# 7-зертханалық жұмыс: Функционалдық принциптерді қолдану арқылы модульдік код құру

**Мақсат:**

Python тілінде функционалдық бағдарламалау негізінде модульдік және кеңейтілетін кодты құру принциптері мен әдістерін меңгеру. Зертхананың мақсаты функционалдық ұғымдарды меңгеру ғана емес, сонымен қатар оларды таза, қолдауға болатын және масштабталатын кодты әзірлеу үшін қолдану болып табылады.

Тапсырмалар:

1. Модульдік және функционалдық бағдарламалау принциптерін зерттеу:
   * Таза функциялар, өзгермейтін, жоғары ретті функциялар және ламбда өрнектері сияқты түсініктерді меңгеру.
   * функционалдық бағдарламалау контекстіндегі модульдік маңыздылығын

талдау.

1. Модульдік функционалдық құрылымдарды жобалау және енгізу:
   * Қолданбаның басқа бөліктеріне оңай біріктірілетін модульдік компоненттерді

құру.

- Кодтың кеңеюін және қайта пайдалануды қамтамасыз ету жолдарын зерттеу.

1. Бар кодты рефакторинг және оңтайландыру:
   * Императивті кодты функционалды стильге түрлендіруге машықтандыру.
   * Модульдік және қолдау мүмкіндігі тұрғысынан кодты оңтайландыру.
2. Функционалдық үлгілерді түсіну және қолдану:
   * Монадалар, карриинг және функционалдық композиция сияқты жалпы

функционалдық үлгілерді зерттеу және қолдану.

- Осы үлгілердің код құрылымы мен ұйымына әсері туралы түсінікті дамыту.

1. Арнайы практикалық тапсырмаларды әзірлеу:
   * функционалдық тәсілдер мен модульдік дизайнды қолдануды талап ететін нақты тапсырмаларды жүзеге асыру.
   * Алынған шешімдерді модульдік, кеңейту және қолдау тұрғысынан бағалау.

Зертханалық жұмыстың маңыздылығы:

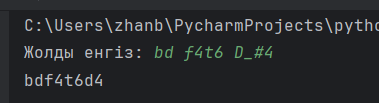
Бұл зертхана студенттердің модульдік және оңай қызмет көрсететін кодты жасау дағдыларын дамытуға бағытталған. Бағдарламалауда функционалдық принциптерді түсіну және қолдану кодтың сапасын жақсартып қана қоймайды, сонымен қатар тестілеуді, жөндеуді және масштабтауды жеңілдетеді. Жұмыс студенттерге жүйелік ойлауды дамытуға және бағдарламалық өнімдердің архитектурасын терең түсінуге көмектеседі.

# Жеке тапсырмалар:

Әрбір студентке топ тізіміндегі санына сәйкес бірегей тапсырма беріледі (SSO қараңыз).

1. Жолды өңдеуге арналған функциялардың құрамы
   * Жолдарды түрлендіру функцияларын жүзеге асыру (мысалы, бос орындарды жою, кіші әріптерге түрлендіру) және осы функциялардың құрамын құру.

import asyncio  
  
async def remove\_spaces(s):  
 await asyncio.sleep(1)  
 return s.replace(" ", "")  
  
async def to\_lower\_case(s):  
 await asyncio.sleep(1)  
 return s.lower()  
  
async def remove\_non\_alphanumeric(s):  
 await asyncio.sleep(1)  
 return ''.join(char for char in s if char.isalnum())  
  
async def compose(\*functions):  
 async def composed(arg):  
 result = arg  
 for func in functions:  
 result = await func(result)  
 return result  
 return composed  
  
async def process\_string(input\_string):  
 string\_processing = await compose(remove\_spaces, to\_lower\_case, remove\_non\_alphanumeric)  
 return await string\_processing(input\_string)  
  
async def run\_main():  
 input\_string = input("Жолды енгіз: ")  
 output\_string = await process\_string(input\_string)  
 print(output\_string)  
  
asyncio.run(run\_main())



# Бағалау критерийлері:

* Жеке есепті шешу үшін код жазу: 1 ұпай
* Қорғау кезінде жазылған кодты түсіндіру және түсіну: 2 ұпай
* Мұғалім таңдаған теориялық сұрақтардың біріне жауап: 1 ұпай

# Дайындық сұрақтары:

1. Бағдарламалаудағы модульдік дегеніміз не және ол қандай артықшылықтар береді?
   * Мақсаты: Студенттің модульдік түсінігін және оның кодты құрылымдау үшін маңызын түсінуін тексеру.

**Модульность в программировании** означает разделение кода на небольшие, самодостаточные блоки (модули), каждый из которых отвечает за определенную функциональность. Преимущества модульности включают повторное использование кода, облегчение отладки и тестирования, а также повышение читаемости и поддерживаемости кода.

1. Python-да функция құрамын қалай жүзеге асырасыз және ол қандай есептер үшін қолайлы?
   * Мақсаты: Студенттің функция құрамы және оны таза және түсінікті код құру үшін қолдану туралы білімін бағалау.

**Композиция функций в Python** означает объединение нескольких функций в одну так, чтобы результат выполнения каждой функции передавался на вход следующей. Этот подход применяется для создания чистого и понятного кода, особенно в случаях, когда набор операций можно разделить на более мелкие и независимые функции.

1. Зертханада қандай асинхронды бағдарламалау әдістерін қолдандыңыз?
   * Мақсаты: Студенттердің асинхронды бағдарламалаудың әртүрлі тәсілдері және оларды пайдалану туралы түсінігі.

В лабораторной работе использовались методы асинхронного программирования, такие как использование асинхронных функций (async/await), использование корутин, и использование библиотек и фреймворков, поддерживающих асинхронные операции, например, asyncio в Python.

1. Асинхронды бағдарламалау Python тіліндегі функционалдық стильмен қалай интеграцияланады?
   * Мақсаты: Студенттің функционалдық және асинхронды тәсілдерді біріктіру қабілетін тексеру.

Асинхронное программирование интегрируется с функциональным стилем в Python путем использования асинхронных функций, которые могут вызываться и комбинироваться как обычные функции, а также путем применения функциональных принципов, таких как композиция функций, к асинхронным операциям.

1. Модульдік бағдарламалауда қандай мәселелер туындауы мүмкін және оларды шешу жолы?
   * Мақсаты: Студенттің модульдікке байланысты жалпы есептерді анықтау және шешу қабілетін бағалау.

Проблемы, возникающие при модульном программировании, могут включать сложности в управлении зависимостями между модулями, конфликты имен, и потерю производительности из-за чрезмерной декомпозиции. Эти проблемы можно решить путем хорошего проектирования интерфейсов модулей, использования принципов инкапсуляции и абстракции, а также тщательного планирования и тестирования.

1. Функция құрамының кодты қайта пайдалану мүмкіндігін арттырудағы рөлі қандай?
   * Мақсаты: Оқушылар модульдік және қайта пайдалануға болатын кодты құру үшін функция құрамының маңыздылығын түсінеді.

Композиция функций способствует повышению переиспользуемости кода, поскольку позволяет создавать независимые и легко комбинируемые блоки функциональности, которые могут быть использованы в различных частях программы или даже в разных проектах.

1. Асинхронды бағдарламалау практикалық емес мысал келтіре аласыз ба?
   * Мақсаты: Студенттердің асинхронды бағдарламалаудың шектеулері туралы түсінігін тексеру.

Асинхронное программирование может быть нецелесообразным в случаях, когда программа выполняет преимущественно вычислительные операции, которые не блокируют поток исполнения, или когда взаимодействие с внешними ресурсами не является узким местом производительности.

1. Асинхронды қолданбаларда күйді қалай басқарасыз?
   * Мақсаты: Асинхронды және функционалдық жүйелердегі мемлекеттік басқару әдістерін білуді бағалау.

Управление состоянием в асинхронных приложениях может осуществляться с помощью корутин, механизмов синхронизации (например, блокировок или очередей), а также структур данных, предназначенных для работы с асинхронным кодом, таких как Future или Task в asyncio.

1. Функционалдық үлгілер дегеніміз не және олар сіздің зертханалық жұмысыңызда қалай қолданылды?
   * Мақсаты: Студенттер бағдарламалық жасақтаманы әзірлеуде функционалдық үлгілерді пайдалануды түсінеді.

Функциональные паттерны - это шаблоны проектирования, основанные на функциональных принципах, таких как чистота функций, неизменяемость данных и отделение состояния. В лабораторной работе функциональные паттерны могли использоваться для управления состоянием, обработки ошибок или применения функциональных преобразований данных.

1. Қандай Python құралдары мен кітапханалары асинхронды бағдарламалауға қолдау көрсетеді?
   * Мақсаты: Python тілінде асинхронды бағдарламалау үшін қолжетімді құралдар туралы студенттердің білімін тексеру.

Для асинхронного программирования в Python доступны различные инструменты и библиотеки, включая asyncio, aiohttp, aiofiles, и др.

1. Модульдік тәсіл тестілеу және жөндеу бағдарламаларына қалай әсер етеді?
   * Мақсаты: Модульдіктің тестілеу және жөндеу процестеріне әсерін түсіну.

Модульный подход облегчает тестирование и отладку программ путем изоляции отдельных модулей, что позволяет проводить модульное тестирование и идентифицировать и исправлять ошибки с минимальными затратами.

1. Функционалдық стильде асинхронияны жүзеге асыру кезінде қандай қиындықтарға тап болдыңыз?
   * Мақсаты: Зертханалық жұмысты орындау барысында студенттер қандай мәселелерге тап болғанын және оларды қалай шешкенін анықтау.

Трудности при реализации асинхронности в функциональном стиле могут включать в себя сложности с управлением состоянием, обработкой ошибок и согласованием асинхронных операций.

1. Таза асинхронды кодты жазуға функционалдық бағдарламалау қалай ықпал ететінін түсіндіре аласыз ба?
   * Мақсаты: Студенттің функционалдық бағдарламалау мен асинхронды өңдеудің өзара әрекеттесуін қаншалықты түсінетінін тексеру.

Функциональное программирование способствует написанию чистого асинхронного кода путем предоставления простых и предсказуемых механизмов композиции и управления состоянием.

1. Асинхронды функцияларды құру кезінде қандай негізгі шақырулар болды?
   * Мақсаты: Асинхронды функция құрамымен байланысты қиындықтарды және олардың шешімдерін түсіну.

Ключевые вызовы при композиции асинхронных функций могут включать в себя управление асинхронными операциями, управление состоянием и обработку ошибок.

1. Модульдік асинхронды қолданбаның архитектурасын қалай жақсартуға болады?
   * Мақсаты: Студенттерді сыни тұрғыдан бағалауға және өз жобаларын жақсартуды ұсынуға ынталандыру.

Для улучшения архитектуры модульного асинхронного приложения можно использовать более четкие интерфейсы между модулями, эффективное управление зависимостями и ресурсами, а также разделение приложения на более мелкие и независимые компоненты.

# Пример для решения

Задача: Создание Модульной Библиотеки для Обработки Текста

Целью этой задачи является разработка модульной библиотеки на Python, использующей функциональные принципы для выполнения различных операций обработки текста, таких как подсчет слов, удаление знаков препинания и обращение порядка слов в предложении.

Цель задачи:

Реализовать серию функций, каждая из которых выполняет определенную задачу обработки текста. Эти функции должны быть чистыми, модульными и легко комбинируемыми.

Решение:

import string

# Функция для удаления знаков препинания из текста def remove\_punctuation(text):

return text.translate(str.maketrans('', '', string.punctuation))

# Функция для подсчета слов в тексте def count\_words(text):

return len(text.split())

# Функция для обращения порядка слов в тексте def reverse\_words(text):

return ' '.join(text.split()[::-1])

# Комбинирование функций для обработки текста def process\_text(text, functions):

for function in functions: text = function(text)

return text

# Пример использования

text = "Hello, world! Welcome to functional programming." functions = [remove\_punctuation, reverse\_words, count\_words] result = process\_text(text, functions)

print(result) # Выводит количество слов после обработки Объяснение:

* В этом примере созданы три отдельные функции: `remove\_punctuation`,

`count\_words`, и `reverse\_words`. Каждая из них выполняет одну операцию обработки текста и является чистой функцией, так как не имеет побочных эффектов и возвращает новое значение на основе входных данных.

* Функция `process\_text` принимает текст и список функций обработки. Она последовательно применяет эти функции к тексту, что демонстрирует модульность и комбинируемость функционального подхода.
* Каждая функция может быть использована независимо или в комбинации с другими, что обеспечивает гибкость и расширяемость библиотеки.

Это решение иллюстрирует, как функциональные принципы могут быть использованы для создания модульного и легко расширяемого кода. Каждая функция выполняет свою задачу независимо, что упрощает тестирование и поддержку кода.