Передовые инженерные школы

МИНОБРНАУКИ РОССИИ

УНИВЕРСИТЕТ ИННОПОЛИС

онлайн университет

Программная инженерия. Разработка ПО (Python для продвинутых специалистов)

Модуль: Введение в Python для анализа: извлечение данных, подготовка и визуализация.

Лекция 2: Структуры данных в Python

Дата: 27.03.2025

ПЕРЕДОВАЯ
ИНЖЕНЕРНАЯ ШКОЛА
УНИВЕРСИТЕТА ИННОПОЛИС

Содержание лекции

ПЕРЕДОВАЯ
ИНЖЕНЕРНАЯ ШКОЛА
УНИВЕРСИТЕТА ИННОПОЛИС

- Доступные в Python структуры данных
- Структуры данных - производительность
- Практическая часть

Что такое структура данных?

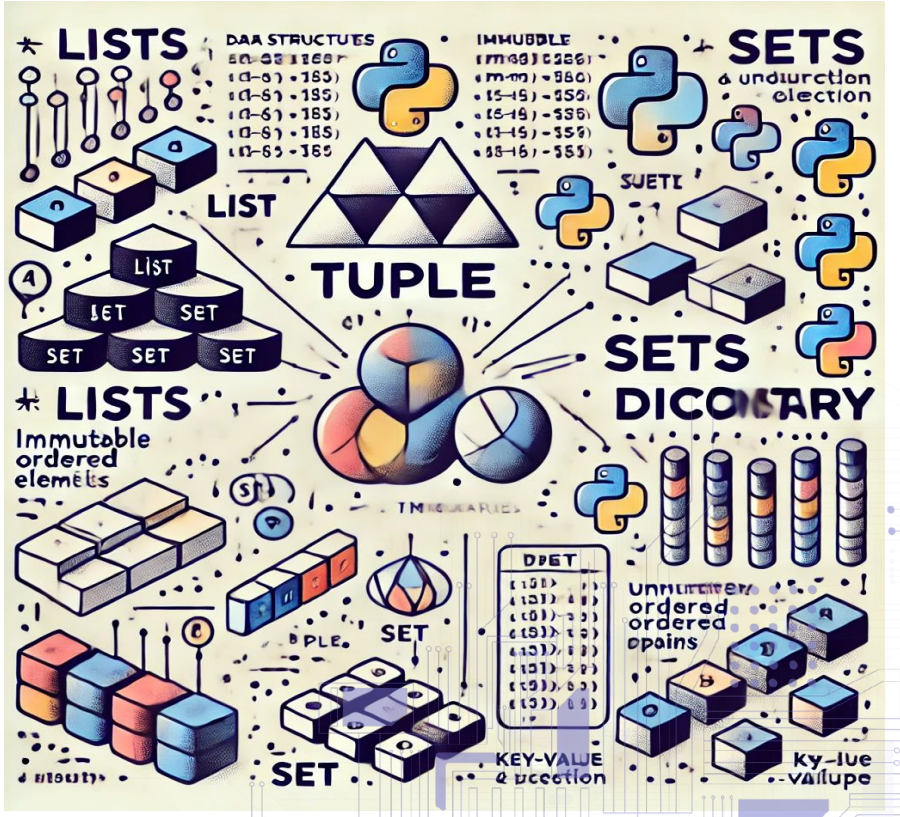
ПЕРЕДОВАЯ ИНЖЕНЕРНАЯ ШКОЛА
УНИВЕРСИТЕТА ИННОПОЛИС

Структура данных — это способ организации и хранения данных для их эффективного использования.

Разные функции лучше работают с разными структурами данных, и важно применять алгоритмы, которые будут работать быстрее всего на вашей структуре данных.

В Python доступны:

- Список (List)
- Кортеж (Tuple)
- Множества (Set/Frosenset)
- Словари (Dictionaries)

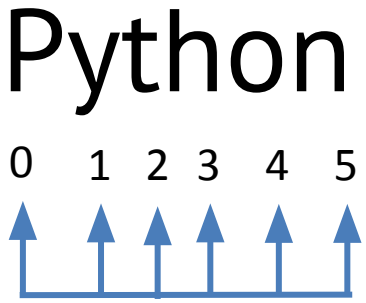


Строка (Str)

ПЕРЕДОВАЯ ИНЖЕНЕРНАЯ ШКОЛА УНИВЕРСИТЕТА ИННОПОЛИС

Строка в Python – неизменяемая последовательность символов

```
# Слайсинг – конкатенация наоборот
a = 'Python'
a[0:3] # 'Pyt'
a[1:] # 'ython'
a[:2] # 'Py'
a[1:-2] # 'yth'
a[2::3] # 'tn'
# Некоторые строковые методы
s = "simpleStrinG"
len(s) # Длина строки
s.find('i') # Индекс первого элемента i
s.rfind('i') # Индекс первого элемента i с конца строки
s.replace('i', 'TT', 1) # Поменяет все элементы t на TT
s.upper() # Переведет строку в верхний регистр
s.lower() # Переведет строку в нижний регистр
s.islower)) # Проверка, если все буквы в нижнем регистре
s.isupper() # Проверка, если все буквы в верхнем регистре
s.istitle() # Проверка, если первая буква заглавная
s.isalpha() # Проверка, если строка состоит только из букв
s.isdigit() # Проверка, если строка состоит только из цифр
s.isalnum() # Проверка, если строка состоит только из цифр или букв
```



адрес каждой буквы = индекс

Список (List)

Список — это изменяемая упорядоченная коллекция элементов.

```
my_list = [1, 2, 3, 4, 5]
empty_list = []

my_list.append(6)  # Добавление в конец списка
my_list.insert(0, 0)  # Вставка на указанную позицию

my_list.remove(3)  # Удаление первого вхождения элемента
popped_element = my_list.pop()  # Удаление и возврат последнего элемента

first_element = my_list[0]  # Первый элемент
sublist = my_list[1:4]  # Срез списка
```

Кортеж (Tuple)

Кортеж — это неизменяемая упорядоченная коллекция элементов. Кортежи неизменяемы, поэтому операции добавления и удаления элементов недоступны.

Кортежи сохраняют порядок элементов, в котором они были добавлены. Это значит, что элементы кортежа можно индексировать и извлекать в том порядке, в котором они были добавлены, что делает их упорядоченными.

```
my_tuple = (1, 2, 3)
single_element_tuple = (1,)
empty_tuple = ()


first_element = my_tuple[0]
subtuple = my_tuple[1:3]
```

Множества (Set/Frosenset)

ПЕРЕДОВАЯ ИНЖЕНЕРНАЯ ШКОЛА УНИВЕРСИТЕТА ИННОПОЛИС

Множество — это неупорядоченная коллекция уникальных элементов.

```
my_set = {1, 2, 3, 4, 5}
empty_set = set()

#Добавление элемента
my_set.add(6)

#Удаление элемента
my_set.remove(3)
popped_element = my_set.pop()

#Операции над множествами
another_set = {3, 4, 5, 6, 7}
union_set = my_set.union(another_set)
intersection_set = my_set.intersection(another_set)
```

Словари (Dictionaries)

ПЕРЕДОВАЯ ИНЖЕНЕРНАЯ ШКОЛА
УНИВЕРСИТЕТА ИННОПОЛИС

Словарь — это неупорядоченная коллекция пар ключ-значение.

```
my_dict = {"a": 1, "b": 2, "c": 3}
empty_dict = {}

my_dict["d"] = 4  # Добавление новой пары
my_dict["a"] = 10  # Обновление значения по ключу

# Удаление элементов
del my_dict["b"]
value = my_dict.pop("c")

# Доступ к элементу
value = my_dict["a"]
```

Сравнение структур данных

ПЕРЕДОВАЯ ИНЖЕНЕРНАЯ ШКОЛА
УНИВЕРСИТЕТА ИННОПОЛИС

	Списки []	Кортежи ()	Множества {}	Словари {k : v}
Элементы уникальны	-	-	V	Уникальны ключи (k)
Можно добавлять/удалять элементы	V	-	V	V
Можно изменять элементы	V	-	-	V
Доступен слайсинг	V	V	-	-
Элементы доступны по индексу/ключу	V	V	-	V
Доступны операции над множествами (&,|,^,-)	-	-	V	-
Можно сортировать/переворачивать	V	-	-	-
Неизменяемый тип (Immutable)	-	V	-	V (key) / - (value)

Передовые инженерные школы

МИНОБРНАУКИ РОССИИ

УНИВЕРСИТЕТ ИННОПОЛИС

онлайн университет

Структуры данных – производительность

Важность выбора структуры данных

ПЕРЕДОВАЯ ИНЖЕНЕРНАЯ ШКОЛА
УНИВЕРСИТЕТА ИННОПОЛИС

Разные структуры данных имеют разные временные и пространственные сложности для операций добавления, удаления, поиска и изменения данных. Выбор правильной структуры данных может упростить код и сделать его более понятным.

Пример:

Список: Хорош для хранения упорядоченных коллекций элементов. Операции добавления и удаления в конец списка выполняются за постоянное время.

Словарь: Хорош для хранения данных с быстрым доступом по ключу. Поиск, добавление и удаление элементов выполняются за постоянное время.

Пример задачи: Управление списком контактов

ПЕРЕДОВАЯ ИНЖЕНЕРНАЯ ШКОЛА УНИВЕРСИТЕТА ИННОПОЛИС

Представьте, что вы разрабатываете приложение для управления списком контактов. Вам нужно хранить информацию о каждом контакте, такую как имя, телефонный номер и адрес электронной почты. Какую структуру данных вы выберете для хранения этой информации?

Пример: Список vs Словарь

ПЕРЕДОВАЯ ИНЖЕНЕРНАЯ ШКОЛА
УНИВЕРСИТЕТА ИННОПОЛИС

```
contacts = [
    ["Alice", "123-456-7890", "alice@example.com"],
    ["Bob", "987-654-3210", "bob@example.com"],
    ["Charlie", "555-555-5555", "charlie@example.com"]
]

def find_contact_list(name):
    for contact in contacts:
        if contact[0] == name:
            return contact
    return None

print(find_contact_list("Alice"))
```

Время поиска контакта в худшем случае линейное (O(n)), так как нужно проверить каждый элемент списка.

```
contacts = {
    "Alice": {"phone": "123-456-7890", "email": "alice@example.com"},
    "Bob": {"phone": "987-654-3210", "email": "bob@example.com"},
    "Charlie": {"phone": "555-555-5555", "email":
"charlie@example.com"}
}

def find_contact_dict(name):
    return contacts.get(name, None)

print(find_contact_dict("Alice"))
```

Время поиска, добавления и удаления контакта постоянное (O(1)), так как операции выполняются через хэш-таблицу.

Что такое O-нотация

O-нотация, или асимптотическая нотация, используется для описания производительности алгоритмов в зависимости от размера входных данных. Она показывает, как изменяется время выполнения или использование памяти по мере роста объема данных при наихудшем сценари.

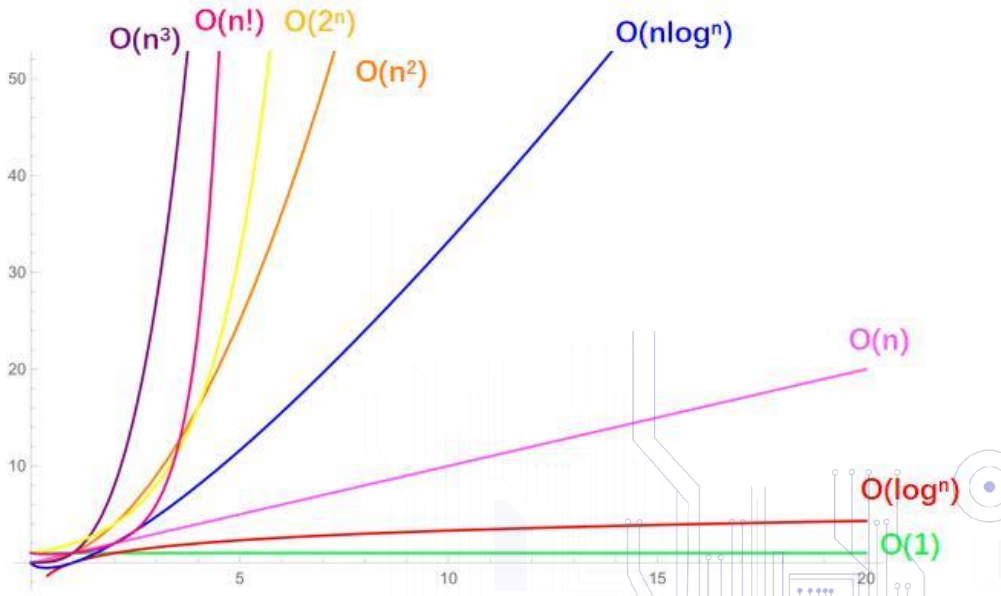
Линейный поиск: Время выполнения зависит от количества элементов в списке. Если список увеличивается в два раза, время выполнения также увеличится в два раза. Это обозначается как O(n).

Бинарный поиск: Время выполнения зависит от логарифма количества элементов. Если список увеличивается в два раза, время выполнения увеличится незначительно. Это обозначается как O(log n).

Таблица различных видов О-нотаций

ПЕРЕДОВАЯ ИНЖЕНЕРНАЯ ШКОЛА
УНИВЕРСИТЕТА ИННОПОЛИС

О-нотация	Описание	Пример	Иллюстрация
O(1)	Время выполнения не зависит от размера входных данных.	Доступ к элементу массива по индексу.	График, где линия времени выполнения остается плоской при увеличении размера данных.
O(n)	Время выполнения растет линейно с увеличением размера входных данных.	Поиск элемента в неотсортированном списке.	График, где время выполнения растет линейно с увеличением размера данных.
O(log n)	Время выполнения растет пропорционально логарифму размера входных данных.	Бинарный поиск в отсортированном списке.	График, где время выполнения растет медленно по мере увеличения размера данных.
O(n^2)	Время выполнения растет пропорционально квадрату размера входных данных.	Сортировка пузырьком.	График, где время выполнения растет быстрее по мере увеличения размера данных.



Полезная статья

Наглядное сравнение

Таблица сравнения структур данных

ПЕРЕДОВАЯ ИНЖЕНЕРНАЯ ШКОЛА
УНИВЕРСИТЕТА ИННОПОЛИС

Операция	Список (List)	Кортеж (Tuple)	Множество (Set)	Словарь (Dictionary)
Поиск элемента	O(n)	O(n)	O(1)	O(1)
Добавление элемента	O(1)	N/A	O(1)	O(1)
Удаление элемента	O(n)	N/A	O(1)	O(1)

Списки (Lists): Подходят для хранения упорядоченных коллекций элементов с быстрым доступом по индексу, но операции поиска и удаления могут быть медленными.
Кортежи (Tuples): Имеют те же преимущества и недостатки, что и списки, но неизменяемы.
Множества (Sets): Отлично подходят для хранения уникальных элементов и выполнения быстрых операций поиска, добавления и удаления.
Словари (Dictionaries): Идеальны для хранения пар ключ-значение с быстрым доступом, поиском, добавлением и удалением элементов.

Рекомендации

Передовые инженерные школы

МИНОБРНАУКИ РОССИИ

УНИВЕРСИТЕТ ИННОПОЛИС

онлайн университет

Практическая часть

Задание 1: Управление библиотекой книг

ПЕРЕДОВАЯ
ИНЖЕНЕРНАЯ ШКОЛА
УНИВЕРСИТЕТА ИННОПОЛИС

Создайте систему управления библиотекой, которая позволяет добавлять книги, удалять книги, искать книги по названию и автору.

Задание 2: Базовая статистика

ПЕРЕДОВАЯ
ИНЖЕНЕРНАЯ ШКОЛА
УНИВЕРСИТЕТА ИННОПОЛИС

Напишите программу для вывода моды, медианы и среднего

Задание 3: Анализ текста

ПЕРЕДОВАЯ
ИНЖЕНЕРНАЯ ШКОЛА
УНИВЕРСИТЕТА ИННОПОЛИС

Напишите программу, которая анализирует текст, введенный пользователем, и выводит количество слов, количество уникальных слов и самое частое слово.

Передовые инженерные школы

МИНОБРНАУКИ РОССИИ

УНИВЕРСИТЕТ ИННОПОЛИС

онлайн университет

Спасибо за внимание