Знакомство со словарями

Словари

- Важным типом данных являются словари. Словарь можно представить в виде некоторого списка, но при этом роль индексов могут играть не только числовые значения. Значения, которые играют роль индексов, называются ключами (ключом, например, может быть текст, число или кортеж — но всегда это значение неизменяемого типа). Другими словами, имеется набор значений, и каждое значение из этого набора идентифицируется с помощью ключа.

Словари

- Создать словарь можно с использованием фигурных скобок: в блоке из этих скобок указываются через запятую выражения вида «ключ: значение», которые определяют ключи и соответствующие им значения в словаре.

Словари

- Также для создания словаря используется функция dict(). Если ключами словаря являются текстовые значения, то в качестве аргументов функции dict() могут передаваться конструкции вида «ключ = значение». Такие выражения разделяются запятыми. В этом случае создается словарь, в котором ключи и значения определяются аргументами, переданными функции.

Подробности

- Ключи в таком случае (хотя они являются текстовыми) указываются без кавычек. Причина в том, что на самом деле в данном случае ключи играют роль именованных аргументов (именованные аргументы функций описываются несколько позже, в одной из следующих глав). В этом случае на ключи накладываются ограничения как на названия аргументов (например, это не может быть два слова). Также стоит отметить, что описанный способ передачи аргументов функции dict() не является единственным.

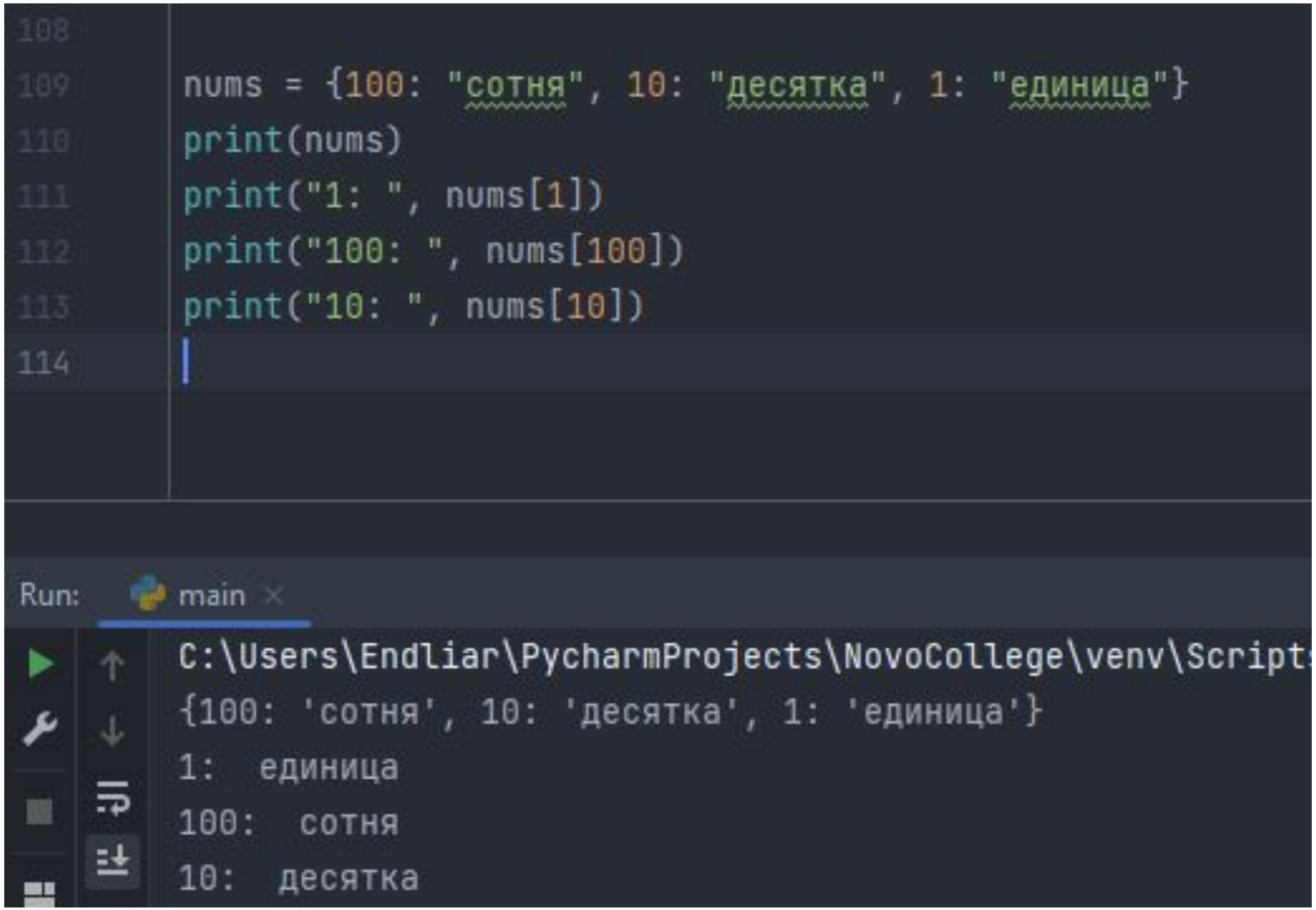
На заметку

- Вызов функции dict() без аргументов приводит к созданию пустого словаря. Пустой блок из фигурных скобок {} также соответствует пустому словарю (не множеству). Как и в случае со списками, кортежами и множествами, количество элементов в словаре можно определить с помощью функции len().

Словари

- Если словарь создан, то доступ к значению из этого словаря можно получить по ключу. Доступ выполняется в формате «словарь [ключ]» - то есть после имени словаря в квадратных скобках указывается ключ для доступа к соответствующему значению. Небольшая программа, в которой создаются словари, представлена скрином ниже:

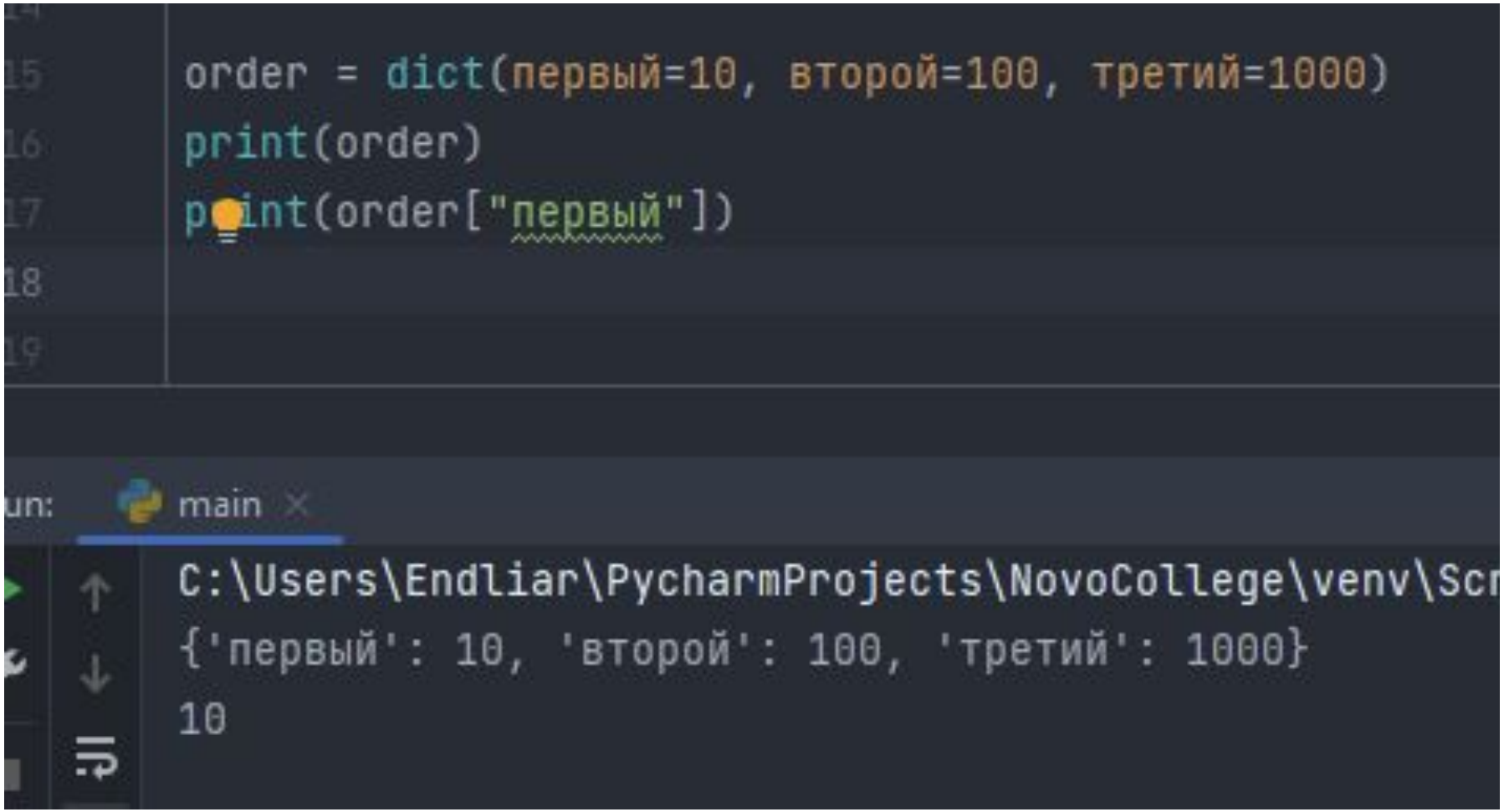
Создание словаря



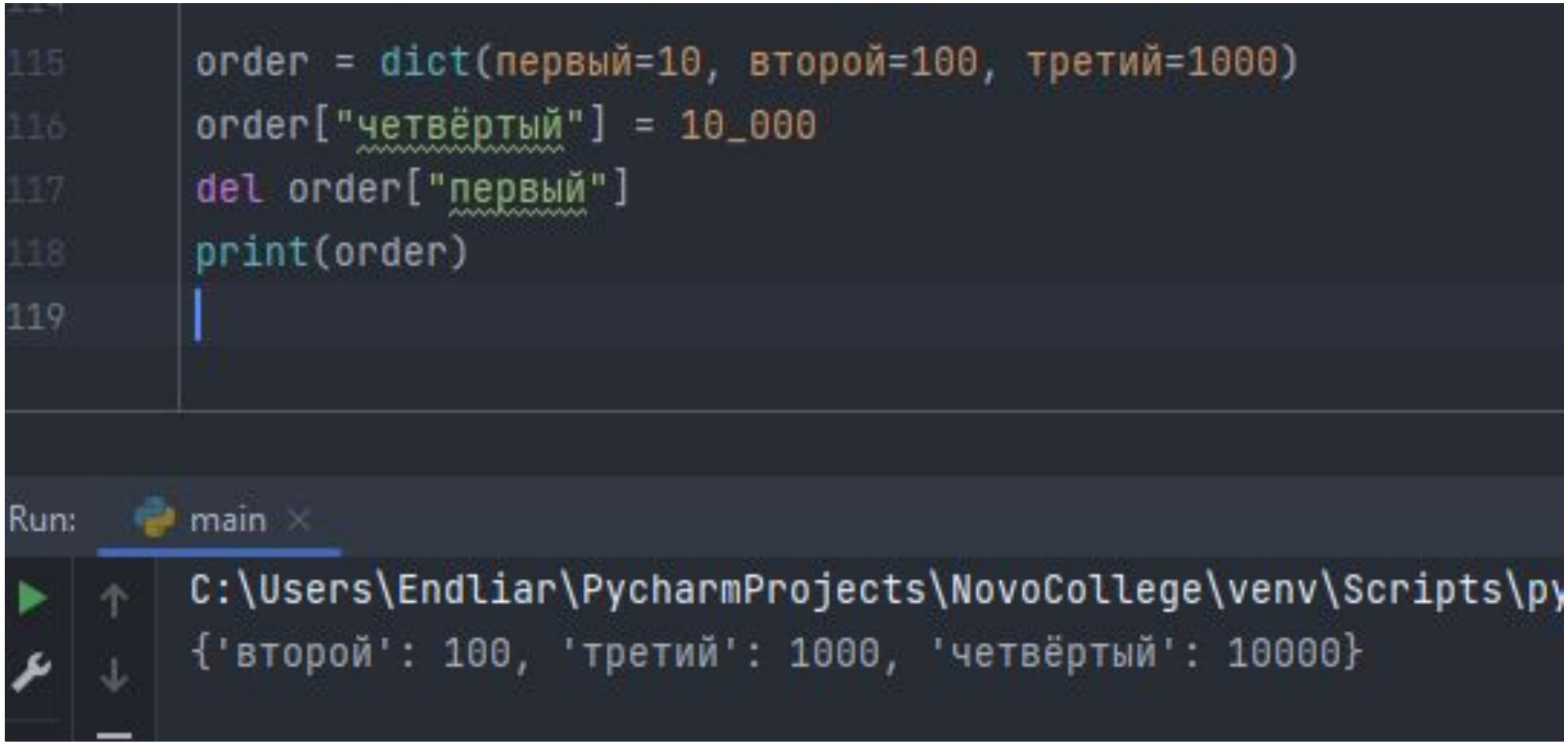
Операции со словарями

```
108
109     nums = {100: "сотня", 10: "десятка", 1: "единица"}
110     nums[3] = "Тройка"
111     nums[15] = "Пятнадцать"
112     nums.pop(100)
113     print(nums)
114
```

Создадим второй словарь



Операции со вторым словарём



Всё о первом словаре

- В программе создается два словаря, и с этими словарями выполняются несложные операции. Первый словарь создается командой nums={100:«сотня", 1:"единица",10:” десятка”}. В этом словаре три элемента. Их ключи — это целые числа 1, 10 и 100, а значения элементов соответственно равны “единица”, “десятка”, “сотня”. Для получения значения элемента словаря после имени словаря в квадратных скобках указывается ключ соответствующего элемента: например, nums[1], nums[10] и nums[100].

Всё о первом словаре

- В данном случае считывается значение элемента. В таком же формате элементу можно присвоить новое значение – примером может служить команда nums[10] = “десять”.
- Если указать ключ, которого в словаре нет, то в словарь будет добавлен новый элемент с таким ключом. Например, при выполнении команды nums[3] = “тройка” в словаре nums появляется элемент с ключом 3 и значением “тройка” в словаре nums появляется элемент с ключом 3 и значением “тройка”. Для удаления элемента с определённым ключом можно использовать метод pop().

Всё о первом словаре

- Еще один словарь создаётся командой order = dict(Первый = 1, Третий = 3, Последний = 10). Ключами в данном случае являются текстовые значения “Первый”, “Третий”, “Последний”, а элементы имеют значения 1, 3 и 10 соответственно.

На заметку

- Еще раз обращаем внимание, что текстовые значения для ключей при создании словаря указываются без кавычек. Однако когда выполняется обращение к элементу, то кавычки для текстовых ключей указываются.

Словари

- Для считывания значений элементов словаря используем команды order[“Первый”], order[“Третий”] и order[“Последний”].
- Командой order [“Последний”] = 12 элементу с ключом “Последний” присваивается значение 12. Командой del order [“Третий”] из словаря удаляется элемент с ключмо “Третий”. А вот при выполнении команды order [“Пятый”] = 5 значение присваивается элементу с ключом “Пятый”, которого в словаре нет. В результате выполнения команды такой элемент в словаре появляется.

Словари

- Описанные выше способы создания словарей не являются исчерпывающими. Так, функции dict() в качестве аргумента можно передать уже существующий словарь — в этом случае создается его поверхностная копия. Словарь можно создавать и на основе списка. Элементами такого списка, который передается аргументом функции dict(), должны быть списки или кортежи, состоящие из двух элементов. Эти элементы определяют ключи и значения для элементов словаря.
- Еще один пример создания словарей представлен в листинге скриншота ниже:

Еще операции со словарями

```
120  age = dict([["Кот Кот", 5], ["Пёс Пёс", 10], ["Кабан Кабан", 15]])
121  for s in age.keys():
122      print(s+":", age[s])
123  for v in age.values():
124      print(v, end=" ")
125
```

```
Run:  main
C:\Users\Endliar\PycharmProjects\NovoCollege\venv\Scripts\python.exe C:/Use
Кот Кот: 5
Пёс Пёс: 10
Кабан Кабан: 15
5 10 15
```

Еще операции со словарями

```
125
126  color = dict([[(225, 0, 0), "Красный"], [(0, 225, 0), "Зелёный"], [(0, 0, 225), "Синий"]])
127  color[(225, 225, 0)] = "Жёлтый"
128  print(color)
129  print(color[(225, 0, 0)])
130
```

```
Run:  main
C:\Users\Endliar\PycharmProjects\NovoCollege\venv\Scripts\python.exe C:/Users/Endliar/PycharmProjec
{(225, 0, 0): 'Красный', (0, 225, 0): 'Зелёный', (0, 0, 225): 'Синий', (225, 225, 0): 'Жёлтый'}
Красный
```

Еще операции со словарями

- Первый словарь создаётся командой age = dict([["Кот Кот", 5], ["Пёс Пёс", 10], ["Кабан Кабан", 15]]).
- В этой команде мы передаём аргументом функции dict() список элементами которого являются списки ([["Кот Кот", 5], ["Пёс Пёс", 10], ["Кабан Кабан", 15].
- Первый элемент в каждом из этих списков определяет ключ элемента словаря, а второй элемент в списке определяет значение элемента словаря.

На заметку

- В данном случае словарь напоминает мини-базу данных, в которой ключ является именем персонажа, а значение элемента словаря определяет возраст персонажа.

На заметку

- Для каждого словаря с помощью метода keys() можно получить итерируемый объект, позволяющий определить значения ключей из словаря. Методом values() возвращается итерируемый объект, позволяющий получить значения элементов из словаря.

Подробности

- Итерируемый объект не является последовательностью в прямом смысле этого понятия. Но это объект, который, как и последовательность, можно перебирать. Поэтому во многих ситуациях итерируемый объект используется так, как если бы это была последовательность вроде списка.
- Что касается метода keys(), то результатом он возвращает итерируемый объект класса dict_keys. Метод values() результатом возвращает итерируемый объект класса dict_values. Классы и объекты обсуждаются в одной из следующих глав.

Операции с словарями

- В операторе цикла for переменная s принимает значения ключей из словаря age (для получения набора ключей использована инструкция age.keys()). Для каждого заданного значения s выполняется команда print(s+":", age[s]), которой отображается значение ключа и значение соответствующего элемента словаря.

Операции с словарями

- В следующем операторе цикла for перебираются значения элементов словаря. Переменная v принимает значения из последовательности ключей (вычисляется инструкцией age.values()). Команда print(v, end=" ") отображает значения элементов из словаря, при этом в качестве разделителя используются пробелы.

Операции с словарями

- Еще один словарь создаётся командой color = dict([[(225, 0, 0), "Красный"], [(0, 225, 0), "Зелёный"], [(0, 0, 225), "Синий"]]). В качестве аргумента функции dict() передан список, элементами которого являются списки [(225, 0, 0), "Красный"], [(0, 225, 0), "Зелёный"], [(0, 0, 225), "Синий"]. Примечательно здесь то, что ключами являются кортежи (225, 0, 0), "Красный"], [(0, 225, 0), "Зелёный"], [(0, 0, 225), "Синий"].
- Значения элементов словаря — текстовые значения с названиями цвета (соответственно “Красный”, “Зелёный” и “Синий”).

На заметку

- Мы воспользовались тем, что в формате RGB (сокращение от Red Green Blue) цвет задается с помощью трех целых чисел, каждое в диапазоне от 0 до 255 включительно.

На заметку

- При обращении к элементам словаря можно, как мы уже делали ранее, указывать ключ в квадратных скобках после названия словаря (например, как в командах color[(255,0,0)] и color[(255,255,0)]). Такой же подход использован в команде color[(255,255,0)]=“Жёлтый” которой в словарь добавляется новый элемент (поскольку указан ключ, который на момент выполнения команды в словаре не представлен).

На заметку

- Однако для получения значения элемента по ключу можно использовать метод get(), при вызове которого ключ передается в качестве аргумента (примером служит команда color.get((0,255,0))). Методу можно передать второй аргумент, который будет возвращаться в качестве результата, если ключа, указанного первым аргументом, в словаре нет. Например, результатом выражения color.get ((0, 0, 255), “Белый”) является значение “Синий” элемента с ключом (0, 0, 255).
- Но элемента с ключом (255, 255, 255) в словаре нет, поэтому результатом выражения color.get((255, 255, 255), “Белый”) является значение “Белый”.

Подробности

- Если при обращении к элементу словаря в квадратных скобках указать несуществующий ключ, это приведет к ошибке класса KeyError. Если доступ к элементу осуществляется с помощью метода get() и аргументом методу передается один аргумент, определяющий значение ключа, то результатом возвращается значение соответствующего элемента. Если указан несуществующий ключ, то результатом возвращается значение None, и ошибка при этом не возникает.

Подробности

- Как и в случае со списками, кортежами и множествами, для создания словарей можно использовать генераторы последовательностей. Небольшой пример того, как это можно было бы сделать, представлен в листинке кода ниже:

Еще пара мегаопераций

```python
days = ["Пн", "Вт", "Ср", "Чт", "Пт", "Сб", "Вс"]
week = {days[s]: s for s in range(len(days))}
myWeek = {d: days.index(d) for d in days}
print(week)
print(myWeek)
sqrs = {k: k ** 2 for k in range(1, 11) if k % 2 != 0}
print(sqrs)
```

```
main
C:\Users\Endliar\PycharmProjects\NovoCollege\venv\Scripts\python.ex
{'Пн': 0, 'Вт': 1, 'Ср': 2, 'Чт': 3, 'Пт': 4, 'Сб': 5, 'Вс': 6}
{'Пн': 0, 'Вт': 1, 'Ср': 2, 'Чт': 3, 'Пт': 4, 'Сб': 5, 'Вс': 6}
{1: 1, 3: 9, 5: 25, 7: 49, 9: 81}
```

Еще пара мегаопераций

- В данном случае создается три словаря, причем два из них одинаковые. Более конкретно, мы создаем список days с текстовыми значениями для ключей. Словарь week создается с использованием генератора последовательности: элемент словаря формируется инструкцией days[s]: s, где переменная s пробегает значения от 0 до величины len(days)-1 (количество элементов в списке days минус 1). Выражение days[s]: s означает, что при заданном значении s ключ элемента определяется элементом списка days[s], а значение элемента словаря при этом равно s.

Еще пара мегаопераций

- Аналогичный словарь myweek создается несколько иначе. Элемент словаря формируется выражением d: days.index(d), а переменная d принимает значения из списка days. Значение переменной задает ключ элемента словаря, а в качестве значения элемента словаря указано выражение days.index(d). Это индекс элемента со значением d в списке days. В итоге мы создаем два одинаковых словаря week и myweek, в чем и убеждаемся при проверке.

Еще пара мегаопераций

- Еще один словарь состоит из числовых значений: ключами элементов являются нечетные натуральные числа, а значением элементов является квадрат соответствующего числа. Словарь создается командой sqrs={k: k**2 for k in range(1,11) if k%2!=0}. Элемент словаря определяется выражением k: k**2, а переменная k пробегает значения из диапазона от 1 до 10 включительно, причем в словарь соответствующий элемент добавляется, только если истинно условие k%2!=0 (остаток от деления значения k на 2 не равен нулю — то есть число нечетное).

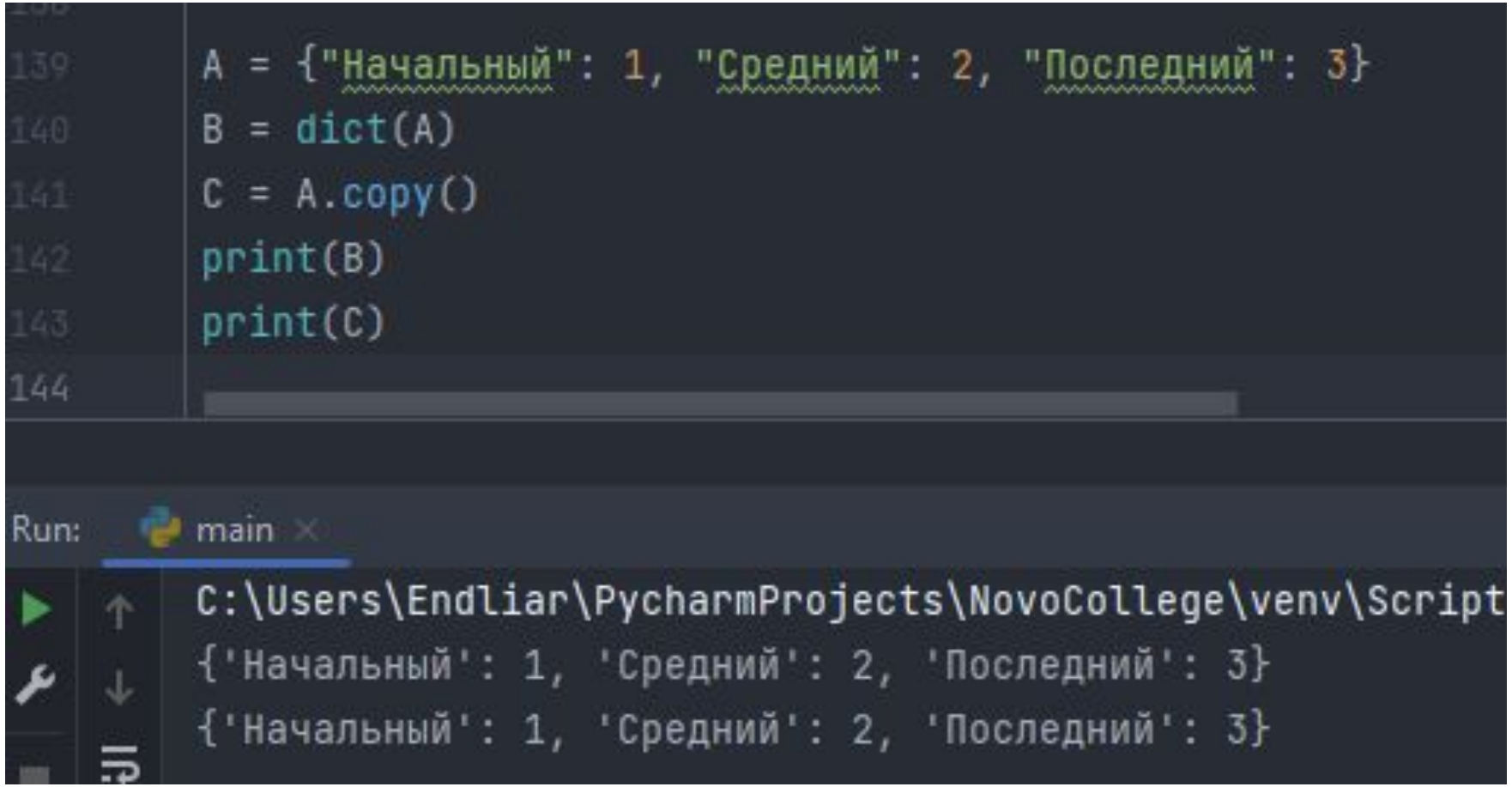
Операции со словарями

- В первую очередь остановимся на способах создания копии словаря. Причём эту задачу мы будем интерпретировать в широком смысле, когда на основе уже существующего словаря создаётся новый словарь (не обязательно совпадающий с исходным). Для этого существуют разные подходы, и некоторые из них реализованы в программе, представленной в листинге скриншота ниже:

Операции со словарями

- В первую очередь остановимся на способах создания копии словаря. Причём эту задачу мы будем интерпретировать в широком смысле, когда на основе уже существующего словаря создаётся новый словарь (не обязательно совпадающий с исходным). Для этого существуют разные подходы, и некоторые из них реализованы в программе, представленной в листинге скриншота ниже:

Операции со словарями



Операции со словарями

```
138
139  A = {"Начальный": 1, "Средний": 2, "Последний": 3}
140  B = dict(A)
141  C = A.copy()
142  D = {k: v * 10 for k, v in A.items()}
143  print("Созданы множества", sep="\n")
144  print("A =", A)
145  print("B =", B)
146  print("C =", C)
147  print("D =", D)
148
```

```
Run:  main
Созданы множества
A = {'Начальный': 1, 'Средний': 2, 'Последний': 3}
B = {'Начальный': 1, 'Средний': 2, 'Последний': 3}
C = {'Начальный': 1, 'Средний': 2, 'Последний': 3}
D = {'Начальный': 10, 'Средний': 20, 'Последний': 30}
```

Операции со словарями

```
A = {"Начальный": 1, "Средний": 2, "Последний": 3}
B = dict(A)
C = A.copy()
D = {k: v * 10 for k, v in A.items()}
print("Созданы множества", sep="\n")
print("A =", A)
print("B =", B)
print("C =", C)
print("D =", D)
for k in A:
    A[k] *= 100
print("После изменения оригинала:")
print("A =", A)
print("B =", B)
print("C =", C)
print("D =", D)
```

```
Run:  main
B = {'Начальный': 1, 'Средний': 2, 'Последний': 3}
C = {'Начальный': 1, 'Средний': 2, 'Последний': 3}
D = {'Начальный': 10, 'Средний': 20, 'Последний': 30}
После изменения оригинала:
A = {'Начальный': 100, 'Средний': 200, 'Последний': 300}
B = {'Начальный': 1, 'Средний': 2, 'Последний': 3}
C = {'Начальный': 1, 'Средний': 2, 'Последний': 3}
D = {'Начальный': 10, 'Средний': 20, 'Последний': 30}
```

Операции со словарями

- Это простая программа, в которой создаётся словарь A с тремя элементами (команда A = {“Начальный”:1, “Средний”:2, “Последний”:3}).
- Затем командами B=dict(A) и C=A.copy() на основе словаря A создаются словари-копии. Еще один словарь D создается на основе словаря A, но на этот раз речь не идет о копии. Словарь создается командой D={k: v*10 for k, v in A.items()}. Здесь мы используем генератор последовательности, причем в качестве перебираемого множества указано выражение A.items(). Метод items() возвращает итерируемый объект класса dict_items. В качестве «элементов» этого итерируемого объекта возвращаются кортежи, состоящие из двух элементов: ключ элемента словаря и значение элемента словаря.

Операции со словарями

- В операторе цикла указаны через запятую две переменные, k и v. Поэтому переменная k в качестве значения получает ключ элемента словаря, а переменная v принимает значение соответствующего элемента словаря.

Подробности

- Методы items(), keys() и values() результатом возвращают итерируемые объекты. Чтобы получить на основе этих итерируемых объектов списки, результат вызова данных методов следует передать аргументом функции list(). В результате получим список, который обрабатывается по правилам работы со списками. Аналогично можно воспользоваться функциями tuple() и set() для создания на основе итерируемого объекта соответственно кортежа и множества. Более того, если результат вызова метода items() передать в качестве аргумента функции dict(), то в результате получим копию словаря, из которого вызывался метод items().

Подробности

- На следующем этапе перебираются элементы словаря A. Причем мы использовали оператор цикла, в котором переменная k перебирает значения из словаря A. Важно то, что в этом случае переменная k принимает значения ключей из словаря A. Поэтому при выполнении команды A[k]*=100 значения элементов из словаря A умножаются на 100. Причем проверка показывает, что изменение содержимого словаря A не влияет на содержимое словарей B, C и D.

Подробности

- 203 страница – если что-то пойдёт не по плану.