Проект API. Алиса. Разработка собственных навыков и связка с другими API

План урока

1. Введение

Сегодня мы попробуем совместить наш опыт по работе со сторонними API (Яндекс.Карты) и знания Алисы, чтобы создать новый навык, который будет сообщать пользователю, в какой стране находится загаданный город, а также вычислять расстояние от одного города до другого.

Алгоритм работы навыка следующий:

- Если мы пишем название одного города, то Алиса сообщит нам, в какой стране находится этот город,

- Если мы пишем названия двух городов, то Алиса посчитает расстояние между ними,
- Если мы вдруг напишем названия трёх и более городов, то Алиса возмутится и сообщит: «Слишком много городов. Я запуталась».

Для получения информации о географических объектах мы воспользуемся геокодером Яндекс.Карт.

2. Получаем координаты города

Мы уже знаем, что Алиса умеет вычленять из текста разные сущности, в том числе и имена городов.

Напишем функцию **get_coordinates(city_name)**, которая получает географические координаты города по его имени.

Напомним, что надо отправить **HTTP-запрос** (например, такой `https://geocode-maps.yandex.ru/1.x/?geocode=Москва&format=json`) Яндекс.Картам, а потом разобрать ответ.

Возвращает эта функция кортеж с координатами города. В случае, если в процессе работы произошла **любая** ошибка, мы вернём исключение.

```
import requests


def get_coordinates(city_name):
    try:
        # url, по которому доступно API Яндекс.Карт
        url = "https://geocode-maps.yandex.ru/1.x/"
        # параметры запроса
        params = {
            # город, координаты которого мы ищем
            'geocode': city_name,
            # формат ответа от сервера, в данном случае JSON
            'format': 'json'
        }
        # отправляем запрос
        response = requests.get(url, params)
        # получаем JSON ответа
        json = response.json()
        # получаем координаты города (там написаны долгота(longitude),
        # широта(latitude) через пробел).
        # Посмотреть подробное описание JSON-ответа можно
        # в документации по адресу
```

```
        # https://tech.yandex.ru/maps/geocoder/
        coordinates_str = json['response']['GeoObjectCollection'][
            'featureMember'][0]['GeoObject']['Point']['pos']
        # Превращаем string в список, так как точка -
        # это пара двух чисел - координат
        long, lat = map(float, coordinates_str.split())
        # Вернем ответ
        return long, lat
    except Exception as e:
        return e
```

3. Получаем страну города

Функция get_country(city_name) вернёт нам страну, в которой находится указанный город. Отличие от предыдущей функции заключается лишь в получении других данных из ответа геокодера.

```
def get_country(city_name):
    try:
        url = "https://geocode-maps.yandex.ru/1.x/"
        params = {
            'geocode': city_name,
            'format': 'json'
        }
        data = requests.get(url, params).json()
        # Все отличие тут, мы получаем имя страны
        return data['response']['GeoObjectCollection']['featureMember'][0][
            'GeoObject']['metaDataProperty']['GeocoderMetaData'][
            'AddressDetails']['Country']['CountryName']
    except Exception as e:
        return e
```

4. Рассчитываем расстояние от города до города

А вот для вычисления расстояний между двумя точками необходимы знания **тригонометрических** функций. Ведь Земля — круглая!

Подробно о расчёте коротких расстояний на Земле можно прочитать тут.

Пока же можно просто воспользоваться приведённой функцией.

```
import math


def get_distance(p1, p2):
    # p1 и p2 - это кортежи из двух элементов - координаты точек
    radius = 6373.0

    lon1 = math.radians(p1[0])
    lat1 = math.radians(p1[1])
    lon2 = math.radians(p2[0])
    lat2 = math.radians(p2[1])

    d_lon = lon2 - lon1
    d_lat = lat2 - lat1

    a = math.sin(d_lat / 2) ** 2 + math.cos(lat1) * \
        math.cos(lat2) * math.sin(d_lon / 2) ** 2
    c = 2 * math.atan2(a ** 0.5, (1 - a) ** 0.5)

    distance = radius * c
    return distance
```

5. Программируем навык

Наша программа будет состоять из двух файлов.

В первом файле мы разместим код, который будет отвечать за общение с Алисой, а во втором — функции, которые связаны с общением с API Яндекс.Карт.

Всю функциональность общения с картами мы уже реализовали, поэтому осталось объединить все функции в один файл — geo.py.

А во втором файле мы расположим код, который отвечает за общение с Алисой — app.py.

Многое из того, что мы делаем тут, уже было сделано в других уроках. Так что объясним только новые моменты.

```
from flask import Flask, request
import logging
```

```
import json
# импортируем функции из нашего второго файла geo
from geo import get_country, get_distance, get_coordinates

app = Flask(__name__)

# Добавляем логирование в файл. Чтобы найти файл,
# перейдите на pythonwhere в раздел files, он лежит в корневой папке
logging.basicConfig(level=logging.INFO, filename='app.log',
                    format='%(asctime)s %(levelname)s %(name)s %(message)s')


@app.route('/post', methods=['POST'])
def main():
    logging.info('Request: %r', request.json)
    response = {
        'session': request.json['session'],
        'version': request.json['version'],
        'response': {
            'end_session': False
        }
    }
    handle_dialog(response, request.json)
    logging.info('Request: %r', response)
    return json.dumps(response)


def handle_dialog(res, req):
    user_id = req['session']['user_id']
    if req['session']['new']:
        res['response']['text'] = \
            'Привет! Я могу показать город или сказать расстояние между городами!
        return
    # Получаем города из нашего
    cities = get_cities(req)
    if not cities:
        res['response']['text'] = 'Ты не написал название не одного города!'
    elif len(cities) == 1:
        res['response']['text'] = 'Этот город в стране - ' + \
            get_country(cities[0])
    elif len(cities) == 2:
        distance = get_distance(get_coordinates(
            cities[0]), get_coordinates(cities[1]))
        res['response']['text'] = 'Расстояние между этими городами: ' + \
            str(round(distance)) + ' км.'
    else:
        res['response']['text'] = 'Слишком много городов!'
```

```
def get_cities(req):
    cities = []
    for entity in req['request']['nlu']['entities']:
        if entity['type'] == 'YANDEX.GEO':
            if 'city' in entity['value']:
                cities.append(entity['value']['city'])
    return cities


if __name__ == '__main__':
    app.run()
```

6. Тестируем программу

Теперь загрузим нашу программу на https://www.pythonanywhere.com/.

Для того чтобы запустить программу из двух файлов, надо просто добавить второй файл в папку к файлу flask_app.py:

Регистрируем навык и запускаем тестирование:

Таким образом у нас вышла программа, которая с одной стороны взаимодействует с API Яндекс Карт, а с другой стороны — с Алисой.

Это демонстрирует, какие у нас есть возможности при создание навыков.

Разворачивать приложение можно не только на https://www.pythonanywhere.com/. Например, Яндекс предоставляет сервис Яндекс.Облако. Инструкция, как развернуть приложение в Яндекс.Облаке, есть здесь.

В отличие от pythonanywhere, Яндекс Облако предоставляет широкие возможности для конфигурирования облака.

Помощь