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

# 3-зертхана: Классикалық есептер үшін рекурсивті функцияларды орындау

**Мақсат:**

Классикалық алгоритмдік және математикалық есептерді шешу үшін Python тілінде рекурсивті функцияларды үйреніңіз және қолданыңыз. Жұмыстың мақсаты студенттердің рекурсивті кодты жазу, оның принциптері мен шектеулерін түсіну, сонымен қатар рекурсивті алгоритмдерді талдау және оңтайландыру дағдыларын дамыту болып табылады.

Тапсырмалар:

1. Рекурсия негіздері:

* Рекурсия ұғымын және оның итеративті тәсілден айырмашылығын түсіну.
* Рекурсивті функциялардың негізгі мысалдарын оқу.

1. Рекурсивті функциялардың дамуы:

* Классикалық рекурсивті есептерді орындау, мысалы, факториалды, Фибоначчи сандарын, екілік іздеуді және т.б.
* Итерациялық әдістермен салыстырғанда рекурсивті функциялардың тиімділігін

талдау.

1. Қоңыраулар стегі мен рекурсиялық шектеулерді түсіну:
   * Python тіліндегі шақыру стек механизмін және оның рекурсивті функцияларға әсерін зерттеу.
   * Қоңыраулар стекінің толып кету мәселесін анықтау және болдырмау.
2. Рекурсивті функцияларды оңтайландыру:
   * Рекурсивті қоңырауларды оңтайландыру үшін есте сақтау және динамикалық бағдарламалау әдістерін пайдалану.
   * оңтайландырылған рекурсивті функциялардың өнімділігін олардың оңтайландырылмаған нұсқаларымен салыстыру.
3. Практикалық есептердегі рекурсияның қолданылуы:
   * Ағаштарды, графиктерді және басқа деректер құрылымдарын аралау сияқты рекурсивті тәсілді қажет ететін күрделірек мәселелердің шешімдерін әзірлеу.

Зертханалық жұмыстың маңыздылығы:

Бұл зертхана студенттерге рекурсияны бағдарламалаудағы қуатты құрал ретінде түсінуді тереңдетуге көмектеседі. Итерациялық әдістерді қолдану арқылы шешуі қиын немесе тиімсіз есептерді шешу үшін жиі рекурсивті әдістер қолданылады. Рекурсияны меңгеру программисттің құралдар жиынтығын кеңейтіп, алгоритмдік есептердің кең ауқымын тиімді шешуге мүмкіндік береді.

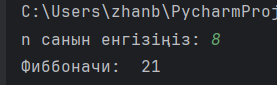
# Жеке тапсырмалар:

Әрбір студентке топ тізіміндегі санына сәйкес бірегей тапсырма беріледі (SSO қараңыз). Студенттерге рекурсияға, деректер құрылымдарын айналып өтуге және рекурсивті функцияларды оңтайландыруға көмектесетін есептер

1. ***Фибоначчи сандары***

***- \(n\)-ші Фибоначчи санын есептеу үшін рекурсивті функцияны жазыңыз.***

def fibonacci(n):  
 if n <= 1:  
 return n  
 else:  
 return fibonacci(n-1) + fibonacci(n-2)  
  
n = int(input("n санын енгізіңіз: "))  
if n < 0:  
 print("оң санын енгізіңіз")  
else:  
 result = fibonacci(n)  
 print("Фиббоначи:", result)





# 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. Рекурсия дегеніміз не және ол бағдарламалау контекстінде қалай жұмыс істейді?

Рекурсия-бұл функция өзін-өзі шақырған кезде бағдарламалаудағы ұғым. Ол бірдей типтегі қарапайым ішкі есептерге бөлуге болатын мәселелерді шешу үшін кеңінен қолданылады. Әрбір функция шақыруы тапсырманың бір бөлігін шешеді, содан кейін қалған жұмысты өзіне тапсырады. Процесс негізгі шартқа қол жеткізілгенге дейін жалғасады, содан кейін рекурсивті қоңыраулар аяқталады.

Мақсаты: Оқушылардың рекурсия ұғымы туралы жалпы түсінігін тексеру.

1. Рекурсивті функцияның негізгі компоненттері қандай?

Рекурсивті функцияның негізгі компоненттеріне мыналар жатады:

1. Негізгі жағдай (base Case): Бұл рекурсияның қашан аяқталатынын анықтайтын шарт. Бағдарлама негізгі жағдайға жеткенде, ол белгілі бір мәнді қайтарады немесе белгілі бір әрекетті орындайды, бірақ өзін қайта шақырмайды. Негізгі жағдай болмаса, рекурсивті функция шексіз шақырылады.

2. Рекурсивті жағдай (Recursive Case): Бұл бірдей функцияны тудыратын шарт немесе шарттар жиынтығы, бірақ негізгі жағдайға жақындайтын дәлелдермен. Рекурсивті жағдайда функция мәселенің бір бөлігін шешеді, содан кейін қалған жұмысты өзіне тапсырады. Әрбір жаңа қоңырау негізгі жағдайға жақындайтындай етіп реттелуі керек.

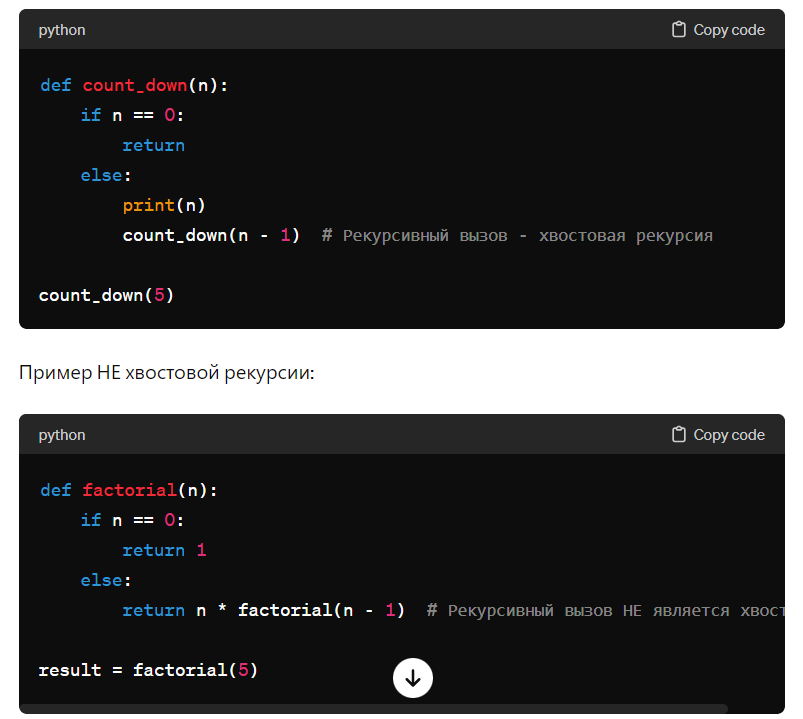
3. Айнымалылар (немесе параметрлер): бұл әр қоңырау кезінде функцияға берілетін мәндер. Олар негізгі жағдайға жақындау үшін әр қоңырауда өзгеруі мүмкін.

4. Логика (немесе алгоритм): бұл функцияның есепті қалай шешетінін анықтайтын функция ішіндегі нұсқаулар. Рекурсивті функция жағдайында бұл негізгі жағдай және рекурсивті жағдай, сондай-ақ тапсырманы өңдеу үшін қажет кез келген басқа логика болуы мүмкін.

5. Мәнді қайтару (немесе әрекетті орындау): сайып келгенде, рекурсивті функция белгілі бір мәнді қайтаруы немесе белгілі бір әрекетті орындауы керек. Негізгі жағдайда функция мәнді қайтаруы немесе қандай да бір әрекетті орындауы мүмкін, ал рекурсивті жағдайда функция әдетте өзін шақырады, содан кейін алгоритмнің келесі қадамдарын орындау үшін қайтарылатын мәнді пайдаланады.

Бұл компоненттер рекурсивті функцияға есепті қарапайым ішкі есептерге бөлу арқылы шешуге мүмкіндік беру үшін бірге жұмыс істейді.- Мақсаты: Студенттің рекурсивті функцияның құрылымын, оның ішінде негізгі жағдай мен рекурсивті қадамды түсінуін қамтамасыз ету.

1. Бағдарламалауда рекурсияны қолданудың артықшылықтары мен кемшіліктері қандай?
   * Бағдарламалауда рекурсияны қолданудың артықшылықтары мен кемшіліктері бар.
   * Артықшылықтары:
   * 1. Кодтың қарапайымдылығы мен айқындылығы: рекурсивті Алгоритмдер қарапайым және түсінікті болуы мүмкін, әсіресе табиғи түрде кішігірім ішкі есептерге бөлінетін есептер үшін.
   * 2. Тапсырмаларды табиғи түрде көрсету: кейбір тапсырмалар үшін рекурсия олардың құрылымын табиғи түрде көрсетеді, бұл кодты интуитивті етеді.
   * 3. Кейбір алгоритмдерге ыңғайлылық: кейбір алгоритмдерді, мысалы, біріктіру сұрыптау немесе ағашты айналып өту, рекурсияны қолдану арқылы жүзеге асыру оңайырақ.
   * 4. Оңтайландыру мүмкіндігі: кейбір рекурсивті алгоритмдерді жақсы өнімділік немесе жадты пайдалану үшін оңтайландыруға болады.
   * \*\* Кемшіліктері:\*\*
   * 1. Стектің толып кетуі (Stack overflow): дұрыс орындалмаған рекурсия, әсіресе үлкен немесе дұрыс емес кірістер үшін қоңырау стегінің толып кетуіне әкелуі мүмкін.
   * 2. Жадты тұтыну: рекурсивті қоңыраулар қайталанатын баламаларға қарағанда көбірек жадты алуы мүмкін, себебі қоңырау деректерін стекке сақтау қажет.
   * 3. Өнімділік: кейбір жағдайларда рекурсивті Алгоритмдер функционалдық қоңырауларға қосымша үстеме шығындарға байланысты олардың итеративті эквиваленттеріне қарағанда өнімділігі төмен болуы мүмкін.
   * 4. Жөндеудің қиындығы: рекурсивті функцияларды олардың жасырын күйіне және бірнеше кірістірілген қоңырауларға байланысты түзету қиынырақ болуы мүмкін.
   * 5. Кодты оқу: кейбір адамдар рекурсивті шешімдерді итеративті әріптестеріне қарағанда аз оқылатын және түсінікті деп санауы мүмкін.
   * Рекурсияны пайдалану белгілі бір тапсырма мен деректер құрылымы контекстінде қарастырылуы керек. Кейбір жағдайларда бұл ең жақсы таңдау болуы мүмкін, ал басқа жағдайларда итерацияны немесе басқа тәсілдерді қолданған дұрыс.Мақсаты: Оқушылардың рекурсияның артықшылықтарын (мысалы, кодтың қарапайымдылығы) және кемшіліктерін (мысалы, стектің толып кету қаупі) түсінуін бағалау.
2. Рекурсия ең жақсы шешім болып табылатын мәселеге мысал келтіріңіз.
   * Мақсаты: Студент рекурсия тиімді тәсіл болатын сценарийлерді анықтай алатынын көру.
3. Рекурсияда қоңыраулар стекінің толып кетуін қалай болдырмауға болады?
   * Рекурсиядағы қоңыраулар стегінің толып кетуі рекурсивті қоңыраулар саны тым көп болған кезде және функционалдық қоңыраулар мен олардың жергілікті айнымалыларын сақтауға арналған жад стегі толған кезде орын алуы мүмкін. Бұл үлкен кірістермен жұмыс істегенде немесе рекурсивті функцияны дұрыс орындамаған кезде орын алуы мүмкін.Мақсаты: Студент есте сақтау немесе итеративті тәсілге көшу сияқты терең рекурсиямен байланысты мәселелерді болдырмау әдістерін түсінеді.
4. Құйрық рекурсиясы дегеніміз не және ол не үшін маңызды?

Құйрық рекурсия-бұл рекурсияның ерекше жағдайы, мұнда рекурсивті қоңырау функцияда орындалатын соңғы операция болып табылады. Басқаша айтқанда, каудальды рекурсия функция денесінің соңында рекурсивті қоңырау болған кезде пайда болады және одан кейін ешқандай операция жасалмайды.

* + Мақсаты: Студенттердің құйрық рекурсиясы түсінігі және оның артықшылықтары туралы білімдерін тексеру.

1. Рекурсивті функцияда есте сақтау қалай жүзеге асырылады?
   * Рекурсивті функциядағы жад әдетте қоңырау стекімен басқарылады (call stack). Функция шақырылған сайын қоңырау мәтінмәні, оның ішінде функция аргументтері, жергілікті айнымалылар және қайтару мекен-жайы (функция шақыруы аяқталғаннан кейін қайтып келетін мекен-жай) стектің жоғарғы жағына орналастырылады. Функцияны шақыру аяқталған кезде, бұл контекст стектен жойылады.
   * Рекурсивті функция өзін-өзі шақырған кезде, қоңыраудың жаңа контексті алдыңғысының үстіне қойылады. Бұл процесс әр рекурсивті қоңырауда қайталанады, әйнекте қоңырау тізбегін жасайды.Мақсаты: есте сақтау сияқты рекурсивті қоңырауларды оңтайландыру механизмдерін түсінуді бағалау.
2. Есептерді шешудің рекурсивті және итеративті тәсілдерін салыстыра аласыз ба?
   * Әрине, бірнеше критерийлер бойынша есептерді шешудің рекурсивті және итеративті тәсілдерін салыстырайық:
   * 1. \*\* Кодтың оқылуы:\*\*
   * - \*\*Рекурсивті тәсіл: \* \* рекурсивті функцияның коды көбінесе оқылады және түсінікті болады, әсіресе егер тапсырма табиғи түрде кішігірім ішкі тапсырмаларға бөлінсе.
   * - \*\*Итеративті тәсіл: \* \* итеративті шешім кодын түсіну қиынырақ болуы мүмкін, әсіресе оның құрамында көптеген циклдар мен шартты операторлар болса.
   * 2. \*\* Өнімділік:\*\*
   * - \*\*Рекурсивті тәсіл: \* \* рекурсивті қоңыраулар есте сақтау қабілеті мен орындалу уақыты бойынша тиімсіз болуы мүмкін, әсіресе рекурсияның тереңдігі үлкен болса немесе қайталанатын қоңыраулар көп болса.
   * - \*\*Итеративті тәсіл: \* \* итеративті шешімдер көбінесе жады мен орындалу уақытында тиімдірек болады, өйткені олар стекке қоңырау контекстін сақтауды қажет етпейді.
   * 3. \*\* Жадты басқару:\*\*
   * - \*\*Рекурсивті тәсіл: \* \* рекурсивті қоңыраулар қоңырау стегін пайдаланады, бұл терең рекурсияларда стектің толып кетуіне әкелуі мүмкін.
   * - \*\*Итеративті тәсіл: \* \* итеративті шешімдер әдетте қоңырау стегін пайдалануды қажет етпейді және жадты тиімдірек басқара алады.
   * 4. \*\* Икемділік:\*\*
   * - \*\*Рекурсивті тәсіл: \* \* рекурсивті функциялар көбінесе икемді және жан-жақты болады, өйткені олар тапсырманы қарапайым ішкі есептерге бөлу принципіне негізделген көптеген мәселелерді шеше алады.
   * - \*\*Итеративті тәсіл: \* \* итеративті шешімдер аз икемді және белгілі бір тапсырмаға тән болуы мүмкін, бірақ олар белгілі бір тапсырманы орындау үшін тиімдірек болуы мүмкін, әсіресе жад пен жұмыс уақытын оңтайландыру қажет болса.
   * Сонымен, рекурсивті және итеративті тәсілдер арасындағы таңдау нақты тапсырмаға, оның ерекшеліктеріне және өнімділік талаптарына байланысты. Екі тәсілдің де артықшылықтары мен кемшіліктері бар және оларды тапсырманың контексті мен шарттарын ескере отырып қолдану керек.Мақсаты: Рекурсивті және итеративті әдістердің айырмашылығын түсіну және оларды салыстыра білу.
3. Рекурсивті функцияларды жазу кезінде қандай қателер болуы мүмкін және оларды қалай болдырмауға болады?

При написании рекурсивных функций могут возникать различные ошибки. Некоторые из наиболее распространенных ошибок включают:

1. \*\*Бесконечная рекурсия:\*\* Это случается, когда условие выхода из рекурсии никогда не достигается, и функция вызывает саму себя бесконечное количество раз. Это приводит к переполнению стека вызовов и ошибке "RecursionError". Чтобы избежать этой ошибки, убедитесь, что ваша рекурсивная функция имеет базовый случай, который гарантирует завершение рекурсии.

2. \*\*Неправильный базовый случай:\*\* Если базовый случай в рекурсивной функции неправильно определен или упущен, это может привести к неправильным результатам или бесконечной рекурсии. Убедитесь, что базовый случай корректно определяет условие выхода из рекурсии.

3. \*\*Неоптимальная рекурсия:\*\* Некоторые рекурсивные алгоритмы могут быть неоптимальными и иметь большую сложность по времени или памяти из-за чрезмерного количества рекурсивных вызовов или повторяющихся вычислений. Попробуйте оптимизировать ваш алгоритм или рассмотреть альтернативные подходы, такие как использование итерации или хвостовой рекурсии.

4. \*\*Проблемы с памятью:\*\* В рекурсивных функциях может произойти переполнение стека вызовов из-за большого количества рекурсивных вызовов или больших объемов данных. Попробуйте уменьшить глубину рекурсии или оптимизировать использование памяти.

5. \*\*Неоптимальное использование ресурсов:\*\* Некоторые рекурсивные алгоритмы могут быть неоптимальными из-за излишнего использования ресурсов, таких как процессорное время или память. Рассмотрите возможность оптимизации вашего алгоритма или выбора другого метода решения задачи.

Чтобы избежать этих ошибок, важно тщательно проектировать и тестировать вашу рекурсивную функцию, убедившись, что она корректно обрабатывает все возможные случаи входных данных и не приводит к нежелательным последствиям.

* + Мақсаты: Студенттің негізгі регистрді жіберіп алу немесе дұрыс емес күй өзгерістері сияқты жиі кездесетін қателерді білуін қамтамасыз ету.

1. Ағаштар мен графиктер сияқты деректер құрылымдарын айналып өту үшін рекурсияны қалай пайдалануға болады?

Рекурсия часто используется для обхода структур данных, таких как деревья и графы, потому что эти структуры естественно рекурсивны. Рекурсивный обход позволяет легко и эффективно обрабатывать каждый узел или вершину структуры данных и выполнять различные операции на каждом уровне вложенности. Вот как можно использовать рекурсию для обхода деревьев и графов:

1. **Обход деревьев:**
   * **Прямой (pre-order) обход:** Сначала обрабатывается корневой узел, затем левое поддерево, затем правое поддерево.
   * **Симметричный (in-order) обход:** Сначала обрабатывается левое поддерево, затем корневой узел, затем правое поддерево. Этот тип обхода часто используется для обхода бинарных деревьев поиска.
   * **Обратный (post-order) обход:** Сначала обрабатывается левое поддерево, затем правое поддерево, затем корневой узел.

**Обход графов:**

* **Обход в глубину (Depth-First Search, DFS):** При обходе в глубину мы идем от стартовой вершины до конца каждой ветки графа перед тем, как вернуться назад и перейти к следующей ветви.
* **Обход в ширину (Breadth-First Search, BFS):** При обходе в ширину мы идем от стартовой вершины к ее соседям, затем к соседям соседей и так далее, пока не посетим все вершины на текущем уровне перед переходом к следующему уровню.

Рекурсивный подход к обходу структур данных обычно более лаконичен и интуитивно понятен, чем итеративные методы, но он может быть менее эффективным в некоторых случаях из-за возможного переполнения стека вызовов при обходе очень больших структур данных.

* + Мақсаты: Студенттердің күрделі деректер құрылымдарын өту үшін рекурсияны пайдалану туралы түсінігін тексеру.

# Примерная Задача для Лабораторной Работы 3

**Задача: Рекурсивное Решение Задачи о Ханойских Башнях**

Задача о Ханойских башнях — классическая задача, решаемая с использованием рекурсии. Цель состоит в том, чтобы переместить диски с одного стержня на другой, придерживаясь определенных правил.

Правила:

1. За один раз можно перемещать только один диск.
2. Диск можно класть только на пустой стержень или на диск большего размера.

Цель задачи:

Написать рекурсивную функцию, которая выводит последовательность шагов для решения задачи о Ханойских башнях для n дисков.

Решение:

def hanoi\_towers(n, source, target, auxiliary): """

Решение задачи о Ханойских башнях. n - количество дисков

source - начальный стержень target - целевой стержень

auxiliary - вспомогательный стержень """

if n > 0:

# Переместить n-1 дисков на вспомогательный стержень hanoi\_towers(n-1, source, auxiliary, target)

# Переместить оставшийся диск на целевой стержень print(f"Переместить диск с {source} на {target}")

# Переместить n-1 дисков с вспомогательного на целевой стержень hanoi\_towers(n-1, auxiliary, target, source)

# Пример использования функции hanoi\_towers(3, 'A', 'C', 'B')

Объяснение:

Функция `hanoi\_towers` решает задачу о Ханойских башнях, используя рекурсивный подход.

* Если `n` больше 0, сначала перемещаем `n-1` дисков на вспомогательный стержень.
* Затем перемещаем самый большой диск на целевой стержень.
* После этого перемещаем `n-1` дисков с вспомогательного стержня на целевой, следуя тем же правилам.

Эта задача иллюстрирует классическое применение рекурсии и помогает студентам лучше понять концептуальные основы и практическую реализацию рекурсивных функций.