

Функциональное программирование




Урок 4

Курс “Парадигмы программирования и языки парадигм”









Цели семинара

-  Понять основные отличия между **функциональной парадигмой** и уже известными парадигмами
-  Научиться принимать решение об использовании **функциональной парадигмы** в конкретной задаче
-  Научиться решать задачи в рамках **функциональной парадигмы**



План семинара

-  Викторина
-  Пишем код
-  Решаем кейс
-  Подведение итогов



Викторина





Регламент

- 1 Прочитать код
- 2 Подумать в какой парадигме написана программа и поделиться своим ответом
- 3 Обсудить решение



Что за парадигма: конвертация

```
1 def celsius_to_fahrenheit(celsius):  
2     return (celsius * 9/5) + 32  
3  
4 temps_celc = [0, 10, 20, 30]  
5 temps_fahr = list(map(celsius_to_fahrenheit, temps_celc))  
6 print(temps_fahr)
```

Ответ: ..?



Что за парадигма: конвертация

```
1 def celsius_to_fahrenheit(celsius):  
2     return (celsius * 9/5) + 32  
3  
4 temps_celc = [0, 10, 20, 30]  
5 temps_fahr = list(map(celsius_to_fahrenheit, temps_celc))  
6 print(temps_fahr)
```

Ответ: *функциональная и процедурная парадигмы.*

Почему это так: создана функция для конвертации единиц температуры и применена с помощью метода `map`, то есть она использована не только как процедура, но и как *функция функциональной парадигмы*, с помощью *map*.



Что за парадигма: стандартизация данных

```
1 from math import sqrt
2 from statistics import mean, variance
3
4 def standardize(data):
5     avg = mean(data)
6     var = variance(data)
7     std = sqrt(var)
8     def standardize_element(x):
9         return (x - avg) / std
10    return list(map(standardize_element, data))
11
12 data = [1, 2, 3, 4, 5]
13 print(standardize(data))
14 # [-1.2649110640673518, -0.6324555320336759, 0.0,
    0.6324555320336759, 1.2649110640673518]
```

Ответ: ... ?



Что за парадигма: стандартизация данных

```
1 from math import sqrt
2 from statistics import mean, variance
3
4 def standardize(data):
5     avg = mean(data)
6     var = variance(data)
7     std = sqrt(var)
8     def standardize_element(x):
9         return (x - avg) / std
10    return list(map(standardize_element, data))
11
12 data = [1, 2, 3, 4, 5]
13 print(standardize(data))
14 # [-1.2649110640673518, -0.6324555320336759, 0.0,
    0.6324555320336759, 1.2649110640673518]
```

Ответ: функциональная, процедурная и структурная парадигмы.

Почему это так: с одной стороны мы используем функцию `standardize_element` как отображение, но с другой - объявляем внешнюю функцию `standardize` как обычную процедуру внутри которой происходит последовательность шагов в рамках *структурной* парадигмы.



Что за парадигма: reduce

```
1 from functools import reduce
2
3 numbers = [1, 2, 3, 4, 5]
4 sum_numbers = reduce(lambda x, y: x + y, numbers)
```

Ответ: .. ?



Что за парадигма: reduce

```
1 from functools import reduce
2
3 numbers = [1, 2, 3, 4, 5]
4 sum_numbers = reduce(lambda x, y: x + y, numbers)
```

Ответ: это *функциональная* парадигма.

Почему это так: суммирование чисел происходит без явного объявления цикла и элементов на каждом шаге (это происходит под капотом метода *reduce*).



Вопросы



Пишем код





Регламент

- 1 Вместе читаем условия задачи
- 2 Вы самостоятельно решаете задачу
- 3 Вместе обсуждаем решение



Нормализация данных



Нормализация данных

- **Контекст**

Есть такая операция в статистике - “нормализация”. Это операция принимающая на вход вектор и возвращающая другой вектор. Смысл этой операции в том, чтобы данные из разных шкал загнать в единый диапазон, как правило - от 0 до 1, тогда с данными становится проще работать.

- **Ваша задача**

Реализовать с использованием функциональной парадигмы процедуру *normalization*, которая выполняет нормализацию полученного массива по приведенной формуле *нормализованного значения элемента*, где

- x_{norm} - нормализованное значение элемента
- x - исходное значение элемента
- x_{max}, x_{min} - максимальное и минимальное значение в массиве

$$x_{norm} = \frac{x - x_{min}}{x_{max} - x_{min}}$$

- **Решение.. ?**



Нормализация данных

- **Ваша задача**

Реализовать с использованием функциональной парадигмы процедуру *normalization*, которая выполняет нормализацию полученного массива по приведенной формуле *нормализованного значения элемента*, где

- x_{norm} - нормализованное значение элемента
- x - исходное значение элемента
- x_{max}, x_{min} - максимальное и минимальное значение в массиве

- **Решение:**

```
1 def normalize(data):
2     min_val = min(data)
3     max_val = max(data)
4
5     def normalize_element(x):
6         return (x - min_val) / (max_val - min_val)
7
8     return list(map(normalize_element, data))
```



Фильтрация данных



Фильтрация данных

- **Контекст**

Предположим, что есть какой-то массив содержащий данные о разных людях и их возрасте и вас попросили ответить на следующий вопрос: “сколько в массиве людей возраста > 30 ?”. Для этого, вы хотите написать программу для фильтрации наблюдений по возрастному признаку.

- **Ваша задача**

Написать скрипт принимающий на вход *массив с данными о людях* и *число - возраст*, а возвращающий *число - количество людей старше указанного возраста*.

- **Решение.. ?**



Фильтрация данных

- Ваша задача

Написать скрипт принимающий на вход массив с данными о людях и число - возраст, а возвращающий число - количество людей старше указанного возраста.

- Решение:

```
1 people = [  
2     {'name': 'Elizaveta', 'age': 25},  
3     {'name': 'Vasiliy', 'age': 30},  
4     {'name': 'Sergey', 'age': 35},  
5     {'name': 'Ivan', 'age': 40}  
6 ]  
7  
8 def filter_by_age(people:list, min_age:int) → list:  
9     return list(filter(lambda pers: min_age ≤ pers['age'], people))  
10  
11 age = 30  
12 filtered_people = filter_by_age(people, age)  
13 print(filtered_people)  
14 print(len(filtered_people))
```



Поиск дубликатов



Поиск дубликатов

- **Контекст**

Важнейшая задача в анализе данных - поиск дубликатов. Дубликат - это наблюдение, встречающееся в данных больше одного раза. Такие наблюдения не просто не улучшают результат анализа или полученных моделей, но и замедляют весь процесс в целом, поэтому аналитики и разработчики предпочитают избавляться от них перед тем как приступить к анализу.

- **Ваша задача**

Реализовать с использованием функциональной парадигмы процедуру для поиска дубликатов. На вход подается массив, где могут присутствовать дубликаты (а могут и не присутствовать). При применении к массиву, дубликаты должны быть выведены на экран в виде списка.

- **Решение.. ?**



Поиск дубликатов

- **Ваша задача**

Реализовать с использованием функциональной парадигмы процедуру для поиска дубликатов. На вход подается массив, где могут присутствовать дубликаты (а могут и не присутствовать). При применении к массиву, дубликаты должны быть выведены на экран в виде списка.

- **Решение:**

```
1 def find_duplicates(numbers):
2     unique_numbers = set()
3     return list(filter(lambda x: x in unique_numbers or
4         unique_numbers.add(x), numbers))
5
6 numbers = [1, 2, 3, 2, 4, 5, 3, 6, 6]
7 duplicates = find_duplicates(numbers)
```



Решаем кейс





Регламент

- 1 Вместе читаем кейс
- 2 Вы думаете как можно его решить
- 3 Вместе обсуждаем решение



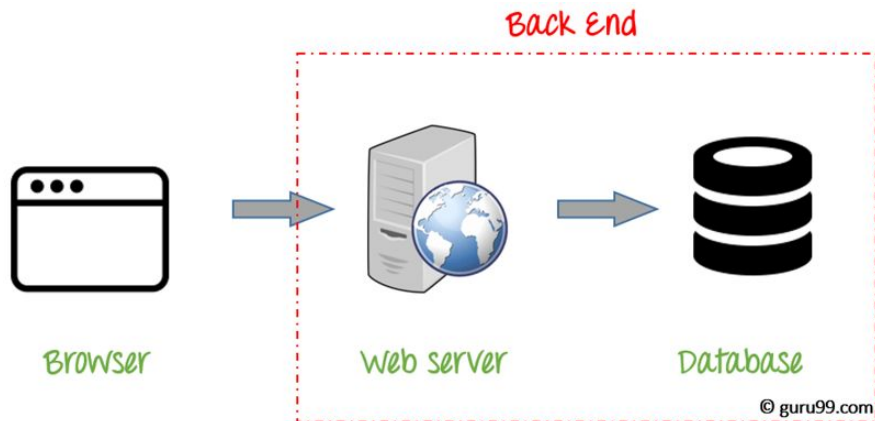
Сервер обработки данных

Кейс. Сервер обработки данных

Контекст: Представьте, что вы работаете бэкенд-разработчиком в ИТ-отделе финансовой организации “Банк Финик”. На ваших серверах ежедневно происходит обработка *терабайтов* данных. Компания хочет создать новый продукт: “мобильный банк”. В рамках данного проекта, компания попросила вашу команду заняться разработкой бэкенда для нового приложения.

ТЗ: Есть мобильное приложение, которое в скором времени начнет слать на ваш сервер сотни тысяч запросов в секунду. Необходимо построить быстрый и устойчивый бэкенд, который справится с таким трафиком.

Задача: Какие парадигмы программирования вы будете использовать для разработки такого ПО и почему именно их?





Итоги семинара



Кейс. Сервер обработки данных

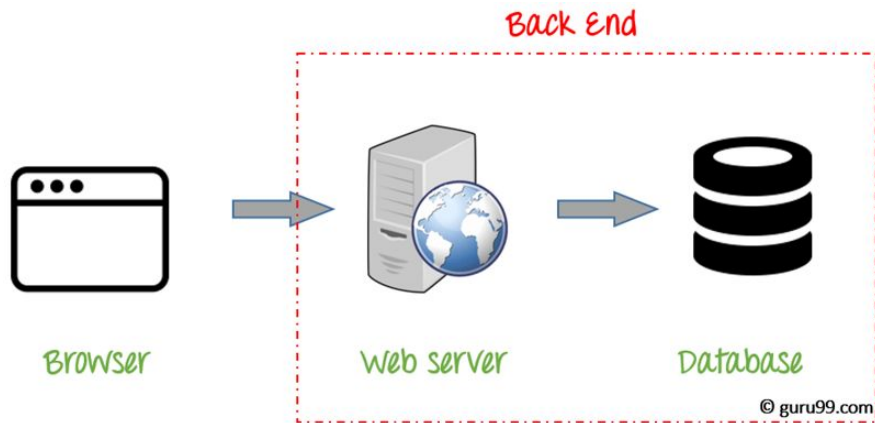
Обсуждение:

Для выполнения подобного проекта скорее всего потребуется немало часов и людей с разной экспертизой. В одном из вариантов реализации, в архитектуре сервера будут присутствовать следующие составляющие:

- База данных
- Сервис для обработки запросов
- Сервис для аутентификации и управления аккаунтами
- Сервис финансовых операций

Каждый из этих сервисов может быть исполнен разными технологиями, поэтому важно учитывать текущий технологический стек. Использование **функциональной парадигмы может быть очень полезным** при обработке сотен тысяч запросов или анализа данных большого объёма. В таком случае, и данные и запросы - являются **массивами**, а то, что мы хотим с ними сделать - **математическими функциями**.

Для финансовых операций и аутентификации скорее всего будет **очень удобно использовать ООП** вместе со **структурным и процедурным стилями**: например, различные *счета клиента* - это **экземпляры класса Счет** с **методом перевод средств**. *Учетная запись клиента* - это **объект**, а *сменить пароль* - это **метод**.





Итоги семинара



Викторина “Что за парадигма”

- Решили 3 задачи на классификацию парадигм



Пишем код

- Решили 3 задачи по программированию



Решаем кейсы

- Решили и обсудили 1 кейс



Подвели итоги



Домашнее задание



Корреляция

- **Контекст**

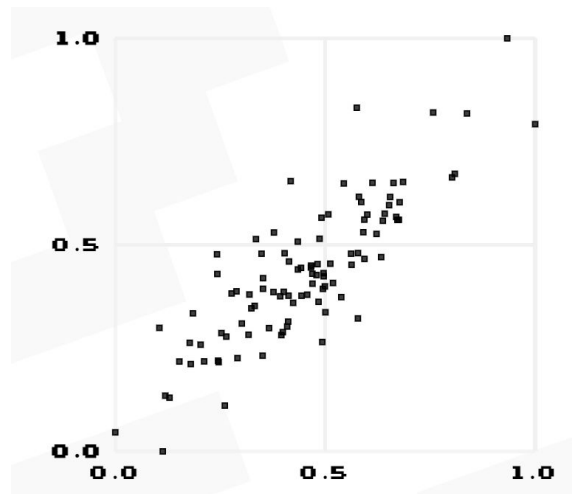
Корреляция - статистическая мера, используемая для оценки связи между двумя случайными величинами.

- **Ваша задача**

Написать скрипт для расчета корреляции Пирсона между двумя случайными величинами (двумя массивами). Можете использовать любую парадигму, но рекомендую использовать функциональную, т.к. в этом примере она значительно упростит вам жизнь.

- **Формула корреляции Пирсона:**

$$r_{xy} = \frac{\sum (x_i - M_x) (y_i - M_y)}{\sqrt{\sum (x_i - M_x)^2 \cdot (y_i - M_y)^2}}$$





Конец семинара
Спасибо за внимание!

