

**Федеральное государственное автономное образовательное
учреждение высшего образования
«Московский физико-технический институт
(национальный исследовательский университет)»**

УТВЕРЖДЕНО

**Директор по цифровизации
образования**

Д.И. Гриц

	Рабочая программа дисциплины (модуля)
по дисциплине:	Программирование на Python
по направлению:	Бизнес-информатика
профиль подготовки:	Финансовые технологии и аналитика центр дополнительного, дополнительного профессионального и онлайн-образования "Пуск" центр дополнительного, дополнительного профессионального и онлайн-образования "Пуск"
курс:	1
квалификация:	магистр

Семестры, формы промежуточной аттестации:

1 (осенний) - Зачет
2 (весенний) - Зачет

Аудиторных часов: 135 всего, в том числе:

лекции: 60 час.
семинары: 75 час.
лабораторные занятия: 0 час.

Самостоятельная работа: 135 час.

Всего часов: 270, всего зач. ед.: 6

Программу составил: Р.С. Кулиев, старший преподаватель

Программа обсуждена на заседании центра дополнительного, дополнительного профессионального и
онлайн-образования "Пуск" 23.09.2022

Аннотация

Дисциплина состоит из 4 модулей:

Модуль 1. Основы программирования на Python

Модуль 2. Объектно-ориентированное программирование (ООП), графический интерфейс и основы работы с базами данных в Python

Модуль 3. Создание web-приложений в Python

Модуль 4. Анализ данных в Python.

По итогам обучения обучающийся будет способен формализовать и алгоритмизировать поставленную задачу, написать программный код с использованием языков программирования, оформить код в соответствии с установленными требованиями.

1. Цели и задачи

Цель дисциплины

Целью реализации дисциплины «Программирование на Python» является формирование/совершенствование компетенций в области решения профессиональных задач по программированию с использованием языка Python, применения шаблонов проектирования на Python, работы с Python библиотеками, применения объектно-ориентированного и функционального программирования.

Задачи дисциплины

- Сформировать умение использовать базовые типы и конструкции языка программирования Python;
- сформировать умение работать со стандартными структурами данных в Python, писать функции на Python, применять функциональные особенности языка, работать с файлами с помощью языка Python;
- сформировать умение применять механизмы наследования, создавать классы и работать с ними, обрабатывать исключения;
- сформировать умение искать и исправлять ошибки в программе на Python, тестировать программы на Python;
- сформировать умение писать многопоточный код на Python, писать асинхронный код на Python, работать с сетью, создать своё серверное сетевое приложение;
- сформировать умение пользоваться структурным программированием, использовать библиотеку unittest;
- сформировать умение создавать корректную иерархию классов, интерпретировать UML-диаграммы, выполнять рефакторинг существующего кода;
- сформировать умение создавать Декоратор класса, создавать адаптер для интерфейса, несовместимого с системой, реализовывать паттерн Наблюдатель;
- сформировать умение создавать цепочку обязанностей, создавать абстрактную фабрику, создавать обработчик YAML файла;
- сформировать умение работать с библиотекой requests;
- сформировать умение работать с регулярными выражениями из Python, выполнять сложный поиск и замену при помощи регулярных выражений;
- сформировать умение извлекать и изменять данные при помощи модуля BeautifulSoup, использовать API для получения данных со сторонних сайтов;
- сформировать умение создавать и изменять базы данных и таблицы в MySQL, получать данные из баз и таблиц в MySQL;
- сформировать умение создавать приложение на Django, работать с Django-шаблонизатором, работать с базой данных при помощи Django ORM;
- сформировать умение отправлять данные из браузера, валидировать данные на клиентской стороне, валидировать данные на серверной стороне, проводить аутентификацию и авторизацию при помощи Django;
- сформировать умение создавать чат-бота на базе Telegram, работать с системой Git, раскладывать проект на облачный хостинг Heroku;
- сформировать умение применять инструменты библиотеки NumPy, применять инструменты библиотеки SciPy, применять инструменты библиотеки Pandas для работы с данными;

- сформировать умение визуализировать данные при помощи инструментов Python, применить на практике инструменты Python для работы со статистическим анализом;
- сформировать умение применять на практике линейную регрессию, применять на практике кросс-валидацию, оценивать качества моделей, обучать на практике ансамблевые модели;
- сформировать умение применять на практике методы кластеризации, применять на практике методы понижения размерности. создавать рекомендательную сеть;
- сформировать умение реализовывать перцептрон, реализовывать свою нейронную сеть.

2. Перечень формируемых компетенций

Освоение дисциплины направлено на формирование следующих компетенций:

Код и наименование компетенции	Индикаторы достижения компетенции
ОПК-1 Способен разрабатывать стратегию развития информационных технологий, инфраструктуры предприятия и управлять её реализацией	ОПК-1.1 Применяет на практике методики оценки качества ресурсов информационных технологий, управления активами и конфигурации информационных технологий, методики определения потребностей в уровне качества ресурсов ИТ

3. Перечень планируемых результатов обучения по дисциплине (модулю)

В результате освоения дисциплины обучающиеся должны знать:

- Базовые сведения о языке, особенности организации кода на Python;
- стандартные структуры данных в Python;
- механизмы наследования, классы;
- особенности объектно-ориентированной модели в Python;
- процессы и потоки ОС;
- модульное тестирование и его преимущества, методику TDD, её особенностях и преимуществах, контрактное программирование;
- основные парадигмы и принципы ООП, терминологию ООП;
- виды паттернов проектирования, основные паттерны и задачи, которые они решают;
- паттерн Chain of responsibility, паттерн Abstract Factory;
- принципы функционирования современного интернета, основные протоколы в web-взаимодействия;
- причины необходимости сбора данных со сторонних сайтов;
- удобные способы получения данных;
- реляционные базы данных, нереляционные базы данных, инструменты Redis;
- архитектуру web-фреймворков, популярные web-фреймворки в Python, устройство view в Django, основы HTML и CSS;
- понятия аутентификации и авторизации;
- отличия Development и Production;
- базовые понятия математического анализа, базовые понятия линейной алгебры;
- понятия математической статистики.

уметь:

- Использовать базовые типы и конструкции языка;
- работать со стандартными структурами данных в Python, писать функции на Python, применять функциональные особенности языка, работать с файлами с помощью языка Python;
- применять механизмы наследования, создавать классы и работать с ними, обрабатывать исключения;
- искать и исправлять ошибки в программе на Python, тестировать программы на Python;
- писать многопоточный код на Python, писать асинхронный код на Python, работать с сетью, создать своё серверное сетевое приложение;
- создавать корректную иерархию классов, интерпретировать UML-диаграммы, выполнять рефакторинг существующего кода;
- создавать Декоратор класса, создавать адаптер для интерфейса, несовместимого с системой, реализовывать паттерн Наблюдатель;
- создавать цепочку обязанностей, создавать абстрактную фабрику, создавать обработчик YAML файла;
- работать с регулярными выражениями из Python, выполнять сложный поиск и замену при помощи регулярных выражений;
- извлекать и изменять данные при помощи модуля Beautiful Soup, использовать API для получения данных со сторонних сайтов;
- создавать и изменять базы данных и таблицы в MySQL, получать данные из баз и таблиц в MySQL;
- создавать приложение на Django, работать с Django-шаблонизатором, работать с базой данных при помощи Django ORM;
- отправлять данные из браузера, валидировать данные на клиентской стороне, валидировать данные на серверной стороне, проводить аутентификацию и авторизацию при помощи Django;
- создавать чат-бота на базе Telegram, работать с системой Git, раскладывать проект на облачный хостинг Heroku;
- визуализировать данные при помощи инструментов Python, применить на практике инструменты Python для работы со статистическим анализом.

владеть:

- Структурным программированием, библиотекой unittest;
- библиотекой requests;
- Django-шаблонизатором;
- системой Git;
- инструментами библиотеки NumPy, инструментами библиотеки SciPy, инструментами библиотеки Pandas для работы с данными.

4. Содержание дисциплины (модуля), структурированное по темам (разделам) с указанием отведенного на них количества академических часов и видов учебных занятий

4.1. Разделы дисциплины (модуля) и трудоемкости по видам учебных занятий

№	Тема (раздел) дисциплины	Трудоемкость по видам учебных занятий, включая самостоятельную работу, час.			
		Лекции	Семинары	Лаборат. работы	Самост. работа
1	Основы программирования на Python	30	30		75
2	Объектно-ориентированное программирование (ООП), графический интерфейс и основы работы с базами данных в Python	10	16		20
3	Создание web-приложений в Python	10	18		20
4	Анализ данных в Python	10	11		20
Итого часов		60	75		135
Подготовка к экзамену		0 час.			

Общая трудоёмкость	270 час., 6 зач.ед.
--------------------	---------------------

4.2. Содержание дисциплины (модуля), структурированное по темам (разделам)

Семестр: 1 (Осенний)

1. Основы программирования на Python

Основы программирования на Python. Структуры данных и функции. Объектно-ориентированное программирование. Углубленный Python. Многопоточное и асинхронное программирование.

Семестр: 2 (Весенний)

2. Объектно-ориентированное программирование (ООП), графический интерфейс и основы работы с базами данных в Python

Тестирование и отладка программ. Объектно-ориентированное проектирование. Паттерны проектирования. Графический интерфейс.

3. Создание web-приложений в Python

Общее представление о WEB. Сбор данных со сторонних сайтов. BeautifulSoup и работа с API. Хранение данных. SQL / NoSQL. Веб интерфейсы с Django и Bootstrap. Работа с данными пользователя. Дополнительный инструментарий.

4. Анализ данных в Python

Математика и Python для анализа данных. Визуализация данных и статистика.

5. Описание материально-технической базы, необходимой для осуществления образовательного процесса по дисциплине (модулю)

Система дистанционного обучения:

Обучающемуся необходимо наличие доступа в сеть интернет, компьютер.

Преподавателю курса необходимо наличие доступа администратора курса и оборудование для проведения дистанционных семинаров (вебинаров), качественный отказоустойчивый доступ в сеть интернет.

6. Перечень рекомендуемой литературы

Основная литература

1. Язык программирования PYTHON [Текст] : учеб. пособие для вузов / Р. А. Сузи .— 2 изд., испр. — М. : Интернет-Ун-т Информ. Технологий : БИНОМ. Лаб. знаний, 2007 .— 326 с.
2. Программирование на языке высокого уровня Python, учебное пособие для вузов / Д. Ю. Федоров. — Москва, Юрайт, 2020.— URL: <https://urait.ru/bcode/454100> (дата обращения: 15.12.2020). - Полный текст (Режим доступа : из сети МФТИ / Удаленный доступ)

Дополнительная литература

1. Программирование на Python 3 : Подробное руководство [Текст] = Programming in Python 3 : [учеб. пособие для вузов] / М. Саммерфилд; пер. с англ. А. Киселева .— СПб : Символ-Плюс, 2015 .— 608 с.
2. Python на практике [Текст], создание качественных программ с использованием параллелизма, библиотек и паттернов/М. Саммерфилд, -М, ДМК Пресс, 2014
3. Изучаем Python / Эрик Мэтиз, Санкт-Петербург, Питер, 2020

4. Python и анализ данных, Первичная обработка данных с применением pandas, NumPy и IPython / У. Маккини. — Москва, ДМК Пресс, 2020.— URL: <https://e.lanbook.com/book/131721> (дата обращения: 26.01.2021). - Полный текст (Режим доступа : из сети МФТИ / Удаленный доступ)

7. Перечень ресурсов информационно-телекоммуникационной сети "Интернет", необходимых для освоения дисциплины (модуля)

Think Python [Электронный ресурс] – Режим доступа - <https://greenteapress.com/wp/think-python-2e/>
Automate the Boring Stuff with Python [Электронный ресурс] – Режим доступа - <https://automatetheboringstuff.com/>
Dive Into Python 3 [Электронный ресурс] – Режим доступа - <http://diveintopython3.problemsolving.io/>
Problem Solving with Algorithms and Data Structures using Python [Электронный ресурс] – Режим доступа - <https://runestone.academy/runestone/static/pythonds/index.html>
Swaroop Chitlur. A Byte of Python [Электронный ресурс] – Режим доступа - <https://wombat.org.ua/AByteOfPython/AByteofPythonRussian-2.02.pdf> – 2020.

8. Перечень информационных технологий, используемых при осуществлении образовательного процесса по дисциплине (модулю), включая перечень необходимого программного обеспечения и информационных справочных систем (при необходимости)

Документация Postgres про сравнение строк - <https://postgrespro.ru/docs/postgrespro/9.5/functions-matching>
Документация Postgres про другие функции работы со строками - <https://postgrespro.ru/docs/postgrespro/9.5/functions-string>
Тестер регулярных выражений - <https://www.regextester.com>
Интерактивный учебник по SQL - <http://www.sql-tutorial.ru/ru/content.html>
Введение в анализ данных с помощью Pandas - <https://habr.com/ru/post/196980/>
Начало работы с Power BI - <https://docs.microsoft.com/ru-ru/power-bi/fundamentals/desktop-getting-started>
Профессиональный информационно-аналитический ресурс, посвященный машинному обучению, распознаванию образов и интеллектуальному анализу данных – <http://www.machinelearning.ru/>
Информационная система «Единое окно доступа к образовательным ресурсам» (ИС «Единое окно») – <http://window.edu.ru/>
Платформа открытое образование – <https://openedu.ru/>

9. Методические указания для обучающихся по освоению дисциплины (модуля)

Самостоятельная работа подразделяется на аудиторную и внеаудиторную. Аудиторную самостоятельную работу составляют практические задания, которые выполняются слушателями во время учебных занятий, результаты ее выполнения проверяются и оцениваются преподавателем в учебном процессе.

Внеаудиторная самостоятельная работа включает формы: изучение дополнительной литературы, подготовка итоговых проектов по модулям, подготовка проекта.

Основными критериями качества организации самостоятельной работы служит наличие контроля результатов самостоятельной работы.

Основными современными формами организации самостоятельной работы являются творческие работы и работа с информационными компьютерными технологиями.

ОЦЕНОЧНЫЕ МАТЕРИАЛЫ ПО ДИСЦИПЛИНЕ (МОДУЛЮ)

по направлению:	Бизнес-информатика		
профиль подготовки:	Финансовые технологии и аналитика	▲	▲
	онлайн-образования "Пуск"	▲	▲
	онлайн-образования "Пуск"		
курс:	1		
квалификация:	магистр		
Семестры, формы промежуточной аттестации:			
	1 (осенний) - Зачет		
	2 (весенний) - Зачет		
Разработчик:	Р.С. Кулиев, старший преподаватель		

1. Компетенции, формируемые в процессе изучения дисциплины

Код и наименование компетенции	Индикаторы достижения компетенции
ОПК-1 Способен разрабатывать стратегию развития информационных технологий, инфраструктуры предприятия и управлять её реализацией	ОПК-1.1 Применяет на практике методики оценки качества ресурсов информационных технологий, управления активами и конфигурации информационных технологий, методики определения потребностей в уровне качества ресурсов ИТ

2. Показатели оценивания компетенций

В результате изучения дисциплины «Программирование на Python» обучающийся должен:

знать:

- Базовые сведения о языке, особенности организации кода на Python;
- стандартные структуры данных в Python;
- механизмы наследования, классы;
- особенности объектно-ориентированной модели в Python;
- процессы и потоки ОС;
- модульное тестирование и его преимущества, методику TDD, её особенностях и преимуществах, контрактное программирование;
- основные парадигмы и принципы ООП, терминологию ООП;
- виды паттернов проектирования, основные паттерны и задачи, которые они решают;
- паттерн Chain of responsibility, паттерн Abstract Factory;
- принципы функционирования современного интернета, основные протоколы в web-взаимодействия;
- причины необходимости сбора данных со сторонних сайтов;
- удобные способы получения данных;
- реляционные базы данных, нереляционные базы данных, инструменты Redis;
- архитектуру web-фреймворков, популярные web-фреймворки в Python, устройство view в Django, основы HTML и CSS;
- понятия аутентификации и авторизации;
- отличия Development и Production;
- базовые понятия математического анализа, базовые понятия линейной алгебры;
- понятия математической статистики.

уметь:

- Использовать базовые типы и конструкции языка;
- работать со стандартными структурами данных в Python, писать функции на Python, применять функциональные особенности языка, работать с файлами с помощью языка Python;
- применять механизмы наследования, создавать классы и работать с ними, обрабатывать исключения;
- искать и исправлять ошибки в программе на Python, тестировать программы на Python;
- писать многопоточный код на Python, писать асинхронный код на Python, работать с сетью, создать своё серверное сетевое приложение;
- создавать корректную иерархию классов, интерпретировать UML-диаграммы, выполнять рефакторинг существующего кода;
- создавать Декоратор класса, создавать адаптер для интерфейса, несовместимого с системой, реализовывать паттерн Наблюдатель;
- создавать цепочку обязанностей, создавать абстрактную фабрику, создавать обработчик YAML файла;
- работать с регулярными выражениями из Python, выполнять сложный поиск и замену при помощи регулярных выражений;
- извлекать и изменять данные при помощи модуля Beautiful Soup, использовать API для получения данных со сторонних сайтов;
- создавать и изменять базы данных и таблицы в MySQL, получать данные из баз и таблиц в MySQL;
- создавать приложение на Django, работать с Django-шаблонизатором, работать с базой данных при помощи Django ORM;
- отправлять данные из браузера, валидировать данные на клиентской стороне, валидировать данные на серверной стороне, проводить аутентификацию и авторизацию при помощи Django;
- создавать чат-бота на базе Telegram, работать с системой Git, раскладывать проект на облачный хостинг Heroku;
- визуализировать данные при помощи инструментов Python, применить на практике инструменты Python для работы со статистическим анализом.

владеть:

- Структурным программированием, библиотекой unittest;
- библиотекой requests;
- Django-шаблонизатором;
- системой Git;
- инструментами библиотеки NumPy, инструментами библиотеки SciPy, инструментами библиотеки Pandas для работы с данными.

3. Перечень типовых (примерных) вопросов, заданий, тем для подготовки к текущему контролю

4. Перечень типовых (примерных) вопросов и тем для проведения промежуточной аттестации обучающихся

Критерии оценивания

Оценка «зачтено» выставляется студенту, если он показал всесторонние, систематизированные, глубокие знания учебной программы дисциплины и умение уверенно применять их на практике при решении конкретных задач, свободное и правильное обоснование принятых решений.

Оценка «не зачтено» выставляется студенту, который не знает большей части основного содержания учебной программы дисциплины, допускает грубые ошибки в формулировках основных понятий дисциплины и не умеет использовать полученные знания при решении типовых практических задач.

5. Методические материалы, определяющие процедуры оценивания знаний, умений, навыков и (или) опыта деятельности

Модуль 1

Тема 1. Введение в Python

Написать программу (скрипт), которая будет запускаться из командной строки. Программа принимает в качестве аргумента строку, состоящую из цифр. Гарантируется, что других символов в переданном параметре нет и на вход всегда подается не пустая строка. Программа должна вычислить сумму цифр из которых состоит строка и вывести полученный результат на печать в стандартный вывод.

Тема 2. Структуры данных и функции

Реализовать собственное **key-values** хранилище. Данные будут сохраняться в файле **storage.data**. Добавление новых данных в хранилище и получение текущих значений осуществляется с помощью утилиты командной строки **storage.py**.

Тема 3. Объектно-ориентированное программирование

Написать python-модуль `solution.py`, внутри которого необходимо поместить код класса `FileReader`. Конструктор этого класса принимает один параметр: путь до файла на диске. В классе `FileReader` должен быть реализован метод `read`, возвращающий строку - содержимое файла, путь к которому был указан при создании экземпляра класса. Python модуль должен быть написан таким образом, чтобы импорт класса `FileReader` из него не вызвал ошибок.

При написании реализации метода `read`, вам нужно учитывать случай, когда при инициализации был передан путь к несуществующему файлу. Требуется обработать возникающее при этом исключение `FileNotFoundException` и вернуть из метода `read` пустую строку.

Тема 4. Углубленный Python

Создать интерфейс для работы с файлами. Интерфейс должен предоставлять следующие возможности по работе с файлами:

- чтение из файла, метод `read` возвращает строку с текущим содержанием файла
 - запись в файл, метод `write` принимает в качестве аргумента строку с новым содержанием файла
 - сложение объектов типа `File`, результатом сложения является объект класса `File`, при этом создается новый файл и файловый объект, в котором содержимое второго файла добавляется к содержимому первого файла. Новый файл должен создаваться в директории, полученной с помощью функции `tempfile.gettempdir`. Для получения нового пути можно использовать `os.path.join`.
 - возвращать в качестве строкового представления объекта класса `File` полный путь до файла
 - поддерживать протокол итерации, причем итерация проходит по строкам файла
- При создании экземпляра класса `File` в конструктор передается полный путь до файла на файловой системе. Если файла с таким путем не существует, он должен быть создан при инициализации.

Тема 5. Многопоточное и асинхронное программирование

В крупных проектах, с большим количеством пользователей, необходимо тщательно наблюдать за всеми процессами, происходящими в нем. Информация о процессах может быть представлена различными численными показателями, например: количество запросов к вашему приложению, время ответа вашего сервиса на каждый

запрос, количество пользователей в сутки и другие. Эти различные численные показатели мы будем называть метриками.

Для сбора, хранения и отображения подобных метрик существуют готовые решения, например Graphite, InfluxDB. Мы в рамках курса разработаем свою систему для сбора и хранения метрик, основанную на клиент-серверной архитектуре.

На этой неделе мы начнем с разработки клиента для отправки и получения метрик. На следующей неделе, в качестве финального задания, вам будет предложено реализовать сервер для хранения метрик.

Протокол взаимодействия

Прежде, чем приступить к описанию задания, рассмотрим протокол взаимодействия, по которому будет происходить обмен данными между клиентом и сервером.

Клиент и сервер взаимодействуют между собой по простому текстовому протоколу через TCP сокет. Текстовый протокол имеет главное преимущество – наглядность, можно просматривать диалог взаимодействия клиентской и серверной стороны без использования дополнительных инструментов.

Общий формат запросов и ответов.

Протокол поддерживает два вида запросов к серверу со стороны клиента:

- отправка данных для сохранения их на сервере
- получения сохраненных данных

Общий формат запроса клиента:

```
<команда> <данные запроса><\n>
```

где:

- <команда> - команда сервера (команда может принимать одно из двух значений: put — сохранить данные на сервере, get — вернуть сохраненные данные с сервера),

- <данные запроса> - данные запроса (их формат мы подробно разберем ниже в примере),

- <\n> - символ переноса строки.

Обратим ваше внимание на пробел между командой и данными запроса и его отсутствием между данными и символом перевода на новую строку.

Общий формат ответов сервера:

```
<статус ответа><\n><данные ответа><\n\n>
```

где:

- <статус ответа> - статус выполнения команды, допустимы два варианта: «ok» - команда успешно выполнена на сервере и «error» - выполнение команды завершилось ошибкой

- <данные ответа> - не обязательное поле (формат ответа и случаи его отсутствия будут рассмотрены в примере ниже)

- <\n\n> - два символа переноса строки.

Обратите внимание, что статус ответа и данные ответа разделены символом перевода строки <\n>.

Метод put

Метод **put** принимает в качестве параметров: название метрики, численное значение и необязательный именованный параметр **timestamp**. Если пользователь вызвал метод **put** без аргумента **timestamp**, то клиент автоматически должен подставить значение временной отметки, полученное с помощью вызова **int(time.time())**.

Метод **put** не возвращает ничего в случае успешной отправки и выбрасывает пользовательское исключение **ClientError** в случае не успешной.

Method get

Метод **get** принимает в качестве параметра имя метрики, значения которой мы хотим получить. В качестве имени метрики можно использовать символ «*», о котором мы упоминали в описании протокола.

Метод **get** возвращает словарь с метриками (смотрите пример ниже) в случае успешного получения ответа от сервера и выбрасывает исключение **ClientError** в случае не успешного.

Клиент получает данные от сервера в текстовом виде, метод **get** должен обработать строку ответа и вернуть словарь с полученными ключами с сервера. Значением ключей в словаре является список кортежей:

```
[(timestamp1, metric_value1), (timestamp2, metric_value2), ...]
```

Значение **timestamp** и **metric_value** должны быть преобразованы соответственно к типам **int** и **float**. Список должен быть отсортирован по значению **timestamp** (по возрастанию).

Модуль 2

Тема 1. Тестирование и отладка программ

Дана функция **factorize(x)** со следующим контрактом:

```
def factorize(x):  
    """  
    Factorize positive integer and return its factors.  
    :type x: int, >=0  
    :rtype: tuple[N], N>0  
    """  
    pass
```

Необходимо написать комплект тестов используя модуль **unittest** стандартной библиотеки Python. Имя тестового класса - **TestFactorize**.

Описание тестов:

test_wrong_types_raise_exception - проверяет, что передаваемый в функцию аргумент типа **float** или **str** вызывает исключение **TypeError**. Тестовый набор входных данных: 'string', 1.5

test_negative - проверяет, что передача в функцию **factorize** отрицательного числа вызывает исключение **ValueError**. Тестовый набор входных данных: -1, -10, -100

test_zero_and_one_cases - проверяет, что при передаче в функцию целых чисел 0 и 1, возвращаются соответственно кортежи (0,) и (1,). Набор тестовых данных: 0 → (0,), 1 → (1,)

test_simple_numbers - что для простых чисел возвращается кортеж, содержащий одно данное число. Набор тестовых данных: $3 \rightarrow (3,)$, $13 \rightarrow (13,)$, $29 \rightarrow (29,)$

test_two_simple_multipliers — проверяет случаи, когда передаются числа для которых функция factorize возвращает кортеж с числом элементов равным 2. Набор тестовых данных: $6 \rightarrow (2, 3)$, $26 \rightarrow (2, 13)$, $121 \rightarrow (11, 11)$

test_many_multipliers - проверяет случаи, когда передаются числа для которых функция factorize возвращает кортеж с числом элементов больше 2. Набор тестовых данных: $1001 \rightarrow (7, 11, 13)$, $9699690 \rightarrow (2, 3, 5, 7, 11, 13, 17, 19)$

ВАЖНО! Все входные данные должны быть такими, как указано в условии. Название переменной в каждом тестовом случае должно быть именно "x". При этом несколько различных проверок в рамках одного теста должны быть обработаны как подслучаи с указанием x: subTest(x=...). В задании необходимо реализовать ТОЛЬКО класс TestFactorize, кроме этого реализовывать ничего не нужно. Импортировать unittest и вызывать unittest.main() в решении также не нужно.

Тема 2. Объектно-ориентированное проектирование

Даны 3 класса A, B, C, имеющие сходный (но не одинаковый) интерфейс. Вам необходимо создать абстрактный базовый класс Base и построить корректную схему наследования.

При выполнении следует избегать дублирования кода, и стараться следовать SOLID принципам ООП.

Кроме того, рекомендуется самостоятельно тестировать код перед отправкой, а также при написании следовать стандарту PEP 8.

```
import math
```

```
class Base:
```

```
    pass
```

```
class A:
```

```
    def __init__(self, data, result):
        self.data = data
        self.result = result
```

```
    def get_answer(self):
        return [int(x >= 0.5) for x in self.data]
```

```
    def get_score(self):
        ans = self.get_answer()
        return sum([int(x == y) for (x, y) in zip(ans, self.result)]) \
            / len(ans)
```

```
    def get_loss(self):
        return sum(
            [(x - y) * (x - y) for (x, y) in zip(self.data, self.result)])
```

```
class B:
```

```
    def __init__(self, data, result):
        self.data = data
```

```

        self.result = result

    def get_answer(self):
        return [int(x >= 0.5) for x in self.data]

    def get_loss(self):
        return -sum([
            y * math.log(x) + (1 - y) * math.log(1 - x)
            for (x, y) in zip(self.data, self.result)
        ])

    def get_pre(self):
        ans = self.get_answer()
        res = [int(x == 1 and y == 1) for (x, y) in zip(ans, self.result)]
        return sum(res) / sum(ans)

    def get_rec(self):
        ans = self.get_answer()
        res = [int(x == 1 and y == 1) for (x, y) in zip(ans, self.result)]
        return sum(res) / sum(self.result)

    def get_score(self):
        pre = self.get_pre()
        rec = self.get_rec()
        return 2 * pre * rec / (pre + rec)

class C:
    def __init__(self, data, result):
        self.data = data
        self.result = result

    def get_answer(self):
        return [int(x >= 0.5) for x in self.data]

    def get_score(self):
        ans = self.get_answer()
        return sum([int(x == y) for (x, y) in zip(ans, self.result)]) \
            / len(ans)

    def get_loss(self):
        return sum([abs(x - y) for (x, y) in zip(self.data, self.result)])

```

Тема 3. Паттерны проектирования (часть 1)

Необходимо написать реализацию системы эффектов игры в стиле фэнтези, которые могут быть наложены на героя игры.

В игре есть герой, который обладает некоторым набором характеристик. Класс героя описан следующим образом:

```

class Hero:
    def __init__(self):
        self.positive_effects = []
        self.negative_effects = []
        self.stats = {
            "HP": 128, # health points

```

```

    "MP": 42, # magic points,
    "SP": 100, # skill points
    "Strength": 15, # сила
    "Perception": 4, # восприятие
    "Endurance": 8, # выносливость
    "Charisma": 2, # харизма
    "Intelligence": 3, # интеллект
    "Agility": 8, # ловкость
    "Luck": 1 # удача
}

def get_positive_effects(self):
    return self.positive_effects.copy()

def get_negative_effects(self):
    return self.negative_effects.copy()

def get_stats(self):
    return self.stats.copy()

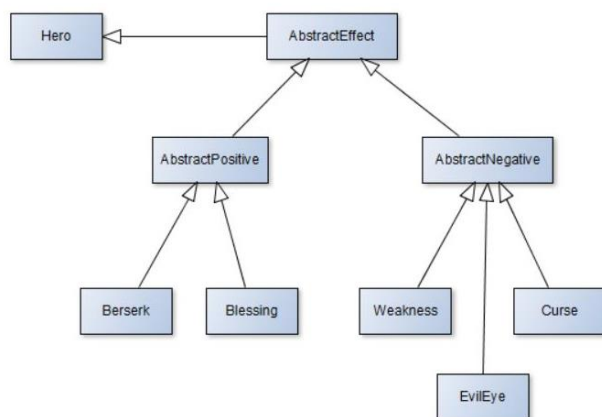
```

К основным характеристикам относятся: Сила (Strength), Восприятие (Perception), Выносливость (Endurance), Харизма (Charisma), Интеллект (Intelligence), Ловкость (Agility), Удача (Luck).

Враги и союзники могут накладывать на героя положительные и отрицательные эффекты. Эти эффекты изменяют характеристики героя, увеличивая или уменьшая значения определенных характеристик, в зависимости от того какие эффекты были наложены. На героя можно накладывать бесконечно много эффектов, действие одинаковых эффектов суммируется. Игрок должен знать, какие положительные и какие отрицательные эффекты на него были наложены и в каком порядке. Названия эффектов совпадают с названиями классов.

За получение данных о текущем состоянии героя отвечают методы `get_stats`, `get_positive_effects`, `get_negative_effects`.

Вам необходимо написать систему декораторов, представленную на UML-диаграмме.



Описание эффектов:

Берсерк (Berserk)

Увеличивает характеристики: Сила, Выносливость, Ловкость, Удача на 7;
уменьшает характеристики: Восприятие, Харизма, Интеллект на 3;

количество единиц здоровья увеличивается на 50.

Благословение (Blessing)

увеличивает все основные характеристики на 2.

Слабость (Weakness)

уменьшает характеристики: Сила, Выносливость, Ловкость на 4.

Сглаз (EvilEye)

уменьшает характеристику Удача на 10.

Проклятие (Curse)

уменьшает все основные характеристики на 2.

При выполнении задания необходимо учитывать, что:

изначальные характеристики базового объекта не должны меняться.

изменения характеристик и накладываемых эффектов (баффов/дебаффов) должны происходить динамически, то есть вычисляться при вызове методов `get_stats`, `get_positive_effects`, `get_negative_effects`

абстрактные классы `AbstractPositive`, `AbstractNegative` и соответственно их потомки могут принимать любой параметр `base` при инициализации объекта (`__init__` (`self`, `base`))

эффекты должны корректно сниматься, в том числе и из середины стека

Для тестирования правильности работы вашего решения вы можете повторить пример работы, приведенный ниже, или использовать тестовый скрипт. Заметим, что данные примеры не покрывают всех тестовых случаев, проверяемых тестовой системой, это всего лишь отправная точка для ваших экспериментов с кодом решения.

Важно! На проверку необходимо сдать фрагмент кода, содержащий только реализацию 8 классов: `AbstractEffect`, `AbstractPositive`, `AbstractNegative`, `Berserk`, `Blessing`, `Weakness`, `Curse`, `EvilEye`. Описывать класс `Hero` и импортировать его в коде НЕ НУЖНО (он уже реализован и будет импортирован тестовой системой).

Пример работы:

```
>>> from deco import *
>>> # создаем героя
>>> hero = Hero()
>>> hero.get_stats()
{'HP': 128, 'MP': 42, 'SP': 100, 'Strength': 15, 'Perception': 4, 'Endurance': 8, 'Charisma': 2, 'Intelligence': 3, 'Agility': 8, 'Luck': 1}
>>> hero.stats
{'HP': 128, 'MP': 42, 'SP': 100, 'Strength': 15, 'Perception': 4, 'Endurance': 8, 'Charisma': 2, 'Intelligence': 3, 'Agility': 8, 'Luck': 1}
>>> hero.get_negative_effects()
[]
>>> hero.get_positive_effects()
[]
>>> # накладываем эффект

>>> brs1 = Berserk(hero)
>>> brs1.get_stats()
{'HP': 178, 'MP': 42, 'SP': 100, 'Strength': 22, 'Perception': 1, 'Endurance': 15, 'Charisma': -1, 'Intelligence': 0, 'Agility': 15, 'Luck': 8}
>>> brs1.get_negative_effects()
[]
>>> brs1.get_positive_effects()
['Berserk']
```



```

>>> # накладываем эффекты
>>> brs2 = Berserk(brs1)

>>> cur1 = Curse(brs2)

>>> cur1.get_stats()
{'HP': 228, 'MP': 42, 'SP': 100, 'Strength': 27, 'Perception': -
4, 'Endurance': 20, 'Charisma': -6, 'Intelligence': -
5, 'Agility': 20, 'Luck': 13}
>>> cur1.get_positive_effects()
['Berserk', 'Berserk']
>>> cur1.get_negative_effects()
['Curse']
>>> # снимаем эффект Berserk
>>> cur1.base = brs1
>>> cur1.get_stats()
{'HP': 178, 'MP': 42, 'SP': 100, 'Strength': 20, 'Perception': -
1, 'Endurance': 13, 'Charisma': -3, 'Intelligence': -
2, 'Agility': 13, 'Luck': 6}
>>> cur1.get_positive_effects()
['Berserk']
>>> cur1.get_negative_effects()
['Curse']

>>>

```

Тема 4. Паттерны проектирования (часть 2)

Вам дан объект класса `SomeObject`, содержащего три поля: **integer_field**, **float_field** и **string_field**:

```

class SomeObject:
    def __init__(self):
        self.integer_field = 0
        self.float_field = 0.0
        self.string_field = ""

```

Необходимо реализовать поведение:

- **EventGet(<type>)** создаёт событие получения данных соответствующего типа
- **EventSet(<value>)** создаёт событие изменения поля типа **type(<value>)**

Необходимо реализовать классы **NullHandler**, **IntHandler**, **FloatHandler**, **StrHandler** так, чтобы можно было создать цепочку:

```
chain = IntHandler(FloatHandler(StrHandler(NullHandler)))
```

Описание работы цепочки:

- `chain.handle(obj, EventGet(int))` — вернуть значение `obj.integer_field`
- `chain.handle(obj, EventGet(str))` — вернуть значение `obj.string_field`
- `chain.handle(obj, EventGet(float))` — вернуть значение `obj.float_field`
- `chain.handle(obj, EventSet(1))` — установить значение `obj.integer_field = 1`
- `chain.handle(obj, EventSet(1.1))` — установить значение `obj.float_field = 1.1`

— chain.handle(obj, EventSet("str")) — установить значение
obj.string_field = "str"

Пример работы:

```
>>> obj = SomeObject()
>>> obj.integer_field = 42
>>> obj.float_field = 3.14
>>> obj.string_field = "some text"
>>> chain = IntHandler(FloatHandler(StrHandler(NullHandler)))
>>> chain.handle(obj, EventGet(int))
42
>>> chain.handle(obj, EventGet(float))
3.14
>>> chain.handle(obj, EventGet(str))
'some text'
>>> chain.handle(obj, EventSet(100))
>>> chain.handle(obj, EventGet(int))
100
>>> chain.handle(obj, EventSet(0.5))
>>> chain.handle(obj, EventGet(float))
0.5
>>> chain.handle(obj, EventSet('new text'))
>>> chain.handle(obj, EventGet(str))
'new text'
```

Модуль 3

Тема 1. Общее представление о WEB

В этом задании вы научитесь работать с библиотекой requests (<https://requests.kennethreitz.org/en/master/>), а также научитесь работать с API сервиса VK и его документаций, что является достаточно частой задачей разработчика.

Необходимо написать клиент к API VK, который будет считать распределение возрастов друзей для указанного пользователя. То есть на вход подается username или user_id пользователя, на выходе получаем **список пар (<возраст>, <количество друзей с таким возрастом>)**, отсортированный по убыванию по второму ключу (количество друзей) и по возрастанию по первому ключу (возраст). Например:

```
[(26, 8), (21, 6), (22, 6), (40, 2), (19, 1), (20, 1)]
```

Для выполнения задания необходимо использовать шаблон проекта: https://github.com/alexopryshko/coursera_assignment_tmp

Решение должно быть файлом **req/friends.py**. В этом файле представлен шаблон функции **calc_age**, реализацию которой нужно написать.

Для этого вам понадобятся два метода API VK:

1. Метод для получения id пользователя (<https://vk.com/dev/users.get>). Он необходим, так как на вход может подаваться username пользователя. URL запроса к API VK: <https://api.vk.com/method/users.get>

2. Метод для получения списка друзей пользователя (<https://vk.com/dev/friends.get>). URL запроса к API VK: <https://api.vk.com/method/friends.get>

Для доступа к этим методам вам понадобится “Сервисный ключ доступа”:

https://vk.com/dev/access_token?f=3.%20%D0%A1%D0%B5%D1%80%D0%B2%D0%B8%D1%81%D0%BD%D1%8B%D0%B9%20%D0%BA%D0%BB%D1%8E%D1%87%20%D0%B4%D0%BE%D1%81%D1%82%D1%83%D0%BF%D0%B0

Получение сервисного ключа:

1. Создать новое приложение, перейдя по ссылке <https://vk.com/apps?act=manage>
2. После создания приложения, перейти в раздел “настройки” и скопировать “Сервисный ключ доступа”.

Если нет возможности получить сервисный ключ, то можно использовать уже созданный:

`ACCESS_TOKEN = '17da724517da724517da72458517b8abce117da17da72454d235c274f1a2be5f45ee711'`

Для получения id пользователя по username или user_id:

[https://api.vk.com/method/users.get?v=5.71&access_token=\[token\]&user_ids=\[user_id\]](https://api.vk.com/method/users.get?v=5.71&access_token=[token]&user_ids=[user_id])

Для получения списка друзей:

[https://api.vk.com/method/friends.get?v=5.71&access_token=\[token\]&user_id=\[user_id\]&fields=bdate](https://api.vk.com/method/friends.get?v=5.71&access_token=[token]&user_id=[user_id]&fields=bdate)

При решении задания обратите внимание на несколько моментов.

- В запросе мы используем версию API VK - «5.71».
- В запросе получения списка друзей добавлен ключ `fields=bdate`. Он необходим для того, чтобы API сразу вернуло пользователей с датами рождения.
- При анализе ответа, полученного методом `friends.get`, можно заметить, что `bdate` есть не у всех пользователей и у некоторых в `bdate` отсутствует год рождения. Поэтому необходимо пропускать этот случай. Примеры возможных значений: `"bdate": "6.6"`, `"bdate": "25.8.1993"`. Для вычисления возраста, необходимо взять текущий год, и вычесть из него год рождения пользователя, полученный из API (без учета месяца и числа).

В своем решении вы можете (если необходимо) определять дополнительные функции и импортировать модули.

Тема 2. Сбор данных со сторонних сайтов

Написать программу, которая найдет в тексте ряд простых арифметических выражений и подсчитает их.

Любое выражение начинается с имени переменной **a**, **b** или **c**, затем может идти **+** или **-**, затем идет **=**, затем может идти имя переменной **a**, **b** или **c**, а затем может идти **+** или **-** и целое число. Если в правой части выражения нет переменной, то число может быть без знака **+** или **-**.

Не бывает названий переменных, кроме **a**, **b** или **c**, и действий, кроме описанных тут. Не бывает пробелов вокруг знаков. В тексте не встречаются некорректные выражения, в которых справа от **=** нет ни переменной, ни числа. Таким образом, список типов выражений, которые могут встречаться, выглядит примерно так:

`a=1`, `a+=1`, `a=-1`, `a=b`, `a=b+100`, `a=b-100`, `b+=10`, `b+=+10`, `b+=-10`, `b+=b`, `b+=b+100`, `b+=b-100`, `c=-101`, `c=-+101`, `c=-101`, `c=-b`, `c=-b+101`, `c=-b-101`

Выражения могут встречаться внутри текста, например `loremc=-a+10ipsuma=-adb+=10olorsitamet`.

В вашу функцию `calculate(data, findall)` будет передан словарь с начальными значениями переменных **a**, **b** и **c**: `data = {"a":1, "b":2, "c": 3}` и ссылка на функцию `findall`, а вы должны вернуть такой же словарь с новыми значениями для **a**, **b** и **c**.

Работает `findall()` аналогично `re.findall()`, только у нее всего один параметр – регулярное выражение. А текст, в котором она будет искать выражения, она знает сама (см. приложенный архив, чтобы понять, о чем речь). С помощью `findall()` нужно как минимум найти все выражения в тексте, а как максимум найти их и разбить на группы так, чтобы было удобно их обработать. *Если `findall()` будет вызвано больше одного раза или если размер списка, который она вернет, будет отличаться от количества выражений в тексте, тест провалится.*

Эталонное решение занимает 11 строк, не содержит импортов, не использует `eval`, а регулярное выражение находит выражения и бьет каждое на четыре группы (некоторые группы для некоторых выражений оказываются пустыми):

1. Имя переменной слева.
2. Знак перед = (если есть).
3. Имя переменной справа (если есть).
4. Число (если есть) со знаком (если есть).

Это позволяет легко (буквально в одну строку) посчитать правую часть, а потом, в зависимости от наличия знака перед =, произвести действие с левой частью. Однако ваш алгоритм может быть другим, требуется только выполнить ограничения на вызов `findall()` и оставить сигнатуру `calculate()` неизменной.

Вы должны скачать архив к этому уроку, изучить оба файла и переписать функцию `calculate()` внутри `regex.py`. Для проверки вашего решения локально запустите `python test.py`. На сервере задание будет проверяться похожим образом, но реализация `findall()` будет иной (с проверкой всех ограничений), и тестов будет несколько. Решением будет файл `regex.py`, который надо загрузить на сервер для проверки.

Тема 3. Beautiful Soup и работа с API

Часть 1:

В этом задании вам необходимо реализовать парсер для сбора статистики со страниц [Википедии](https://en.wikipedia.org/wiki/Stone_Age). Чтобы упростить вашу задачу, необходимые страницы уже скачаны и сохранены на файловой системе в директории `wiki/` (Например, страница https://en.wikipedia.org/wiki/Stone_Age сохранена файле `wiki/Stone_Age`). Архив страниц:

Парсер реализован в виде функции `parse`, которая принимает на вход один параметр: `path_to_file` — путь до файла, содержащий html код страницы википедии. Гарантируется, что такой путь существует. Ваша задача — прочитать файл, пройтись **Beautiful Soup** по статье, найти её тело (это `<div id="bodyContent">`) и внутри него подсчитать:

1. Количество картинок (`img`) с шириной (`width`) не меньше 200. Например: ``, но не `` и не ``
2. Количество заголовков (`h1`, `h2`, `h3`, `h4`, `h5`, `h6`), первая буква текста внутри которых соответствует заглавной букве E, T или C. Например: `<h1>End</h1>` или `<h5>Contents</h5>`, но не `<h1>About</h1>` и не `<h2>end</h2>` и не `<h3>1End</h3>`
3. Длину максимальной последовательности ссылок, между которыми нет других тегов, открывающихся или закрывающихся. Например: `<p><a>, <a>, <a></p>` - тут 2 ссылки подряд, т.к. закрывающийся `span` прерывает последовательность. `<p><a>, <a>, <a></p>` - а тут 3 ссылки подряд, т.к. `span` находится внутри ссылки, а не между ссылками.

4. Количество списков (ul, ol), не вложенных в другие списки. Например: ``, `` - два не вложенных списка (и один вложенный)

Результатом работы функции parse будет **список четырех чисел**, посчитанных по формулам выше. В качестве шаблона для вашего решения **используйте следующий код**:

```
from bs4 import BeautifulSoup
import unittest

def parse(path_to_file):
    # Поместите ваш код здесь.
    # ВАЖНО!!!
    # При открытии файла, добавьте в функцию open необязательный параметр
    # encoding='utf-8', его отсутствие в коде будет вызывать падение вашего
    # решения на грейдере с ошибкой UnicodeDecodeError
    return [imgs, headers, linkslens, lists]

class TestParse(unittest.TestCase):
    def test_parse(self):
        test_cases = (
            ('wiki/Stone_Age', [13, 10, 12, 40]),
            ('wiki/Brain', [19, 5, 25, 11]),
            ('wiki/Artificial_intelligence', [8, 19, 13, 198]),
            ('wiki/Python_(programming_language)', [2, 5, 17, 41]),
            ('wiki/Spectrogram', [1, 2, 4, 7]),)

        for path, expected in test_cases:
            with self.subTest(path=path, expected=expected):
                self.assertEqual(parse(path), expected)

if __name__ == '__main__':
    unittest.main()
```

В пункте про последовательность ссылок вы можете ошибиться с результатом, если решите использовать метод `find_next()`. Обратите внимание, что хотя `find_next` находит тег, идущий сразу за текущим, этот тег может оказаться вложенным в текущий, а не быть его следующим соседом. Возможно, нужно использовать другой метод или алгоритм.

Так же, не упустите момент, что данные во всех пунктах нужно искать внутри `<div id="bodyContent">`, а не по всей странице.

Пример работы функции parse:

```
>>> parse("wiki/Stone_Age")
>>> [13, 10, 12, 40]
```

Несколько других примеров работы вы можете посмотреть в тестах из шаблона кода выше.

Во время проверки на сервере будут доступны только стандартные модули и bs4, сеть не доступна. Ваше решение будет проверяться, как на наборе страниц из приложенного архива, так и на дополнительном наборе страниц википедии.

"Важное замечание! Не используйте для сбора статистики по странице регулярные выражения. BeautifulSoup в своей работе использует специализированные парсеры, которые позволяют корректно обрабатывать невалидные (содержащие ошибки, например: незакрытый тег) html страницы. Статистика, собранная с помощью модуля re, на таких страницах будет возвращать не верное значение."

Часть 2:

В этом задании продолжаем работать со страницами из wikipedia. Необходимо реализовать механизм сбора статистики по нескольким страницам. Сложность задачи состоит в том, что сначала нужно будет определить страницы, с которых необходимо собирать статистику. В качестве входных данных служат названия двух статей(страниц). Гарантируется, что файлы обеих статей есть в папке **wiki** и из первой статьи можно попасть в последнюю, переходя по ссылкам только на те статьи, копии которых есть в папке **wiki**.

Например, на вход подаются страницы: **Stone_Age** и **Python_(programming language)**. В статье **Stone_Age** есть ссылка на **Brain**, в ней на **Artificial_intelligence**, а в ней на **Python_(programming language)** и это кратчайший путь от **Stone_Age** до **Python_(programming language)**. Ваша задача — **найти самый короткий путь** (гарантируется, что существует только один путь минимальной длины), а затем с помощью функции **parse** из предыдущего задания собрать статистику по всем статьям в найденном пути.

Результат нужно вернуть в виде словаря, ключами которого являются имена статей, а значениями списки со статистикой. Для нашего примера правильный результат будет:

```
{ 'Stone_Age': [13, 10, 12, 40],
  'Brain': [19, 5, 25, 11],
  'Artificial_intelligence': [8, 19, 13, 198],
  'Python_(programming language)': [2, 5, 17, 41]
}
```

Вам необходимо реализовать две функции: **build_bridge** и **get_statistics**

```
def build_bridge(path, start_page, end_page):
    """возвращает список страниц, по которым можно перейти по ссылкам со start_page на
    end_page, начальная и конечная страницы включаются в результирующий список
    """

    # напишите вашу реализацию логики по вычисления кратчайшего пути здесь

def get_statistics(path, start_page, end_page):
    """собирает статистику со страниц, возвращает словарь, где ключ - название
    страницы,
    значение - список со статистикой страницы"""
```

```

# получаем список страниц, с которых необходимо собрать статистику
pages = build_bridge(path, start_page, end_page)
# напишите вашу реализацию логики по сбору статистики здесь

return statistic

```

Обе функции принимают на вход три параметра: path - путь до директории с сохраненными файлами из wikipedia, start_page - название начальной страницы, end_page - название конечной страницы.

Функция build_bridge вычисляет кратчайший путь и возвращает список страниц в том порядке, в котором происходят переходы. Начальная и конечная страницы включаются в результирующий список. В случае, если название стартовой и конечной страницы совпадают, то результирующий список должен содержать только стартовую страницу. Получить все ссылки на странице можно разными способами, в том числе и с помощью регулярных выражений, например так:

```

with open(os.path.join(path, page), encoding="utf-8") as file:
    links = re.findall(r"(?<=/wiki/)[\w()]+", file.read())

```

Обратите внимание, что что на страницах wikipedia могут встречаться ссылки на страницы, которых нет в директории wiki, такие ссылки должны игнорироваться.

Пример работы функции build_bridge:

```

>>> result = build_bridge('wiki/', 'The_New_York_Times', 'Stone_Age')
>>> print(result)
['The_New_York_Times', 'London', 'Woolwich', 'Iron_Age', 'Stone_Age']

```

Функция get_statistics использует функцию parse и собирает статистику по страницам, найденным с помощью функции build_bridge. Пример работы функции get_statistics:

```

>>> from pprint import pprint
>>> result = get_statistics('wiki/', 'The_New_York_Times', "Binyamina_train_station_suicide_bombing")
>>> pprint(result)
{'Binyamina_train_station_suicide_bombing': [1, 3, 6, 21],
 'Haifa_bus_16_suicide_bombing': [1, 4, 15, 23],
 'Second_Intifada': [9, 13, 14, 84],
 'The_New_York_Times': [5, 9, 8, 42]}

```

Вы можете использовать для проверки вашего решения тесты:

```

# Набор тестов для проверки студентами решений по заданию "Практическое задание
# по Beautiful Soup - 2". По умолчанию файл с решением называется solution.py,
# измените в импорте название модуля solution, если файл с решением имеет другое имя.

```

```
import unittest
```

```
from solution import build_bridge, get_statistics
```



```

STATISTICS = {
    'Artificial_intelligence': [8, 19, 13, 198],
    'Binyamina_train_station_suicide_bombing': [1, 3, 6, 21],
    'Brain': [19, 5, 25, 11],
    'Haifa_bus_16_suicide_bombing': [1, 4, 15, 23],
    'Hidamari_no_Ki': [1, 5, 5, 35],
    'IBM': [13, 3, 21, 33],
    'Iron_Age': [4, 8, 15, 22],
    'London': [53, 16, 31, 125],
    'Mei_Kurokawa': [1, 1, 2, 7],
    'PlayStation_3': [13, 5, 14, 148],
    'Python_(programming_language)': [2, 5, 17, 41],
    'Second_Intifada': [9, 13, 14, 84],
    'Stone_Age': [13, 10, 12, 40],
    'The_New_York_Times': [5, 9, 8, 42],
    'Wild_Arms_(video_game)': [3, 3, 10, 27],
    'Woolwich': [15, 9, 19, 38]}

TESTCASES = (
    ('wiki/', 'Stone_Age', 'Python_(programming_language)',
     ['Stone_Age', 'Brain', 'Artificial_intelligence', 'Python_(programming_la
     nguage)']),

    ('wiki/', 'The_New_York_Times', 'Stone_Age',
     ['The_New_York_Times', 'London', 'Woolwich', 'Iron_Age', 'Stone_Age']),

    ('wiki/', 'Artificial_intelligence', 'Mei_Kurokawa',
     ['Artificial_intelligence', 'IBM', 'PlayStation_3', 'Wild_Arms_(video_gam
     e)',
      'Hidamari_no_Ki', 'Mei_Kurokawa']),

    ('wiki/', 'The_New_York_Times', "Binyamina_train_station_suicide_bombing",
     ['The_New_York_Times', 'Second_Intifada', 'Haifa_bus_16_suicide_bombing',
      'Binyamina_train_station_suicide_bombing']),

    ('wiki/', 'Stone_Age', 'Stone_Age',
     ['Stone_Age', ]),
)

class TestBuildBrige(unittest.TestCase):
    def test_build_bridge(self):
        for path, start_page, end_page, expected in TESTCASES:
            with self.subTest(path=path,
                               start_page=start_page,
                               end_page=end_page,
                               expected=expected):
                result = build_bridge(path, start_page, end_page)
                self.assertEqual(result, expected)

class TestGetStatistics(unittest.TestCase):
    def test_build_bridge(self):
        for path, start_page, end_page, expected in TESTCASES:
            with self.subTest(path=path,
                               start_page=start_page,
                               end_page=end_page,
                               expected=expected):
                result = build_bridge(path, start_page, end_page)
                self.assertEqual(result, expected)

```



```

        end_page=end_page,
        expected=expected):
    result = get_statistics(path, start_page, end_page)
    self.assertEqual(result, {page: STATISTICS[page] for page in e
xpected})

if __name__ == '__main__':
    unittest.main()

```

Ваше решение должно содержать реализацию функций `get_statistics`, `build_bridge` и `parse`. Вы можете дополнительно объявить в коде другие функции, если этого требует логика вашего решения.

Тема 4. Хранение данных. SQL / NoSQL

Подготовка к заданию

Скачайте архив:



Внутри архива `mysql_sample.zip` находится файл `sales.sql`, который создаст базу данных `test` с тремя таблицами и заполнит их данными.

1. Если у вас установлен и запущен `mysql 5.7` сервер и клиент, в `mysql` есть пользователь `root` с паролем `passwd`, вы можете загрузить приложенный файл командой:

```
$ mysql -uroot -ppassw < sales.sql
```

После этого можно подключиться к серверу с помощью `MySQL Workbench` или команды:

```
$ mysql -uroot -ppassw test
```

2. Если у вас установлен `docker`, то можно установить и запустить `mysql 5.7` так:

```
$ docker run --name mysql57 -p 3306:3306 --rm -
de MYSQL_ROOT_PASSWORD=passwd mysql:5.7
```

загрузить приложенный файл:

```
$ docker exec -i mysql57 mysql -uroot -ppassw < sales.sql
```

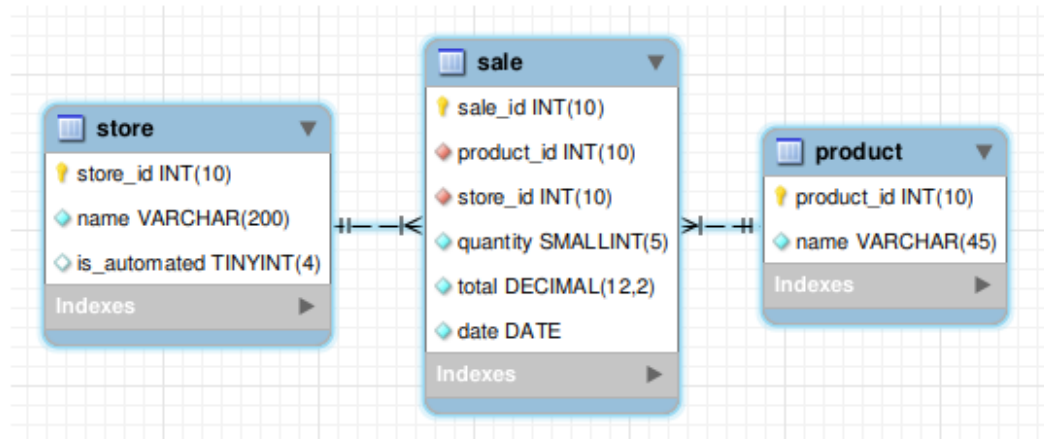
после этого можно подключиться к серверу с помощью `MySQL Workbench` или команды:

```
$ docker exec -it mysql57 mysql -uroot -ppassw test
```

Для корректного отображения русских букв надо выполнить SQL-запрос: `set names utf8`

Задание

База данных test содержит три таблицы: store(склады), product (товары) и sale(продажи). Если с данными в таблицах store, product все очевидно, то в таблице продаж sale каждая строка – это данные о продаже товара(product_id), со склада(store_id), в количестве(quantity), на общую сумму(total) и датой продажи(date).



Таблицы заполнены некоторыми тестовыми данными. Вам дается 10 заданий, в каждом необходимо написать ровно один запрос SELECT (возможно, с вложенными подзапросами), чтобы получить требуемую выборку. Далее необходимо сравнить результат, который возвращает ваш SELECT, с тем, что приводится в качестве правильного результата к заданию. Они должны совпадать.

После того как вы напишете все 10 запросов, их нужно добавить в файл results.sql. Строка каждого запроса должна располагаться строго под соответствующим комментарием (с его номером). Как будет происходить проверка решения? Тестовая система содержит аналогичную по структуре базу данных, но с другим набором тестовых данных. Запросы из файла с вашим решением будут выполнены по одному к базе данных, а результаты сверены с эталонными.

Тема 5. Веб интерфейсы с Django и Bootstrap

Часть 1

В этом задании вы научитесь формировать данные для шаблона (контекст), передавать эти параметры в шаблон и изучите базовые основы языка шаблонизации Django.

Для выполнения задания необходимо использовать шаблон проекта https://github.com/alexopryshko/coursera_assignment_tmp

В **views.py** нужно создать функцию **echo(request)**, которая принимает **request** и возвращает **HttpResponse** всегда со статусом **200**, а также она должна возвращать эхо, переданных в запросе, параметров и значение заголовка **X-Print-Statement**. View должна быть доступна по пути **/template/echo/**

View которые необходимо отредактировать находятся в **template/views.py**, **template - template/templates/echo.html**.

Решение должно быть **zip** архивом. В архиве должно содержаться два файла, которые вы редактировали ранее (обратите внимание, название файлов должно быть именно таким): **views.py**, **echo.html**

Примеры запросов и ответов:

Метод	Параметры	Заголовки	Ответ
get	GET параметры: a=1	-	get a: 1 statement is empty
get	GET параметры: c=2	-	get c: 2 statement is empty
post	POST параметры: b=1	-	post b: 1 statement is empty
post	POST параметры: d=3	-	post d: 3 statement is empty
get.post	-	X-Print-Statement = test	statement is test
get.post	-	-	statement is empty

Замечание: заголовки находятся в **META** параметрах запроса.
<https://docs.djangoproject.com/en/2.0/ref/request-response/#django.http.HttpRequest.META>

Часть 2

В этом задании вы научитесь создавать темплейт фильтры. Фильтры часто используются на практике, например, когда необходимо привести данные к конкретному виду (<https://docs.djangoproject.com/en/1.11/ref/templates/builtins/>), в частности:

- изменить формат даты
- привести строку к нижнему регистру

Для выполнения задания необходимо использовать шаблон проекта https://github.com/alexopryshko/coursera_assignment_tmp

Templatetags которые необходимо отредактировать находятся в **template/templatetags/extras.py**. Решение должно быть файлом **extras.py**.

2.1. Фильтр inc. Необходимо в файле **extras.py** создать фильтр “inc” который принимает 2 аргумента: 1-й - число которое нужно увеличить, 2-й - на сколько нужно увеличить первое число. Пример использования фильтра “inc” представлен в файле **template/templates/filters.html**

2.2. Тег division. Необходимо в файле **extras.py** создать тег “division” (то есть тег для деления), который принимает 3 параметра: 1-ый - делимое, 2-ой - делитель, 3-ий — флаг определяющий тип возвращаемого значения для результата деления (именованный аргумент to_int). Если переданное значение to_int равно False, необходимо выполнить вещественное деление. Если передано True результат вещественного деления необходимо привести к целому. Значение to_int по-умолчанию — False.

Обратите внимание, что делимое и делитель целые числа, но передаются в тег в формате string.

Пример использования тега “division” представлен в файле **template/templates/filters.html**

```
{% division a b to_int=True %}
{% division a b %}
```

Часть 3

В этом задании вы научитесь работать с наследованием шаблонов. Наследование повсеместно используется. Например, header и footer у сайта одинаковый, не стоит дублировать код header и footer на каждой странице, имеет смысл сделать базовый шаблон и наследоваться от него.

Для выполнения задания необходимо использовать шаблон проекта https://github.com/alexopryshko/coursera_assignment_tmp

Шаблон, который необходимо отредактировать **template/templates/extend.html**. Решение должно быть файлом **extend.html**.

Нужно написать шаблон которые наследуется от **template/templates/base.html** и переопределяет блоки block_a, block_b. В блоках block_a, block_b должно остаться, то

что было у родительского шаблона и дополнительно вывести параметр “a” и параметр “b”

Тема 6. Работа с данными пользователя

В этом задании вам требуется отправить POST запрос на следующий <https://datasend.webpython.graders.eldf.ru/submissions/1/>

В запросе должен содержаться заголовок *Authorization* со способом аутентификации *Basic* логином *alladin* и паролем *opensesame* закодированными в base64.

Authorization: Basic YWxsYWRpbjpvvcGVuc2VzYW11

Запрос можно отправить любым удобным для вас способом, но мы рекомендуем использовать библиотеку *requests*, так как она понадобится вам при выполнении последующих заданий.

В ответе на запрос вы получите инструкции для последующего запроса который приведет вас к специальному коду который является ответом на это задание.

Тема 7. Дополнительный инструментарий

В этом задании вы должны клонировать репозиторий, создать ветку, влить в нее существующие ветки, решить конфликты и во время решения конфликтов подправить код так, чтобы привести код из разных веток к одному стилю.

Задачи

1. Есть общедоступный [репозиторий](#), вы должны клонировать его для локальной работы.

2. В проекте три ветки, в которых одновременно шла работа над разными алгоритмами шифрования: *master*, в которой лежит шифр Цезаря, и две ветки созданные от нее на разных стадиях работы, в которых лежит вариации шифра цезаря: *rot13* и шифр Виженера. Вам нужно переключиться (*checkout*) в каждую из веток, посмотреть код и запустить тесты (*python -m unittest -v*), убедившись, что в каждой из веток тесты проходят.

3. Вам нужно вернуться в ветку *master*, создать от нее новую ветку *dev* и перейти туда.

4. В ветку *dev* вам нужно влить ветку с шифром *rot13*, с этим не должно возникнуть проблем, но не забудьте про ключ *--no-ff*. С момента ответвления от *master*, в ветках изменялись только разные файлы, поэтому конфликтов при слиянии не будет. После слияния, в *dev* должны быть шифр Цезаря и *rot13*, тесты на них, и они должны проходить.

5. Теперь в ветку *dev* нужно влить ветку с шифром Виженера, не забудьте про ключ *--no-ff*. Эта ветка была ответвлена от *master* на раннем этапе работы над программой, и в ней оказался старый вариант кода, который потом правился и в ней и *master* параллельно, но по разному, поэтому будут конфликты слияния. К тому же код в этой ветке слегка устарел, так как в *master* был существенно унифицирован код как шифров, так и тестов. Это привело к тому, что шифры Цезаря и *rot13* используют общую кодовую базу для тестов, а шифр Виженера нет. При решении конфликтов в *cipher.py* и *test.py* имейте в виду, что код, который у вас в *dev* новее и лучше, и все конфликтующие изменения нужно оставить именно из него, а из Виженера взять только классы *Vigenere* и *TestVigenere* (и то, последний придется потом переделать почти полностью).

6. Убрав все конфликтующие изменения, не спешите делать *commit*, чтобы завершить *merge*. Сначала вам нужно привести код шифра Виженера и его

теста к новой кодовой базе. Т.е. сделать их максимально похожими на код шифра Цезаря: переименовать методы шифрации и дешифрации в `encode` и `decode`, использовать `cls.get_char`, `if`, `cls.alpha` (из 26 букв) и генератор списка, а в тестах наследоваться от `TestCipherMixin`. Учтите, что теперь ключ шифрования в шифре Виженера не может содержать пробел, уберите его из строки ключа в тестах, а вот строка для шифрации, наоборот, может содержать любые символы, а не только буквы (и последняя версия шифра Цезаря прилетевшая из ветки `rot13` с этим справляется)

7. После всех исправлений завершите `merge` с помощью `commit`.

8. Последним шагом перейдите в ветку `master` и влейте в нее `dev`, не забудьте необходимые ключи `merge`. Проблем не должно возникнуть. Еще раз запустите тесты, их должно быть 6 и они должны пройти.

9. Папку с проектом целиком запакуйте в `zip`-архив и отправьте на проверку. На сервере проект будет распакован и с помощью `git` проверено, что ветки объединялись именно так, как описано. Затем будет проверено, что классы `Caesar` и `Vigenere` наследуются от `Cipher` (`Rot13` может не наследоваться), методы шифрования у всех классов называются `encode` и `decode` и действительно реализуют нужные алгоритмы шифрования, и вызывают при этом `cls.get_char`. Будет проверено, что все 6 тестов проходят, наследуются от `TestCipherMixin` и `unittest.TestCase` и определяют только методы `encode` и `decode` (могут содержать еще константы и атрибуты).

Вот для сравнения вывод `git log --graph --oneline --decorate --all` в ветке `master`, после того как проделано все описанное выше (наглядно показаны все ветки, откуда ответвились и куда влиты:

```
> git log --graph --oneline --decorate --all
* 8b892ac (HEAD -> master) Merge branch 'dev'

|
| * 13453de (dev) Merge branch 'feature/vigenere' into dev
|
| * 3c3d40c (origin/feature/vigenere, feature/vigenere) add vigenere cipher & tests
| * 7743fbd bugfix in caesar & refactor
| * 1a5e66b Merge branch 'feature/rot13' into dev
|
|
| * 7a06ceb (origin/feature/rot13, feature/rot13) Rot13 cipher and tests
| * 4d463b9 (origin/master) readme
|
|
| * 23cfb94 refactor tests
| * eac88d2 fix for tests
| * c4ee2cc bugfix, refactor, decoder function
|
|
* d278c40 caesar cipher & tests
```

Модуль 4

Тема 1.

Ответьте на вопросы.

1. Какие из следующих методов `numpy` **не создают массивы**, заполненные значениями (например массивы, состоящие из единиц или случайных чисел)?

`full()`

Все методы создают массивы с значениями

`eye()`

randint()

flatten()

2. Поменять порядок элементов в массиве на обратный и вернуть 4-й элемент нового массива $Z = \text{np.array}([92, 13, 44, 555, 1, -3])$. Напомним, что индексация массива начинается с нуля.

1

13

555

-3

3. Создайте два вектора - вектор, состоящий из 5 единиц, и другой вектор размером 5, содержащий только значение 2. Найдите скалярное произведение векторов.

[2,2,2,2,2]

10

3

нельзя найти произведение векторов из-за разной размерности

Тема 2. Визуализация данных и статистика

В качестве задания на отработку навыков по визуализации данных, мы предлагаем вам провести различные визуализации датасета с большим количеством атрибутов.

Это датасет с информацией о ~15 000 игроков из футбольного симулятора FIFA 18, особо актуальный в год проведения Чемпионата Мира в России. Ознакомиться с ним и скачать можно по ссылке с Kaggle, платформы для соревнований по машинному обучению.

<https://www.kaggle.com/thec03u5/fifa-18-demo-player-dataset>

Для выполнения этого задания, вам нужно будет сделать как минимум 4 визуализации для указанного датасета. Вы можете проявить свою фантазию или же использовать предложенные нами варианты визуализаций.

Примеры визуализации:

- Гистограмма возраста игроков
- Сравнение зависимости зарплаты от возраста игрока. Можно найти самые большие команды и сравнить эти зависимости между ними.
- Гистограмма распределения одной из статистик по странам — например, средний и максимальный Performance

Решение должно соответствовать следующим критериям:

1. Минимум две визуализации с помощью matplotlib
2. Минимум две визуализации с помощью Plotly
3. В каждой визуализации должен использоваться как минимум один атрибут, не использованный в других визуализациях
4. Решением к заданию должен быть архив, в котором лежат IPYNB файлы визуализаций. Помимо самого кода визуализации и изображения, в ноутбуке должно присутствовать текстовое описание визуализации.
5. В каждой визуализации должен быть указан заголовок и подписаны оси, если они есть
6. К каждой визуализации должен быть описан вопрос, на который она отвечает, например:
 - Гистограмма возраста укажет на доминирующий возраст и особенности распределения

– Гистограмма среднего Performance по странам определит стабильные и надежные страны, а гистограмма максимума — родины особо ярких игроков

Задача

Оцените решения сокурсников по критериям ниже

1. В решении присутствуют минимум две визуализации с помощью matplotlib
2. В решении присутствуют минимум две визуализации с помощью Plotly
3. В каждой визуализации используется как минимум один атрибут, не использованный в других визуализациях
4. К каждой визуализации описан вопрос, на который она отвечает
5. В каждой визуализации указан заголовок и подписаны оси, если они есть
6. Решением к заданию является архив, в котором лежат IPYNB файлы визуализаций. Помимо самого кода визуализации и изображения, в ноутбуке присутствует текстовое описание визуализации.

Нет – 0

Да – 1

Тема 3.

Ответьте на вопросы.

1. Выберите верные утверждения
 - 1) Решающие деревья склонны к переобучению
 - 2) Результатом классификации нового объекта в решающем дереве является класс, содержащийся в листе этого дерева
 - 3) Решающие деревья не используются в качестве регрессионных моделей
2. Каким образом может достигаться различие между решающими деревьями в алгоритме случайного леса?
 - 1) Использованием случайного подмножества признаков при разбиении объектов в вершинах решающего дерева
 - 2) Выбором случайного подмножества объектов при обучении решающего дерева
 - 3) Всем из вышеперечисленного
3. Какие из пунктов описывают достоинства случайного леса?
 - 1) Существуют методы оценки предсказательной способности признаков в модели
 - 2) Случайный лес не склонен к переобучению при увеличении числа базовых алгоритмов
 - 3) В случайном лесе обычно используются глубокие решающие деревья
 - 4) В этой модели решающие деревья строятся независимо, что делает возможным использование методов параллельных вычислений

Тема 4.

Ответьте на вопросы.

1. В каком из следующих сценариев может пригодиться рекомендательная система?
 - 5) Оценка вероятности возврата кредита по анкетным данным
 - 6) Рекомендация прически в парикмахерской
 - 7) Составление списка музыкальных композиций, удовлетворяющих вкусам пользователя
 - 8) Составление списка статей, удовлетворяющим интересам пользователя
2. В методах коллаборативной фильтрации
 - 1) Для расчета похожести между пользователями/товарами пропуски надо заменить на "0"
 - 2) Заложена устойчивость к проблеме "холодного старта"
 - 3) предполагается что оценки похожих пользователей/товаров будут похожими
 - 4) Предполагается только 5-и балльная система оценок
3. Выберите верные утверждения
 - 1) Latent Factor Model обучается на всевозможных парах пользователь-товар
 - 2) Размерность векторов p и q в Latent Factor Model должна быть одинаковой
 - 3) Так как Latent Factor Model обучается с помощью стохастического градиентного спуска, то для двух разных запусков могут обучиться разные p и q
 - 4) Перед применением SVD надо избавиться от пропусков в матрице рейтингов

Тема 5.

Ответьте на вопросы.

1. Предположим, что на вход сверточного слоя поступает цветное изображение размером 32×32 пикселей. Количество каналов для каждого изображения в данном случае равно 3. Мы используем фильтры с ядром свертки 5×5 . Какое количество параметров (настраиваемых весов) содержится в одном фильтре без учета bias параметра?
 - 1) 102
 - 2) 3072
 - 3) 75
 - 4) 675
2. Каким образом качество восстанавливаемых данных зависит от размерности скрытого пространства в модели автоэнкодера при обучении на одной и той же выборке? Мы считаем каждую модель достаточно точной по отношению к величине ошибки восстановления
 - 1) При увеличении размерности скрытого пространства качество восстанавливаемых данных увеличивается
 - 2) При уменьшении размерности скрытого пространства качество восстанавливаемых данных увеличивается

3) Качество восстанавливаемых данных не зависит от размерности скрытого пространства

3. В чем заключается основная идея архитектуры ResNet?

1) Использование Batch normalization после сверточного слоя нейронной сети

2) Дублирование сверточных слоев по отношению к количеству каналов

3) Добавление дополнительного соединения, которое обходит цепочку преобразований и создает дополнительный поток данных, который складывается с основным

Проект

Проект предполагает очную защиту перед экзаменационной комиссией. Перед комиссией студент готовит материалы согласно заданию и направляет экзаменационной комиссии не менее чем за 2 дня до назначенной даты экзамена

В данном задании необходимо будет закончить разработку полноценной ролевой игры «Рыцарь в подземелье». В данной игре необходимо будет играть за рыцаря, который путешествует по многоэтажному подземелью, борется с врагами и собирает сокровища.

Вам будет дан код движка игры, текстуры и игровая логика. Некоторые из классов и методов изначально не реализованы в предоставленном коде. Их и необходимо будет реализовать в данном задании. В помощь вам будут даны полные диаграммы классов, которые должны быть представлены в проекте.

Кроме реализации новых методов, необходимо будет дописать недостающий код в некоторые из существующих функций и методов. Места, в которых требуется исправление, помечены меткой **#FIXME**.

Кроме основных заданий, Вам будут предложены дополнительные задания. В них вам нужно будет немного усовершенствовать игру. При выполнении дополнительных заданий старайтесь быть креативными.

В качестве ответа вам необходимо будет прикрепить архив, содержащий всю структуру папок Вашего проекта.

Проект

Проект предполагает очную защиту перед экзаменационной комиссией. Перед комиссией студент готовит материалы согласно заданию и направляет экзаменационной комиссии не менее чем за 2 дня до назначенной даты экзамена

Вашей задачей будет реализовать на Django сервер управления умным домом, имеющий web-интерфейс для настройки и ручного управления, который будет производить периодический опрос датчиков и осуществлять автоматическую реакцию в случае определенных ситуаций, используя API контроллера умного дома

Авторы курса сделали виртуальные контроллер, датчики и устройства, которыми он управляет. Зайдя на сайт и зарегистрировавшись, вы получите уникальный ключ (KEY), который нужно будет использовать при работе с API контроллера. Используя этот же ключ можно получить изображение с “виртуальной камеры” умного дома, на которой будет наглядно видно, какие устройства работают или почему сработал тот или иной датчик, и вручную управлять устройствами. Документацию по API можно посмотреть на этом же сайте.



Устройства, подключенные к контроллеру, доступны на запись (обычно true - включить/открыть, false - выключить/закрыть, но бывают варианты). И датчики и устройства доступны на чтение. Устройства, при чтении с них, работают как датчики и возвращают свое состояние, которое может отличаться от записанного.

Устройства (запись):

air_conditioner – Кондиционер (true – вкл, false – выкл). При включении постепенно понижает температуру в спальне, пока она не достигнет 16 градусов и сильнее охладить уже не может.

bedroom_light – Лампа в спальне (true – вкл, false – выкл).

bathroom_light – Лампа в ванной (true – вкл, false – выкл).

curtains – Занавески string (“open” – открыть, “close” – закрыть).

boiler – Бойлер (true – вкл, false – выкл). При включении постепенно повышает температуру воды, пока она не достигнет 90 градусов. Для работы должен быть открыт входной кран холодной воды.

cold_water – Входной кран холодной воды (true – открыть, false – закрыть).
Позволяет открыть/перекрыть подачу холодной воды в квартиру

hot_water – Входной кран горячей воды (true – открыть, false – закрыть).

washing_machine – Стиральная машина string (“on” – вкл, “off” – выкл). При включении начинает стирать, потом самостоятельно отключается. Может сломаться и протечь.

Датчики (чтение):

air_conditioner – Кондиционер. (true – вкл, false – выкл).

bedroom_temperature – Температура в спальне. Int (0 – 80).

bedroom_light – Лампа в спальне. (true – вкл, false – выкл).

smoke_detector – Датчик задымления на потолке. (true – задымление, false – нет).

bedroom_presence – Датчик присутствия в спальне. (true – есть человек, false – нет).

bedroom_motion – Датчик движения в спальне. (true – есть движение, false – нет).

curtains – Занавески. string (“open” – открыты, “close” – закрыты, “slightly_open” – приоткрыты вручную).

outdoor_light – Датчик освещенности за окном (0 – 100).

boiler – Бойлер. (true – вкл, false – выкл).

boiler_temperature – Температура горячей воды бойлере. Int (0 – 100 / null). Если

перекрыта холодная вода, то воды в бойлере нет, и датчик возвращает null.

cold_water – Входной кран холодной воды. (true – открыт, false – закрыт).

hot_water – Входной кран горячей воды. (true – открыт, false – закрыт).

bathroom_light – Лампа в ванной. (true – вкл, false – выкл).

bathroom_presence – Датчик присутствия в ванной. (true – есть человек, false – нет).

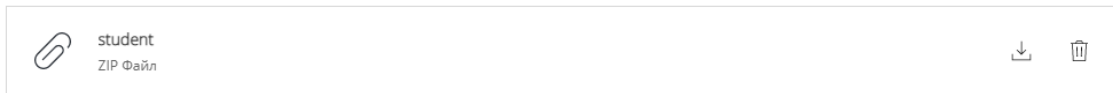
bathroom_motion – Датчик движения в ванной. (true – есть движение, false – нет)

washing_machine – Стиральная машина. string (“on” – вкл, “off” – выкл, “broken” – сломана).

leak_detector – Датчик протечки воды (true – протечка, false – сухо).

Приложение студента

Используя приложенный к этому заданию “скелет” приложения на Django (файл student.zip), вам нужно реализовать свой интерфейс управления умным домом.



По адресу / должна открываться веб-форма, со следующими контролами:

bedroom_target_temperature – input type=number, желаемая температура в спальне, запоминать настройку в базе, текущую настройку из базы выводить на форму. Допустимое значение от 16 до 50, default: 21

hot_water_target_temperature – input type=number, желаемая температура горячей воды в доме, запоминать настройку в базе, текущую настройку из базы выводить на форму. Допустимое значение от 24 до 90, default: 80

bedroom_light – checkbox, включает/выключает свет в спальне, синхронизировать значение с контроллером

bathroom_light – checkbox, включает/выключает свет в ванной, синхронизировать значение с контроллером

Там же отображать текущие значения всех датчиков, прочитанные из контроллера. Для рендеринга шаблона прочитанные из контроллера значения должны быть в словаре в context.data.

Реализовать автоматически опрос контроллера в фоне каждую секунду (django celery) и реакцию на некоторые события.

Реакция на события:

Если есть протечка воды (leak_detector=true), закрыть холодную (cold_water=false) и горячую (hot_water=false) воду и отослать письмо в момент обнаружения.

Если холодная вода (cold_water) закрыта, немедленно выключить бойлер (boiler) и стиральную машину (washing_machine) и ни при каких условиях не включать их, пока холодная вода не будет снова открыта.

Если горячая вода имеет температуру (boiler_temperature) меньше чем hot_water_target_temperature - 10%, нужно включить бойлер (boiler), и ждать пока она не достигнет температуры hot_water_target_temperature + 10%, после чего в целях экономии энергии бойлер нужно отключить

Если шторы частично открыты (curtains == “slightly_open”), то они находятся на ручном управлении - это значит их состояние нельзя изменять автоматически ни при каких условиях.

Если на улице (outdoor_light) темнее 50, открыть шторы (curtains), но только если не горит лампа в спальне (bedroom_light). Если на улице (outdoor_light) светлее 50, или горит свет в спальне (bedroom_light), закрыть шторы. Кроме случаев когда они на

ручном управлении

Если обнаружен дым (`smoke_detector`), немедленно выключить следующие приборы [`air_conditioner`, `bedroom_light`, `bathroom_light`, `boiler`, `washing_machine`], и ни при каких условиях не включать их, пока дым не исчезнет.

Если температура в спальне (`bedroom_temperature`) поднялась выше `bedroom_target_temperature + 10%` - включить кондиционер (`air_conditioner`), и ждать пока температура не опустится ниже `bedroom_target_temperature - 10%`, после чего кондиционер отключить.

Опрос контроллера и отправка ему ответа должны происходить внутри функции `core.tasks.smart_home_manager`. Эта функция должна вызываться периодически с интервалом в 5 секунд, например, с помощью `celery`. В начале своей работы функция запрашивает данные из контроллера, используя `requests.get` в API, затем анализирует настройки пользователя по желаемой температуре из БД, и текущую ситуацию, и в конце, если требуется коррекция, делает `requests.post` в API с командами для контроллера, если необходимо отправить письмо, то отправляет его.

Для отсылки писем нужно использовать `send_mail` из `django.core.mail`, а в `settings` нужно задать настройки `EMAIL_HOST`, `EMAIL_PORT` и другие `EMAIL_*`, так чтобы во время разработки вы отправляли письма через какую-нибудь почтовую систему и могли проверить их работу. Во время проверки задания на сервере эти настройки будут переопределены.

Для сохранения настроек в базе, нужно использовать модель `Settings` (`name`, `value`), заготовка для которой уже есть в приложенных исходниках.

Веб-форма для настройки и управления умным домом должна открываться в корне сайта, содержать 4 `input`'а с именами (`name=...`): `bedroom_target_temperature`, `hot_water_target_temperature`, `bedroom_light`, `bathroom_light`.

В `settings` нужно добавить переменные `SMART_HOME_API_URL` и `SMART_HOME_ACCESS_TOKEN` и задать их значения, затем использовать их для взаимодействия с умным домом. Еще можно добавить `EMAIL_RECEPIENT`, в котором задать получателя писем от системы.

В приложенных исходниках есть несколько тестов `django tests`, которые тестируют некоторые базовые вещи. После того, как реализуете функционал, обязательно запустите `manage.py test`. На бою мы проверяем задание аналогичным образом через тесты контроллера по урлу `/`, и ручной вызов `core.tasks.smart_home_manager`.

Что будем проверять

Во время разработки вы можете в реальном времени наблюдать на сайте за реакцией умного дома на управляющие воздействия вашего приложения, а также руками выставлять показания датчиков для моделирования различных ситуаций.

Во время проверки на сервере интернет будет недоступен, `requests.get` будет заменен на `mock`, который будет возвращать приложению разные состояния контроллера (показания датчиков), а тесты будут проверять результат работы в `requests.post` и факт отправки письма, если оно требуется. Также будет проверяться работа веб-формы. Например если будет передана низкая температура горячей воды, в ответ будет ожидаться команда на включение бойлера, если он выключен. Но если к предыдущему примеру добавить еще и обнаруженную протечку, или пожар, то команда на включение бойлера подаваться не должна, а наоборот бойлеру должна подаваться команда на выключение, если он включен. Все возможные пограничные ситуации описаны в разделе “Реакция на события”. В случае если внешний сервер вернул ошибку или не отвечает нужно вернуть страницу с ошибкой со статус кодом 502.

Вам необходимо реализовать весь проект основываясь на скелете приложенном

к этому заданию. Весь код должен находиться в приложении **core**. После реализации **заархивируйте содержимое папки core в zip архив и отправьте на проверку.**

Реализовывать валидацию данных можно с использованием библиотек

marshmallow

jsonschema

или с помощью класса Forms. Также в проекте установлен фреймворк для тестирования py.test <https://docs.pytest.org/en/latest/>

Проект

Проект предполагает очную защиту перед экзаменационной комиссией. Перед комиссией студент готовит материалы согласно заданию и направляет экзаменационной комиссии не менее чем за 2 дня до назначенной даты экзамена

В качестве итогового проекта вам предлагается решить задачу классификации на наборе данных Fashion-MNIST, который вы можете скачать по следующей ссылке

Fashion-MNIST

Данные проект является учебным и его цель – научиться работать с различными моделями машинного обучения. Вам необязательно строить самую лучшую модель и получать лучшее качество на тестовых данных. Самое важное – это научиться обучать модели и анализировать полученные результаты. В результате, вы научитесь использовать и подготавливать данные для решения задачи классификации. Так же вы изучите на практике различные алгоритмы машинного обучения, такие как логистическая регрессия, полносвязные нейронные сети и сверточные нейронные сети. Более того, вы научитесь обучать эти модели и анализировать результаты работы этих алгоритмов на новых данных

Fashion-MNIST – это датасет состоящий из 70000 черно-белых изображений одежды 28x28 пикселей каждое. 60000 из них содержатся в тренировочной выборке, и 10000 – в тестовой. Этот набор данных представляет из себя альтернативу обычному датасету рукописных цифр MNIST. Существовало несколько предпосылок для создания такого набора данных.

Большинство исследователей в области машинного и глубинного обучения используют MNIST при первой проверки своих моделей, что не всегда является хорошей идеей. Если какая-то модель не работает на датасете MNIST – это не значит, что она не работает в принципе. Если какая-то модель прекрасно работает на датасете MNIST – это не означает, что она так же хорошо будет работать на других датасетах. Известны случаи в отказе публикации некоторых статей из-за недостаточно хорошего качества модели на MNIST. Поэтому Zalando Research предложили альтернативу.

В Fashion-MNIST содержится 10 классов разной одежды по аналогии с 10 цифрами из MNIST.

В зачетно-экзаменационную ведомость оценка выставляется в соответствии с нижеприведенной таблицей 1.

Таблица 1

Сумма баллов	Оценка
50-100	Зачтено
менее 50	Не зачтено

Составляющие процесса обучения, которые оцениваются в ходе обучения, и их вклад в итоговую оценку представлены в таблице 2.

Таблица 2

№ п/п	Основные показатели оценки	Вклад в итоговую оценку
1	Практические занятия	30%
2	Выполнение домашних заданий	40%
3	Промежуточная аттестация	30%

3.2. Оценочные материалы

Таблица 3

Наименование модуля	Формы и методы контроля и оценки	Вес задания
Основы программирования на Python	Практические задания по темам лекций Домашние задания в системе дистанционного обучения по темам лекций	14
Структуры данных и функции	Практические задания по темам лекций Домашние задания в системе дистанционного обучения по темам лекций	14
Объектно-ориентированное программирование	Практические задания по темам лекций Домашние задания в системе дистанционного обучения по темам лекций	14
Углубленный Python	Практические задания по темам лекций Домашние задания в системе дистанционного обучения по темам лекций	14
Многопоточное и асинхронное программирование	Практические задания по темам лекций Домашние задания в системе дистанционного обучения по темам лекций	14
Промежуточная аттестация	Итоговое тестирование	30