ҚАЗАҚСТАН РЕСПУБЛИКАСЫНЫҢ БІЛІМ ЖӘНЕ ҒЫЛЫМ МИНИСТРЛІГІ

СӘТБАЕВ УНИВЕРСИТЕТІ

Автоматика және Информациялық Технологиялар институты

Программалық Инженерия кафедрасы



ЛАБОРАТОРИЯЛЫҚ ЖҰМЫС #4

Тақырыбы: Жетілдірілген рекурсия және функционалдық үлгілер

|  |  |  |  |
| --- | --- | --- | --- |
| № | Жұмысты орындау сапасы | Баға диапазоны | Орындалған  % |
| 1 | Орындалған жоқ | 0% |  |
| 2 | Орындалды | 0-50% |  |
| 3 | Материялдық өзіндік жүйелендіру | 0-10% |  |
| 4 | Талап етілген көлемде және көрсетілген мерзімде орындау | 0-5% |  |
| 5 | Қосымша ғылыми әдебиеттерді пайдалану | 0-5% |  |
| 6 | Орындаған тапсырманың ерекшелігі | 0-10% |  |
| 7 | СӨЖ-ді қорғау | 0-20% |  |
|  | Қорытынды: | 0-100% |  |

Оқытушы: Шаяхметов Д

Студент: Ұлдақан А

Мамандығы: Computer Science

Тобы: Дс 7:50 – 9:45

Алматы 2024 ж

**Мақсат:**

Python тіліндегі озық рекурсивті әдістер мен функционалдық үлгілерді терең меңгеру және қолдану. Зертхананың мақсаты рекурсивті кодты жазу дағдыларын жақсарту ғана емес, сонымен қатар бағдарламалардың оқылуын, тиімділігін және масштабталуын жақсартатын әртүрлі функционалдық үлгілерді түсіну және қолдану болып табылады.

Тапсырмалар:

1. Рекурсияны терең зерттеу:

- Күрделі рекурсивті есептер мен алгоритмдерді талдау.

- Python тілінде құйрықты рекурсия және оны жүзеге асыру принциптерін зерттеу.

- мәліметтерді өңдеу және алгоритмдік есептерді шешу үшін рекурсияны практикалық қолдану.

2. Функционалдық үлгілерді меңгеру:

- Монадалар, карриинг және функционалдық композиция сияқты функционалдық үлгілерді зерттеу және қолдану.

- Функционалдық үлгілердің кодты ұйымдастыруға және құрылымына қалай әсер ететіні туралы түсінікті дамыту.

3. Функционалдық программалауды нақты есептерде қолдану:

- Практикалық есептерге күрделі функционалды шешімдерді енгізу, мысалы, деректерді талдау, мәтінді өңдеу немесе веб-қызметтерді құру.

- Модульділік пен кодтың сыналуын жақсарту үшін функционалдық тәсілдерді қолдану.

4. Жақсартылған код өнімділігі және оқылу мүмкіндігі:

- Рекурсия мен функционалдық заңдылықтардың бағдарламаның жұмысына әсерін талдау.

- Функционалдық бағдарламалау стилінің артықшылықтары мен кемшіліктерін кодты оқу және техникалық қызмет көрсету тұрғысынан бағалау.

5. Сыни тұрғыдан ойлау және проблемалық талдау дағдыларын дамыту:

- Күрделі есептерді шешу кезінде сыни талдау мен тәуелсіз ізденістерді ынталандыру.

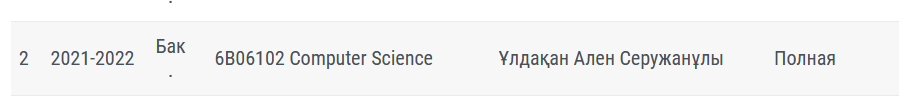
- Теориялық білімдерін практикада қолдану және бейімдеу қабілетін дамыту.

Зертханалық жұмыстың маңыздылығы:

Зертханалық жұмыс функционалдық бағдарламалау мен рекурсияда алдыңғы қатарлы дағдыларды дамытуға бағытталған. Бұл студенттерге кодының сапасы мен ауқымдылығын жақсарта отырып, функционалды бағдарламалау принциптерін нақты әлемдегі мәселелерге қалай қолдануға болатынын жақсы түсінуге көмектеседі. Сондай-ақ жұмыс аналитикалық ойлауды дамытуға және әртүрлі бағдарламалық тапсырмалар мен талаптарға бейімделу қабілетіне ықпал етеді.

**Жеке тапсырмалар:**

Әрбір студентке топ тізіміндегі санына сәйкес бірегей тапсырма беріледі (SSO қараңыз). Студенттерге рекурсияда қосымша дағдыларды дамытуға және функционалдық заңдылықтарды түсінуге көмектесетін есептер:



2. Сөмке мәселесін шешуге арналған рекурсиямен динамикалық бағдарламалау

- Сөмке мәселесін шешу үшін есте сақтау арқылы рекурсияны пайдаланыңыз.

def knapsack(weights, values, capacity):

    memo = {}

    def knapsack\_helper(index, current\_capacity):

        if index < 0 or current\_capacity == 0:

            return 0

        if (index, current\_capacity) in memo:

            return memo[(index, current\_capacity)]

        if weights[index] > current\_capacity:

            memo[(index, current\_capacity)] = knapsack\_helper(index - 1, current\_capacity)

            return memo[(index, current\_capacity)]

        included = values[index] + knapsack\_helper(index - 1, current\_capacity - weights[index])

        not\_included = knapsack\_helper(index - 1, current\_capacity)

        memo[(index, current\_capacity)] = max(included, not\_included)

        return memo[(index, current\_capacity)]

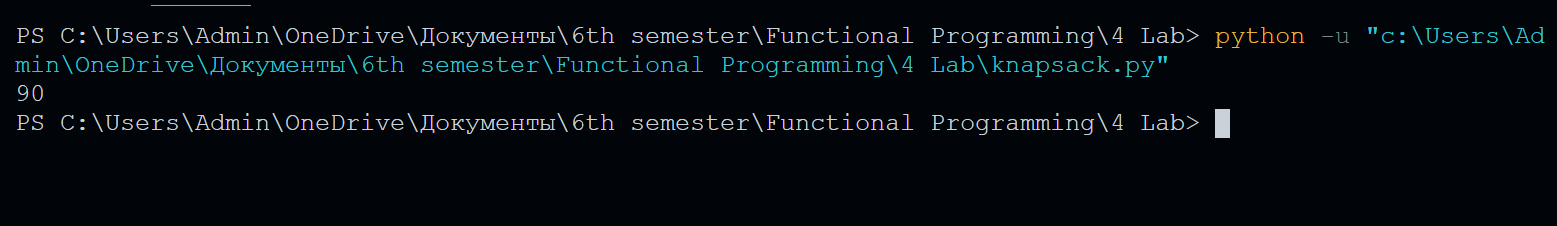
    return knapsack\_helper(len(weights) - 1, capacity)

weights = [5, 4, 6, 3]

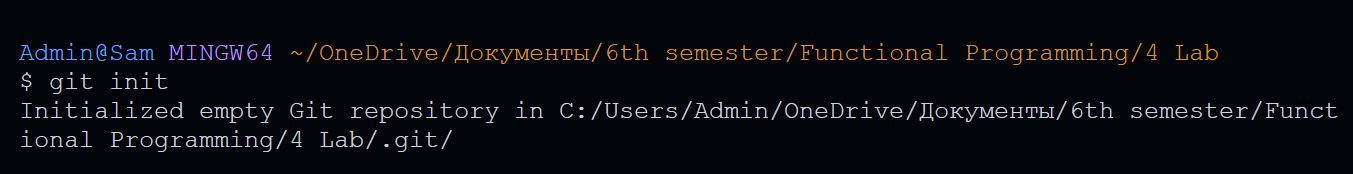
values = [10, 40, 30, 50]

capacity = 10

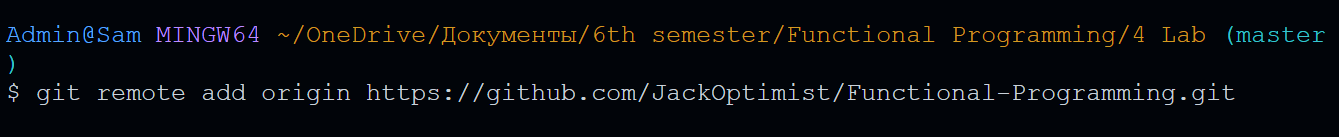
print(knapsack(weights, values, capacity))



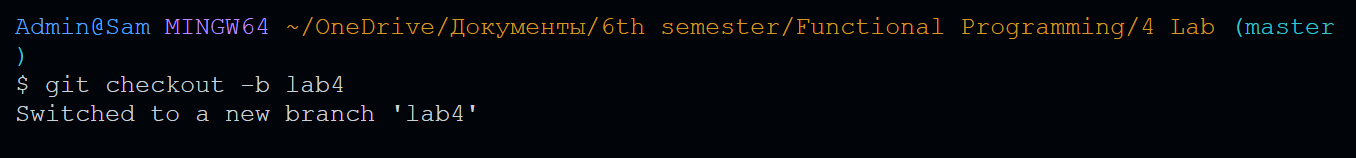
**$ git init**



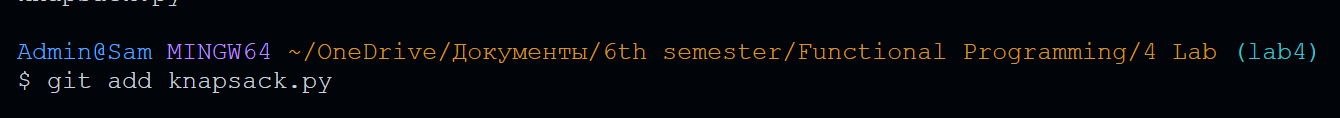
**$ git remote add origin <URL>**



**$ git checkout -b branch name (lab4)**



**$ git add knapsack.py**

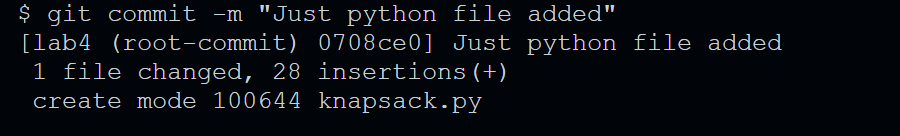


**$ git status**

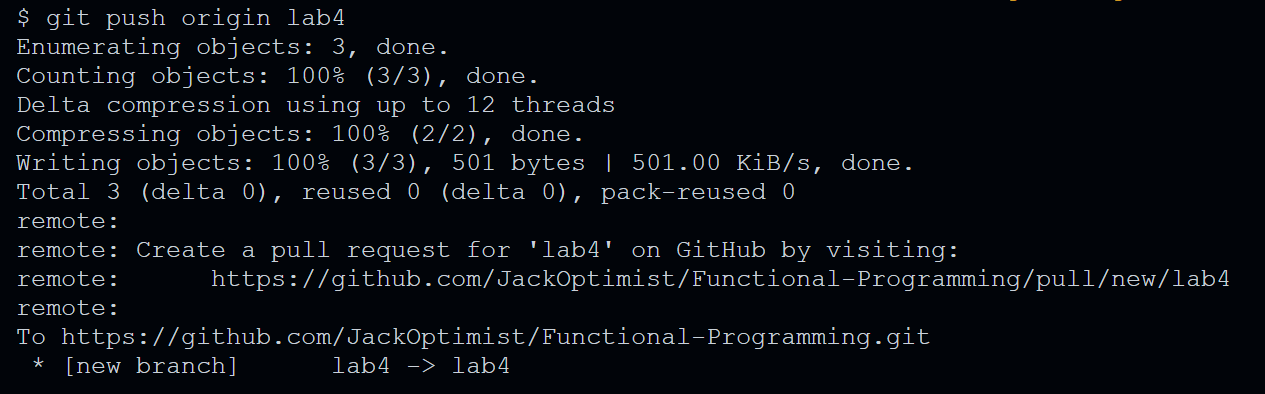
Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

**$ git commit -m “<COMMIT>”**



**$ git push origin lab4 (branch name)**



Изображение выглядит как текст, снимок экрана, программное обеспечение, Мультимедийное программное обеспечение

Автоматически созданное описание