ПМ3 Разработка модулей ПО.

РО 3.1 Понимать и применять принципы объектно-ориентированного и асинхронного программирования.

Тема 10. Коллекции данных. Словари. Методы Словарей.

Цель занятия:

Сформировать представление о словарях в Python как о структуре данных для хранения пар «ключ–значение», научиться создавать словари, работать с их элементами и использовать основные методы словарей.

Учебные вопросы:

1. Понятие словаря в Python
2. Создание словарей.
3. Доступ к элементам словаря.
4. Изменение словаря.
5. Основные методы словарей.

1. Понятие словаря в Python

Словарь (dict) — это изменяемая коллекция данных, предназначенная для хранения информации в виде пар «ключ – значение».

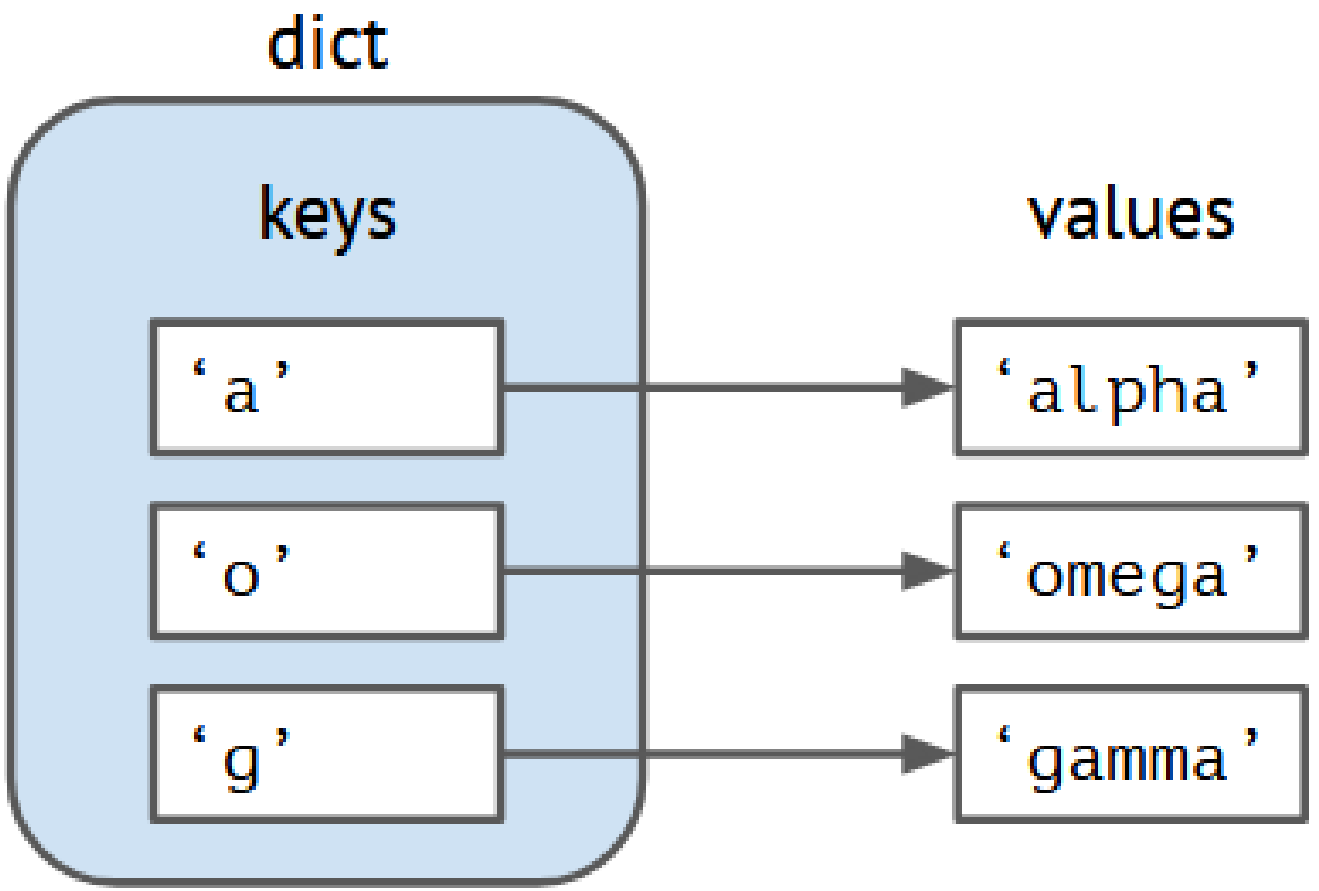
Каждому ключу соответствует определённое значение, по которому осуществляется быстрый доступ к данным.

Пример словаря:

```
student = {
    "name": "Kirill",
    "age": 18,
    "group": "11-TIS"
}
```

Элементы в словарях хранятся в формате ***key:value***.

```
dict = {'a': 'alpha', 'o': 'omega', 'g': 'gamma'}
```



Основные свойства словаря

1. Хранение данных по принципу «ключ – значение»

Ключ используется для доступа к значению.

Значение может быть любого типа данных.

```
print(student["name"])  # Kirill
```

2. Уникальность ключей

В словаре каждый ключ уникален.

При повторном использовании ключа его значение будет перезаписано.

3. Изменяемость (mutable)

Словарь можно изменять после создания:

- добавлять новые элементы;
- изменять значения;
- удалять элементы.

```
student = {
    "name": "Kirill",
    "age": 18,
    "group": "11-TIS"
}

student["age"] = 19
student["email"] = "kirill@mail.com"
del student["group"]
print(student)
# {'name': 'Kirill', 'age': 19, 'email': 'kirill@mail.com'}
```

4. Типы данных ключей

Ключами словаря могут быть только неизменяемые (immutable) типы данных:

- числа (int, float);
- строки (str);
- кортежи (tuple).

Нельзя использовать списки и множества в качестве ключей.

```
d = {
    1: "one",
    "two": 2,
    (3, 4): "tuple key"
}
```

5. Упорядоченность словарей

В современных версиях Python словари сохраняют порядок добавления элементов.

Логически словарь рассматривается как структура с доступом по ключу, а не по индексу.

Области применения словарей

Словари применяются для:

- хранения и передачи структурированных данных;
- подсчёта и группировки данных.
- Web-разработка
- работы с API;

JSON (JavaScript Object Notation) — это текстовый формат обмена данными, широко используемый в веб-разработке для передачи информации между клиентом и сервером.

Пример JSON:

```
{
   "name": "Ivan",
   "age": 18,
   "is_student": true
}
```

Структура JSON практически полностью соответствует словарю Python:

JSON	Python
объект	dict
ключ	ключ словаря
значение	значение словаря
массив	list

Пример словаря Python:

```
data = {
    "name": "Ivan",
    "age": 18,
    "is_student": True
}
```

Использование словарей при работе с веб-данными:

- Данные, получаемые с веб-сервера (API), часто приходят в формате JSON.
- В Python такие данные автоматически преобразуются в словари и списки.
- Работа с веб-данными фактически сводится к работе со словарями.

2. Создание словарей.

Создание словаря с помощью литерала {}

Самый распространённый способ создания словаря — использование фигурных скобок {} с перечислением пар «ключ: значение».

Ключи отделяются от значений двоеточием :.

Пары «ключ–значение» разделяются запятыми.

```
student = {
    "name": "Kirill",
    "age": 18,
    "group": "11-TIS"
}
```

Создание пустого словаря

```
d = {}
```

Важно: {} без элементов — это пустой словарь, а не множество.

Создание словаря с помощью функции dict()

```
d = dict(name="Anna", age=20)
```

Ограничение: ключи должны быть строками без пробелов.

Создание словаря из других коллекций

Из списка или кортежа пар:

```
pairs = [("a", 1), ("b", 2)]
d = dict(pairs)
```

С помощью zip():

```
keys = ["name", "age", "city"]
values = ["Ivan", 18, "Moscow"]

d = dict(zip(keys, values))
```

3. Доступ к элементам словаря.

Доступ по ключу

Основной способ доступа к значению — обращение по ключу.

```
student = {"name": "Ivan", "age": 18}
print(student["name"])  # Ivan
```

⚠ Если ключ отсутствует, возникнет ошибка KeyError.

Проверка наличия ключа

Перед обращением к ключу рекомендуется проверить его наличие:

```
if "age" in student:
    print(student["age"])
```

Метод get()

Метод get() возвращает значение по ключу и не вызывает ошибку, если ключ отсутствует.

```
print(student.get("age"))        # 18
print(student.get("grade"))      # None
print(student.get("grade", 0))   # 0 — значение по умолчанию
```

Преимущества get():

- безопасный доступ к данным;
- удобно использовать при работе с внешними источниками (JSON, файлы, API).

Добавление и изменение элементов

- Если ключ уже существует — значение изменяется
- Если ключа нет — он добавляется.

```
student = {"name": "Ivan", "age": 18}
student["age"] = 19
student["city"] = "Moscow"
print(student)
# {'name': 'Ivan', 'age': 19, 'city': 'Moscow'}
```

Вывод:

- Словари можно создавать разными способами: через {}, dict(), из других коллекций.
- Доступ к элементам осуществляется по ключу.
- Метод get() является безопасным способом доступа к значениям.
- Добавление и изменение элементов выполняется одинаковым образом.

4. Изменение словаря

Добавление элементов в словарь

Чтобы добавить новый элемент, достаточно указать новый ключ и присвоить ему значение.

```
student = {
    "name": "Ivan",
    "age": 18
}
student["group"] = "11-TIS"
```

Если ключа не было — элемент добавляется.

Изменение значений элементов

Если указанный ключ уже существует, его значение будет перезаписано.

```
student = {
    "name": "Ivan",
    "age": 18
}
student["group"] = "11-TIS"
student["age"] = 19

print(student)
# {'name': 'Ivan', 'age': 19, 'group': '11-TIS'}
```

Удаление элементов из словаря.

Оператор del

Удаляет элемент по ключу.

```
del student["age"]
```

Если ключ отсутствует — возникнет ошибка KeyError.

Метод pop()

Удаляет элемент по ключу и возвращает его значение.

```
age = student.pop("age")
print(age)
```

Можно указать значение по умолчанию:

```
student = {}
age = student.pop("grade", 0)
print(age) # 0
```

Метод popitem()

Удаляет и возвращает последнюю добавленную пару (ключ, значение).

```
key, value = student.popitem()
```

Часто используется при поочерёдной обработке элементов словаря.

Метод clear()

Удаляет все элементы словаря, делая его пустым.

```
student.clear()
```

Метод update()

Добавляет элементы из другого словаря или набора пар.

- Новые ключи добавляются.
- Существующие ключи обновляются.

```
student.update({"age": 20, "city": "Moscow"})
```

Безопасное добавление: setdefault()

Метод setdefault() возвращает значение по ключу, если он есть;

если ключа нет — создаёт его с указанным значением.

```
student = {
    "name": "Ivan",
    "age": 18
}

email = student["email"] # KeyError: 'email'
email = student.setdefault("email", "not defined")
print(email) # not defined
```

Сортировка словаря

Сортировка по ключам:

```
my_dict = {'banana': 3, 'apple': 4, 'pear': 1, 'orange': 2}

# Сортировка по ключам
sorted_dict = dict(sorted(my_dict.items()))
print(sorted_dict)
# Вывод: {'apple': 4, 'banana': 3, 'orange': 2, 'pear': 1}
```

Сортировка по значениям:

```
my_dict = {'banana': 3, 'apple': 4, 'pear': 1, 'orange': 2}

# sorted() с key параметром
sorted_dict = dict(sorted(my_dict.items(), key=lambda x: x[1]))
print(sorted_dict)
# Вывод: {'pear': 1, 'orange': 2, 'banana': 3, 'apple': 4}
```

Вывод:

- Словари можно свободно изменять после создания.
- Добавление и изменение элементов выполняются одинаковым способом.
- Для удаления используются del, pop(), popitem() и clear().
- Методы update() и setdefault() упрощают массовую и безопасную работу со словарями.

5. Основные методы словарей

Словари Python содержат встроенные методы, которые позволяют получать доступ к ключам и значениям, изменять словарь и удобно обрабатывать данные.

Метод	Значение
dict()	создание словаря
len()	возвращает число пар
clear()	удаляет все значения из словаря
copy()	создает псевдокопию словаря
deepcopy()	создает полную копию словаря
fromkeys()	создание словаря
get()	получить значение по ключу
has_key()	проверка значения по ключу
items()	возвращает список значений
iteriyems()	возвращает итератор
keys()	возвращает список ключей
iterkeys()	возвращает итератор ключей
pop()	извлекает значение по ключу

Метод keys()

keys() — ключи словаря. Возвращает объект-представление всех ключей словаря.

```
student = {
    "name": "Ivan",
    "age": 18
}

student_keys = student.keys()
print(student_keys) # dict_keys(['name', 'age'])

student_keys_ls = list(student.keys())
print(student_keys_ls) # ['name', 'age']

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

values() — значения словаря

Возвращает объект-представление всех значений словаря.

```
student = {
    "name": "Ivan",
    "age": 18
}

student_val = student.values()
print(student_val) # dict_values(['Ivan', 18])

student_val_ls = list(student.values())
print(student_val_ls) # ['Ivan', 18]

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

items() — пары «ключ – значение»

Возвращает объект-представление пар (ключ, значение).

```
student_items = student.items()
print(student_items) # dict_items([('name', 'Ivan'), ('age', 18)])

student_items_ls = list(student.items())
print(student_items_ls) # [('name', 'Ivan'), ('age', 18)]

for key, value in student.items():
    print(key, value)
```

Эти методы возвращают не списки, а специальные представления (views), которые автоматически обновляются при изменении словаря.

Метод fromkeys()

Метод **fromkeys()** используется для создания нового словаря на основе заданной последовательности ключей и одного общего значения для всех ключей.

```
keys = ["math", "physics", "english"]
scores = dict.fromkeys(keys, 0)

print(scores)
# {'math': 0, 'physics': 0, 'english': 0}
```

Если второй аргумент не указан, всем ключам присваивается **None**.

Вывод:

- Методы словарей позволяют эффективно получать доступ к данным и управлять ими.
- keys(), values(), items() используются при переборе словаря.
- get() и setdefault() обеспечивают безопасную работу с ключами.
- update() упрощает массовое изменение словаря.

Контрольные вопросы:

- Что такое словарь в Python?
- Какие типы данных могут быть ключами словаря?
- Как создать словарь разными способами?
- Чем отличается обращение по ключу от метода get()?
- Как добавить и удалить элемент словаря?
- Для чего используются методы keys(), values(), items()?
- Как правильно перебирать словарь в цикле for?
- Какие ошибки чаще всего возникают при работе со словарями?

Домашнее задание:

https://ru.hexlet.io/courses/js-asynchronous-programming

Материалы лекций:

https://github.com/ShViktor72/Education

Обратная связь:

colledge20education23@gmail.com