Петербургский государственный университет путей сообщения Императора Александра I

Введение в Python

Ведет: *аспирант 2 года* **Волков Егор Алексеевич**
gole00201@gmail.com
ауд. 11 - 304

Санкт-Петербург 2024

Оглавление

Введение в коллекции

- Коллекции — это структуры данных, которые позволяют хранить и манипулировать множеством элементов.
- В Python 3 есть несколько встроенных типов коллекций:
  - Список (`list`)
  - Кортеж (`tuple`)
  - Множество (`set`)
  - Словарь (`dict`)

Основные операции:

- Добавление элемента: `my_list.append(4)`
- Удаление элемента: `my_list.remove(2)`
- Доступ по индексу: `my_list[0]`

Кортежи (tuple)

Определение: неизменяемая последовательность объектов любого типа.

Объявление:

```
my_tuple = (1, 2, 3, "текст")
```

Особенности:

- Неизменяемость
- Быстрее списков при доступе к элементам
- Используются для защиты данных от изменений

Множества (set)

Определение: неупорядоченная коллекция уникальных элементов.
Объявление:

```
my_set = {1, 2, 3, 4}
```

Основные операции:

- Добавление: my_set.add(5)
- Удаление: my_set.discard(3)
- Операции над множествами: объединение, пересечение, разность

Множество можно создать несколькими способами:

С помощью фигурных скобок {}:

```
my_set = {1, 2, 3, 4}
```

Используя функцию set():

```
my_set = set([1, 2, 3, 4])
```

Если передать в `set()` список с повторяющимися элементами, они будут автоматически удалены, так как множество хранит только уникальные элементы:

```
my_set = set([1, 2, 2, 3, 4])
print(my_set)  # {1, 2, 3, 4}
```

Основные операции над множествами

1. Добавление и удаление элементов

Добавление элемента:

```
my_set.add(5)
print(my_set)  # {1, 2, 3, 4, 5}
```

Удаление элемента:

- `remove()` удаляет элемент, но вызывает ошибку, если элемента нет в множестве:

```
my_set.remove(3)
```

- `discard()` удаляет элемент, если он существует, и ничего не делает, если элемента нет:

```
my_set.discard(3)
```

2. Объединение множеств

Объединение объединяет все элементы двух или более множеств. Если элементы дублируются, они включаются только один раз, так как множество хранит только уникальные значения.

Использование оператора | или метода union():

python

```
set1 = {1, 2, 3}
set2 = {3, 4, 5}
result = set1 | set2
print(result)  # {1, 2, 3, 4, 5}
```

3. Пересечение множеств

Пересечение возвращает только те элементы, которые присутствуют в обоих множествах.

Использование оператора & или метода intersection():

```
python

set1 = {1, 2, 3}
set2 = {2, 3, 4}
result = set1 & set2
print(result)  # {2, 3}
```

4. Разность множеств

Разность возвращает элементы, которые находятся в первом множестве, но отсутствуют во втором.

Использование оператора - или метода difference():

```
set1 = {1, 2, 3, 4}
set2 = {3, 4, 5}
result = set1 - set2
print(result)  # {1, 2}
```

5. Симметрическая разность множеств

Симметрическая разность возвращает элементы, которые присутствуют в одном из множеств, но не одновременно в обоих.

Использование оператора ^ или метода symmetric_difference():

```
set1 = {1, 2, 3}
set2 = {3, 4, 5}
result = set1 ^ set2
print(result)  # {1, 2, 4, 5}
```

Словари (dict)

Определение: неупорядоченная коллекция пар "ключ-значение".

Объявление:

```
my_dict = {"ключ1": "значение1", "ключ2": "значение2"}
```

Основные операции:

- Добавление/изменение: `my_dict["ключ3"] = "значение3"`
- Удаление: `del my_dict["ключ1"]`
- Доступ по ключу: `my_dict["ключ2"]`

Представление в памяти

Задача 1: Уникальные элементы

Условие: Дана строка, содержащая несколько слов. Найдите все уникальные слова в строке и выведите их в алфавитном порядке.

```
sentence = "apple banana apple orange banana kiwi"
# Вывод: ['apple', 'banana', 'kiwi', 'orange']
```

Задача 2: Подсчёт частоты элементов

Условие: Дана строка, состоящая из символов. Необходимо подсчитать частоту каждого символа в строке и вывести её.

```
string = "abracadabra"
# Вывод: {'a': 5, 'b': 2, 'r': 2, 'c': 1, 'd': 1}
```

Задача 3: Пересечение списков

Условие: Даны два списка целых чисел. Найдите их пересечение — элементы, которые присутствуют в обоих списках, и выведите их в порядке возрастания.

```
list1 = [1, 2, 3, 4, 5]
list2 = [4, 5, 6, 7, 8]
# Вывод: [4, 5]
```

Задача 4: Словарь квадратов

Условие: Дано целое число n. Создайте словарь, где ключами будут числа от 1 до n, а значениями — их квадраты.

```
n = 5
# Вывод: {1: 1, 2: 4, 3: 9, 4: 16, 5: 25}
```

Задача 5: Проверка на анаграмму

Условие: Даны две строки. Необходимо определить, являются ли они анаграммами (содержат одинаковые символы с одинаковой частотой).

```
string1 = "listen"
string2 = "silent"
# Вывод: True
```