МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РЕСПУБЛИКИ КАЗАХСТАН

УНИВЕРСИТЕТ САТПАЕВ

Институт\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Кафедра\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_



ЛАБОРАТОРНАЯ РАБОТА

Тема: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

|  |  |  |  |
| --- | --- | --- | --- |
| № | Качество выполнения работы | Диапазон  оценки | Получено,  % |
| 1 | Не выполнено | 0% |  |
| 2 | Выполнено | 0-50% |  |
| 3 | Самостоятельная систематизация материала | 0-10% |  |
| 4 | Выполнение требуемого объема и в указанный срок | 0-5% |  |
| 5 | Использование дополнительной научной литературы | 0-5% |  |
| 6 | Уникальность выполненного задания | 0-10% |  |
| 7 | Защита работы | 0-20% |  |
|  | Итого: | 0-100% |  |

Преподаватель Қасенхан Арай

Студент Жаңбырбайқызы Қарлығаш

Группа СР 12:10-14:05

Алматы 2023 г

# Жаңбырбайқызы Қарлығаш

# 2-зертхана: Python тіліндегі функциялар және өзгермейтіндік

**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.

1. 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.

1. 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.

1. 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.

1. 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. ***Функция генераторы***
   * ***Санды көрсетілген қуатқа көтеру үшін басқа функцияны қайтаратын жоғары ретті функцияны жасаңыз.***

def quat\_gen(exponent):  
  
 def quat\_func(x):  
 return x \*\* exponent  
  
 return quat\_func  
  
kvadrat = quat\_gen(2)  
  
kub = quat\_gen(3)  
  
number = 5  
kvadrat\_result = kvadrat(number)  
kub\_result = kub(number)  
  
print(f"{number} квадраты: {kvadrat\_result}")  
print(f"{number} кубы: {kub\_result}")

# Изображение выглядит как текст, Шрифт, снимок экрана, типография Автоматически созданное описание

# 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?***

В функциональном программировании чистая функция - это функция, которая обладает двумя основными свойствами: отсутствие побочных эффектов и детерминированность. **Отсутствие побочных эффектов:** Чистая функция не взаимодействует с внешними данными и не изменяет состояние программы или окружающего контекста. Она принимает аргументы и возвращает результат, но не вносит изменений в глобальные переменные, файлы, базы данных и т.д. Это свойство делает код, использующий чистые функции, более предсказуемым и легким в тестировании.

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

1. ***Деректердің өзгермейтіндігі функционалдық бағдарламалауға қалай әсер етеді?***

Неизменность данных (immutability) представляет собой принцип функционального программирования, который заключается в том, что данные после создания не могут быть изменены. Этот принцип оказывает значительное влияние на функциональное программирование и приносит несколько важных выгод:

**Предсказуемость и стабильность, Поток данных и отладка, Параллелизм и конкурентность, Функции высшего порядка и чистые функции**

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

1. Python тілінде жоғары ретті функцияға мысал келтіріңіз.
   * Мақсаты: Оқушының жоғары ретті функциялар ұғымын қаншалықты түсінетінін және нақты мысал келтіре алатынын тексеру.
2. Python-да өзгермейтін деректер құрылымдарын қалай жүзеге асыра аламыз?

**Кортежи (Tuples), Строки (Strings), Неизменяемые множества (frozenset),**

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

1. Таза функцияларды қолдану қандай есептерді шешеді?

Чистые функции, то есть функции, которые не имеют побочных эффектов и всегда возвращают одинаковый результат для одних и тех же входных данных, имеют ряд преимуществ и могут быть использованы для решения различных задач. Вот несколько типов задач, которые хорошо решаются с использованием чистых функций

**Преобразование данных, Фильтрация данных, Рекурсивные алгоритмы, Генерация потоков данных**

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

1. 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.

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

Побочные эффекты в программировании относятся к изменениям, которые функция может внести во внешнее состояние программы, кроме возвращаемого значения. Побочные эффекты могут быть видны в виде изменений глобальных переменных, взаимодействия с файловой системой, изменениями в сетевых запросах, выводом на экран, и так далее. В отличие от чистых функций, функции, вызывающие побочные эффекты, могут оказывать влияние на состояние программы или окружающей среды.

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

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

Функции высшего порядка могут значительно повлиять на удобство и гибкость программирования. они могут сделать код более гибким и удобным. Функции высшего порядка могут принимать другие функции в качестве аргументов. Это позволяет создавать более абстрактные и гибкие функции, которые могут работать с различными логиками в зависимости от переданной функции.

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

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

В Python кортежи являются неизменяемыми структурами данных, и они могут быть использованы для обеспечения неизменности данных. Кортежи создаются с использованием круглых скобок **()** или функции **tuple()**. Основные свойства кортежей, которые обеспечивают неизменность данных:

**Нельзя изменить элементы, Нельзя добавить или удалить элементы, Неизменяемость вложенных структур**

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

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

Неизменность данных играет важную роль в параллельном и асинхронном программировании, внося в этот процесс ряд преимуществ и облегчая управление состоянием в многозадачных сценариях. Вот как неизменность данных влияет на параллельное и асинхронное программирование:

**Избегание состояния гонки (Race Conditions), Безопасность в многозадачных сценариях, Функциональная структура кода, Отсутствие побочных эффектов**

- 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.