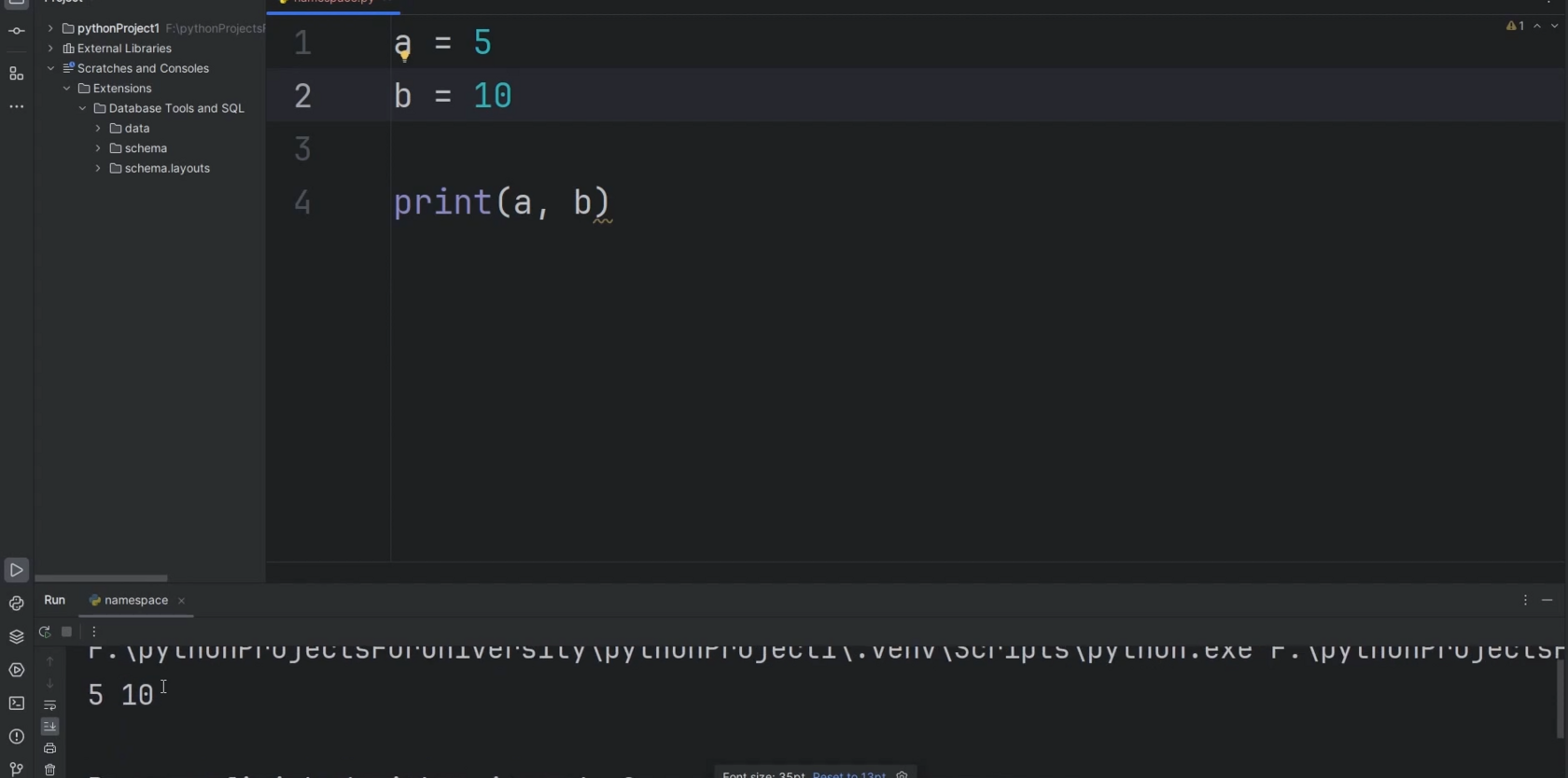
**2023/10/04 00:00|Лекция. Пространство имен**

**Пространство имен**

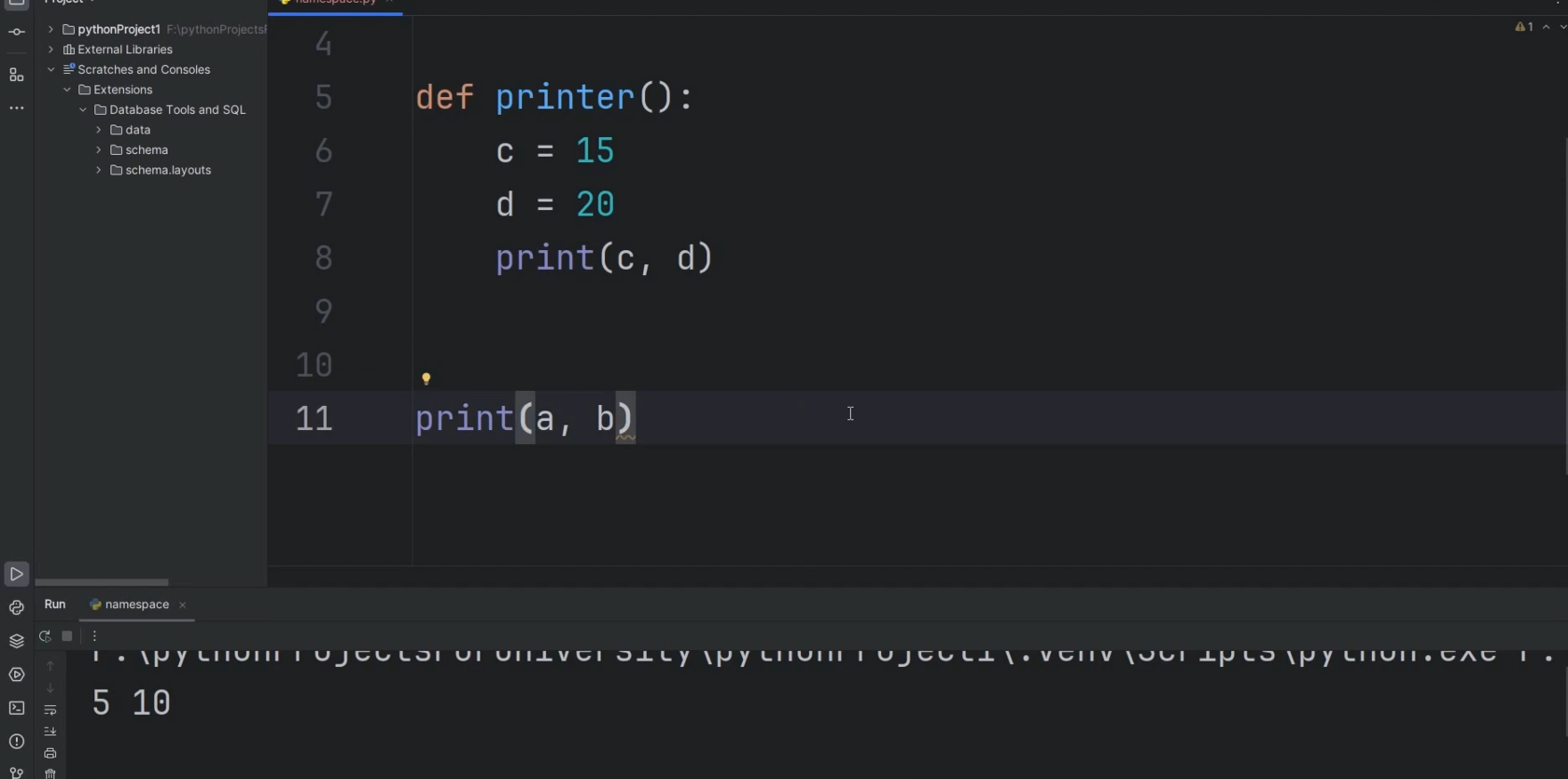
Пространство имен – это своего рода система, которая отвечает за то, чтобы все имена в программе были уникальными и могли использоваться без каких-либо проблем или конфликтов.  
  
**Существует несколько видов пространства имен:**

* Локальное пространство имен
* Глобальное пространство имен
* Встроенное пространство имен

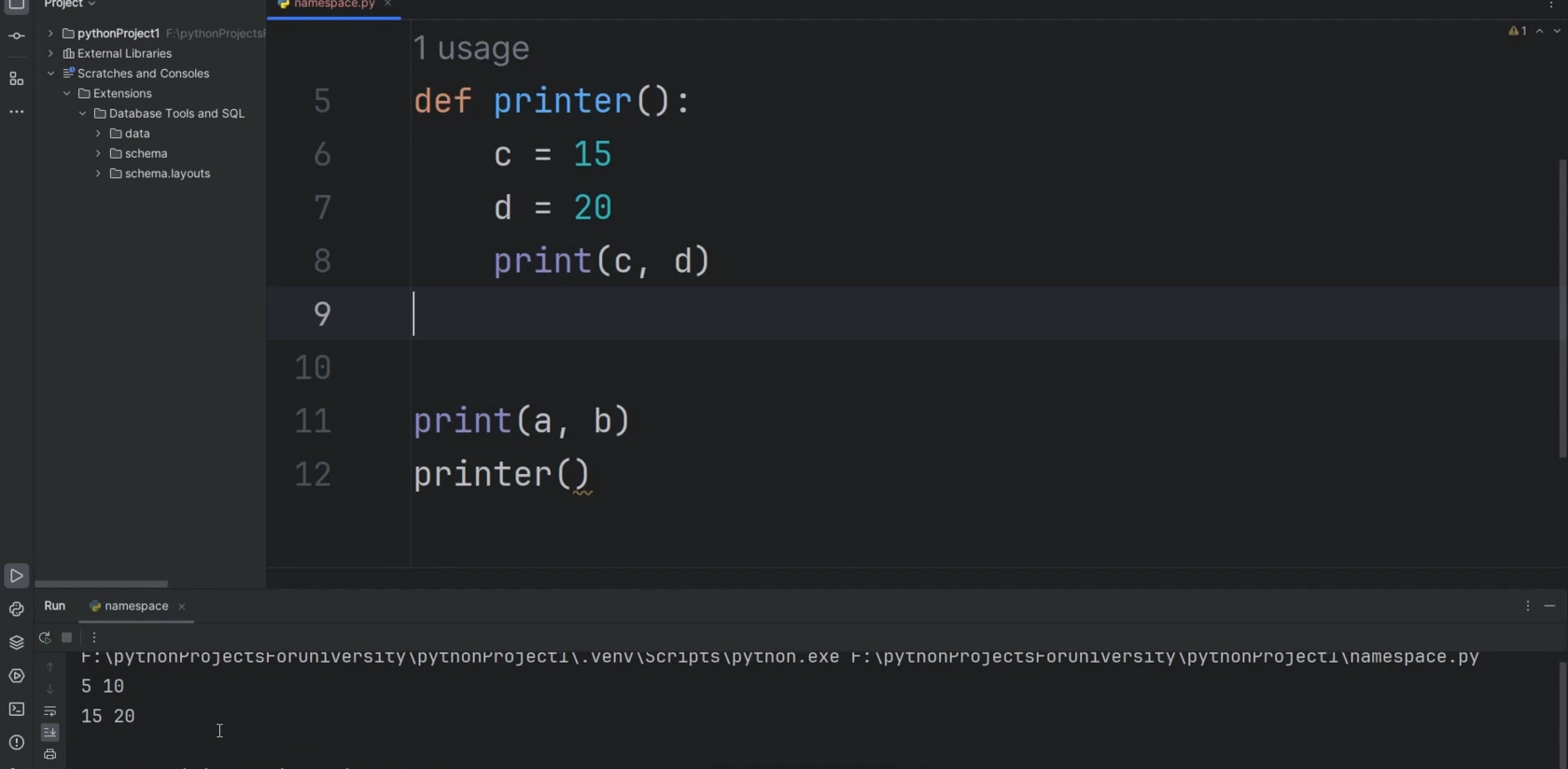
**Встроенное** пространство имен включает в себя имена каких-либо встроенных функций. **Локальное** представляет собой имена, которые мы используем внутри каких-либо функций, которые мы используем и создаем самостоятельно. **Глобальное** пространство имен включает в себя все, что есть в нашей программе, исключая лишь локальное пространство.Рассмотрим на примере. У нас есть переменная «a» и переменная «b» (Рис.1). Сейчас они находятся в глобальном пространстве имен. Мы можем легко к ним обращаться, запуская программу.



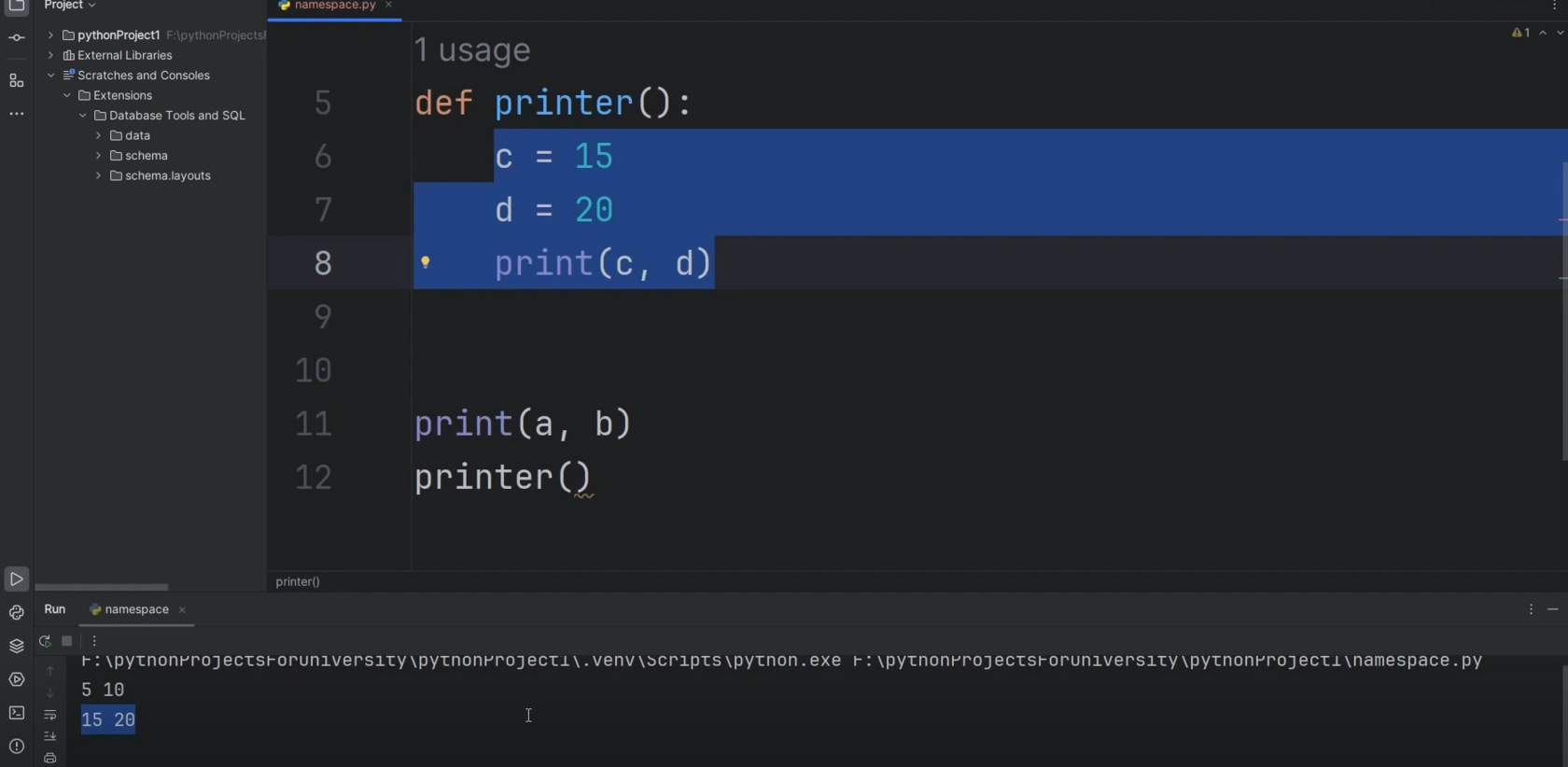
(Рис.1)  
Но представим ситуацию, когда мы создаем какую-либо функцию. Например, назовем ее «printer», она будет создавать переменные «с» и «d» и выводить значение этих переменных (Рис.2).



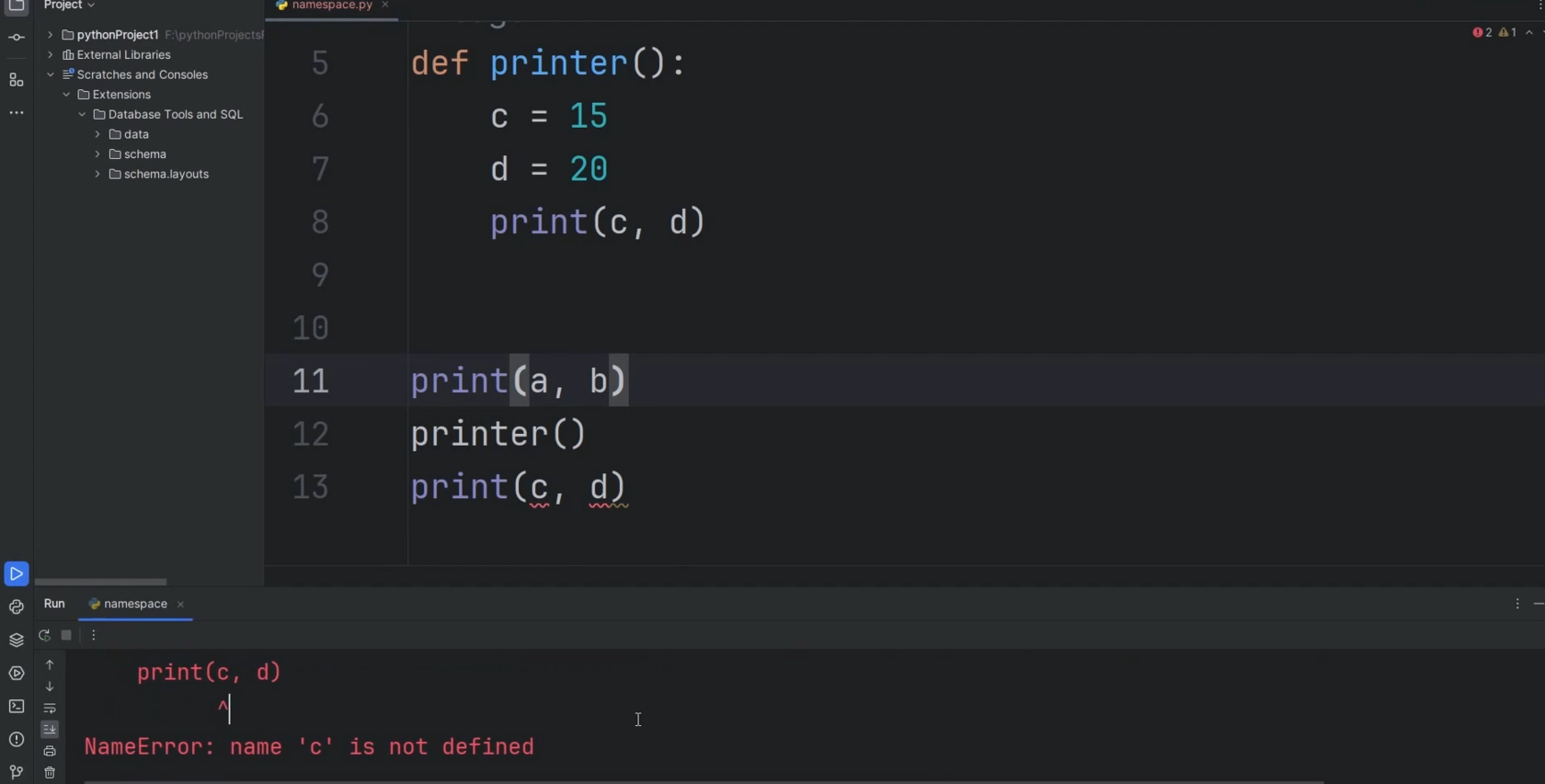
(Рис.2)  
Как мы знаем, чтобы функция отработала код, который находится внутри нее, нам нужно ее вызвать, то есть написать ее имя, открыть и закрыть скобки.  
Если мы запустим программу, то увидим, что переменные «c» и «d» тоже выведены на экран (Рис.3).



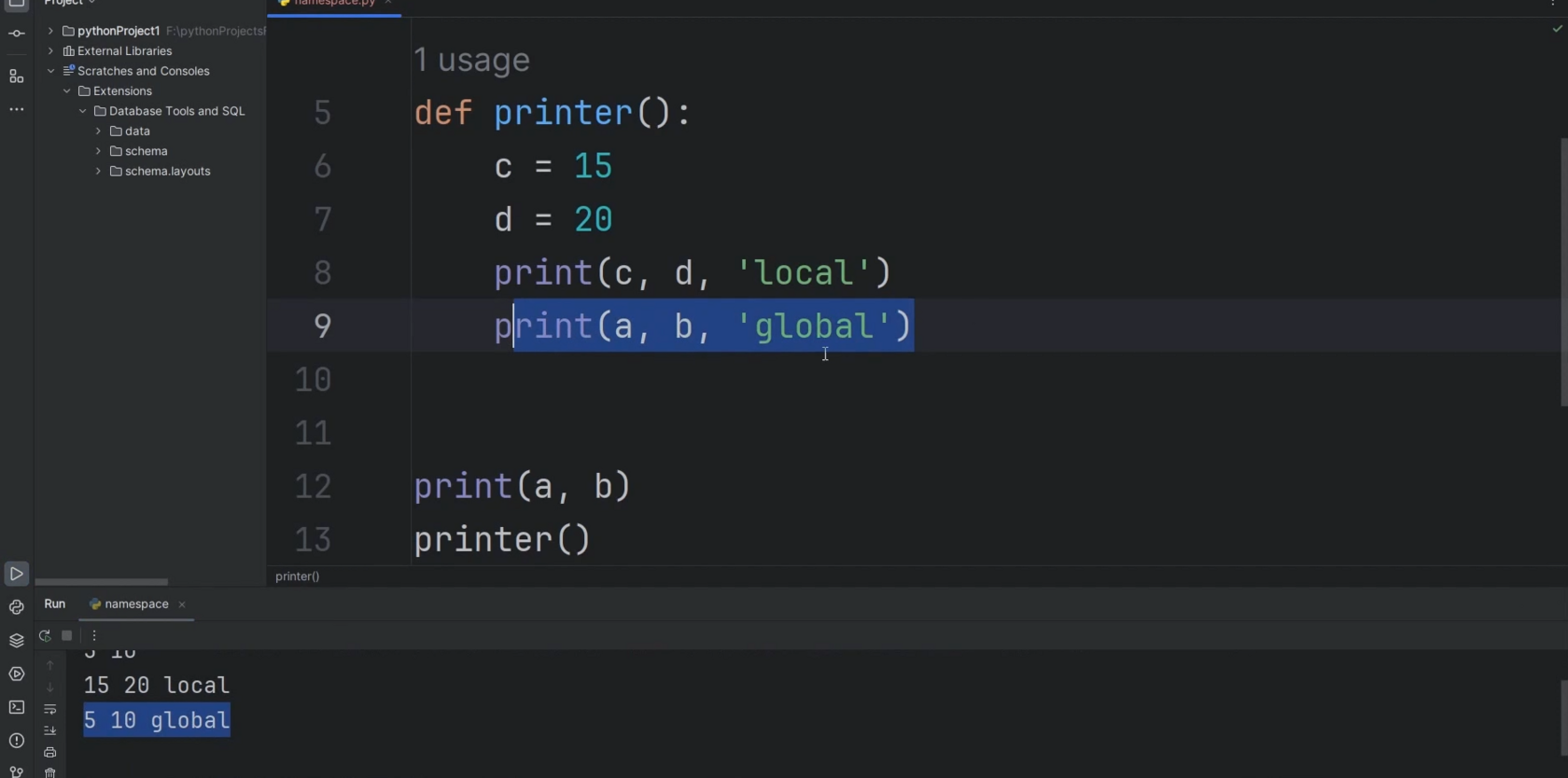
(Рис.3)  
Часть, которая находится внутри функции, и является локальным пространством имен (Рис.4). Переменные «c» и «d» существуют до тех пор, пока функция не завершит свою работу.



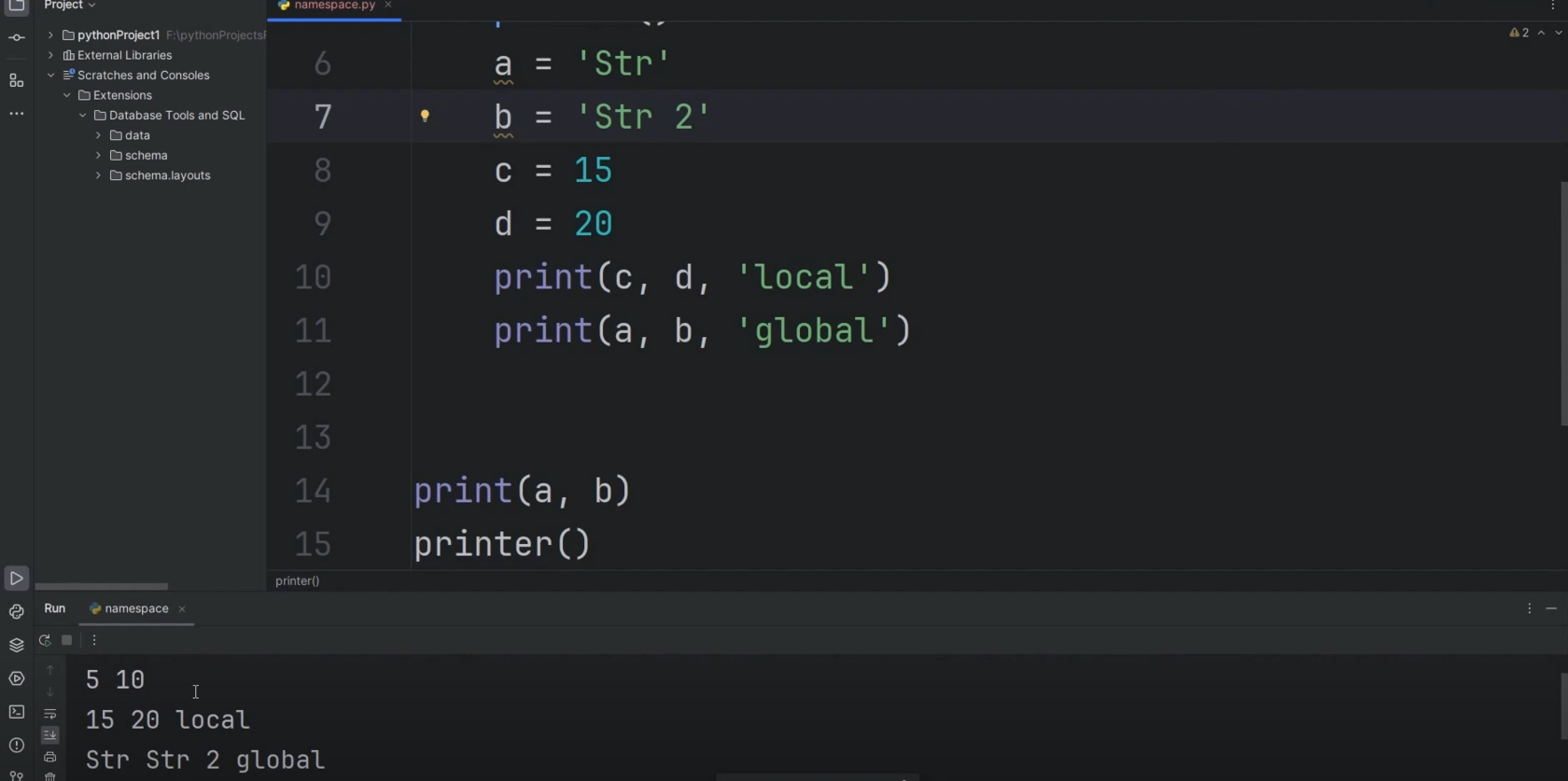
(Рис.4)  
Если мы попытаемся обратиться к этим элементам в нашей основной программе, то сразу получим ошибку, говорящую о том, что таких имен у нас нет (Рис.5). Таким образом, эти имена мы можем использовать лишь внутри функции.



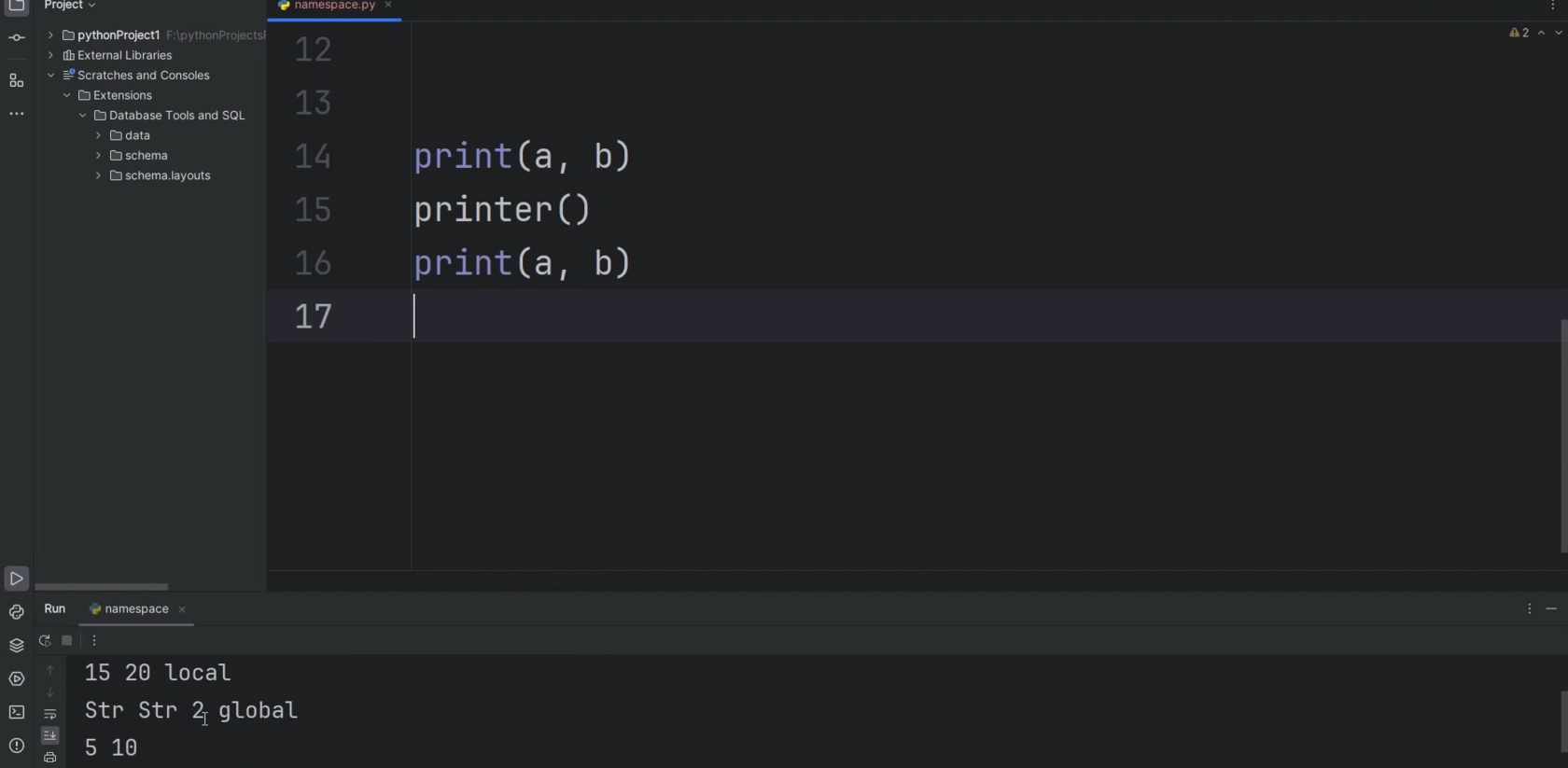
(Рис.5)  
В свою очередь есть такая особенность, что к глобальному пространству имен мы можем обращаться напрямую из локального (Рис.6). Сделаем для себя пояснение, что «c» и «d» у нас local, а «a» и «b» это global.



(Рис.6)  
То есть внутри локального пространства имен функции «printer» мы обратились к значениям из глобального пространства.  
А теперь давайте попробуем создать переменную «a» со значением «Str» и переменную «b» со значением «Str 2» внутри функции. Запускаем (Рис.7).



(Рис.7)  
В итоге мы получаем значения не глобальных переменных «a» и «b», а переменных, которые были объявлены в теле функции «printer».  
Тогда что будет, если мы обратимся к переменным «a» и «b» снова, но уже после вызова функции? Можно было предположить, что после переопределения этих переменных нам выдаст их значения «Str» и «Str 2», но как мы видим, значения вывелись из глобального пространства (Рис.8). Все потому, что те переменные были созданы внутри функции, а значит после ее завершения они не имеют силы. Получается, что несмотря на то, что имена переменных одинаковы, они находятся в разных пространствах имен и, по сути, уникальны.



(Рис.8)  
Но если нам необходимо взаимодействовать с переменными из глобального пространства внутри локального пространства имен функции «printer»? В таком случае мы пропишем в самом начале функции, что используем глобальные переменные «a» и «b» (Рис.9).



(Рис.9)  
При запуске программы мы видим, что теперь даже после завершения функции используются значения, присвоенные переменным внутри нее. Таким образом во время вызова функции мы переопределили значения переменных «a» и «b» из глобального пространства имен.