1.3 Реализация скрипта, позволяющего выполнять импорт удаленных файлов по URL:

import requests

def download\_file(url, save\_path):

response = requests.get(url)

with open(save\_path, 'wb') as file:

file.write(response.content)

print("Файл успешно скачан!")

# Пример использования

file\_url = "http://example.com/sample.pdf"

save\_path = "path/to/save/sample.pdf"

download\_file(file\_url, save\_path)

2.1. Разработать функцию, возвращающую стасов чисел ряда Фибоначчи с бесконечных итератором (мазуль itertools):

import itertools

def fibonacci():

a, b = 0, 1

while True:

yield a

a, b = b, a + b

# Пример использования

fib\_iter = fibonacci()

for \_ in range(10):

print(next(fib\_iter))

В данном примере функция fibonacci реализует бесконечный генератор чисел Фибоначчи. Она использует конструкцию yield для возвращения числа и сохранения состояния итерации. При каждом вызове next генератор продолжает выполнение с места остановки, возвращая следующее число Фибоначчи.

3.1. Пример фрагмента кода, реализующего шаблон проектирования "Декоратор":

from functools import wraps

def make\_blink(function):

@wraps(function)

def decorator():

msg = function()

return "<blink>" + msg + "</blink>"

return decorator

@make\_blink

def hello\_world():

return "Hello, World!"

print(hello\_world())

В данном примере функция make\_blink принимает другую функцию в качестве аргумента и возвращает декоратор, который добавляет тег <blink> к результату вызова этой функции.

Декоратор @make\_blink применяется к функции hello\_world, после чего при вызове hello\_world() будет возвращаться строка "Hello, World!" внутри тега <blink>.

4.1. Визуализация данных о ценах на недвижимость с помощью библиотеки matplotlib:

import matplotlib.pyplot as plt

# Исходные данные

prices = [100, 200, 300, 400, 500]

months = ['Jan', 'Feb', 'Mar', 'Apr', 'May']

# Создание линейного графика

plt.plot(months, prices, label='Prices')

plt.xlabel('Months')

plt.ylabel('Price')

plt.title('Price Changes')

plt.legend()

# Создание графика полинома второй степени

poly\_coeffs = np.polyfit(range(len(prices)), prices, deg=2)

poly\_prices = np.polyval(poly\_coeffs, range(len(prices)))

plt.plot(months, poly\_prices, label='Quadratic')

plt.legend()

# Отображение графиков

plt.show()

В данном примере используется библиотека matplotlib для создания линейного графика цен на недвижимость и графика полинома второй степени.

Сначала определяются исходные данные - список цен prices и список месяцев months. Затем с помощью метода plot создаются линейный график и график полинома второй степени.