**31.8 Практическая работа**

Цели практической работы

* Закрепить понятие «регулярные выражения», отработать умение писать их и создавать шаблоны.
* Отработать использование методов регулярных выражений.
* Отработать умение получать данные с сайта и API.
* Закрепить работу с форматом JSON, использованием методов для сериализации и десериализации.

Что входит в работу

* Задача 1. Lorem ipsum.
* Задача 2. Регистрационные знаки.
* Задача 3. Во все тяжкие.
* Задача 4. Телефонные номера.
* Задача 5. Web scraping.
* Задача 6. Поиск разницы между двумя JSON-файлами (пример из реального тестового задания на должность Python-разработчика уровня Junior).

Задача 1. Lorem ipsum

Что нужно сделать

Для макетов веб-страниц часто используется какой-нибудь текст-рыба — это условный, нередко бессмысленный текст-заполнитель. Пусть дан следующий сгенерированный текст:

text = """ Lorem ipsum dolor sit amet, consectetuer adipiscing elit.   
Aenean commodo ligula eget dolor. Aenean massa. Cum sociis natoque penatibus et magnis dis parturient montes,  
nascetur ridiculus mus. Donec quam felis, ultricies nec, pellentesque eu, pretium quis, sem.   
Nulla consequat massa quis enim. Donec pede justo, fringilla vel, aliquet nec, vulputate   
"""

Напишите программу, которая обрабатывает этот текст и выводит список слов, состоящих ровно из четырёх букв.

**Результат:**  
['amet', 'elit', 'eget', 'quam', 'quis', 'quis', 'enim', 'pede']

Что оценивается

* Результат вычислений корректен.
* Формат вывода соответствует примеру.
* Решение опирается на использование регулярных выражений и их методов.
* Переменные, функции и собственные методы классов имеют значащие имена (не a, b, c, d).

Задача 2. Регистрационные знаки

Что нужно сделать

В России для транспорта применяются регистрационные знаки нескольких видов.

Общее в них то, что они состоят из цифр и букв. Причём используются только 12 букв кириллицы, имеющих графические аналоги в латинском алфавите: А, В, Е, К, М, Н, О, Р, С, Т, У и Х.

У частных легковых автомобилей номера — это буква, три цифры, две буквы, затем две или три цифры с кодом региона.

У такси — две буквы, три цифры, затем две или три цифры с кодом региона.

Напишите программу, которая в перечне номеров находит номера частных автомобилей и номера такси.

**Пример перечня:**

'А578ВЕ777 ОР233787 К901МН666 СТ46599 СНИ2929П777 666АМР666'

**Результат:**  
Список номеров частных автомобилей: ['А578ВЕ777', 'К901МН666']  
Список номеров такси: ['ОР233787', 'СТ46599']

Что оценивается

* Результат вычислений корректен.
* Формат вывода соответствует примеру.
* Решение опирается на использование регулярных выражений и их методов.
* Переменные, функции и собственные методы классов имеют значащие имена (не a, b, c, d).

Задача 3. May the force be with you

Что нужно сделать

Фанаты «Звёздных войн» (Star Wars) написали API по своей любимой вселенной. Ссылка на документацию: [Documentation](https://swapi.dev/documentation" \t "_blank).

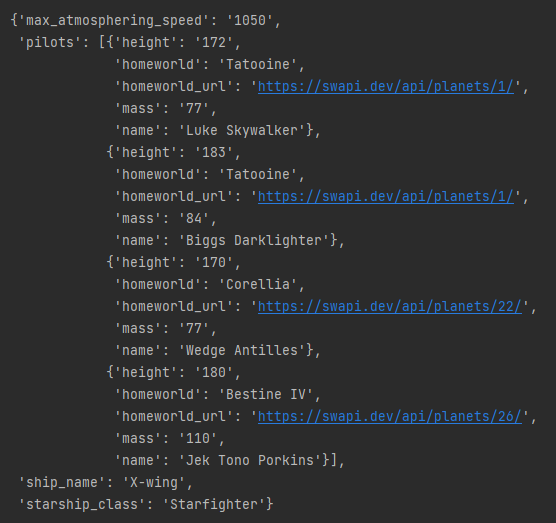
Внимательно изучите документацию этого API и напишите программу, которая выводит на экран (и в JSON-файл) информацию о пилотах легендарного корабля Millennium Falcon.

Информация о корабле должна содержать следующие пункты:

* название,
* максимальная скорость,
* класс,
* список пилотов.

Внутри списка о каждом пилоте должна быть следующая информация:

* имя,
* рост,
* вес,
* родная планета,
* ссылка на информацию о родной планете.

Пример вывода информации по кораблю X-wing в консоль:  


Информация в файле будет выглядеть так:



Что оценивается

* Результат вычислений корректен.
* Информация собрана в словарь с правильной структурой.
* Переменные, функции и собственные методы классов имеют значащие имена (не a, b, c, d).

Задача 4. Телефонные номера

Что нужно сделать

В одной организации перед записью телефонного номера в базу данных его проверяют на соответствие следующим критериям:

* Длина номера — ровно десять знаков.
* Номер начинается с цифры 8 или 9.
* Остальные знаки — только цифры.

На вход в программу подаётся список номеров (можно взять готовый или запросить у пользователя). Реализуйте код, который проверяет каждый номер из списка на соответствие критериям и выводит на экран соответствующие сообщения.

**Пример списка:**  
['9999999999', '999999-999', '99999x9999']

**Результат:**  
первый номер: всё в порядке  
второй номер: не подходит  
третий номер: не подходит

Что оценивается

* Результат вычислений корректен.
* Формат вывода соответствует примеру.
* Решение опирается на использование регулярных выражений и их методов.
* Переменные, функции и собственные методы классов имеют значащие имена (не a, b, c, d).

Задача 5. Web scraping

Что нужно сделать

Дан несложный пример HTML-страницы: [Sample Web Page.](http://www.columbia.edu/~fdc/sample.html" \t "_blank)

Изучите код этой страницы и реализуйте программу, которая получает список всех подзаголовков сайта (они заключены в теги <h3>).

Ожидаемый результат:

['CONTENTS', '1. Creating a Web Page', '2. HTML Syntax', '3. Special Characters', '4. Converting Plain Text to HTML', '5. Effects', '6. Lists', '7. Links', '8. Tables', '9. Viewing Your Web Page', '10. Installing Your Web Page on the Internet', '11. Where to go from here', '12. Postscript: Cell Phones']

Сделайте так, чтобы программа работала для любого сайта, где есть такие теги.

Что оценивается

* Результат вычислений корректен.
* Формат вывода соответствует примеру.
* Решение опирается на использование регулярных выражений и их методов.
* Переменные, функции и собственные методы классов имеют значащие имена (не a, b, c, d).

Задача 6. Поиск разницы между двумя JSON-файлами (пример из реального тестового задания на должность Python-разработчика уровня Junior)

Что нужно сделать

Найдите различия между двумя JSON-файлами. Если различающиеся параметры входят в diff\_list, выведите различие. Иными словами, вам нужно отловить изменение определённых параметров и вывести значение: что изменилось и на что. Набор ключей в обоих файлах идентичный, различаются лишь значения.

Напишите программу, которая:

* загружает данные из двух предложенных JSON-файлов (находятся в репозитории);
* выполняет сравнение параметров, указанных в diff\_list;
* формирует результат в виде словаря;
* записывает словарь в JSON-файл с названием result.json.

Исходные данные

Файлы:

* json\_old.json,
* json\_new.json.

Список параметров для отслеживания (можно сформировать инпутом или ввести вручную):

diff\_list = [‘services’, ‘staff’, ‘datetime’]

Формат итогового словаря с результатом:

Словарь {параметр: новое\_значение, ….}

**Пример**

**Данные, загруженные из json\_old.json:**

{'company\_id': 111111, 'resource': 'record', 'resource\_id': 406155061, 'status': 'create', 'data': {'id': 11111111, 'company\_id': 111111, 'services': [{'id': 9035445, 'title': 'Стрижка', 'cost': 1500, 'cost\_per\_unit': 1500, 'first\_cost': 1500, 'amount': 1}], 'goods\_transactions': [], 'staff': {'id': 1819441, 'name': 'Мастер'}, 'client': {'id': 130345867, 'name': 'Клиент', 'phone': '79111111111', 'success\_visits\_count': 2, 'fail\_visits\_count': 0}, 'clients\_count': 1, 'datetime': '2022-01-25T11:00:00+03:00', 'create\_date': '2022-01-22T00:54:00+03:00', 'online': False, 'attendance': 0, 'confirmed': 1, 'seance\_length': 3600, 'length': 3600, 'master\_request': 1, 'visit\_id': 346427049, 'created\_user\_id': 10573443, 'deleted': False, 'paid\_full': 0, 'last\_change\_date': '2022-01-22T00:54:00+03:00', 'record\_labels': '', 'date': '2022-01-22 10:00:00'}}

**Данные, загруженные из json\_new.json:**

{'company\_id': 111111, 'resource': 'record', 'resource\_id': 406155061, 'status': 'create', 'data': {'id': 11111111, 'company\_id': 111111, 'services': [{'id': 22222225, 'title': 'Стрижка', 'cost': 1500, 'cost\_per\_unit': 1500, 'first\_cost': 1500, 'amount': 1}], 'goods\_transactions': [], 'staff': {'id': 1819441, 'name': 'Мастер'}, 'client': {'id': 130345867, 'name': 'Клиент', 'phone': '79111111111', 'success\_visits\_count': 2, 'fail\_visits\_count': 0}, 'clients\_count': 1, 'datetime': '2022-01-25T13:00:00+03:00', 'create\_date': '2022-01-22T00:54:00+03:00', 'online': False, 'attendance': 2, 'confirmed': 1, 'seance\_length': 3600, 'length': 3600, 'master\_request': 1, 'visit\_id': 346427049, 'created\_user\_id': 10573443, 'deleted': False, 'paid\_full': 1, 'last\_change\_date': '2022-01-22T00:54:00+03:00', 'record\_labels': '', 'date': '2022-01-22 10:00:00'}}

diff\_list = [‘services’, ‘staff’, ‘datetime’]

**Результат**

print(result)

В консоли должно вывестись следующее сообщение:

{'services': [{'id': 22222225, 'title': 'Стрижка', 'cost': 1500, 'cost\_per\_unit': 1500, 'first\_cost': 1500, 'amount': 1}], 'datetime': '2022-01-25T13:00:00+03:00'}

Помимо вывода в консоль, должен быть сформирован JSON-файл с получившимся словарём (result.json).

**Обратите внимание:** в result представлены не все изменившиеся поля, а лишь те, что объявлены в diff\_list.

Что оценивается

* Данные корректно загружены из файлов.
* Формат вывода соответствует примеру.
* Выведены только те различия, которые указаны в diff\_list.
* Выполнение кода приводит к созданию файла result.json с найденными различиями.
* Переменные, функции и собственные методы классов имеют значащие имена (не a, b, c, d).

Что оценивается в практической работе

* Практическая работа сдана через GitLab.
* Структура папок и файлов репозитория соответствует репозиторию python\_basic.
* Все задачи выполнены в соответствующих папках и файлах main.py.
* Описания коммитов осмысленные и понятные: 111, done, «я сделалъ» — неверно; added m15 homework, 14.3 fix: variables naming — верно.
* Использованы именованные индексы, не просто i (подробнее об этом в видео 7.2).
* Использованы правильные числа, без дополнительных действий со стороны пользователя, без +1 (подробнее об этом в видео 7.4).
* Правильно оформлен input, без пустого приветствия для ввода (подробнее об этом в видео 2.3).
* Переменные и функции имеют значащие имена, не только a, b, c, d (подробнее об этом в видео 2.3).
* Есть пробелы после запятых и при бинарных операциях.
* Нет пробелов после имён функций и перед скобками: print (),input () — неверно; print() — верно.
* Правильно оформлены блоки if-elif-else, циклы и функции; отступы одинаковы во всех блоках одного уровня.
* Все входные и выходные файлы называются так, как указано в задачах.
* Работа с файлами осуществляется с помощью контекстного менеджера with.
* Для обработки исключений используются блоки try-except.
* Модели реализованы в стиле ООП, основной функционал описан в методах классов и в отдельных функциях.
* При написании классов соблюдаются основные принципы ООП: инкапсуляция, наследование и полиморфизм.
  + Для получения и установки значений у приватных атрибутов используются сеттеры и геттеры с соответствующими декораторами.
  + Для создания нового класса на основе уже существующего используется наследование.
  + Для статических и классовых методов используется декоратор classmethod.
* Если классы вынесены в отдельный модуль, то импортируются определённые классы (запись вида from garden import \* считается плохим тоном).
* Классы и методы/функции имеют прописанную документацию (хотя бы минимальную).
* Есть аннотация типов для методов/функций и их аргументов (кроме args и kwargs). Если функция/метод ничего не возвращают, то используется None.
* Во всех декораторах используется functools.wraps.
* В скрипте используется блок if \_\_name\_\_ == "\_\_main\_\_".

Советы и рекомендации

* [Подсказка по регулярным выражениям.](https://docs.google.com/document/d/1ScusW0q7PljdvhI2mQrmYNJwSejey-3gS4ZmSZj_OaE/edit)
* Арифметические операции [PEP 8](https://docs.python.org/3.7/reference/expressions.html#operator-precedence) остаются в приоритете. Необходимо вводить and, or.
* Руководство по стилю Python [PEP 8](https://www.python.org/dev/peps/pep-0008/) на английском языке.
* Руководство по стилю Python [PEP 8](https://pythonworld.ru/osnovy/pep-8-rukovodstvo-po-napisaniyu-koda-na-python.html) на русском языке.
* [Список встроенных функций.](https://docs.python.org/3.7/library/functions.html)