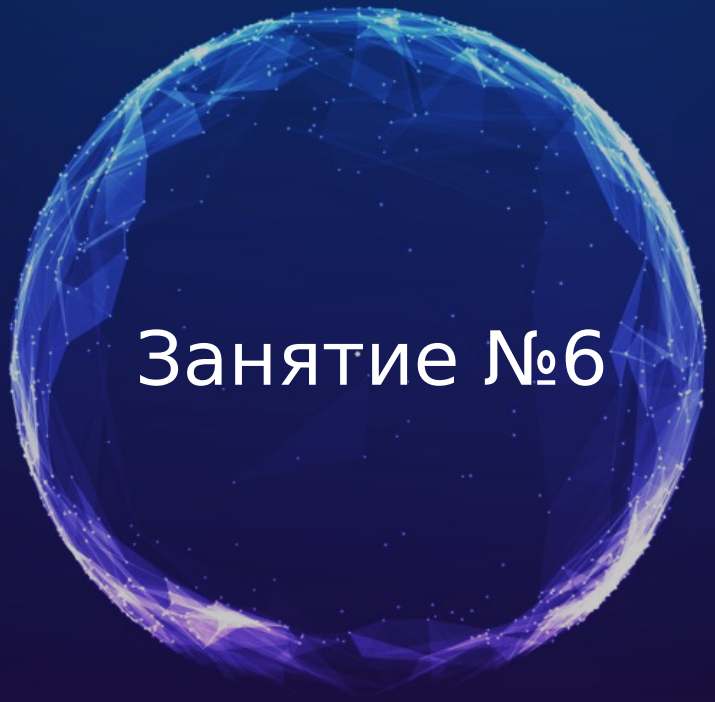


Занятие №6

I T E A

IT Education Academy

WWW.ITEA.UA

I T E A

www.itea.ua

Поляков Антон

Фото инструктора

- Python backend developer

КОНТАКТНЫЕ ДАННЫЕ

Telegram: @polyakov1

Итераторы

Основное место использования итераторов – это цикл for. Если вы перебираете элементы в некотором списке или символы в строке с помощью цикла for, то ,фактически, это означает, что при каждой итерации цикла происходит обращение к итератору.

Итераторы

itr = iter(num_list) #Получаем итератор объекта

next(itr)#Проходимся по итератору
next(itr)#Далее
next(itr)#Далее

Когда итератор исчерпал себя получаем StopIteration Exception.

Итераторы

Если мы хотим построить класс, объекты которого должны поддерживать итерирование, мы должны реализовать методы __next__(), и __iter__() для поддержки цикла for.

Разница между итератором и итерируемым объектом.

Итеририруемый объект — сущность которая поддерживает итерацию.

Итератор — объект который используется для итерации по итерируемой сущности. Итераторы содержат метод __next__(), который возвращает следующий элемент последовательности.

Итератор == итерируемый объект.
Итерируемый объект не всегда итератор.

Итераторы

Для того, чтобы получить итератор от итерируемой сущности используется функция iter(), при успешном выполнении мы получаем объект, который поддерживает метод __next__().

На примере с list.

```
list_var = [1, 2, 3]
```

```
iter_var = iter(list_var)
```

```
next(iter_var)
```

Генератор

Брат итератора, который позволяет писать итераторы, но красивее и проще.

```
def series_generator(low, high):
    while low <= high:
        yield low
        Low += 1
```

Выше была описана функция генератор.

```
(x**2 for x in range(10))#Генераторное выражение
```

Создаём свою структуру данных

```
class CustomList(object):
    def __init__(self, elements=0):
        self.my_custom_list = [0] * elements

    def __setitem__(self, index, value):
        self.my_custom_list[index] = value

    def __getitem__(self, index):
        return "Hey you are accessing {} element whose value is:
{}".format(index, self.my_custom_list[index])

    def __str__(self):
        return str(self.my_custom_list)
```

Задача

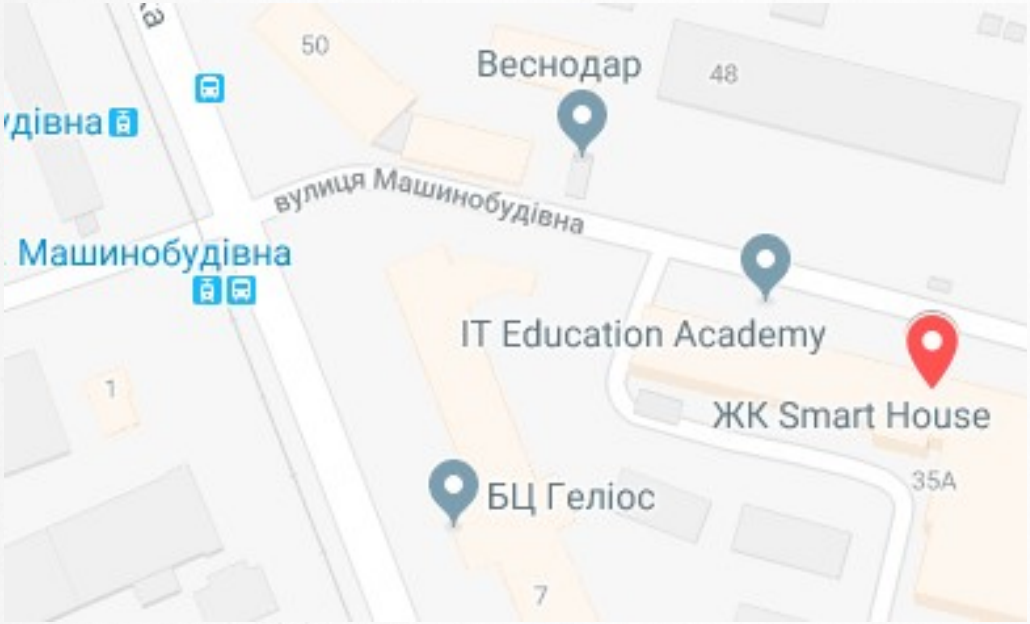
1) Создать свою структуру данных Список, которая поддерживает индексацию. Методы pop, append, insert, remove, clear. Перегрузить операцию сложения для списков, которая возвращает новый расширенный объект.

2) Создать свою структуру данных Словарь, которая поддерживает методы, get, items, keys, values. Так же перегрузить операцию сложения для словарей, которая возвращает новый расширенный объект.

Указанные методы описываем самостоятельно, без использования стандартных.

Домашняя работа

1) Создать консольную программу-парсер, с выводом прогноза погоды. Дать возможность пользователю получить прогноз погоды в его локации (по умолчанию) и в выбраной локации, на определенную пользователем дату. Можно реализовать, как консольную программу, так и веб страницу. Используемые инструменты: requests, beatifulsoup, остальное по желанию. На выбор можно спарсить страницу, либо же использовать какой-либо API.



КОНТАКТНЫЕ ДАННЫЕ

ITEA

ул. Машиностроительная, 41, ЖК «Smart House», Киев

ул. Срибнокильская, 1, офис 269, Киев

пр. Академика Глушкова, 1, корп.17, Киев

+38 (044) 599-01-79
facebook.com/Itea
info@itea.ua
itea.ua