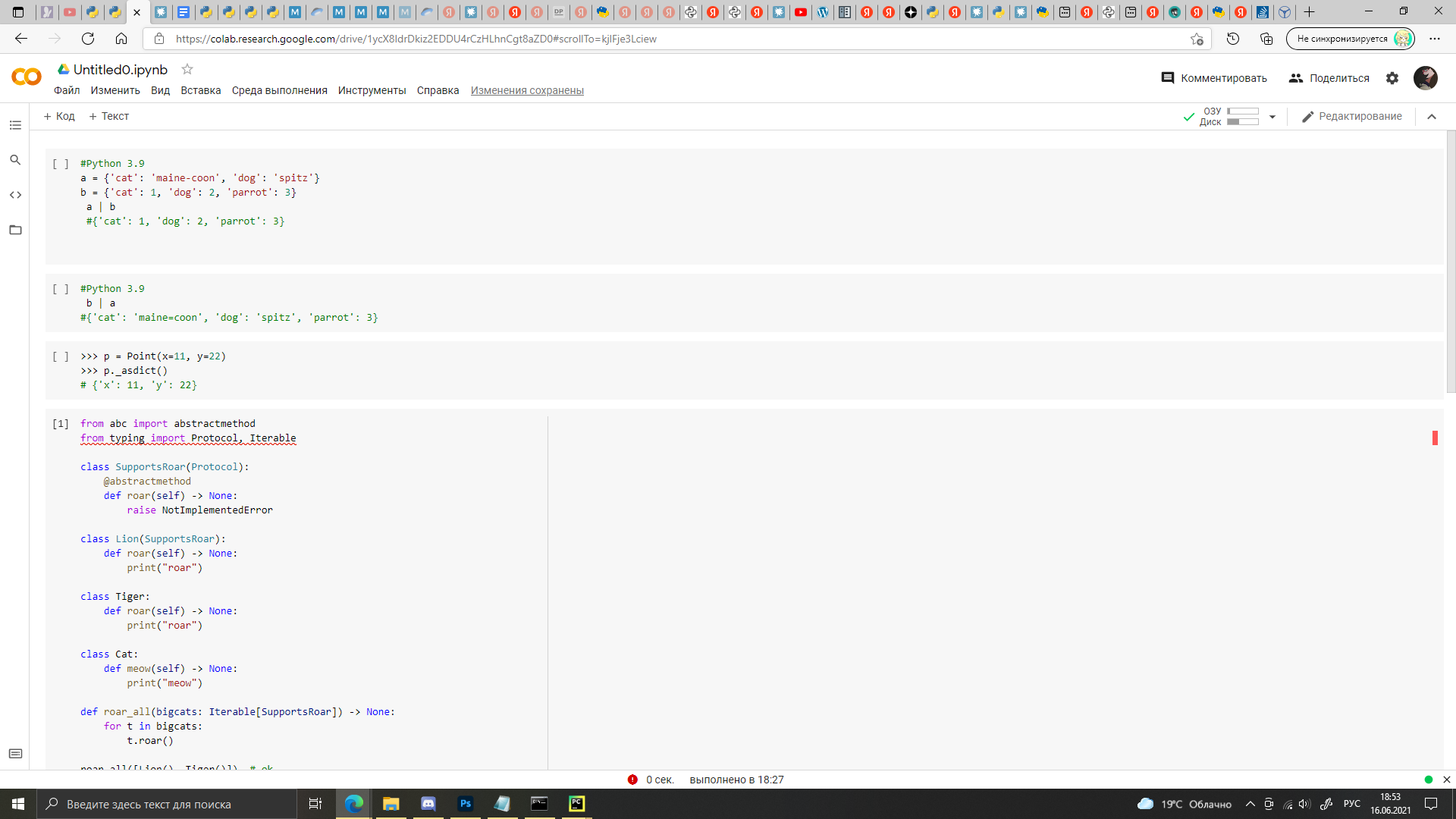
Лабораторная работа 3.

[*Ссылка на Google Colab*](https://colab.research.google.com/drive/1ycX8IdrDkiz2EDDU4rCzHLhnCgt8aZD0?usp=sharing)

Коллекции (*изменения, НЕ новые типы*):

1. **dict (словарь)**
2. подключение не требуется
3. dict используется для создании коллекции произвольных объектов с доступом по ключу.



1. Так как dict имеет много методов, и с ними мы знакомились в прошлом семестре, здесь они перечисляться не будут.

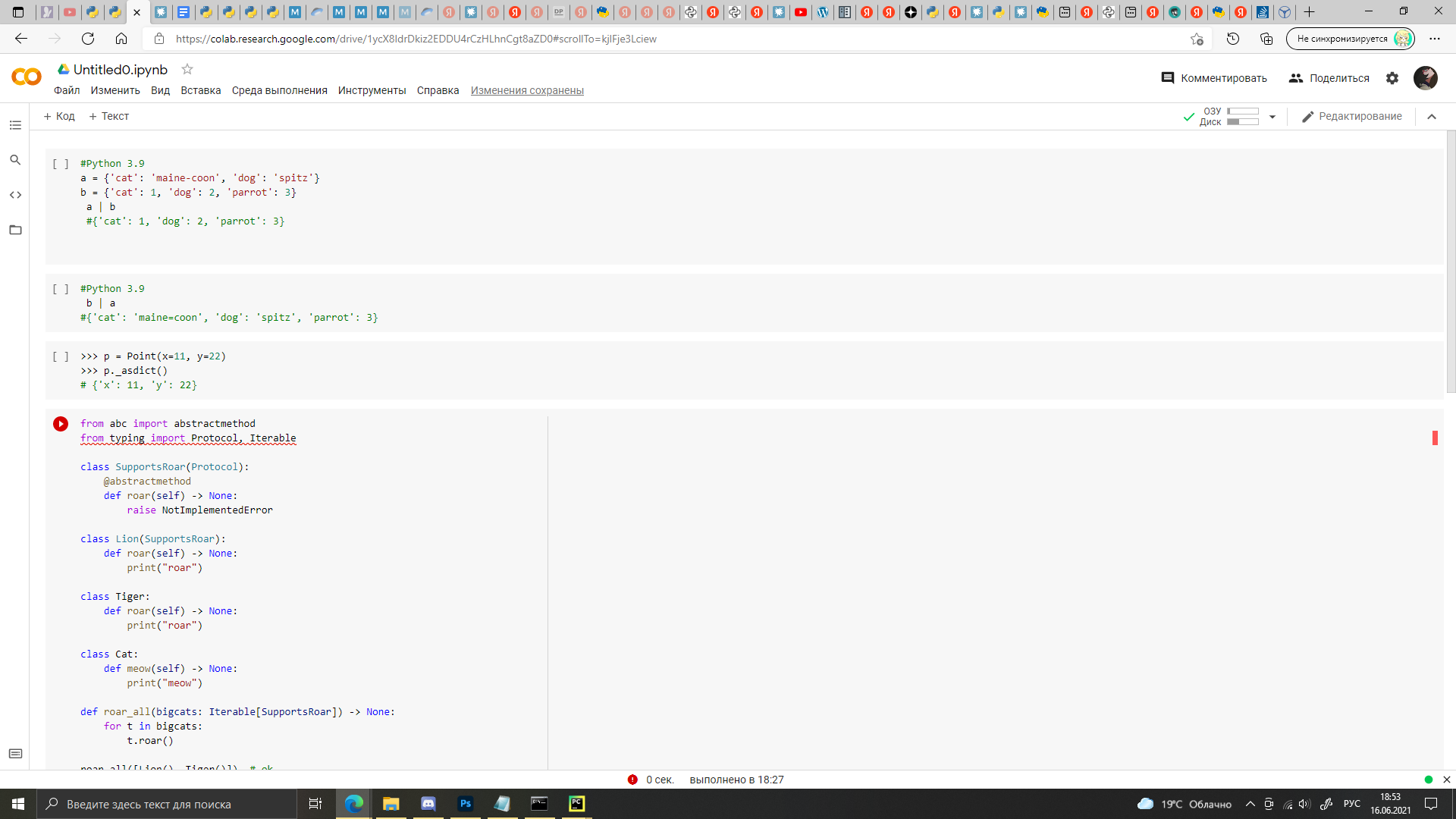
#### Операции со словарями и их временная сложность

* Получение элемента: O(1).
* Установка элемента: O(1).
* Удаление элемента: O(1).
* Проход по словарю: O(n).

Изменения

Python 3.9 - PEP 584

Этот PEP предлагает добавить операторы слияния ( | ) и обновления ( | = ) во встроенный класс dict.



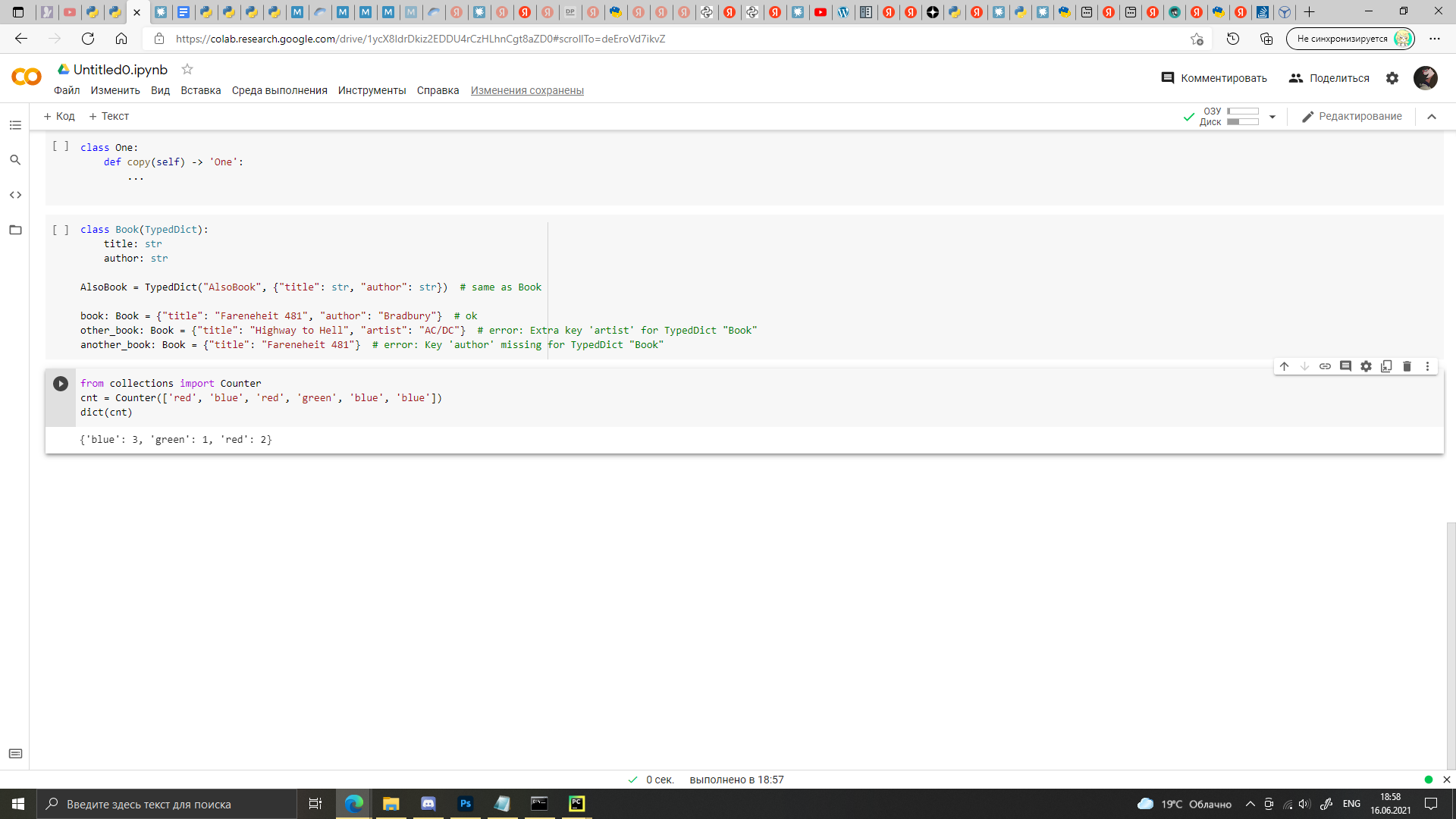
1. **Подкласс dict - Counter()**
2. Подключается отдельно:

from collections import Counter - вызывается как Counter()

или

from collections - вызывается как collections.Counter()

1. Класс [Counter()](https://docs-python.ru/standart-library/modul-collections-python/klass-counter-modulja-collections/) модуля [collections](https://docs-python.ru/standart-library/modul-collections-python/) - это подкласс [словаря dict](https://docs-python.ru/tutorial/osnovnye-vstroennye-tipy-python/tip-dannyh-dict-slovar/) для подсчета хеш-объектов (неизменяемых, каких как [строки](https://docs-python.ru/tutorial/osnovnye-vstroennye-tipy-python/tip-dannyh-str-tekstovye-stroki/), числа, [кортежи](https://docs-python.ru/tutorial/osnovnye-vstroennye-tipy-python/tip-dannyh-tuple-kortezh/) и т.д.). Это коллекция, в которой элементы хранятся в виде словарных ключей, а их счетчики хранятся в виде значений словаря.



#### .elements(), most\_common([n]),subtract([iterable-or-mapping]), fromkeys(iterable), update([iterable-or-mapping])

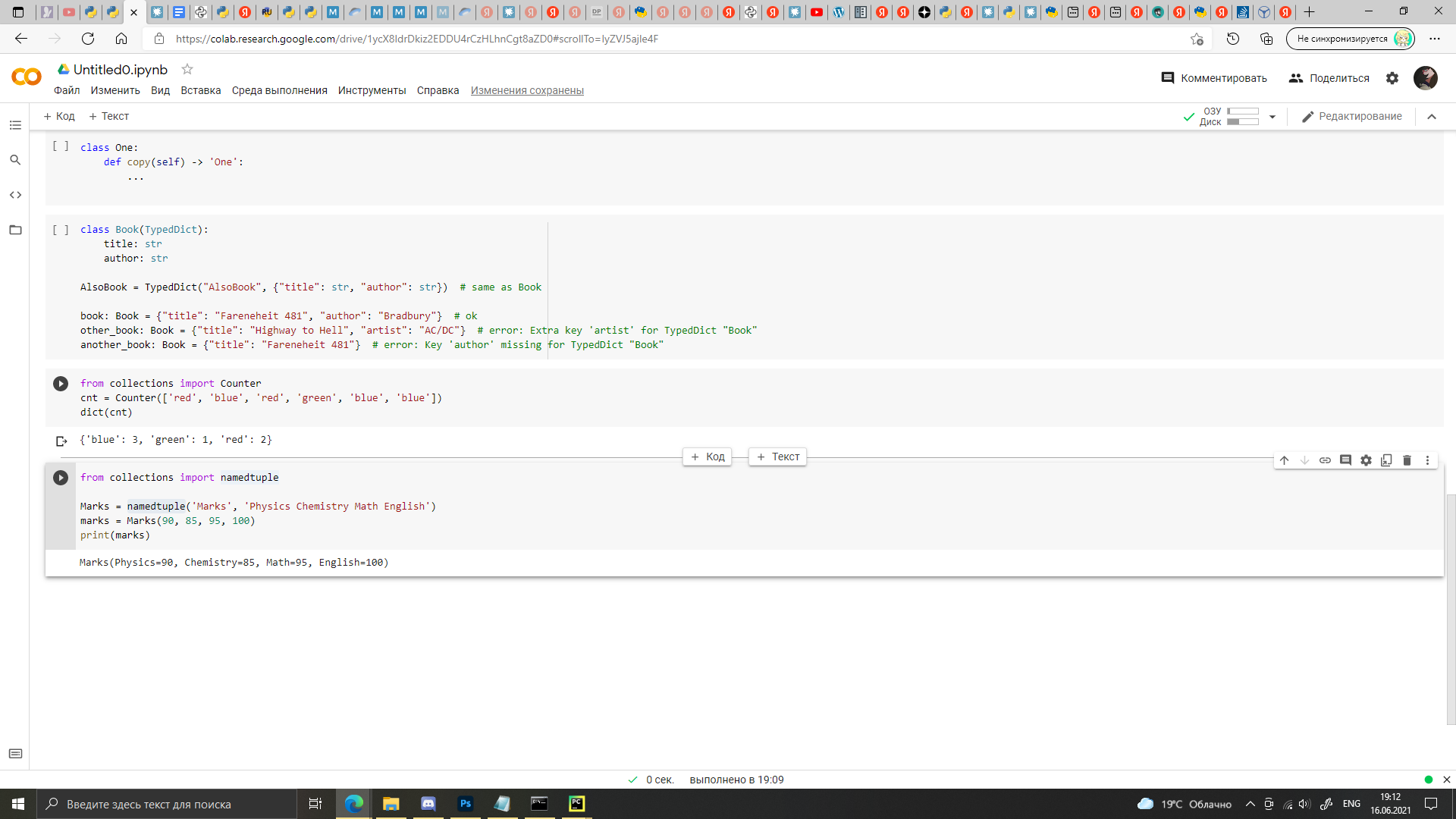
#### Выбрасываемые ошибки в основном касаются синтаксиса

Изменения

Python 3.7.

В качестве подкласса dict(), класс Counter() унаследовал возможность запоминания порядка вставки. Математические операции над объектами Counter() также сохраняют порядок. Результаты упорядочены в соответствии с тем, когда элемент сначала встречается в левом операнде, а затем в порядке, в котором встречается правый операнд.

1. **Именнованный кортеж (**namedtuple, подкласс кортежей)
2. from collections import namedtuple - вызывается как namedtuple()
3. или
4. import collections - вызывается как collections.namedtuple()
5. [Именованные кортежи](https://docs-python.ru/standart-library/modul-collections-python/klass-namedtuple-modulja-collections/) присваивают имя каждому значению элемента в [кортеже](https://docs-python.ru/tutorial/osnovnye-vstroennye-tipy-python/tip-dannyh-tuple-kortezh/) и тем самым создают более читаемый код. Они могут использоваться везде, где используются [обычные кортежи](https://docs-python.ru/tutorial/osnovnye-vstroennye-tipy-python/tip-dannyh-tuple-kortezh/) и добавляют возможность доступа к полям по имени вместо [индекса позиции](https://docs-python.ru/tutorial/obschie-operatsii-posledovatelnostjami-list-tuple-str-python/izvlechenie-elementa-sequence-indeksu/).



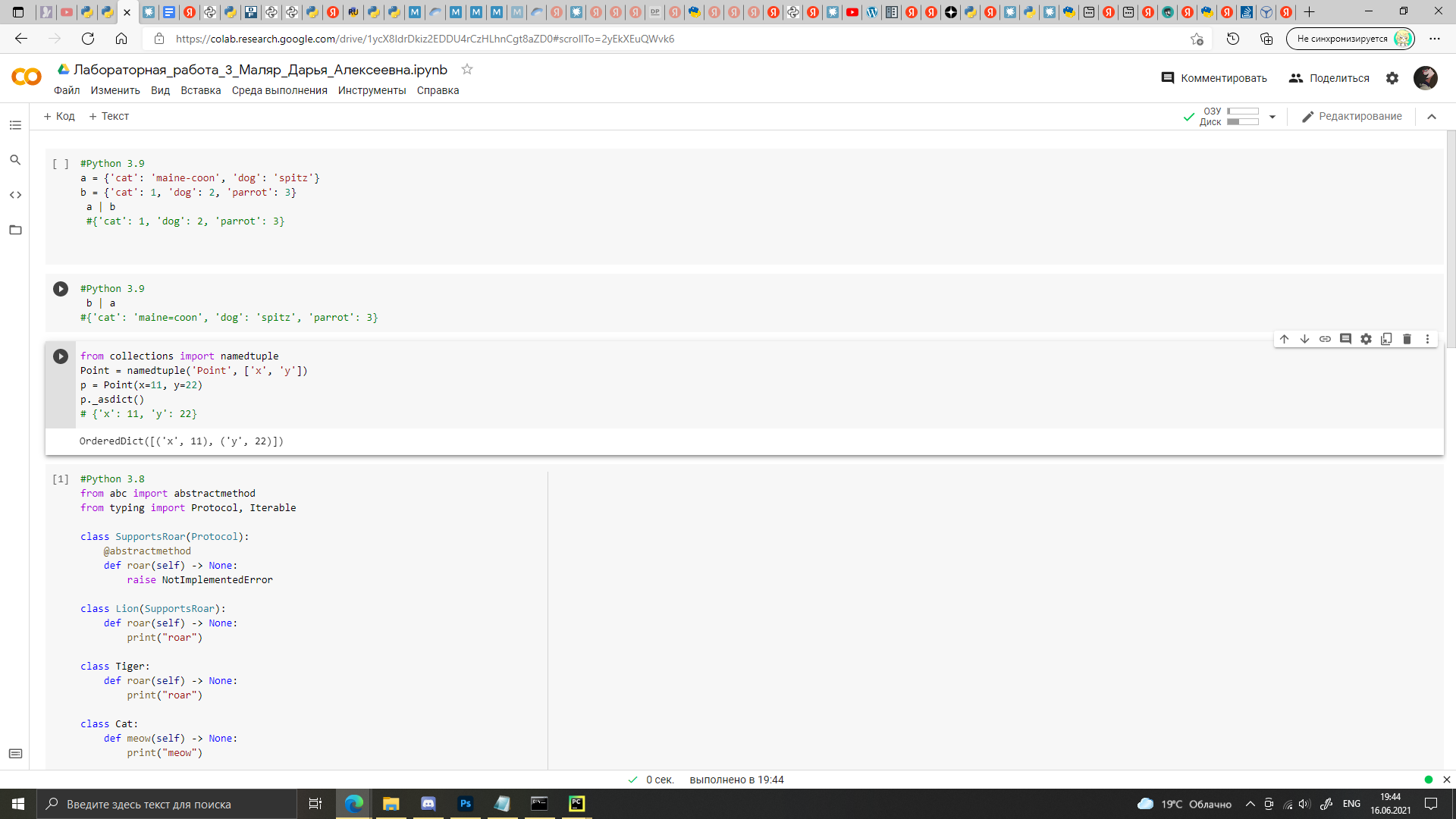
#### .\_make(iterable), \_asdict(), \_replace(\*\*kwargs), \_fields, \_field\_defaults

#### TypeError (если указать на одно значение больше/меньше), Syntax Error

Изменения

Python 3.8

* Удален verbose параметр и свойство .\_source
* Добавлен параметр defaults - [итерируемая последовательность](https://docs-python.ru/tutorial/osnovnye-vstroennye-tipy-python/tip-dannyh-iterator-iterator/), значения по умолчанию имен кортежа.
* Добавлено свойство \_field\_defaults - оно вернет [словарь](https://docs-python.ru/tutorial/osnovnye-vstroennye-tipy-python/tip-dannyh-dict-slovar/), который сопоставляет имена полей со значениями по умолчанию.
* Изменения метода [NamedTuple.\_asdict()](https://docs-python.ru/standart-library/modul-collections-python/klass-namedtuple-modulja-collections/#ntuple._asdict) - Метод вернет новый [словарь dict](https://docs-python.ru/tutorial/osnovnye-vstroennye-tipy-python/tip-dannyh-dict-slovar/), который отображает имена полей в соответствии с их значениями (вместо [OrderedDict](https://docs.python.org/3/library/collections.html#collections.OrderedDict), как было в предыдущих версиях) (в Google Colab версия 3.7, поэтому выдается соответствующий результат)



Изменения, связанные с аннотацией типов (3.8):

как в некоторых других языках.

К сожалению, до версии 3.8 мы не могли выразить необходимые требования к объекту с помощью аннотаций типов.

[PEP 544](https://www.python.org/dev/peps/pep-0544/) призван решить эту проблему.

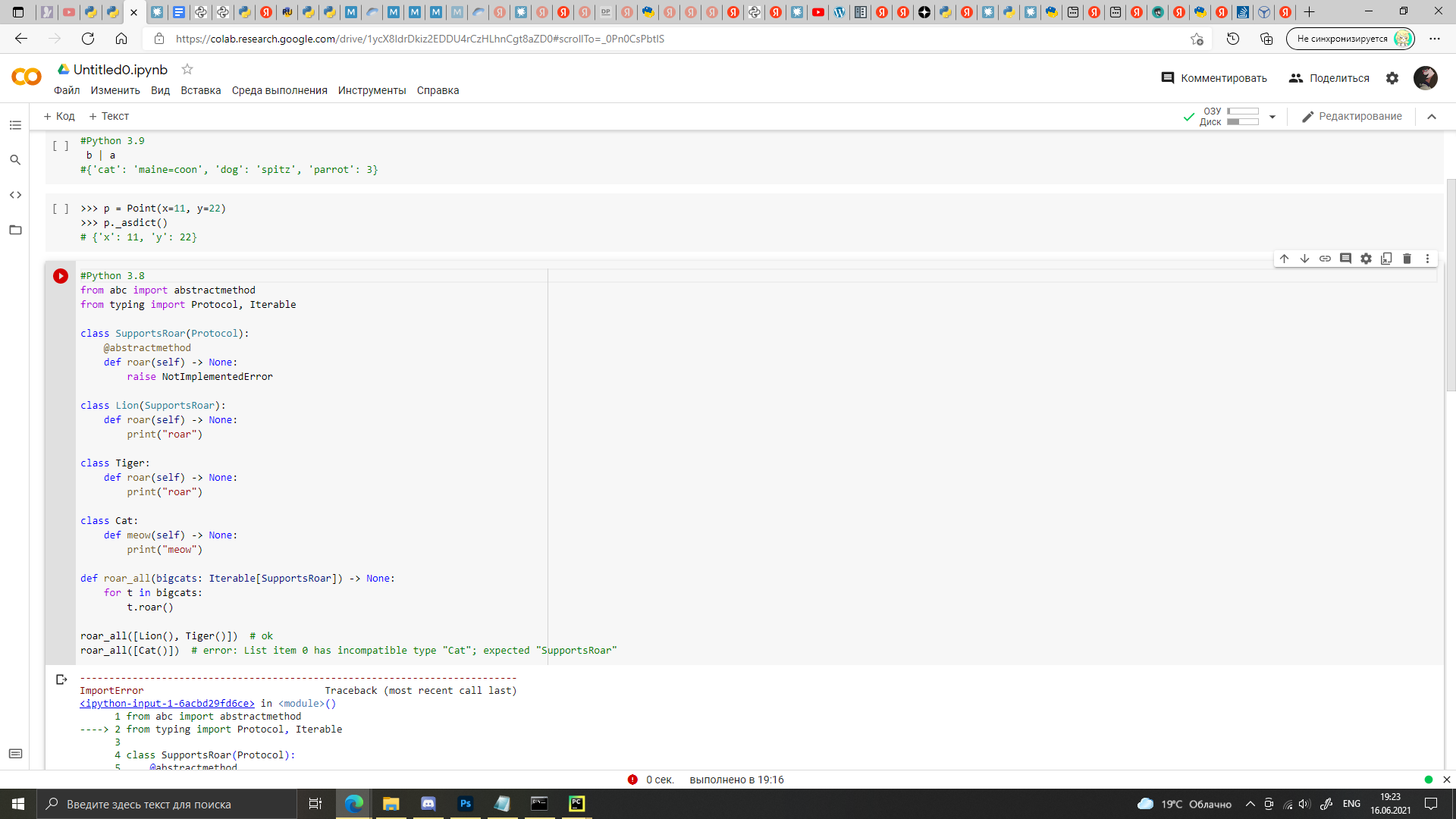
*Изменения, связанные с типами/ Python 3.8*

*Аннотации типов*

1. Протокол
2. from typing import Protocol
3. Протокол описывается как обычный класс, наследующийся от Protocol. Он может иметь методы (в том числе с реализацией) и поля.

Реальные классы, реализующие протокол могут наследоваться от него, но это не обязательно.

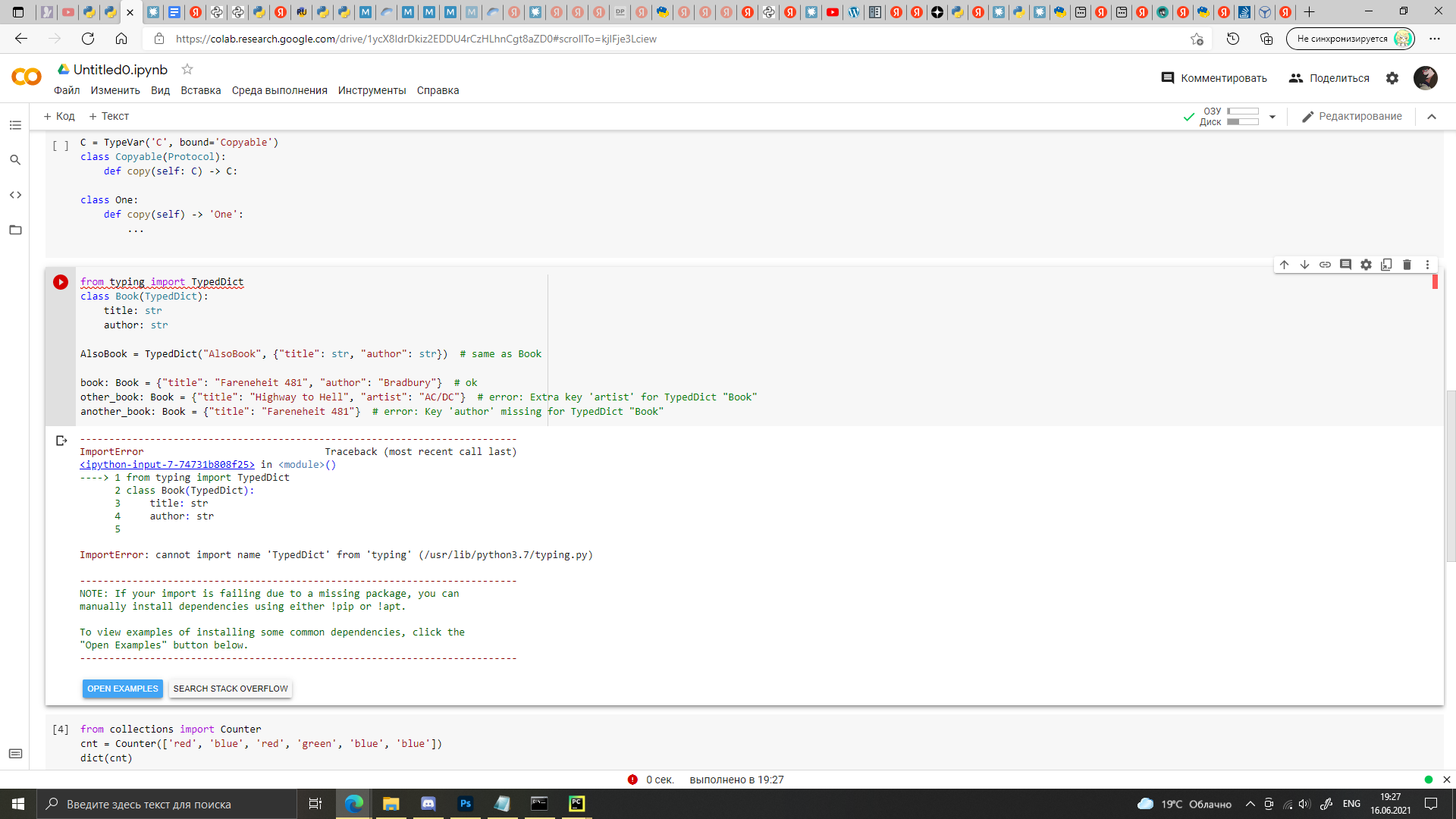
Один из важных типов протоколов — self-typed ([PEP 484](https://www.python.org/dev/peps/pep-0544/#id10)).



**1.TypedDict**

2.from typing import TypedDict

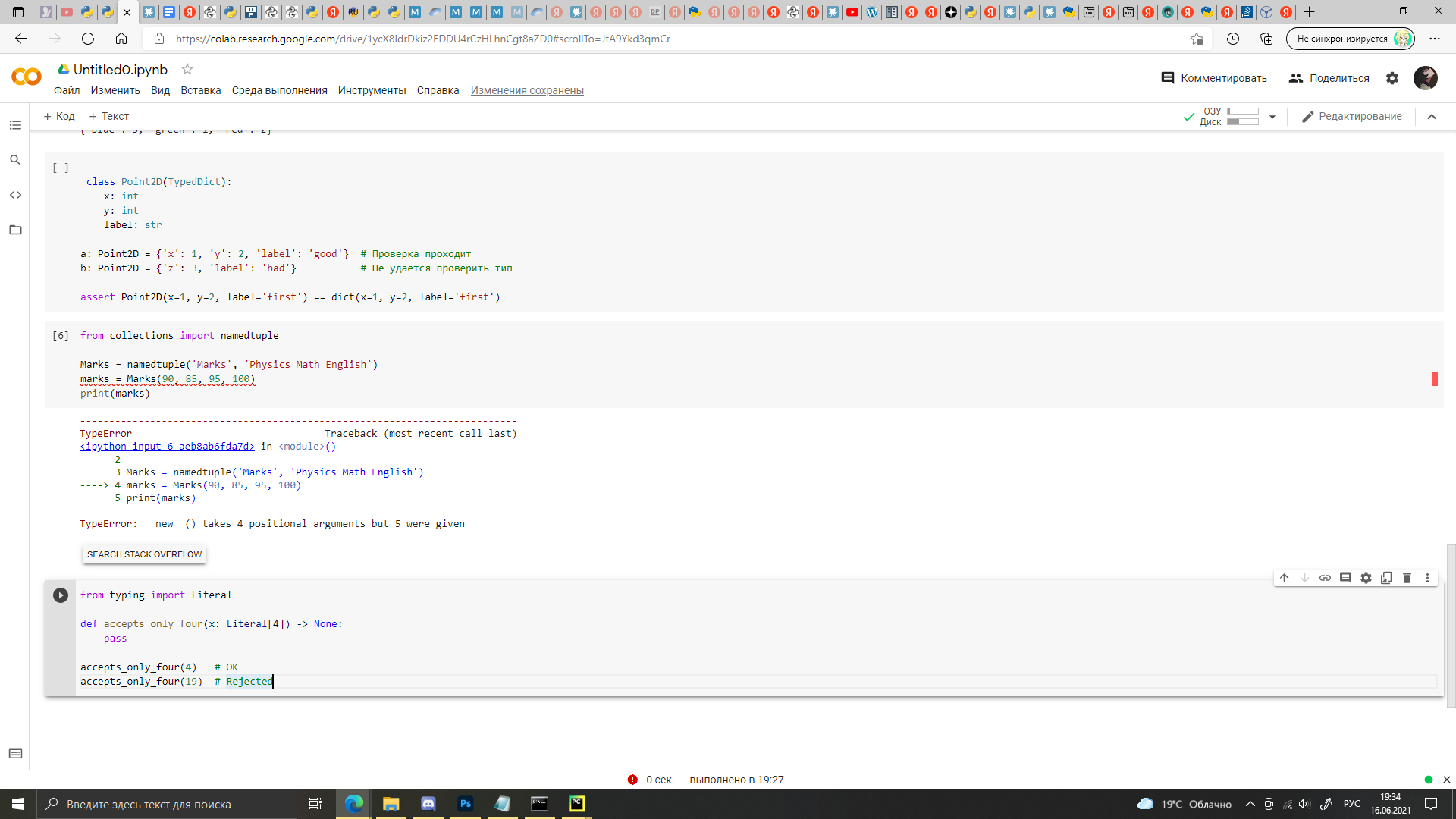
3. Тип аннотации [TypedDict()](https://docs-python.ru/standart-library/modul-typing-python/tip-annotatsii-typeddict-modulja-typing/) модуля [typing](https://docs-python.ru/standart-library/modul-typing-python/) представляет собой специальную конструкцию для добавления аннотаций типов в словарь. Во время выполнения это простой [словарь dict](https://docs-python.ru/tutorial/osnovnye-vstroennye-tipy-python/tip-dannyh-dict-slovar/). TypedDict объявляет тип словаря, который ожидает, что все его экземпляры будут иметь определенный набор ключей, где каждый ключ связан со значением согласованного типа. Это ожидание не проверяется во время выполнения, а выполняется только средствами проверки типизации.



## 5. Выбрасываемые исключения можно увидеть на скриншоте.

## **Literal**

1. from typing import Literal
2. Literal-тип, определенный в [PEP 586](https://www.python.org/dev/peps/pep-0586/) используется когда нужно проверить на конкретным значениям буквально (literally) Например, Literal[4] означает, что в качестве значения ожидается только 4. Важно, что проверяется не только равенство значения, но и его тип (например, нельзя будет использовать False, если ожидается 0).



5. *Изменено в Python 3.9.1*: Тип аннотации Literal теперь дедуплицирует параметры. При сравнении на равенство объектов Literal больше не зависит от порядка допустимых литералов. Литеральные объекты теперь будут вызывать [исключение TypeError](https://docs-python.ru/tutorial/vstroennye-iskljuchenija-interpretator-python/vstroennye-iskljuchenija/) во время сравнения на равенство, если один из их параметров является изменяемым типом.