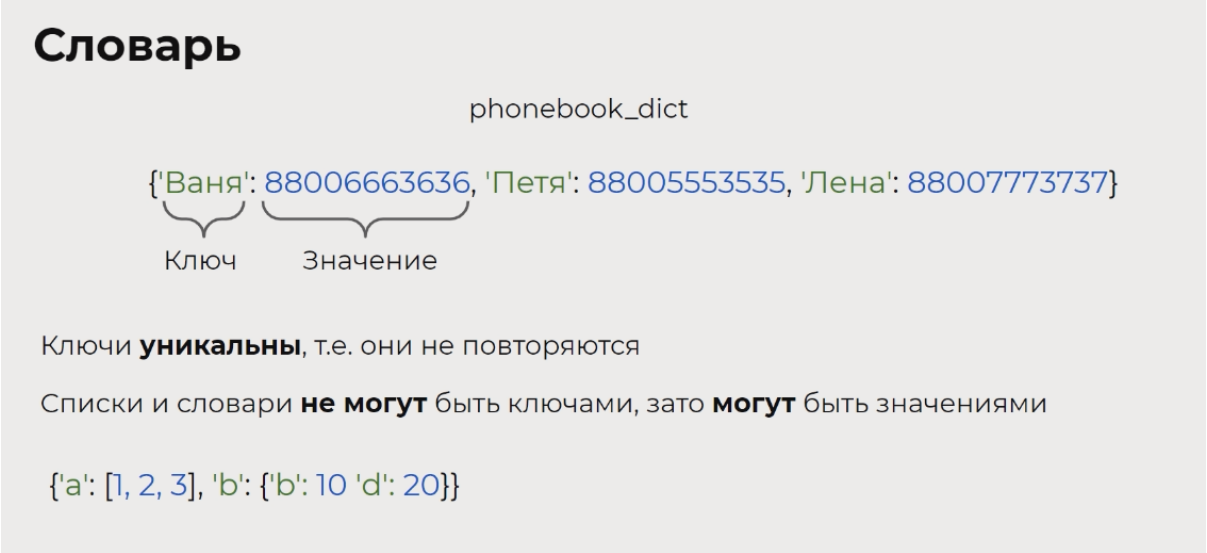
Самое важное

Словари



Словарь — это структура данных, в которой значения хранятся не под номерами в виде индексов, а под ключами, в качестве которых могут выступать уникальные неизменяемые объекты, примером которых могут быть строки.

Значениями могут быть и изменяемые типы данных, и неизменяемые, то есть мы можем создавать вложенные структуры, как в случае со вложенными списками.

Так как нет индексов, то следует обращаться к элементам по ключам: phoneboot_dict[‘Ваня’] → вернёт нам из словаря соответствующий номер 88006663636.

Создание новых элементов словаря: phoneboot_dict[‘Максим’] = 88008889898 — эта операция создаст внутри словаря пару ‘Максим’: 88008889898.

У словаря так же есть методы, которые расширяют наши возможности по работе с ним. Полный их перечень можно найти тут.

Примеры методов словарей

словарь.keys() — этот метод вернёт все ключи словаря
словарь.values() — этот метод вернёт все значения словаря

```
exam_book = {'Вася': 3,
             'Петя': 4,
             'Катя': 5}

print(exam_book.keys()) → dict_keys(['Вася', 'Петя', 'Катя'])
print(exam_book.values()) → dict_values([3, 4, 5])
```

Таким образом, можно отдельно обращаться как к ключам, так и к значениям. В основном эти методы нужны для прохождения в цикле по ключам/значениям.

update — позволит обновить наш словарь новыми данными.

Синтаксис
dict.update([other])
other — другой объект словаря или список пар (key, value) в виде кортежа.

Пример
exam_book.update({'Лера': 3})
print(exam_book)
{'Вася': 3, 'Петя': 4, 'Катя': 5, 'Лера': 3}

pop — позволяет удалить элемент из словаря по его ключу.

Синтаксис
dict.pop(key[, default])
key — ключ словаря.
default — значение по умолчанию.

Пример
print(exam_book.pop('Вася')) — удаляем элемент по ключу ‘Вася’.
print(exam_book)

3
{'Петя': 4, 'Катя': 5, 'Лера': 3} — проверяем, что его больше нет в словаре.

print(exam_book.pop('Игорь', 0)) — пробуем удалить элемент по ключу, которого не было в словаре. Без указания второго параметра мы получим ошибку, но, указав его, мы получим ответ 0, если ключ не будет найден:

print(exam_book)
0 — вот и наш ответ:
{'Петя': 4, 'Катя': 5, 'Лера': 3}.

get — этот метод позволяет получить из словаря элемент по ключу.

Синтаксис
dict.get(key[, default])
key — ключ словаря.
default — значение по умолчанию.

Пример
print(exam_book.get('Петя'))
4 — так мы получили значение из словаря, указав для поиска ключ «Петя».

print(exam_book.get('Игорь', 0)) — если же мы опять попробуем получить элемент по ключу, которого не было в словаре, то мы получим ошибку, если не будем использовать параметр default для установки значения по умолчанию.
Это значение вернётся, если в словаре не будет найден указанный ключ.

Вложенные словари
Как уже известно, в качестве значений могут быть не только числа, но и любые другие объекты. В том числе это могут быть списки или другие словари:

```
exam = {'Студенты 1-го курса': {'Вася': 3, 'Лера': 3},
        'Студенты 2-го курса': {'Петя': 4, 'Катя': 5}}
```

Внимание: вложенные элементы разделяются запятой, как и в случае с обычными цифрами. При этом, чтобы получить доступ к вложенным значениям, нам сперва нужно обратиться к вложенным словарям.

Пример задачи: попробуем узнать какую оценку получила Катя, студентка второго курса.

1. Обратимся к студентам 2-го курса:
   print(exam['Студенты 2-го курса'])
   {'Петя': 4, 'Катя': 5} — получим в качестве ответа словарь со студентами.
2. Обратимся к конкретному студенту, добавив к текущему словарю ещё один ключ:
   print(exam['Студенты 2-ого курса']['Катя'])
   5 — получим оценку Кати.

Множества
Множества — это ещё одна структура данных, главной особенностью которой является то, что она хранит только уникальные объекты, а все дубли в неё просто не записываются.

Создать множество можно при помощи функции set(). Сделать это можно и просто через фигурные скобки {1,2,3}, но таким способом лучше пренебречь, чтобы не путаться между словарём и множеством.

elements = [1, 2, 3, 4, 3, 2, 1] — допустим, у нас есть какая-то последовательность чисел с повторами. unique_elements = set(elements) — если мы создадим из неё множество print(unique_elements) и распечатаем его, то увидим: {1, 2, 3, 4} — все повторяющиеся элементы исчезли!

При помощи этой структуры данных мы также можем использовать некоторые интересные математические операции по работе с множествами:

```
a = {1, 2, 3}
b = {4, 8, 3}
```

print(a & b) → {3} — так мы получим множество с общими элементами двух множеств (теми, которые есть и в одном множестве, и в другом).
print(a | b) → {1, 2, 3, 4, 8} — так получим множество со всеми элементами двух множеств.
print(a - b) → {1, 2} — так получим элементы первого множества за вычетом элементов, которые повторяются во втором (в нашем случае повторяется только 3, это число и пропадёт).

Генерация словарей
Как и в случае с генерацией списков, мы можем сокращать подобные записи:

```
d = {}

for num in range(1, 11):
        d[num] = num ** 2

print(d)  → {1: 1, 2: 4, 3: 9, 4: 16, 5: 25, 6: 36, 7: 49, 8: 64, 9: 81, 10: 100}
```

до подобных:

```
d = {num: num**2 for num in range(1, 11)}

print(d)  → {1: 1, 2: 4, 3: 9, 4: 16, 5: 25, 6: 36, 7: 49, 8: 64, 9: 81, 10: 100}.
```

Как и со списками, можно выделить две части
{num: num**2 for num in range(1, 11)}

В зелёной мы описываем ключи и значения, которые будут содержаться в словаре.
В жёлтом мы задаем цикл, который будет определять количество элементов в словаре и может задавать некие изменения, связанные с переменной цикла.

Не допускай следующих ошибок!

Не забывайте, что ключами словарей не могут выступать изменяемые объекты!

```
d = {}
x = [1, 2, 3]
d[x] = 0
```

При попытке такой записи мы получим ошибку: TypeError: unhashable type: 'list'.

Помните, что, если мы обращаемся к словарю по ключу, которого не было в словаре, мы получим ошибку (как и в случае со списком), но при помощи метода get и параметра default мы можем избежать подобной ошибки, указав стандартное значение, которое нужно вернуть, если заданный ключ не будет обнаружен.