Python

3-я неделя

Множества в языке Python

- Множество – это не упорядоченная коллекция элементов.
- Когда мы имели дело со списками или кортежами, то там, когда мы добавляли элементы. Допустим, создавали список и в него добавляли три элемента, то они идут в том же порядке, в котором мы их добавляли.
- По индексно. Для примера: 1, 2, 3
- Т.е как мы клали элементы в этот список, либо в этот кортеж… Соответственно, в таком же порядке эти элементы и находятся в данной коллекции.

Множества в языке Python

- В случае с множествами всё несколько иначе. Множества – это, повторяемся, неупорядоченная коллекция. Это нужно понимать, это нужно запомнить.
- Т.е если вы кладёте всё те же 1, 2, 3, то они могут идти в произвольном порядке. Мало того, если вы перезапустите код, то порядок может снова измениться и быть совершенно другим.
- Поэтому не стоит рассчитывать на то, что если вы положили во множество два элемента, условно говоря, какие нибудь «яблоко» или «банан». Разумеется, не стоит думать, что «яблоко» будет идти первым, а «банан» вторым.

Множества в языке Python

- Второй момент отличия от списков и кортежей это то, что множество не содержит повторяющихся элементов. Все дубликаты удаляются при попытке добавления их во множества. Удаляются без ошибок, т.е просто если вы кладёте, скажем, 5 элементов и из них один или два элемента повторяются несколько раз, то в множества попадут только уникальные элементы.
- Все дубли будут выкинуты.

Множества в языке Python

- Например, если кладёте, условно, те же «два яблока» и «один банан», то во множестве будут только одно яблоко и один банан.
- И т.к множество это неупорядоченная коллекция, то, соответственно множества не поддерживают индексирование.

Множества в языке Python

- Давайте попробуем поработать со множествами.
- Учимся определять множества и их создавать.
- Первый способ:

```
51
52  s = {'apple', 'orange', 'apple', 'bear', 'orange', 'banana'}
53  print(s)
54
```

- Что у нас выведет при распечатке?
- Что будет если несколько раз запускать код?

Множества в языке Python

- Окей, мы создали множество с помощью литерала. Теперь попробуем создать его с помощью конструктора:

- Обратите внимание на вывод.

Множества в языке Python

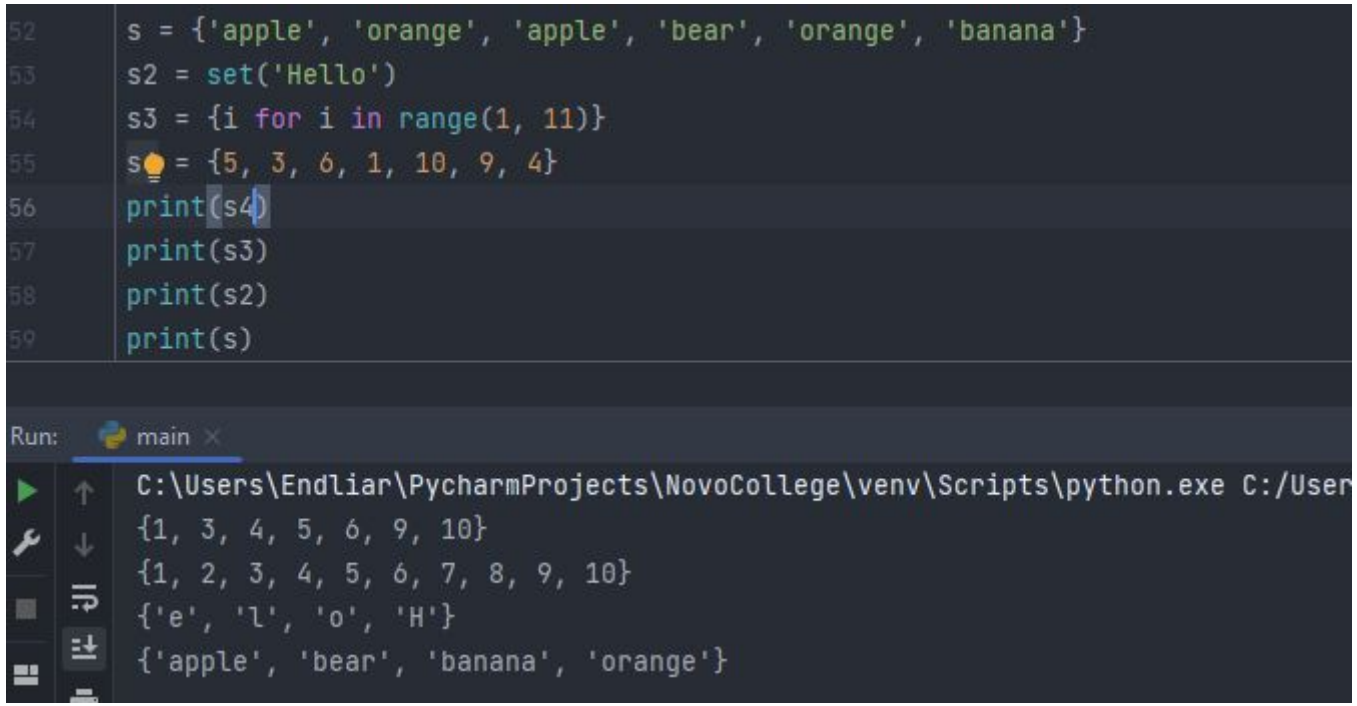
- Третий способ создания множества: с помощью генератора.

```
50
51
52  s = {'apple', 'orange', 'apple', 'bear', 'orange', 'banana'}
53  s2 = set('Hello')
54  s3 = {i for i in range(1, 11)}
55  print(s3)
56  print(s2)
57  print(s)
```

- Снова обращаем внимание на вывод. В данном случае порядок сохраняется.

Множества в языке Python

- И если мы просто числа сунем обычные в генератор. То он также выведет нам все числа по порядку, по возрастанию.



Множества в языке Python

- Как создать пустое множество?

Множества в языке Python

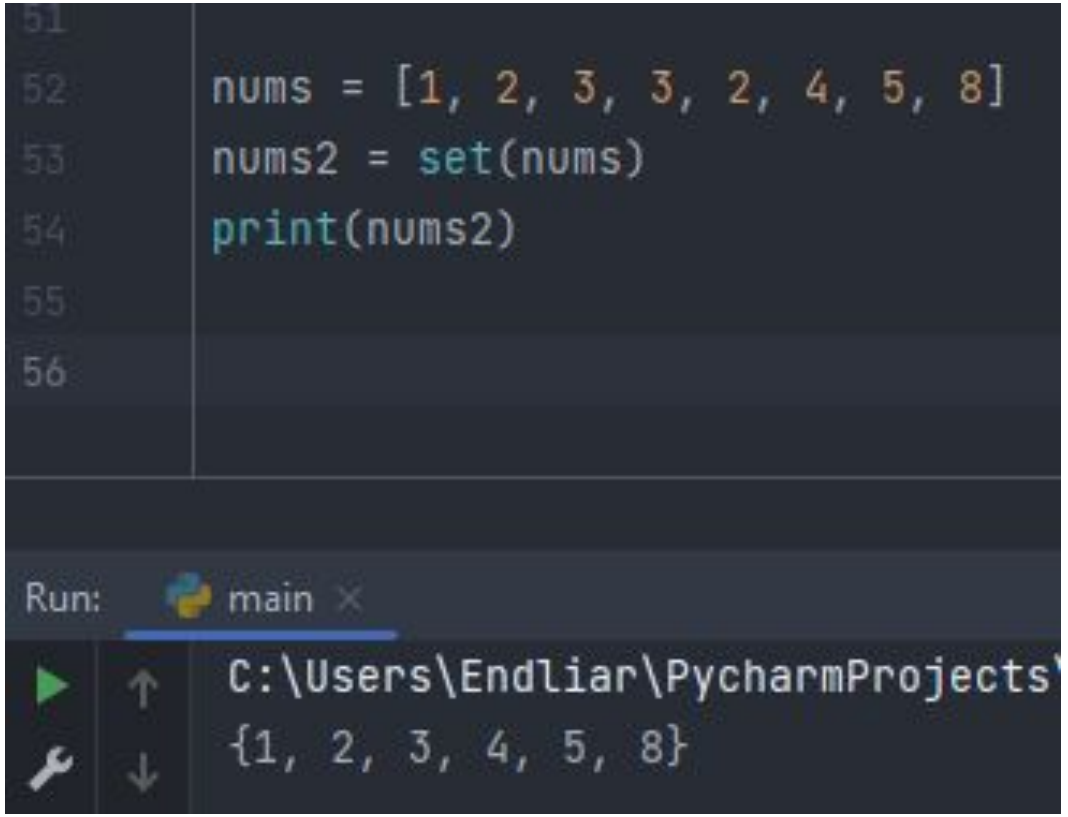
- Чтобы создать пустое множество, достаточно заюзать s5 = set()

Множества в языке Python

- На что еще стоит обратить внимание?
- Множества очень удобно использовать для удаление дубликатов из списка. Поскольку во множестве добавляются только уникальные элементы, все дубли отбрасываются… Соответственно, перед нами встаёт задача из списка взять только уникальные элементы, убрав дубли.
- Можем воспользоваться командой set для этой задачи.

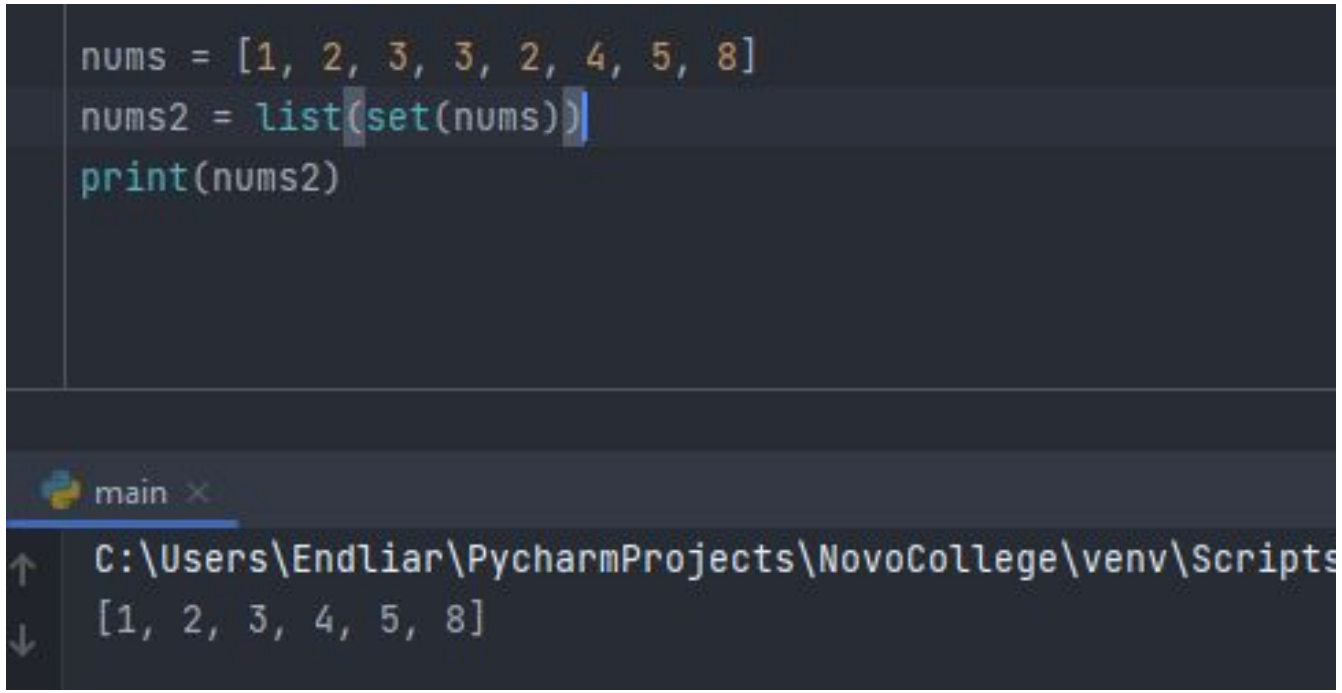
Множества в языке Python

- Например, у нас есть список цифр. И из него нам надо получить только уникальные элементы.



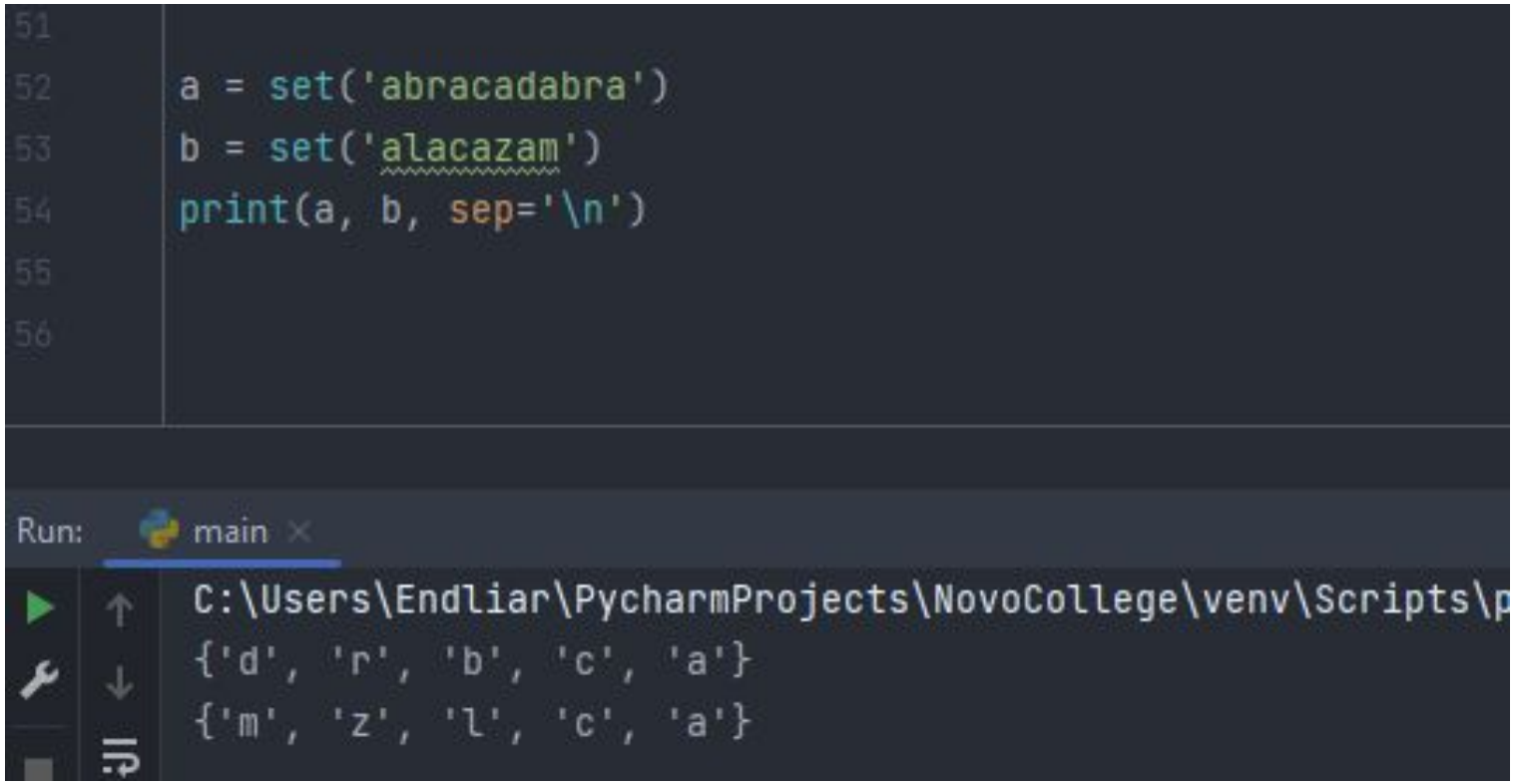
Множества в языке Python

- Соответственно, как можно преобразовать полученное множество в список изначальный?



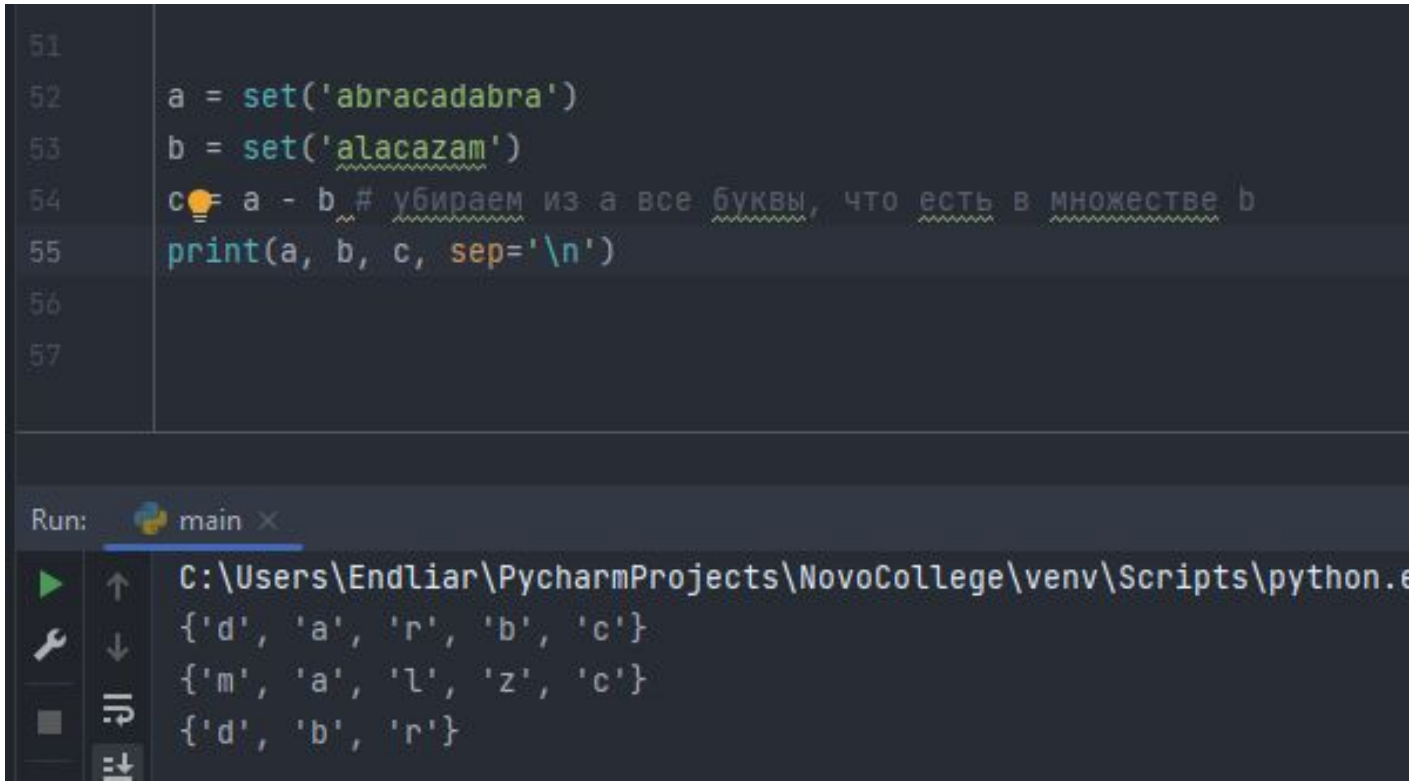
Множества в языке Python

- Окей, посмотрим, какие операции над множествами мы можем производить.



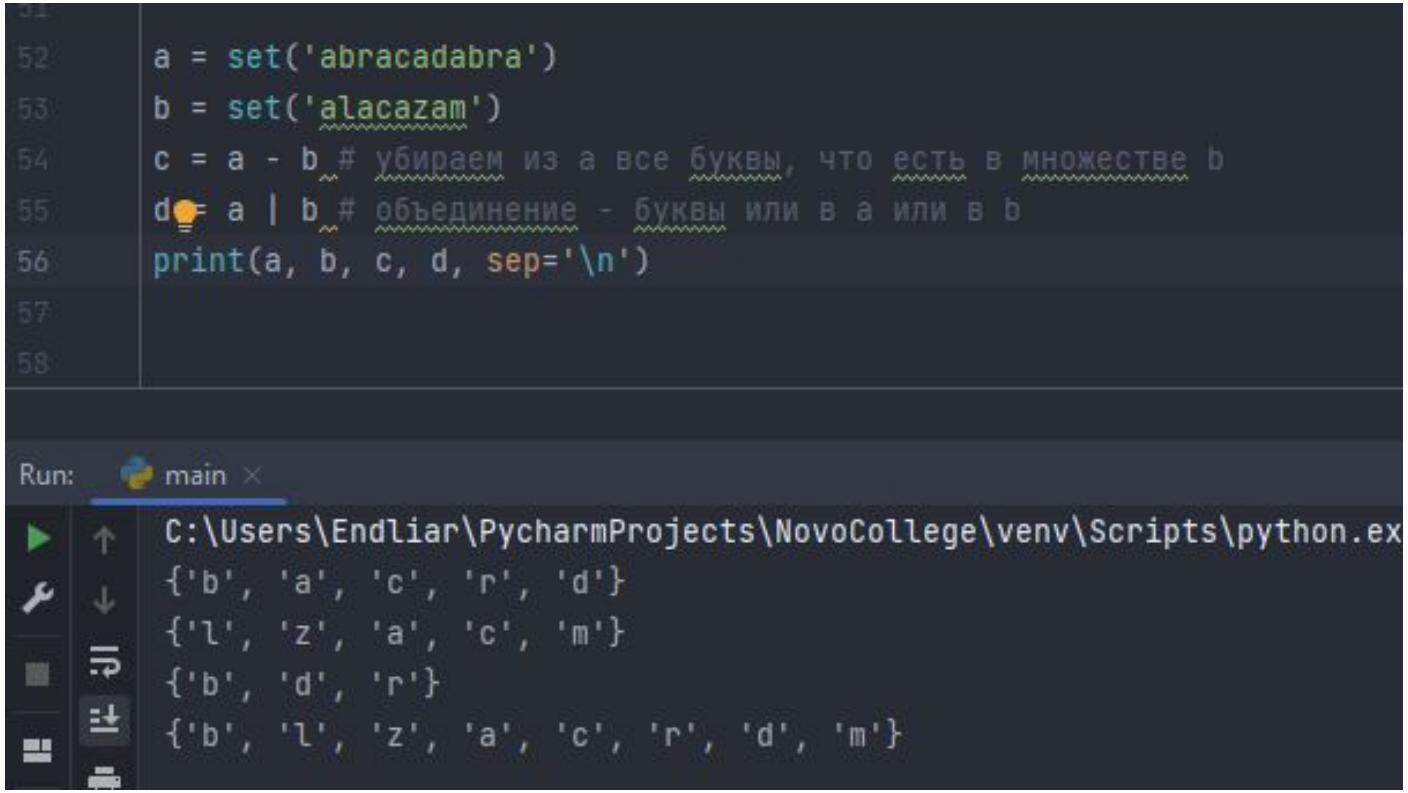
Множества в языке Python

- Еще мы можем проводить операции вычитания. Мы можем из одного множества вычесть другое множество.



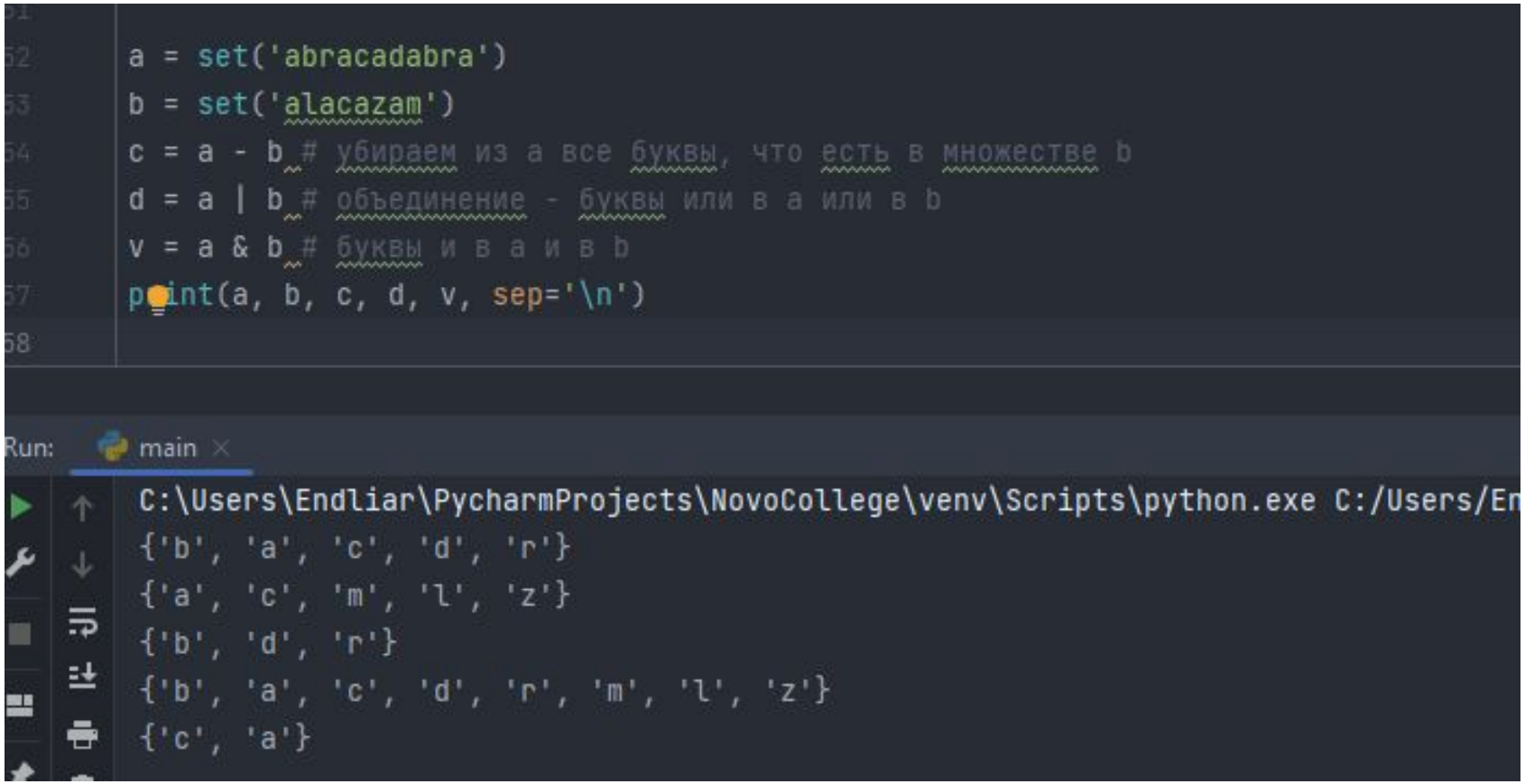
Множества в языке Python

- Операция объединения.



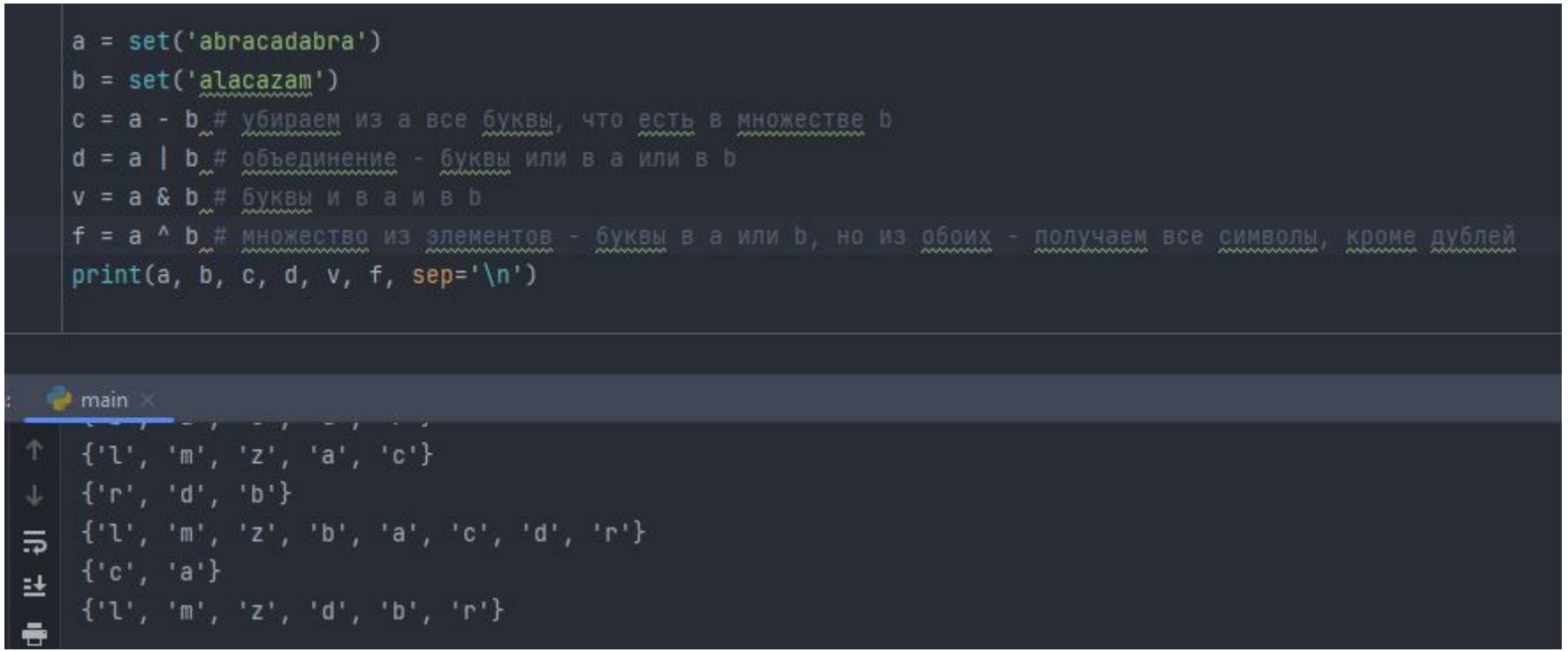
Множества в языке Python

- Операция пересечения.



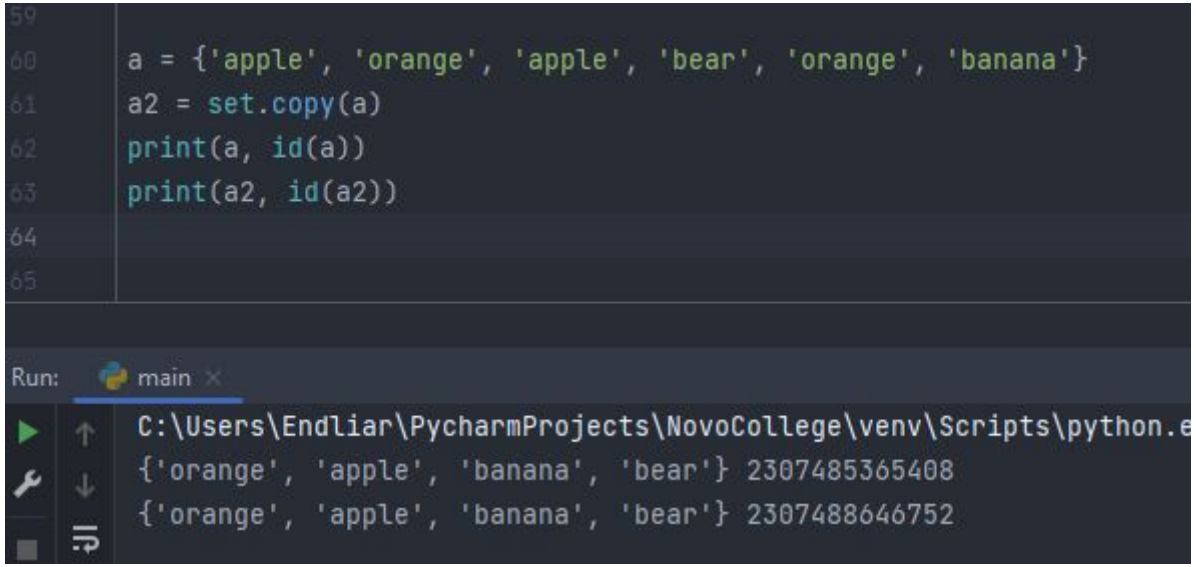
Множества в языке Python

- Получаем множество из элементов. Получаем все символы, кроме дублей.



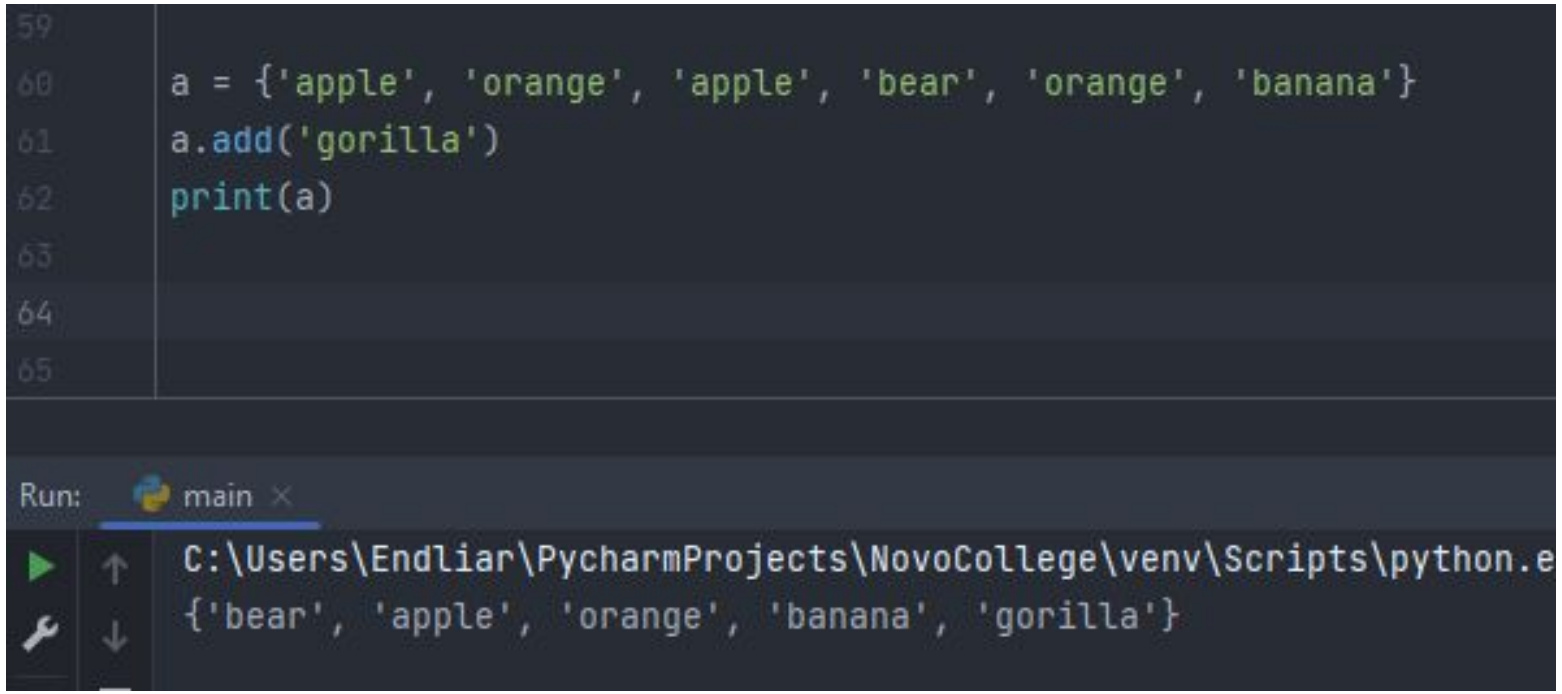
Множества в языке Python

- Итак. С операциями над множествами мы разобрались.
- Теперь посмотрим на методы, которые предлагают нам для работы со множествами.
- Первый метод: set.copy(): получилось два !разных! множества.



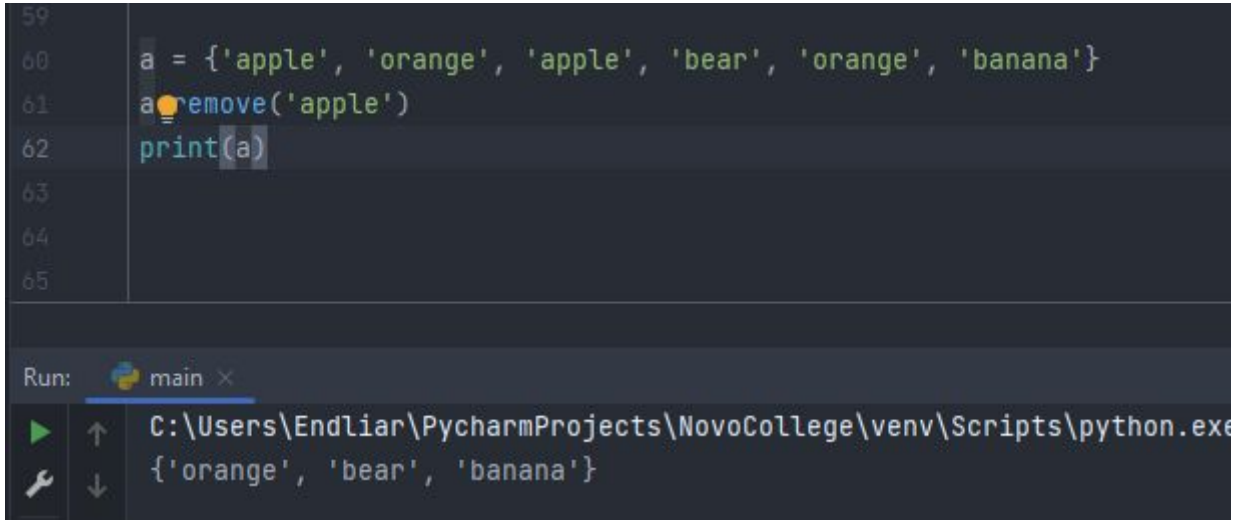
Множества в языке Python

- Второй метод set.add(elem): ну тут и так всё понятно.



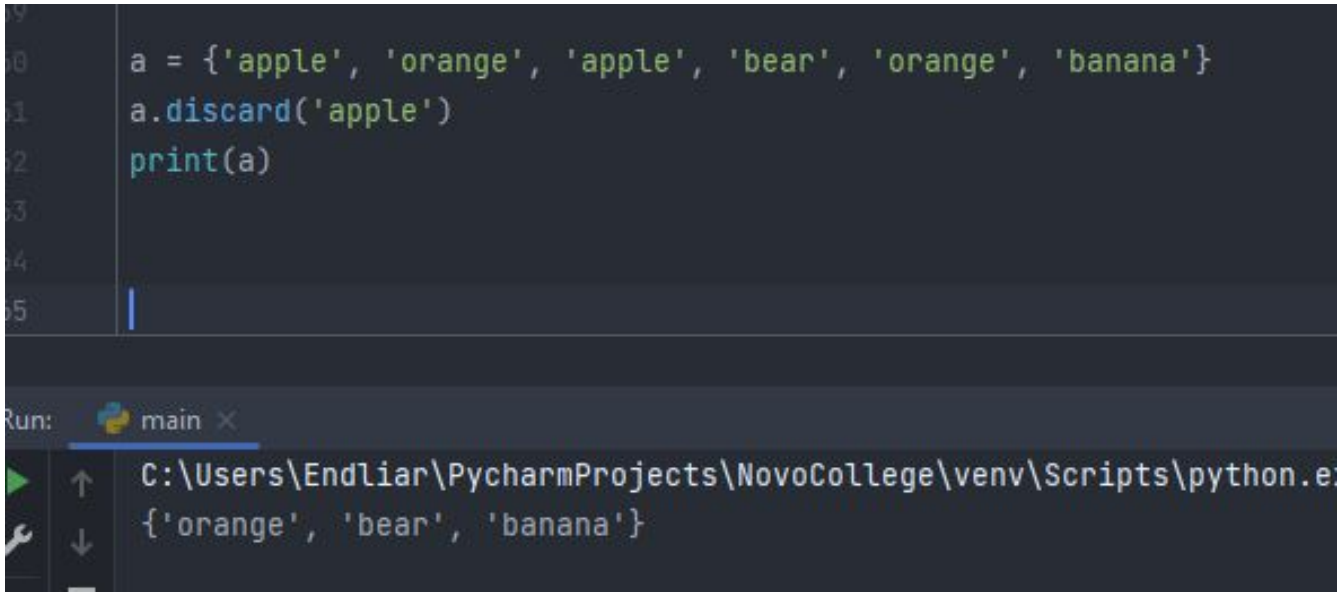
Множества в языке Python

- Третий метод set.remove() – удаляет элемент. В то же время если удаляемого элемента во множестве и не существовало – то будет выдана ошибка.



Множества в языке Python

- Четвёртый метод set.discard() – удаляет элемент, если он находится во множестве. Ошибки не будет, в отличие от метода remove.

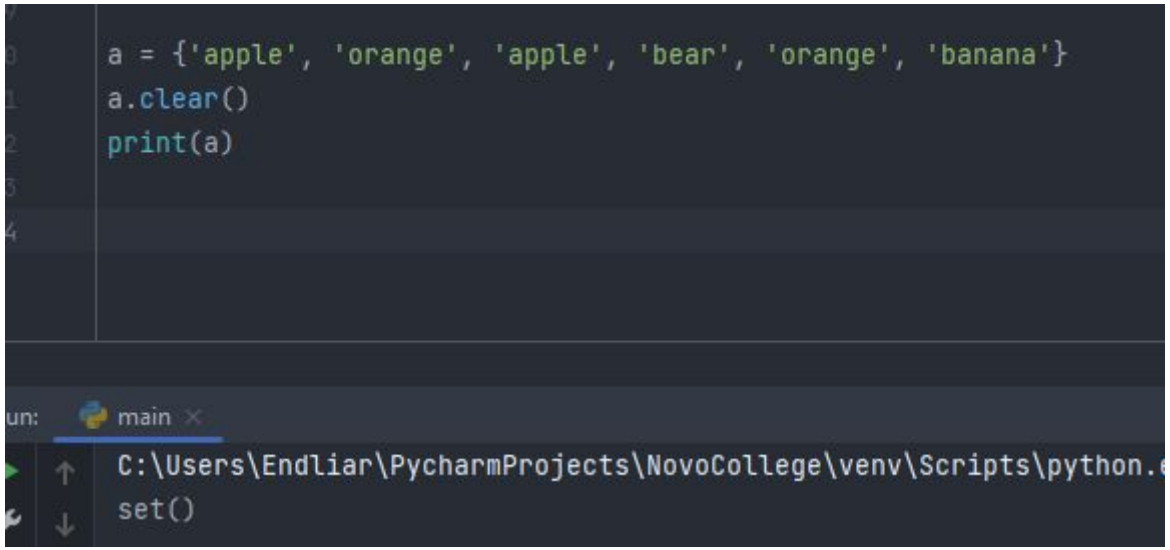


Множества в языке Python

- Пятый метод set.pop() – возвращает и удаляет первый элемент из множества. Так как множества не упорядочены, нельзя точно сказать, какой элемент будет первым.

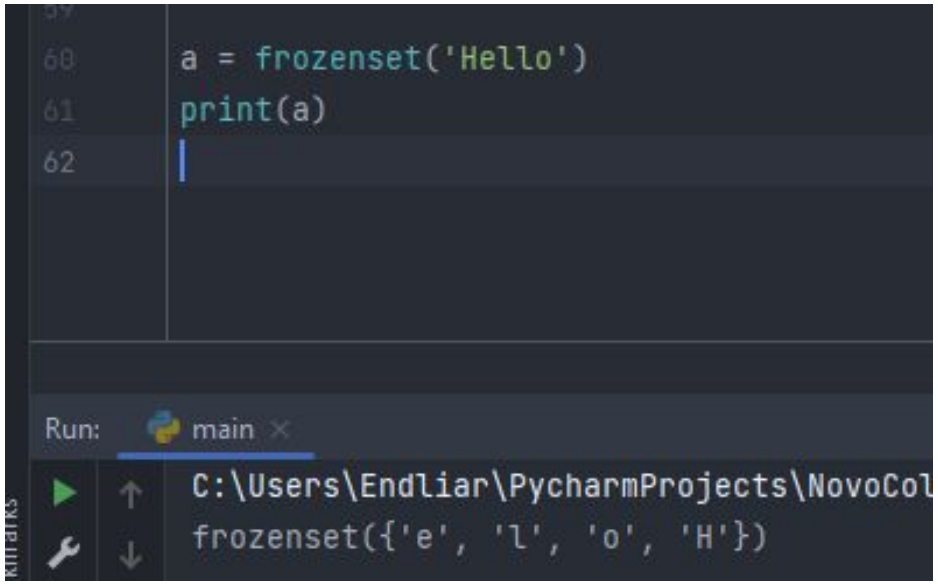
Множества в языке Python

- Шестой метод set.clear() – очистка множества.



Множества в языке Python

- Последний set – frozenset – замороженное множество. Короче говоря, множество, которое мы не можем изменить. Т.е если мы создадим замороженное множество и попробуем в него что-то добавить, то нам даже не будет предложено подсказками метод add.



Множества в языке Python

- Последний set – frozenset – замороженное множество. Короче говоря, множество, которое мы не можем изменить. Т.е если мы создадим замороженное множество и попробуем в него что-то добавить, то нам даже не будет предложено подсказками метод add.

Словари

- Словари в питоне – это еще одна коллекция элементов, так же как и множество неупорядоченная коллекция элементов, произвольных объектов с доступом по ключу.
- Т.е у словарей уже есть ключ и получить доступ к его элементам можно по ключу. Словари в питоне еще называют иногда ассоциативными массивами или хеш-таблицами.
- Т.е есть индексированные списки, и есть ассоциативные массивы, в которых элементы доступны по специальным ключам-строкам.
- Т.е есть какие-либо ассоциации.

Словари

- Способы создания словарей с помощью литерала.

Словари

- Создадим что-нибудь полезное, например, продукты:

Словари

- Теперь попробуем создать словарь через конструктор:

Словари

- Обратим внимание на то, что в конструкторе мы используем уже знакомые нам именованные аргументы и когда мы работает с именованными аргументами, то заключать их в кавычки не нужно.
- Более того, если мы попытаемся это сделать, то Python выдаст нам ошибку. И скажет, что мы дурачки.

Словари

- Следующие, более экстравагантные способы, которые, тем не менее, могут нам так или иначе пригодиться: - это создание словаря из списка или кортежа.
- Несмотря на то, что словарь и список – это, по сути, не похожие типы, не похожие по структуре, но тем не менее существует такая возможность для отдельных видов списков преобразовать их в словарь.
- Делается с помощью уже использованного конструктора dict и для этого список должен хранить набор вложенных списков.

Словари

- Каждый вложенный список при этом должен состоять из двух элементов и при конвертации в словарь, первый элемент станет ключом, а второй значением.
- Соответственно, попробуем создать список users и преобразовать его в словарь.

Словари

```
product2 = dict(title='iPhone', price=120)

users = [
    ['bob@gmail.com', 'Bob'],
    ['sergo@gmail.com', 'Sergo'],
    ['igor@gmail.com', 'Igor']

]

users_d = dict(users)
print(users_d)
print(users)

print(type(d))
print(product1)
```

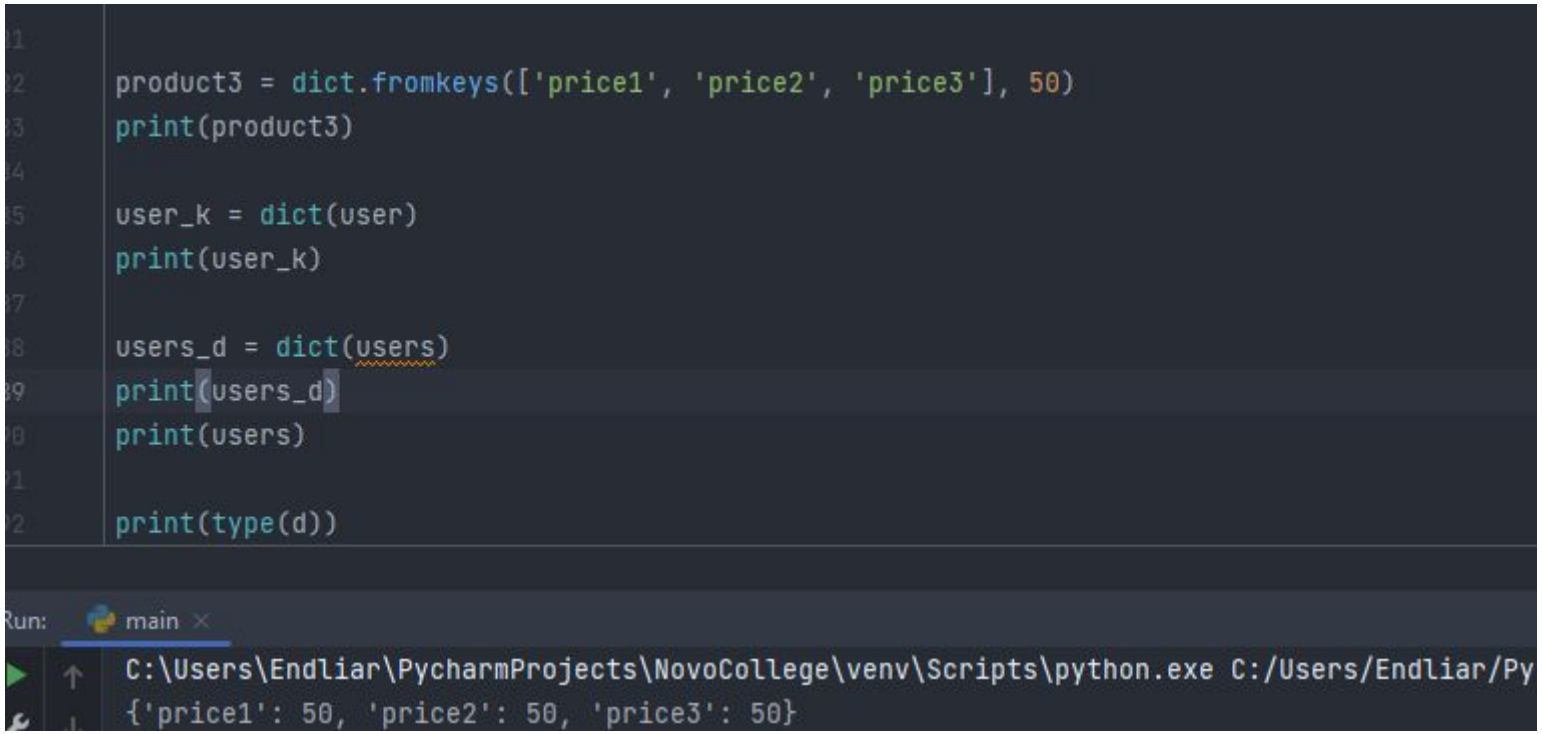
```
main
C:\Users\Endliar\PycharmProjects\NovoCollege\venv\Scripts\python.exe C:/Users/Endliar/Pych
{'bob@gmail.com': 'Bob', 'sergo@gmail.com': 'Sergo', 'igor@gmail.com': 'Igor'}
[['bob@gmail.com', 'Bob'], ['sergo@gmail.com', 'Sergo'], ['igor@gmail.com', 'Igor']]
```

Словари

- Аналогичное можно сделать и с кортежами.

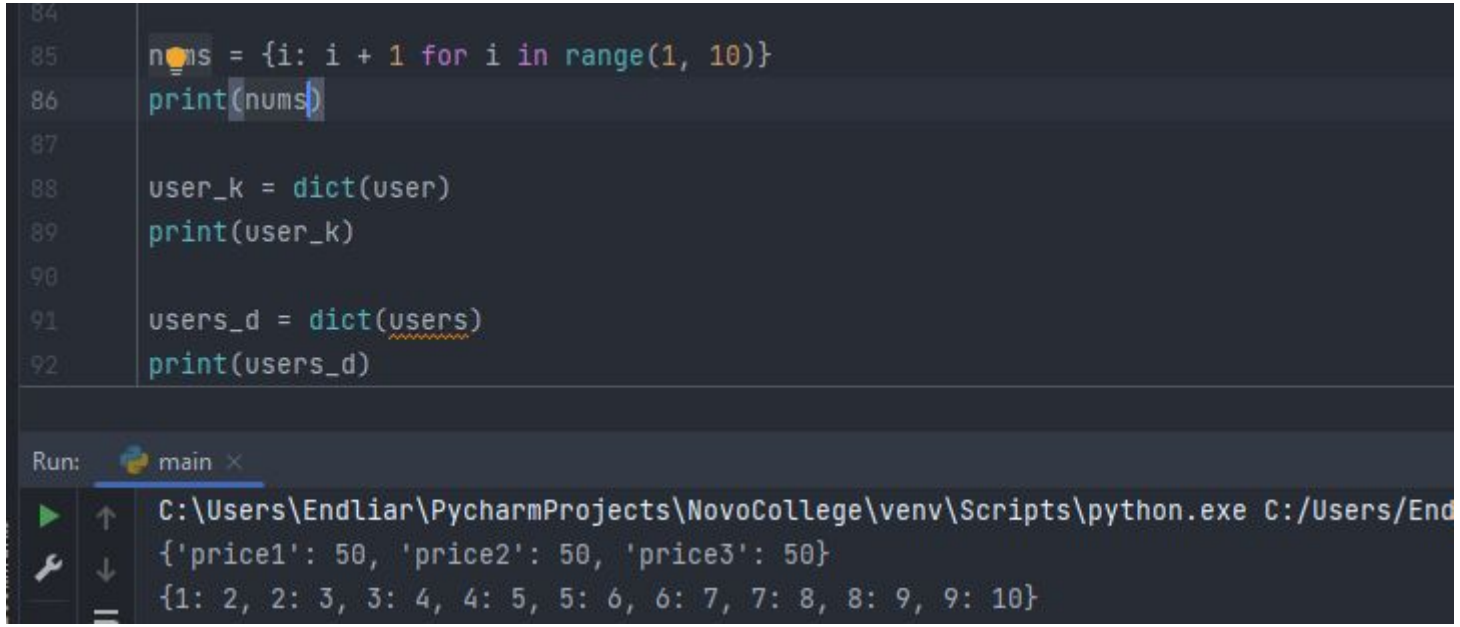
Словари

- Далее рассмотрим уже метод словарей fromcase и с помощью данного метода мы можем быстро создать какой-нибудь однотипный словарь с однотипными значениями.



Словари

- Также для создания словарей мы так же можем использовать уже знакомые нам генераторы.



Словари

- Как мы можем попробовать обратиться к значению в словаре?

- Разумеется, следует помнить, что если мы обратимся к несуществующему ключу, то получим ошибку.

Словари

- Соответственно, обратиться к нашему недавно созданному циклу можно…?