# Практическая работа №3: Коллекции и встроенные функции Python

**Ссылка:https://github.com/IvanKuluzaev/Vanika1**

**Задание №1**

1.Постановка задачи.

1. Заказчик требует реализовать API-приложение, предоставляющее

текущую погоду в городах республики Хакасия.

2. Приложение должно быть протестировано через консоль.

3. Приложение должно предоставлять погоду для следующих

населенных пунктов: Абакан, Сорск, Усть-Абакан, Таштып, Боград, Копьёво,

Абаза, Черногорск.

4. Заказчику нужна погода в градусах в трёх единицах измерения

погоды (Цельсий, Фаренгейт, Кельвин).

5. Приложение должно иметь функцию вывода погоды в градусах в

трёх различных единицах измерения и с корректными показателями погоды

по названию города.

− входные данные: название города (например, «Абакан»);

− выходные данные: список показателей погоды в трёх

различных единицах измерения (например, [30.0C, 86.0F,

303.15K])

6. Приложение должно иметь функцию фильтрации городов по

показателям: больше нуля или ноль, меньше нуля (только по Цельсию).

− входные данные: символы ‘+’ или ‘-‘;

− выходные данные: список городов в формате: [Город, Значение

погоды, Сокращенный формат единицы измерения] (например:

`[[Абакан, 28, C], [Боград, 65, F]]`)

7. Приложение должно иметь функцию сортировки по температуре

по Цельсию как по убыванию, так и по возрастанию.

− входные данные: команда о вызове сортировки;

− выходные данные: отсортированный список городов.

2.Листинг программы:

class TemperatureUnit(Enum):

Celsius = 'C'

Fahrenheit = 'F'

Kelvin = 'K'

CityWeather = Tuple[str, float] # (город, температура в Цельсиях)

cities = [

'Абакан', 'Сорск', 'Усть-Абакан', 'Таштып',

'Боград', 'Копьёво', 'Абаза', 'Черногорск'

]

def generate\_weather(city: str) -> CityWeather:

temperature\_celsius = random.randint(-30, 40) # Генерация случайной температуры от -30 до 40°C

return (city, temperature\_celsius)

def convert\_temperature(temp\_celsius: float, unit: TemperatureUnit) -> float:

if unit == TemperatureUnit.Fahrenheit:

return round(temp\_celsius \* 9/5 + 32, 1)

elif unit == TemperatureUnit.Kelvin:

return round(temp\_celsius + 273.15, 1)

return temp\_celsius # Для Цельсия, возвращаем саму температуру

def get\_weather(city: str) -> List[str]:

city\_weather = generate\_weather(city) # Получаем температуру в Цельсиях

city\_name, temp\_celsius = city\_weather

# Получаем температуру в других единицах

fahrenheit = convert\_temperature(temp\_celsius, TemperatureUnit.Fahrenheit)

kelvin = convert\_temperature(temp\_celsius, TemperatureUnit.Kelvin)

return [

f"{temp\_celsius}C",

f"{fahrenheit}F",

f"{kelvin}K"

]

def filter\_cities\_by\_temperature(cities: List[str], sign: str) -> List[List[str]]:

def filter\_condition(city: str) -> bool:

city\_weather = generate\_weather(city)

city\_name, temp\_celsius = city\_weather

if sign == '+':

return temp\_celsius > 0

elif sign == '-':

return temp\_celsius < 0

return False

filtered\_cities = filter(filter\_condition, cities)

# Преобразуем в нужный формат вывода

return [[city, generate\_weather(city)[1], 'C'] for city in filtered\_cities]

def sort\_cities\_by\_temperature(cities: List[str], reverse: bool = False) -> List[List[str]]:

def get\_temperature(city: str) -> float:

city\_weather = generate\_weather(city)

return city\_weather[1]

sorted\_cities = sorted(cities, key=get\_temperature, reverse=reverse)

return [[city, generate\_weather(city)[1], 'C'] for city in sorted\_cities]

def main():

# Получаем погоду для каждого города

print("Погода в городах Хакасии:")

for city in cities:

print(f"{city}: {get\_weather(city)}")

sign = input("Введите '+' для фильтрации городов с температурой выше 0°C, '-' для ниже 0°C: ")

print(f"Города с температурой {sign} 0°C: {filter\_cities\_by\_temperature(cities, sign)}")

order = input("Введите 'asc' для сортировки по возрастанию, 'desc' для убывания: ")

reverse = True if order == 'desc' else False

print(f"Города, отсортированные по температуре ({order}): {sort\_cities\_by\_temperature(cities, reverse)}")

if \_\_name\_\_ == "\_\_main\_\_":

main()

3.Результаты:

