Кортежи, словари и множества в Python.

Словари

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

Формат записи:
dictionary = { ключ1:значение1, ключ2:значение2, ...}

Пример:

Ключ целый тип данных
```
users = {1: "Tom", 2: "Bob", 3: "Bill"}
```

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

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

Пустой словарь:
```
objects = {}
objects = dict()
```

2. Преобразование списков и кортежей в словарь

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

3. Получение и изменение элементов

Формат записи:
dictionary[ключ]

Пример:

```
print(users[1])      # Tom
print(users["tom@gmail.com"])      # Tom

users[" tom@gmail.com "] = "Sam"
```

Словари (Методы и функции)

4. Получение данных из словаря

Выражение ключ in словарь.

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

Метод get

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

```
user1 = users.get("+55555555")
```

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

```
user2 = users.get("+33333333", "Unknown user")
```

5. Удаление значений из словаря

Оператор del

```
del users["+55555555"]
```

Оператор pop

pop(key): удаляет элемент по ключу key и возвращает удаленный элемент.

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

```
user = users.pop(key)
```

```
user = users.pop("+4444444", "Unknown user")
```

Метод clear

```
users.clear()
```

Словари (Методы и функции)

6. Копирование и объединение словарей

Метод copy() копирует содержимое словаря

```
users = {"+1111111": "Tom", "+3333333": "Bob”}
students = users.copy()
```

Метод update() объединяет два словаря

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

7. Перебор словаря

```
for key in users:
```

метод keys(), values()

```
for key in users.keys():
for value in users.values():
```

Комплексные словари

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

Указываем два ключа

```
old_email = users["Tom"]["email"]
users["Tom"]["email"] = "supertom@gmail.com"
```

Примеры

Пример 1: Перебор Элементов

```
# Создание словаря
person = {'name': 'John', 'age': 30, 'city': 'New York'}

# Перебор ключей
for key in person:
    print(key)

# Перебор значений
for value in person.values():
    print(value)

# Перебор ключей и значений
for key, value in person.items():
    print(f'{key}: {value}')
```

Пример 2: Добавление и Изменение Элементов

```
# Создание словаря
person = {'name': 'John', 'age': 30, 'city': 'New York'}

# Добавление нового элемента
person['email'] = 'john@example.com'

# Изменение существующего элемента
person['age'] = 31

print(person)
# Выведет: {'name': 'John', 'age': 31, 'city': 'New York', 'email': 'john@example.com'}
```

Пример 3: Проверка Наличия Ключа

```
person = {'name': 'John', 'age': 30, 'city': 'New York'}

# Проверка наличия ключа
if 'name' in person:
    print('Ключ "name" присутствует')
```

Множества

Множество (set) представляют еще один вид набора, который хранит только уникальные элементы.

Формат записи:
Set = {значение1, значение2, ...}

users = {"Tom", "Bob", "Alice", "Tom"}

Для определения множества может применяться функция **set()**

users = set()

Длина множества:

Len(set)

2. Добавление элементов, метод add()

users = set()
users.add("Sam")

3. Удаление элементов

Метод remove() – возвращает ошибку, если элемента нет

```
users = {"Tom", "Bob", "Alice"}
if "Tom”  in users:
        users.remove(user)
```

Метод discard() – удаление без возврата ошибки

```
users = {"Tom", "Bob", "Alice"}
users.discard("Tim")
```

метод clear() – отчистка всего множества

users.clear()

Множества

4. Перебор множества

```
users = {"Tom", "Bob", "Alice"}
for user in users:
```

5. Копирование множества, метод copy()

```
users = {"Tom", "Bob", "Alice"}
students = users.copy()
```

6. Объединение множеств метод union()

```
users = {"Tom", "Bob", "Alice"}
users2 = {"Sam", "Kate", "Bob"}

users3 = users.union(users2)
```

7. Пересечение множеств метод intersection()

```
users = {"Tom", "Bob", "Alice"}
users2 = {"Sam", "Kate", "Bob"}
users3 = users.intersection(users2) # # {"Bob"}
```

С помощью логического И

```
users = {"Tom", "Bob", "Alice"}
users2 = {"Sam", "Kate", "Bob"}
print(users & users2)   # {"Bob"}
```

Обновленный метод **intersection_update()**

```
users = {"Tom", "Bob", "Alice"}
users2 = {"Sam", "Kate", "Bob"}
users.intersection_update(users2)
```

Множества

8. Разность множеств

метод difference

```
users = {"Tom", "Bob", "Alice"}
users2 = {"Sam", "Kate", "Bob"}
users3 = users.difference(users2)
```

метод symmetric_difference()

```
users = {"Tom", "Bob", "Alice"}
users2 = {"Sam", "Kate", "Bob"}
users3 = users.symmetric_difference(users2) # {"Tom", "Alice", "Sam", "Kate"}
```

9. Отношения между множествами

Метод issubset

```
users = {"Tom", "Bob", "Alice"}
superusers = {"Sam", "Tom", "Bob", "Alice", "Greg"}

print(users.issubset(superusers))   # True
print(superusers.issubset(users))   # False
```

Метод issuperset

```
users = {"Tom", "Bob", "Alice"}
superusers = {"Sam", "Tom", "Bob", "Alice", "Greg"}

print(users.issuperset(superusers))   # False
print(superusers.issuperset(users))   # True
```

Примеры

1. Удаление дубликатов из списка сохраняя порядок:

```
def duplicate_order(lst):
    seen = set()
    result = []
    for item in lst:
        if item not in seen:
            result.append(item)
            seen.add(item)
    return result

my_list = [1, 2, 3, 4, 2, 3, 5, 6, 1]
s = duplicate_order(my_list)
```

9. Нахождение минимального и максимального элемента

```
my_set = {10, 20, 5, 40, 30}

min_element = min(my_set)
max_element = max(my_set)

print(f"Минимальный элемент: {min_element}")
print(f"Максимальный элемент: {max_element}")
```

Самостоятельная работа

Словари:

1. Составить программу, выводящую на экран меню детского кафе (наименование изделия, вес, стоимость).
2. Составить программу, выводящую на экран студенческую ведомость (Ф. И. О., оценки за три экзамена, средний балл).
3. Составить программу, выводящую на экран расписание движения поездов (станция отправления, станция прибытия, время прибытия, время в пути).
4. Составить программу, выводящую на экран меню ресторана "Дракон" (наименование изделия, вес, стоимость).
5. Составить программу, выводящую на экран анкетные данные учеников (Ф. И. О., год рождения, адрес, сведения о родителях).

Множества:

1. Уникальные элементы:

Напишите программу, которая принимает на вход список и выводит на экран все уникальные элементы этого списка.
Вход: [1, 2, 3, 2, 4, 3, 5]
Выход: {1, 2, 3, 4, 5}

2. Пересечение множеств:

Напишите программу, которая находит пересечение двух множеств и выводит результат на экран.
Вход: set([1, 2, 3, 4]), set([3, 4, 5, 6])
Выход: {3, 4}

3. Объединение множеств:

Напишите программу, которая принимает два множества и выводит на экран их объединение.

Вход: set([1, 2, 3]), set([3, 4, 5])
Выход: {1, 2, 3, 4, 5}