Словари

Словарь (dictionary) в языке Python хранит коллекцию элементов, где каждый элемент имеет уникальный ключ и ассоциированное с ним некоторое значение.

Определение словаря имеет следующий синтаксис:

```
dictionary = { ключ1:значение1, ключ2:значение2, ....}
```

В фигурных скобках через запятую определяется последовательность элементов, где для каждого элемента сначала указывается ключ и через двоеточие его значение.

Определим словарь:

```
users = {1: "Tom", 2: "Bob", 3: "Bill"}
```

В словаре users в качестве ключей используются числа, а в качестве значений - строки. То есть элемент с ключом 1 имеет значение "Tom", элемент с ключом 2 - значение "Bob" и т.д.

Другой пример:

```
emails = {"tom@gmail.com": "Tom", "bob@gmai.com": "Bob", "sam@gmail.com": "Sam"}
```

В словаре emails в качестве ключей используются строки - электронные адреса пользователей и в качестве значений тоже строки - имена пользователей.

Но необязательно ключи и строки должны быть однотипными. Они могу представлять разные типы:

```
objects = {1: "Tom", "2": True, 3: 100.6}
```

Мы можем также вообще определить пустой словарь без элементов:

```
objects = {}
```

или так:

```
objects = dict()
```

Преобразование списков и кортежей в словарь
Несмотря на то, что словарь и список - непохожие по структуре типы, но тем не менее существует возможности для отдельных видов списков преобразования их в словарь с помощью встроенной функции dict(). Для этого список должен хранить набор вложенных списков. Каждый вложенный список должен состоять из двух элементов - при конвертации в словарь первый элемент станет ключом, а второй - значением:

```
users_list = [
  ["+111123455", "Tom"],
  ["+384767557", "Bob"],
  ["+958758767", "Alice"]
]
users_dict = dict(users_list)
print(users_dict)      # {"+111123455": "Tom", "+384767557": "Bob", "+958758767": "Alice"}
```

Подобным образом можно преобразовать в словарь двухмерные кортежи, которые в свою очередь содержать кортежи из двух элементов:

```
users_tuple = (
   ("+111123455", "Tom"),
   ("+384767557", "Bob"),
   ("+958758767", "Alice")
)
users_dict = dict(users_tuple)
print(users_dict)
```

Получение и изменение элементов
Для обращения к элементам словаря после его названия в квадратных скобках указывается ключ элемента:

```
dictionary[ключ]
```

Например, получим и изменим элементы в словаре:

```
users = {
   "+11111111": "Tom",
   "+33333333": "Bob",
   "+55555555": "Alice"
}
# получаем элемент с ключом "+11111111"
print(users["+11111111"])      # Tom
# установка значения элемента с ключом "+33333333"
users["+33333333"] = "Bob Smith"
print(users["+33333333"])   # Bob Smith
```

Если при установки значения элемента с таким ключом в словаре не окажется, то произойдет его добавление:

```
users["+4444444"] = "Sam"
```

Но если мы попробуем получить значение с ключом, которого нет в словаре, то Python сгенерирует ошибку KeyError:

```
user = users["+4444444"]    # KeyError
```

И чтобы предупредить эту ситуацию перед обращением к элементу мы можем проверять наличие ключа в словаре с помощью выражения ключ in словарь. Если ключ имеется в словаре, то данное выражение возвращает True:

```
key = "+4444444"
if key in users:
    user = users[key]
    print(user)
else:
    print("Элемент не найден")
```

Также для получения элементов можно использовать метод get, который имеет две формы:

get(key): возвращает из словаря элемент с ключом key. Если элемента с таким ключом нет, то возвращает значение None

get(key, default): возвращает из словаря элемент с ключом key. Если элемента с таким ключом нет, то возвращает значение по умолчанию default

```
users = {
  "+11111111": "Tom",
  "+33333333": "Bob",
  "+55555555": "Alice"
}
user1 = users.get("+55555555")
print(user1)    # Alice
user2 = users.get("+33333333", "Unknown user")
print(user2)    # Bob
user3 = users.get("+44444444", "Unknown user")
print(user3)    # Unknown user
```

Удаление
Для удаления элемента по ключу применяется оператор del:

```
users = {
  "+11111111": "Tom",
  "+33333333": "Bob",
  "+55555555": "Alice"
}
```

```
del users["+55555555"]

print(users)    # { "+11111111": "Tom", "+33333333": "Bob"}
```

Но стоит учитывать, что если подобного ключа не окажется в словаре, то будет выброшено исключение KeyError. Поэтому опять же перед удалением желательно проверять наличие элемента с данным ключом.

```
users = {
   "+11111111": "Tom",
   "+33333333": "Bob",
   "+55555555": "Alice"
}
key = "+55555555"
if key in users:
   del users[key]
   print(f"Элемент с ключом {key} удален")
else:
   print("Элемент не найден")
```

Другой способ удаления представляет метод pop(). Он имеет две формы:

pop(key): удаляет элемент по ключу key и возвращает удаленный элемент. Если элемент с данным ключом отсутствует, то генерируется исключение KeyError

pop(key, default): удаляет элемент по ключу key и возвращает удаленный элемент. Если элемент с данным ключом отсутствует, то возвращается значение default

```
users = {
  "+11111111": "Tom",
  "+33333333": "Bob",
  "+55555555": "Alice"
}
key = "+55555555"
user = users.pop(key)
print(user)    # Alice
user = users.pop("+4444444", "Unknown user")
print(user)    # Unknown user
```

Если необходимо удалить все элементы, то в этом случае можно воспользоваться методом clear():

```
users.clear()
```

Копирование и объединение словарей
Метод copy() копирует содержимое словаря, возвращая новый словарь:

```
users = {"+1111111": "Tom", "+3333333": "Bob", "+5555555": "Alice"}
students = users.copy()
print(students)    # {"+1111111": "Tom", "+3333333": "Bob", "+5555555": "Alice"}
```

Метод update() объединяет два словаря:

```
users = {"+1111111": "Tom", "+3333333": "Bob"}
users2 = {"+2222222": "Sam", "+6666666": "Kate"}
users.update(users2)
```

```
print(users)    # {"+1111111": "Tom", "+3333333": "Bob", "+2222222": "Sam", "+6666666": "Kate"}
print(users2)   # {"+2222222": "Sam", "+6666666": "Kate"}
```

При этом словарь users2 остается без изменений. Изменяется только словарь users, в который добавляются элементы другого словаря. Но если необходимо, чтобы оба исходных словаря были без изменений, а результатом объединения был какой-то третий словарь, то можно предварительно скопировать один словарь в другой:

```
users3 = users.copy()
users3.update(users2)
```

Перебор словаря
Для перебора словаря можно воспользоваться циклом for:

```
users = {
  "+11111111": "Tom",
  "+33333333": "Bob",
  "+55555555": "Alice"
}
for key in users:
  print(f"Phone: {key}  User: {users[key]} ")
```

При переборе элементов мы получаем ключ текущего элемента и по нему можем получить сам элемент.

Другой способ перебора элементов представляет использование метода items():

```
users = {
  "+11111111": "Tom",
  "+33333333": "Bob",
  "+55555555": "Alice"
}
for key, value in users.items():
  print(f"Phone: {key}  User: {value} ")
```

Метод items() возвращает набор кортежей. Каждый кортеж содержит ключ и значение элемента, которые при переборе мы тут же можем получить в переменные key и value.

Также существуют отдельно возможности перебора ключей и перебора значений. Для перебора ключей мы можем вызвать у словаря метод keys():

```
for key in users.keys():
  print(key)
```

Правда, этот способ перебора не имеет смысла, так как и без вызова метода keys() мы можем перебрать ключи, как было показано выше.

Для перебора только значений мы можем вызвать у словаря метод values():

```
for value in users.values():
  print(value)
```

Комплексные словари

Кроме простейших объектов типа чисел и строк словари также могут хранить и более сложные объекты - те же списки, кортежи или другие словари:

```
users = {
  "Tom": {
     "phone": "+971478745",
     "email": "tom12@gmail.com"
  },
  "Bob": {
     "phone": "+876390444",
     "email": "bob@gmail.com",
     "skype": "bob123"
  }
}
```

В данном случае значение каждого элемента словаря в свою очередь представляет отдельный словарь.

Для обращения к элементам вложенного словаря соответственно необходимо использовать два ключа:

```
old_email = users["Tom"]["email"]
users["Tom"]["email"] = "supertom@gmail.com"
print(users["Tom"])     # { phone": "+971478745", "email": "supertom@gmail.com }
```

Но если мы попробуем получить значение по ключу, который отсутствует в словаре, Python сгенерирует исключение KeyError:

```
tom_skype = users["Tom"]["skype"]   # KeyError
```

Чтобы избежать ошибки, можно проверять наличие ключа в словаре:

```
key = "skype"
if key in users["Tom"]:
   print(users["Tom"]["skype"])
else:
   print("skype is not found")
```
