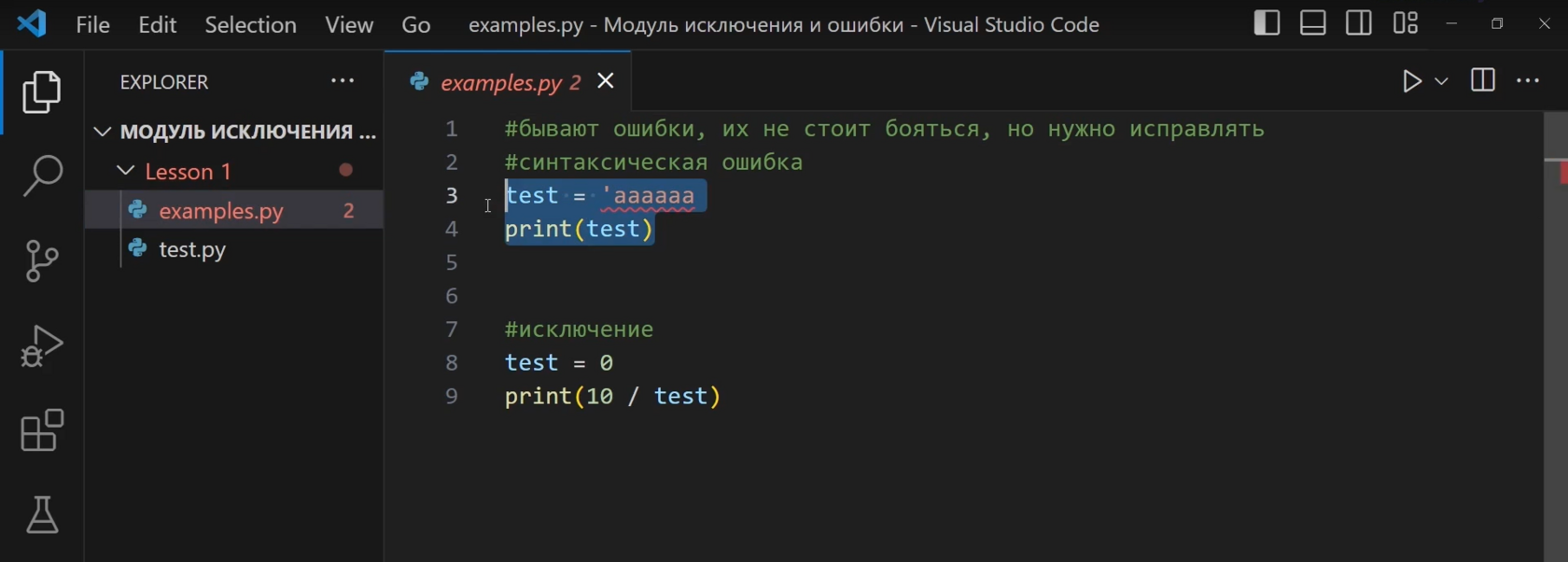
Какие бывают ошибки, сейчас тоже поговорим. Два типа ошибок у нас бывает, а именно синтаксическая, a второй тип — это и есть исключение, которым мы будем заниматься (Рис.1).



(Рис.1)

Синтаксическая ошибка — это довольно просто. Это тогда, когда у нас код, который мы написали, он был написан неправильно, например, строчки не закрыли кавычку или число написали неправильно, хотя это странно. Ну, допустим, написали вместо запятой двоеточие, что бывает, как-то неправильно указали цикл и так далее. То есть это синтаксическая ошибка. Это ошибка прям на уровне кода. То есть она даже ещё не спустилась. Т.е интерпретатор видит, что у нас здесь ошибка в синтаксисе, подсвечивает это и даже не начинает программу.

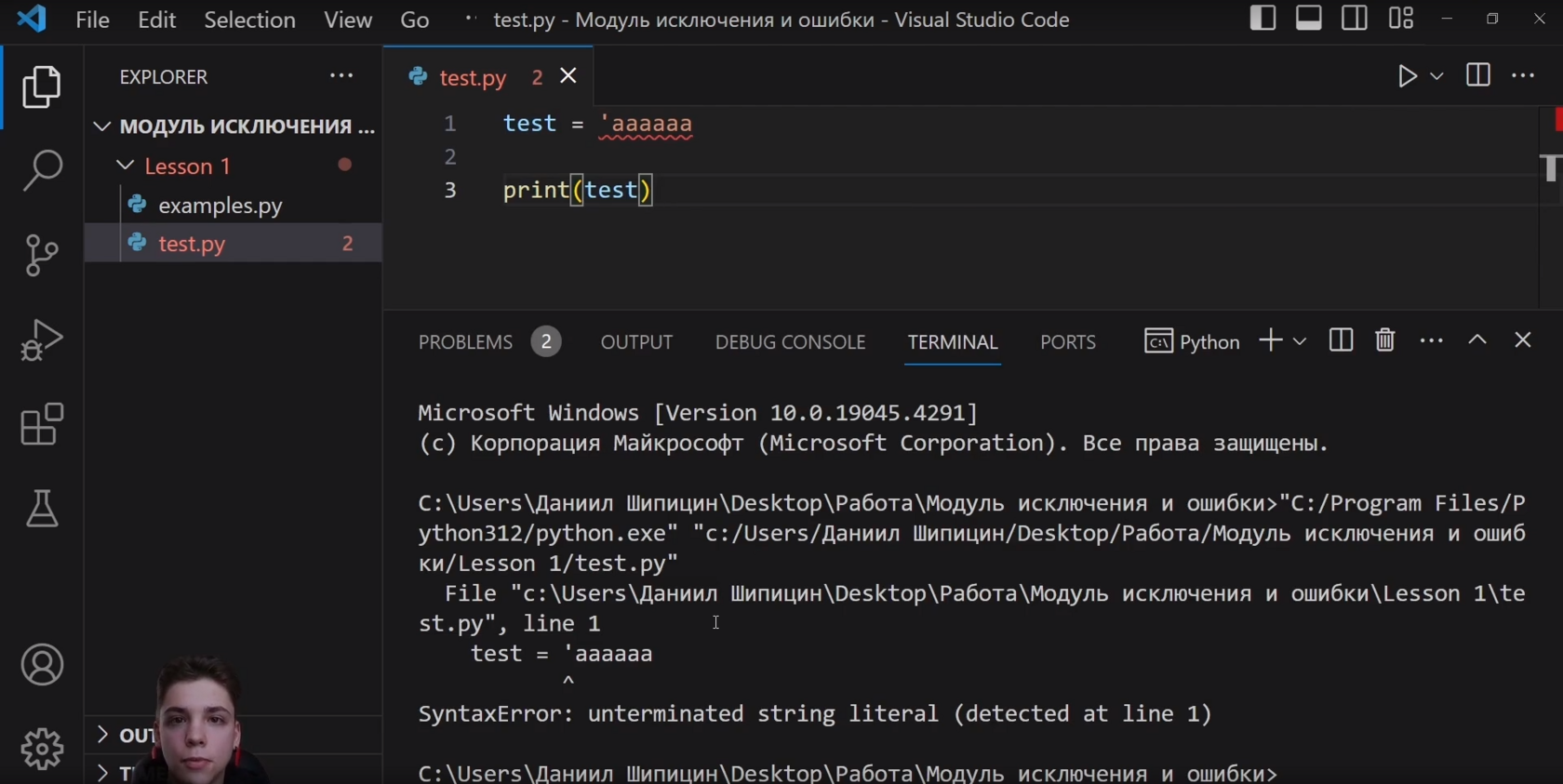
Давайте посмотрим на синтетическую ошибку. Вот в нашем файлике есть небольшой кусочек кода, я его скопирую (Рис.2).



(Рис.2)

И вставим в файл «test.py».

Здесь у нас написано, что переменная «test» сохранена строчка, но кавычку мы специально не закрываем, т.е. как бы сами пытаемся сделать для нас ошибку. В следующей строчке просим программу напечатать переменную тест, как и ожидаемо выводится ошибка (Рис.3).



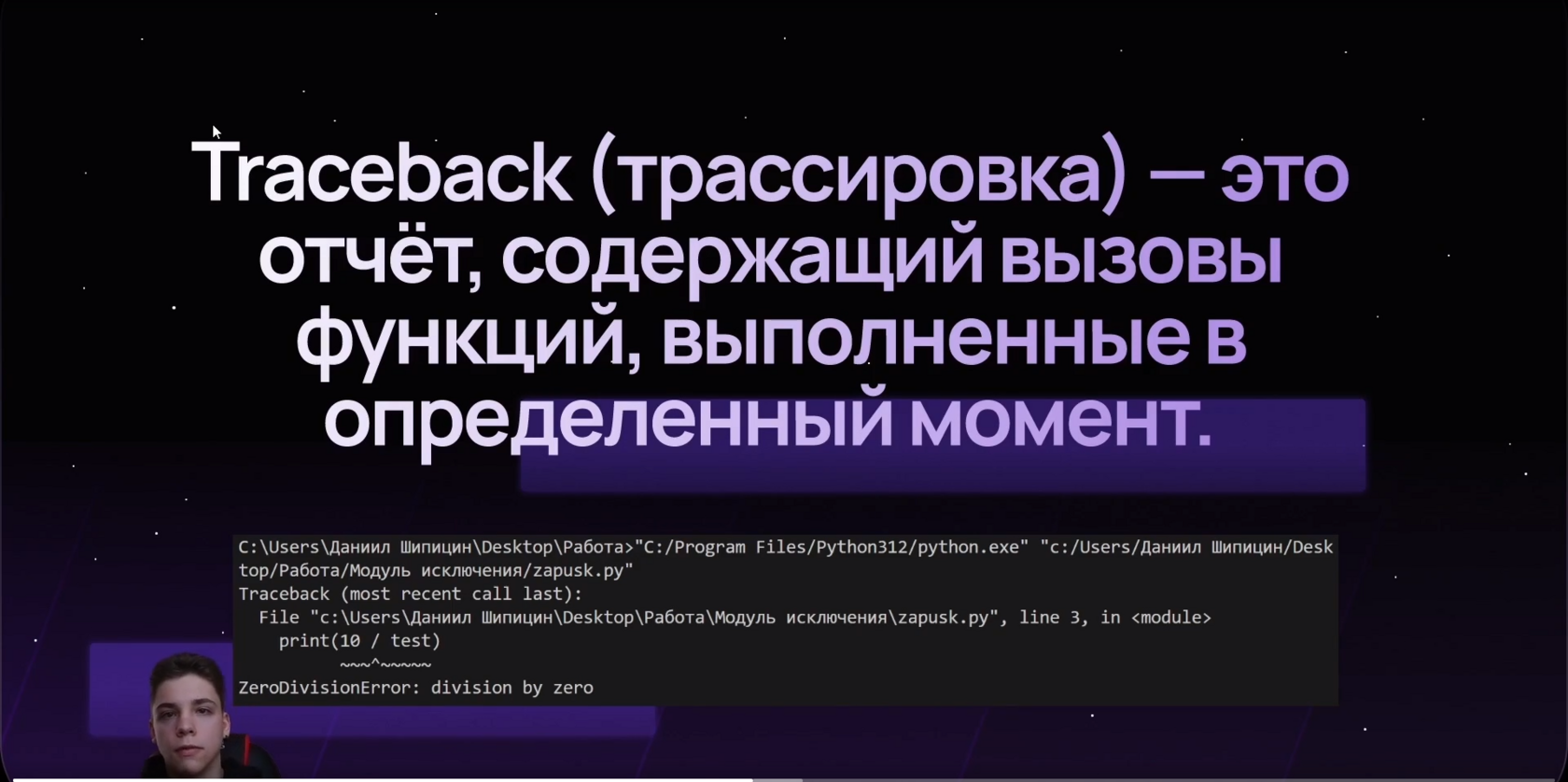
(Рис.3)

Но тут же не написано просто ошибка. У нас высветилось одно большое окно. Называется оно «traceback», что в переводе с английского это трассировка (Рис.4).



(Рис.4)

**«traceback»** — это, отчёт, который содержит информацию о функциях, которые в данный момент выполнялись. Мы можем посмотреть, в каком порядке они выполнялись, в принципе, в каком порядке их программа выполняла. А также там содержатся сведения об ошибке, которая у нас возникла. Вот на презентации показано, как у нас выглядит «traceback» (Рис.5).



(Рис.5)

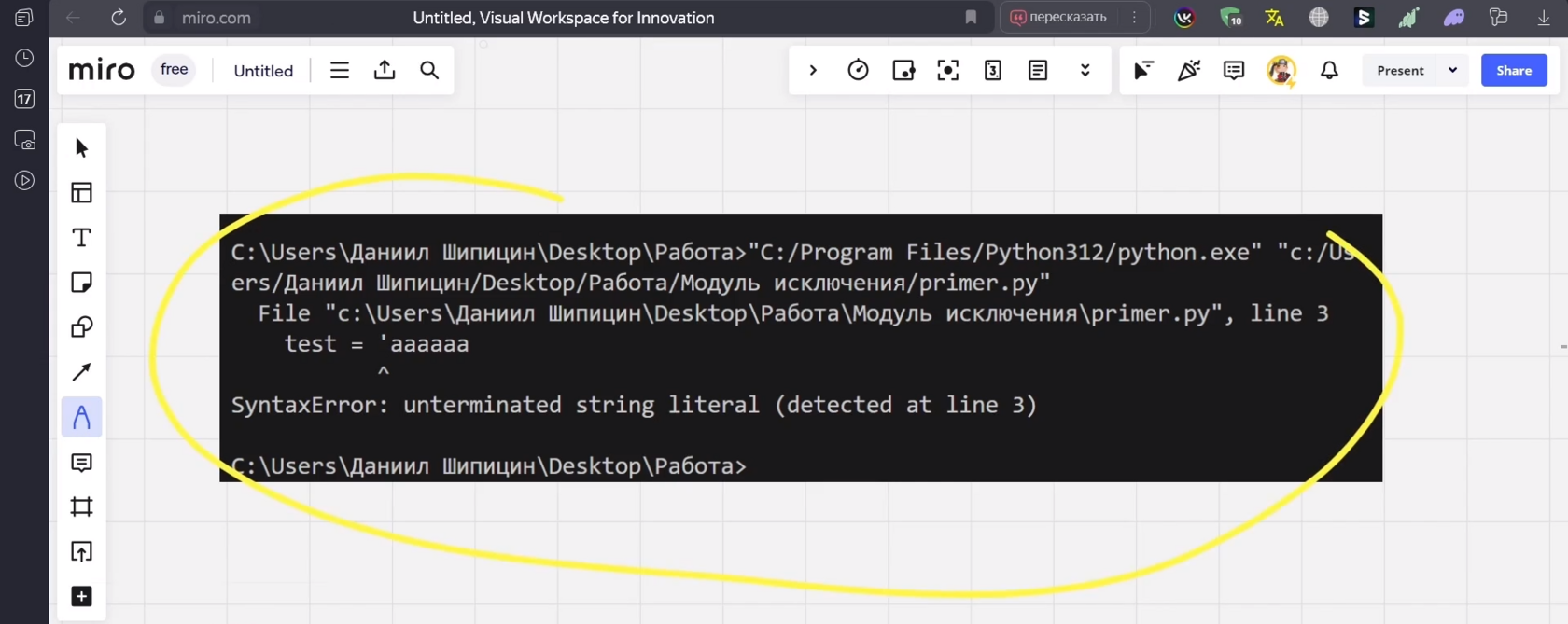
Что самое важное нужно понять в «traceback», во-первых, это то, что его нужно читать снизу вверх. Сейчас это рассмотрим, а во-вторых, то, что в «traceback» сохраняется класс ошибки и описание ошибки, именно той самой, которая у нас в данный момент возникла.

Самое ценное и важное для программиста — это название класса ошибки, а также её подробное описание (Рис.6).



(Рис.6)

Вот посмотрим, сейчас в программе «miro» сохранён скрин нашего «traceback» (Рис.7).



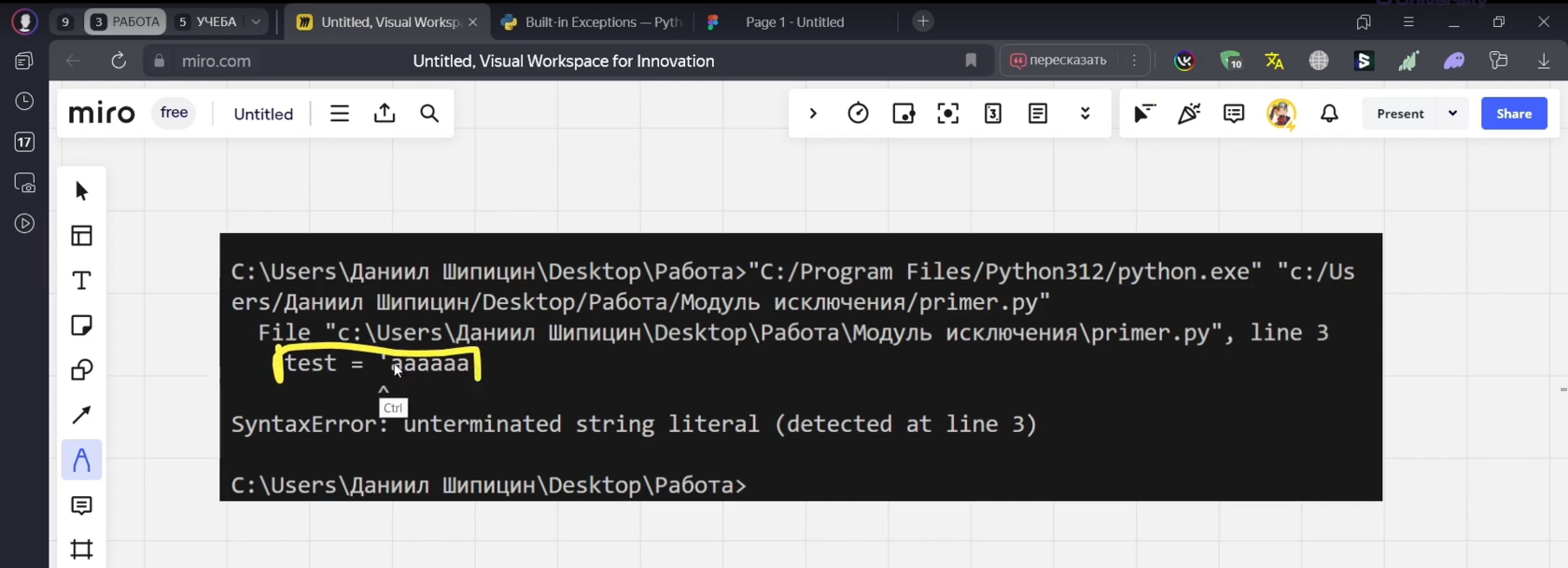
(Рис.7)

Вот указан класс ошибки «syntaxError». То есть это ошибка синтаксиса. В принципе, чего мы и пытались добиться. Через двоеточие, указано более подробное описание ошибки. (Рис.8).



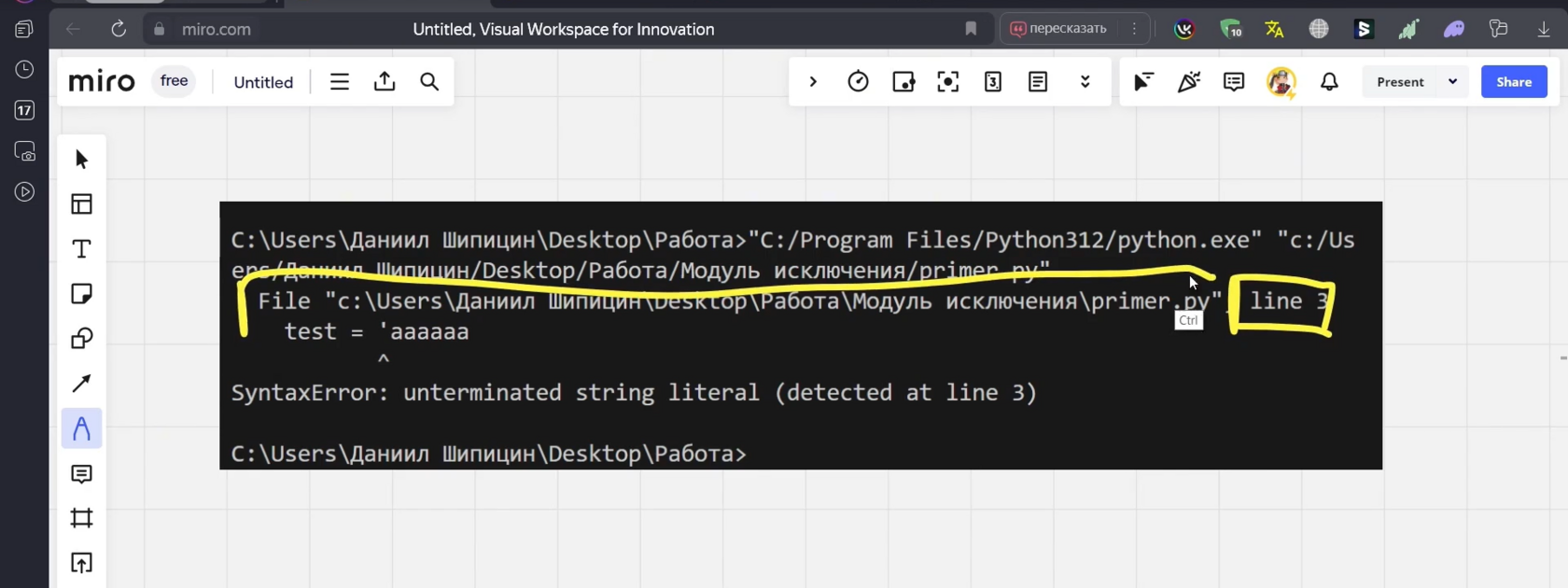
(Рис.8)

Также чуть-чуть выше уже указывается место кода, в котором у нас была допущена ошибка. Ну, у нас было там, в принципе, две строчки, поэтому понять было бы просто, но все равно указывается, где, прям даже стрелочка указывает на кавычки, значит, мы не закрыли, скорее всего, кавычки (Рис.9).



(Рис.9)

Еще выше указана строчка, где у нас возникла ошибка, а также полный путь файла, в том файле, в котором у нас возникла ошибка (Рис.10).

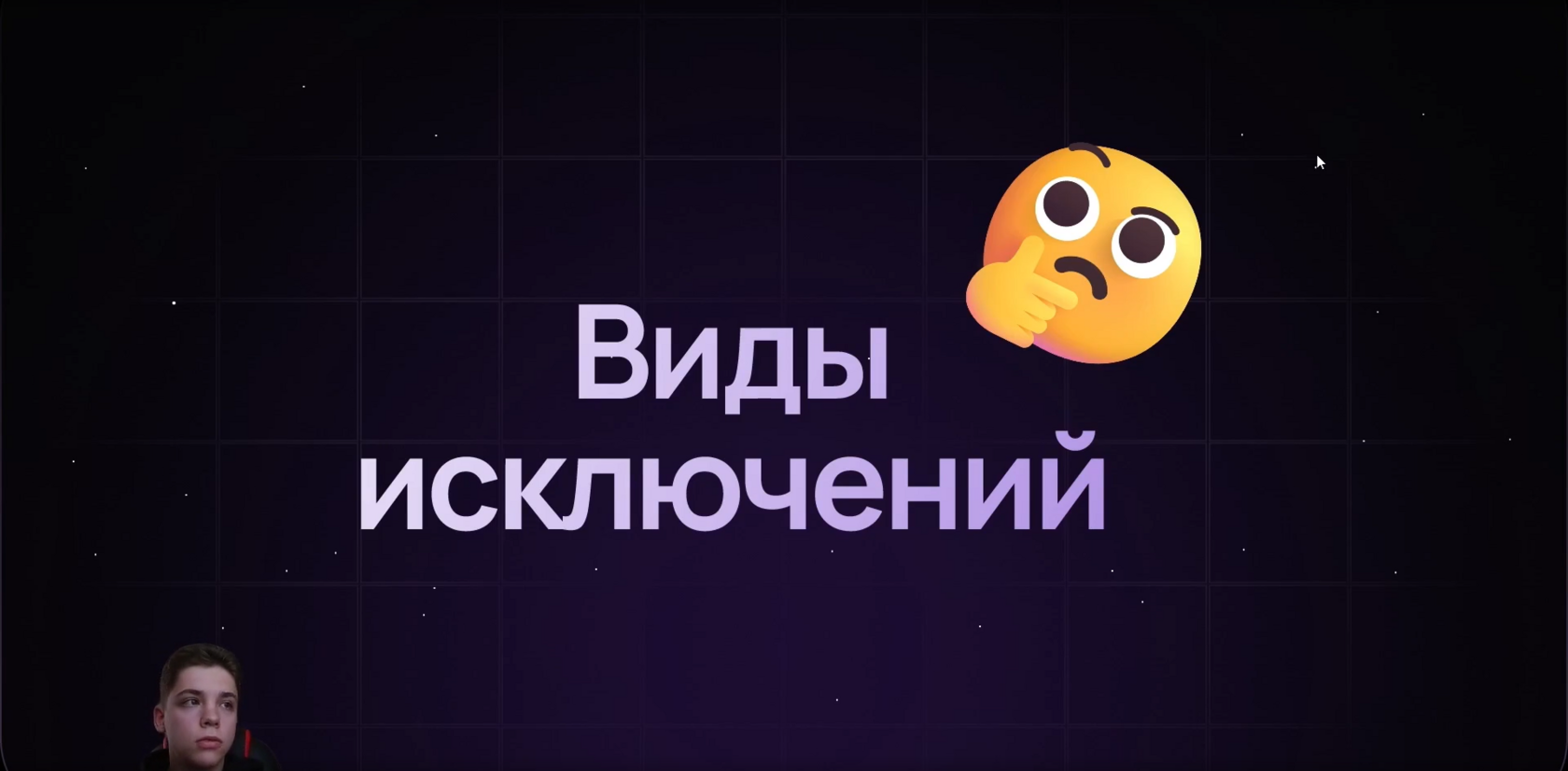


(Рис.10)

Зачем это надо, а затем, потому что приложение в основном состоят из очень больших количества файлов. Запускаются они тоже через один большой файл. И когда у нас возникает ошибка немного непонятно, где, вот поэтому указывается полный путь у файла, где именно возникла ошибка, а также у нас ещё указывается линия.

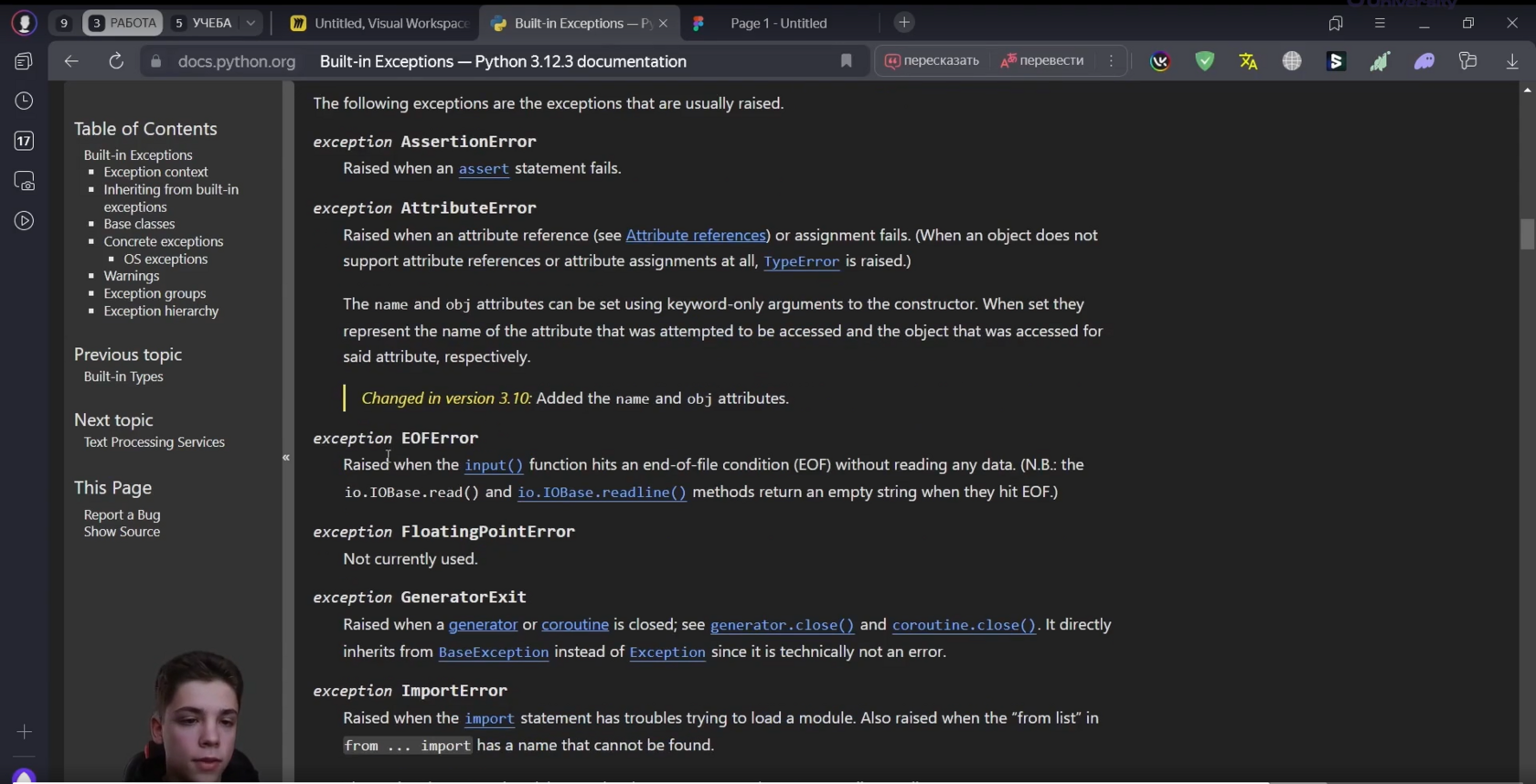
Так что исправить ошибку и найти её очень даже нетрудно, но ещё лучше не исправлять её, естественно, а попытаться перехватить, но об этом мы будем говорить на следующих уроках.

**Какие же виды исключения бывают?** (Рис.11)



(Рис.11)

Их в целом достаточно много, все они наследуются от базового класса «baze exceptions» и намного легче уже просто зайти на официальный сайт официальной документации и немножечко прочитать про них. Смотрите, вот здесь официальный сайт с документацией про «exceptions», то есть про исключения (Рис.12).



(Рис.12)

Например **«AssertionError», «AttributeError», «GeneratorExit», «ImportError»**и так далее. Их на сайте полный список перечислен, это все можно прочитать, можно посмотреть более подробное описание, заучивать это, конечно же, не стоит. Просто, если вам понадобится, вы можете к этому обратиться, но как раз таки в этом блоке мы будем некоторые из них более часто использовать, но это вы уже увидите на следующих занятиях.

**Итог**

Сегодня мы узнали про виды ошибок: что такое ошибка? Что такое исключение? Также научились читать «traceback». Это очень важный и полезный навык для программиста. Поэтому обязательно подключайтесь к следующему видео. Смотрите его, выполняйте домашние задания, а на этом пока что все всем пока и всем удачи (Рис.13).



(Рис.13)