Лабораторная Работа 2: Функции и Неизменяемость в Python

Цель:

Исследование концепций функций и неизменяемости данных в контексте функционального программирования на языке Python. Целью является углубление понимания студентов о том, как функции могут использоваться для создания более модульного, эффективного и безопасного кода, а также о важности неизменяемости данных для предотвращения побочных эффектов.

Задачи:

1. Изучение Особенностей Функций в Python:
   - Понимание различий между функциями и методами.
   - Исследование локальных и глобальных переменных в контексте функций.
   - Понимание концепции области видимости переменных.
2. Понимание и Применение Неизменяемости Данных:
   - Изучение, какие типы данных в Python являются неизменяемыми.
   - Понимание влияния неизменяемости на безопасность и чистоту функций.
   - Рассмотрение практических примеров использования неизменяемых данных.
3. Разработка и Использование Чистых Функций:
   - Понимание, что делает функцию "чистой".
   - Изучение преимуществ использования чистых функций.
   - Реализация примеров чистых функций и сравнение их с нечистыми функциями.
4. Практика Работы с Функциями и Неизменяемыми Структурами Данных:
   - Применение полученных знаний на практических примерах и задачах.
   - Анализ и рефакторинг существующего кода для улучшения его функциональных характеристик.
5. Обсуждение Влияния Функционального Стиля на Производительность и Читаемость Кода:
   - Анализ компромиссов между функциональным стилем и производительностью.
   - Рассмотрение вопросов удобочитаемости и поддержки кода, написанного в функциональном стиле.

Индивидуальные задачи:

Каждому студенту предоставляется уникальная задача в соответствии с его номером в списке группы (смотреть в SSO).Эти задачи ориентированных на реализацию чистых функций, создание функций высшего порядка и использование неизменяемых структур данных:

1. Функция Конвертации Валют
   - Написать чистую функцию для конвертации валюты, которая не изменяет исходные данные.
2. Генератор Функций
   - Создать функцию высшего порядка, которая возвращает другую функцию для возведения числа в указанную степень.
3. Безопасный Список
   - Реализовать функцию, которая копирует список перед выполнением операций, чтобы оригинальный список оставался неизменным.
4. Функция Суммирования
   - Написать чистую функцию для суммирования всех элементов в кортеже.
5. Функция Композиции

- Создать функцию высшего порядка, которая принимает две функции и возвращает их композицию.

6. Функция Фильтрации
   - Реализовать чистую функцию для фильтрации списка чисел, возвращающую новый список без изменения исходного.

7. Калькулятор Налогов
   - Написать функцию, которая рассчитывает налоги на основе дохода, не изменяя исходные данные.

8. Функция Обратного Вызова
   - Создать функцию высшего порядка, которая принимает функцию обратного вызова и применяет её к элементам списка.

9. Генератор Случайных Чисел
   - Написать чистую функцию, которая генерирует список случайных чисел, не изменяя глобальное состояние.

10. Функция Мемоизации
    - Реализовать функцию мемоизации для ускорения вычисления рекурсивных функций.

11. Функция Разделения Строки
    - Создать чистую функцию, которая принимает строку и возвращает список слов, не изменяя исходную строку.

12. Функция Поиска Минимума и Максимума
    - Написать функцию, которая находит минимальное и максимальное число в неизменяемом кортеже.

13. Функция Преобразования Данных
    - Реализовать функцию, которая преобразует список словарей, не изменяя исходный список.

14. Функция Сортировки
    - Создать чистую функцию для сортировки копии списка, оставляя исходный список без изменений.

15. Генератор Предикатных Функций
    - Написать функцию высшего порядка, которая создает и возвращает функцию-предикат на основе заданного условия.

Эти задачи способствуют углубленному пониманию чистоты функций, неизменяемости данных и созданию функций высшего порядка в контексте функционального программирования на Python.

Критерии Оценки:

- Написание кода для решения индивидуальной задачи: 1 балл
- Объяснение и понимание написанного кода во время защиты: 2 балла
- Ответ на один из теоретических вопросов, выбранный преподавателем: 1 балл

Вопросы для Подготовки:

1. Что такое чистая функция в контексте функционального программирования?
   - Цель: Проверить понимание студентами концепции чистых функций, включая отсутствие побочных эффектов и неизменность входных данных.

2. Как неизменяемость данных влияет на функциональное программирование?
   - Цель: Оценить понимание студентом важности неизменяемости данных для предсказуемости и надежности программ.

3. Приведите пример функции высшего порядка в Python.

- Цель: Проверить, насколько хорошо студент понимает концепцию функций высшего порядка и может ли он привести конкретный пример.

4. Как можно реализовать неизменяемые структуры данных в Python?

- Цель: Понимание студентом способов создания и использования неизменяемых структур данных.

5. Какие проблемы решает использование чистых функций?

- Цель: Убедиться, что студент осознает преимущества использования чистых функций, такие как упрощение тестирования и отладки.

6. Могут ли чистые функции улучшить производительность программы? Если да, то как?

- Цель: Оценить понимание студентами потенциальных преимуществ чистых функций в контексте производительности, например, через мемоизацию.

7. Что такое побочные эффекты в программировании и как они связаны с чистыми функциями?

- Цель: Проверка понимания студентом понятия побочных эффектов и их отношения к чистым функциям.

8. Как функция высшего порядка может влиять на удобство и гибкость программирования?

- Цель: Оценка понимания студентами преимуществ функций высшего порядка в написании более модульного и переиспользуемого кода.

9. Объясните, как можно использовать кортежи в Python для обеспечения неизменяемости данных.

- Цель: Понимание использования неизменяемых типов данных, таких как кортежи, для обеспечения неизменяемости.

10. Каким образом неизменяемость данных влияет на параллельное и асинхронное программирование?

- Цель: Проверить, осознают ли студенты влияние неизменяемости на упрощение параллельных и асинхронных операций в программировании.

Пример для решения

Задача: Реализация Функции для Создания Объекта "Замороженного Словаря"

В Python словари (`dict`) являются изменяемым типом данных, что может приводить к побочным эффектам при изменении одного экземпляра словаря, используемого в разных частях программы. Целью этой задачи является создание функции, которая принимает обычный словарь и возвращает его "замороженный" аналог, то есть неизменяемый объект, представляющий те же данные.

Основные Пункты Задачи:

1. Разработать функцию, принимающую словарь в качестве аргумента.
2. Функция должна возвращать неизменяемый объект, который содержит те же пары ключ-значение, что и исходный словарь.
3. Реализация должна быть выполнена с учетом принципов чистой функции.

Решение:

Для решения этой задачи мы можем использовать встроенный в Python тип `frozenset`, который представляет собой неизменяемый аналог обычного множества (`set`). Мы будем создавать `frozenset` из пар ключ-значение исходного словаря.

```
def freeze_dict(input_dict):
    """
    Принимает словарь и возвращает неизменяемое представление его содержимого.
    """
    # Создание списка пар ключ-значение
    items = input_dict.items()

    # Преобразование списка в frozenset для обеспечения неизменяемости
    frozen_dict = frozenset(items)

    return frozen_dict

# Пример использования функции
original_dict = {"apple": "green", "banana": "yellow"}
frozen_dictionary = freeze_dict(original_dict)

print(frozen_dictionary)
```

Объяснение:

- Функция `freeze_dict` принимает словарь `input_dict` в качестве аргумента.
- С помощью метода `items()` получаем список пар ключ-значение из исходного словаря.
- Затем мы преобразуем этот список в `frozenset`, что делает его неизменяемым.
- Возвращаемый `frozenset` представляет собой "замороженный" аналог исходного словаря, который теперь не может быть изменен.

Это решение демонстрирует, как можно обеспечить неизменяемость данных в Python, а также как писать чистые функции, которые не вызывают побочных эффектов и не зависят от внешнего состояния.