

Методические указания

Урок 7.2 Коллекции и их методы

Задачи урока:

- Изучить множества
- Изучить словари

0. Подготовка к уроку

До начала урока преподавателю необходимо:

- 1) Просмотреть, как ученики справились с домашним заданием
- 2) Прочитать методичку

1. Повторение пройденного материала

Учитель: Давайте вспомним методы списков и как с ними работать, а также чем кортеж отличается от списка. Задаёт вопросы по прошлому занятию ученикам.

2. Словари

Учитель: Пора познакомиться одним из важнейших типов данных - словари. Словари очень часто применяются в программировании, особенно когда наша программа работает через интернет. Если списки и кортежи достаточно простой тип данных, т.к как хранят значения через запятую, то синтаксис словарей немного другой.

```
a = {ключ: значение}
```

Словарь, как мы видим имеет более сложную структуру и состоит из ключа и значения. Обратиться как в списке или кортеже по индексу в данном случае мы не можем. Создается словарь с помощью фигурных скобок {}

```
a = {}  
print(a)  
print(type(a))
```

В данном примере создается пустой словарь, а команда `type()`, позволяет проверить нам тип данных, к которому относится переменная.

Для создания словаря с уже существующими данными достаточно указать ключ и значение

```
a = {'a': 1, 'b': 2}
```

Также создать словарь мы можем из существующих списков и кортежей с помощью функции `dict()`, которая преобразует в словарь.

```
a = dict([(1, 2), (4, 5)])  
print(a)
```

Для добавления или изменения значения в словаре мы должны указать ключ, для которого хотим указать значение. Если данного ключа нет, то он будет создан

```
a = {'a': 1}  
a['a'] = 4  
a['b'] = 2  
print(a)
```

Результат

```
{'a': 4, 'b': 2}
```

Рассмотрим основные методы словарей:

- **clear()** - очищает словарь.
- **copy()** - возвращает копию словаря.
- **get(key[, default])** - возвращает значение ключа, но если его нет, не бросает исключение, а возвращает default (по умолчанию None).
- **items()** - возвращает пары (ключ, значение).
- **keys()** - возвращает ключи в словаре.
- **pop(key[, default])** - удаляет ключ и возвращает значение. Если ключа нет, возвращает default (по умолчанию бросает исключение).
- **popitem()** - удаляет и возвращает пару (ключ, значение). Если словарь пуст, бросает исключение `KeyError`.
- **setdefault(key[, default])** - возвращает значение ключа, но если его нет, не бросает исключение, а создает ключ со значением default (по умолчанию None).
- **update([other])** - обновляет словарь, добавляя пары (ключ, значение) из other. Существующие ключи перезаписываются. Возвращает None (не новый словарь!).
- **values()** - возвращает значения в словаре.

1. clear

```
a = {2: 1}
print(a)
a.clear()
print(a)
```

Результат

```
{2:1}
{}
```

2. copy

```
a = {2: 1}
b = a.copy()    # создаем копию
c = a
print(b)
print(b is a)   # b не является a
print(c is a)   # c является a
```

Результат

```
{2: 1}
False
True
```

3. get

```
a = {2: 1}
print(a.get(2, 3))
print(a.get(1, 3))
```

Результат

```
1
3
```

4. items

```
a = {2: 1, 'a': 4}
print(a.items())
```

Результат

```
dict_items([(2, 1), ('a', 4)])
```

5. keys

```
a = {2: 1, 'a': 4}
print(a.keys())
```

Результат

```
dict_keys([2, 'a'])
```

6. pop

```
a = {2: 1, 'a': 4}
b = a.pop('a')
print(b)
print(a)
```

Результат

```
4
{2: 1}
```

7. popitem

```
a = {2: 1, 'a': 4}
b = a.popitem()
print(b)
print(a)
```

Результат

```
('a', 4)
{2: 1}
```

8. setdefault

```
a = {2: 1, 'a': 4}
a.setdefault(3)
print(a)
```

Результат

```
{2: 1, 'a': 4, 3: None}
```

9. update

```
a = {2: 1}
a.update({'a': 2})
print(a)
```

Результат

```
{2: 1, 'a': 2}
```

10. values

```
a = {2: 1, 'a': 2}
print(a.values())
```

Результат

```
dict_values([1, 2])
```

Словари также можно перебирать с помощью циклов. По умолчанию перебор идет по ключам, но мы можем это изменить с помощью методов `values`, `items`

```
a = {2: 1, 'a': 2}
for key in a:
    print(key)

for key, value in a.items():
    print(key, value)

for item in a.items():
    print(item)
```

Результат

```
2
a
2 1
a 2
(2, 1)
('a', 2)
```

3. Множества

Множества — это неупорядоченная коллекция уникальных элементов, сгруппированных под одним именем. Множество может быть неоднородным — включать элементы разных типов. Множество всегда состоит только из уникальных элементов (дубли запрещены) в отличие от списков и кортежей в Python.

Создать объект множества(`set`) в Python можно двумя путями:

```
a = {1, 2, 3}
a = set()
```

Нет ограничений на количество элементов в объекте `set`, но запрещено добавлять элементы изменяемых типов, такие как список или словарь. Если попробовать добавить список (с набором элементов), интерпретатор выдаст ошибку.

```
a = {1, 2, 3, [1, 2, 3]}  
print(a)
```

Добавить элемент в множество мы можем с помощью add

```
set1 = {1, 3, 4}  
set1.add(2)  
print(set1)
```

Добавить несколько элементов с помощью update

```
set2 = {1, 2, 3}  
set2.update([4, 5, 6])  
print(set2) # {1, 2, 3, 4, 5, 6}
```

Один или несколько элементов можно удалить из объекта set с помощью следующих методов. Их отличие в виде возвращаемого значения.

1. remove()
2. discard()
3. pop()

Метод remove() полезен в тех случаях, когда нужно удалить из множества конкретный элемент и вернуть ошибку в том случае, если его нет в объекте.

```
set1 = {1, 2, 3, 4, 'a', 'p'}  
set1.remove(2)  
print(set1)
```

Метод discard() полезен, потому что он удаляет конкретный элемент и не возвращает ошибку, если тот не был найден во множестве.

```
set1 = {1, 3, 4, 'a', 'p'}  
set1.discard('a')  
print(set1)
```

Метод pop() удаляет по одному элементу за раз в случайном порядке. Set — это неупорядоченная коллекция, поэтому pop() не требует аргументов (индексов в этом случае). Метод pop() можно воспринимать как неконтролируемый способ удаления элементов по одному из множеств в Python.

```
set1 = {1, 3, 4, 'p'}  
set1.pop()
```

Ну и самым часто используемым свойством множеств, которыми пользуются, является, то что множество не может хранить дубли

```
list1 = [1, 2, 1, 3]
list1 = set(list1)
list1 = list(list1)
print(list1)
```

3. Решение задач

Задача 1

Есть 3 словаря:

dict_a = {1:10, 2:20}

dict_b = {3:30, 4:40}

dict_c = {5:50, 6:60}

Напишите программу для слияния нескольких словарей в один.

Решение

```
dict_a = {1:10, 2:20}
dict_b = {3:30, 4:40}
dict_c = {5:50, 6:60}
result = {}
for d in (dict_a, dict_b, dict_c):
    result.update(d)
```

Дополнительно

Если на уроке остается время, то ученикам можно предложить начать прорешивать домашнее задание.

Домашняя работа

Задача 1

Если выписать все натуральные числа меньше 10, кратные 3 или 5, то получим 3, 5, 6 и 9. Сумма этих чисел равна 23.

Найдите сумму всех чисел меньше 1000, кратных 3 или 5.

Решение

```
sum = 0
for i in range(1000):
    if i % 3 == 0 or i % 5 == 0:
        sum += i
print(sum)
```