**4-зертхана: Python тіліндегі функциялар және өзгермейтіндік (2 апта)**

**Мақсат:**

Python-да функционалдық бағдарламалау контекстінде функциялар және деректердің өзгермейтіндігі туралы түсініктерді зерттеу. Мақсаты студенттердің модульдік, тиімді және қауіпсіз кодты жасау үшін функцияларды қалай пайдалануға болатынын, сондай-ақ жанама әсерлердің алдын алу үшін деректердің өзгермейтіндігінің маңыздылығын түсінуді тереңдету.

Тапсырмалар:

1. Python тіліндегі функцияларды үйрену ерекшеліктері:

- Функциялар мен әдістердің айырмашылығын түсіну.

- Функциялар контекстінде жергілікті және ғаламдық айнымалыларды зерттеу.

- Ауыспалы ауқым ұғымын түсіну.

2. Деректердің өзгермейтіндігін түсіну және қолдану:

- Python тілінде қандай деректер түрлері өзгермейтінін білу.

- Функциялардың қауіпсіздігі мен тазалығына өзгермейтіндіктің әсерін түсіну.

- өзгермейтін деректерді пайдаланудың практикалық мысалдарын қарастыру.

3. Таза функцияларды әзірлеу және пайдалану:

- Функцияны «таза» ететін нәрсені түсіну.

- Таза функцияларды пайдаланудың артықшылықтарын зерттеу.

- таза функциялардың мысалдарын орындау және оларды таза емес функциялармен салыстыру.

4. Функциялармен және өзгермейтін деректер құрылымдарымен жұмыс істеуге машықтандыру:

- Алған білімдерін практикалық мысалдар мен есептер шығаруда қолдану.

- функционалдық сипаттамаларын жақсарту үшін бар кодты талдау және рефакторинг.

5. Функционалдық стильдің кодтың өнімділігі мен оқылуына әсерін талқылау:

- Функционалдық стиль мен өнімділік арасындағы сәйкестіктерді талдау.

- Функционалдық стильде жазылған кодтың оқылу және қолдау мәселелерін қарастыру.

**Жеке тапсырмалар:**

Әрбір студентке топ тізіміндегі оның нөміріне сәйкес бірегей тапсырма беріледі (SSO қараңыз).Бұл тапсырмалар таза функцияларды жүзеге асыруға, жоғары ретті функцияларды жасауға және өзгермейтін деректер құрылымдарын пайдалануға бағытталған:

1. Валюта айырбастау функциясы

- Бастапқы деректерді өзгертпейтін валютаны айырбастауға арналған таза функцияны жазыңыз.

2. Функция генераторы

- Санды көрсетілген қуатқа көтеру үшін басқа функцияны қайтаратын жоғары ретті функцияны жасаңыз.

3. Қауіпсіз тізім

- Бастапқы тізім өзгеріссіз қалуы үшін әрекеттерді орындамас бұрын тізімді көшіретін функцияны орындаңыз.

4. Қорытындылау функциясы

- Кортеждегі барлық элементтерді қосу үшін таза функцияны жазыңыз.

5. Композиция функциясы

- Екі функцияны қабылдайтын және олардың құрамын қайтаратын жоғары ретті функцияны жасаңыз.

6. Сүзгі функциясы

- Түпнұсқаны өзгертпей, жаңа тізімді қайтарып, сандар тізімін сүзуге арналған таза функцияны іске қосыңыз.

7. Салық калькуляторы

- Түпнұсқа мәліметтерді өзгертпей, салықты табыс негізінде есептейтін функцияны жазыңыз.

8. Кері шақыру функциясы

- Кері шақыру функциясын қабылдайтын және оны тізім элементтеріне қолданатын жоғарырақ ретті функцияны жасаңыз.

9. Кездейсоқ сандар генераторы

- Ғаламдық күйді өзгертпей, кездейсоқ сандар тізімін тудыратын таза функцияны жазыңыз.

10. Есте сақтау функциясы

- Рекурсивті функцияларды есептеуді жылдамдату үшін есте сақтау функциясын орындаңыз.

11. Жолды бөлу функциясы

- Жолды қабылдайтын және бастапқы жолды өзгертпей сөздер тізімін қайтаратын таза функция жасаңыз.

12. Минималды және максималды іздеу функциясы

- Өзгермейтін кортеждегі ең кіші және ең үлкен сандарды табатын функцияны жазыңыз.

13. Деректерді түрлендіру функциясы

- Бастапқы тізімді өзгертпей, сөздіктер тізімін түрлендіретін функцияны іске асыру.

14. Сұрыптау функциясы

- Түпнұсқа тізімді өзгеріссіз қалдырып, тізімнің көшірмесін сұрыптау үшін таза функция жасаңыз.

15. Предикаттар функциясының генераторы

- Берілген шарт негізінде предикат функциясын жасайтын және қайтаратын жоғары ретті функцияны жазыңыз.

**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. Функционалдық программалау контекстіндегі таза функция дегеніміз не?

- Мақсаты: Студенттердің таза функциялар түсінігін, оның ішінде жанама әсерлердің жоқтығын және енгізілген деректердің өзгермейтіндігін тексеру.

2. Деректердің өзгермейтіндігі функционалдық бағдарламалауға қалай әсер етеді?

- Мақсаты: Бағдарламалардың болжамдылығы мен сенімділігі үшін деректердің өзгермейтіндігінің маңыздылығын студенттің түсінуін бағалау.

3. Python тілінде жоғары ретті функцияға мысал келтіріңіз.

- Мақсаты: Оқушының жоғары ретті функциялар ұғымын қаншалықты түсінетінін және нақты мысал келтіре алатынын тексеру.

4. Python-да өзгермейтін деректер құрылымдарын қалай жүзеге асыра аламыз?

- Мақсаты: Студенттердің өзгермейтін деректер құрылымдарын құру және пайдалану жолын түсінуі.

5. Таза функцияларды қолдану қандай есептерді шешеді?

- Мақсаты: Студентке оңай тестілеу және жөндеу сияқты таза функцияларды пайдаланудың артықшылықтарын түсінуін қамтамасыз ету.

6. Таза функциялар бағдарлама өнімділігін жақсарта ала ма? Егер солай болса, қалай?

- Мақсаты: Оқушылардың өнімділік контекстіндегі таза функциялардың әлеуетті артықшылықтарын түсінуін бағалау, мысалы, есте сақтау арқылы.

7. Бағдарламалаудағы жанама әсерлер қандай және олар таза функциялармен қалай байланысты?

- Мақсаты: Студенттің жанама әсерлер түсінігін және олардың таза функциялармен байланысын тексеру.

8. Жоғары ретті функция бағдарламалаудың ыңғайлылығы мен икемділігіне қалай әсер етуі мүмкін?

- Мақсаты: Модульдік және қайта пайдалануға болатын кодты жазуда студенттердің жоғары ретті функциялардың артықшылықтарын түсінуін бағалау.

9. Деректердің өзгермейтіндігін қамтамасыз ету үшін Python тілінде кортеждерді қалай пайдалануға болатынын түсіндіріңіз.

- Мақсаты: өзгермейтіндікке қол жеткізу үшін кортеждер сияқты өзгермейтін деректер түрлерін пайдалануды түсіну.

10. Деректердің өзгермейтіндігі параллельді және асинхронды бағдарламалауға қалай әсер етеді?

- Мақсаты: Студенттердің бағдарламалаудағы параллельді және асинхронды операцияларды жеңілдетудегі өзгермейтіндік әсерін түсінетіндігін тексеру.

**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.