ҚазақстанРеспубликасыОқу-ағартуминистрлігі

«Білім» кәсіби гуманитарлық-техникалық колледжі



**Зертханалық жұмыс №4**

**Тақырыбы: Кіріктірілген циклдерді ұйымдастыру, программаны өңдеу**

|  |  |  |  |
| --- | --- | --- | --- |
| № | Жұмыстың орындау сапасы | Баға диапазоны | Орындаған % |
| 1 | Орындалған жоқ, сабақта себепсіз болмады. | 0 % |  |
| 2 | Жұмыстың орындалуы және студенттің белсенділігі | 0-50% |  |
| 3 | Жұмысты рәсімдеу | 0-20% |  |
| 4 | Анықтамалар мен техникалық әдістемелерді, пәннің оқу-әдістемелік кешенін, лекция конспектілерін қолдана білу. | 0-5% |  |
| 5 | Техникалық құралдарды пайдалана білу | 0-5% |  |
| 6 | Жұмысты қорғау | 0-20% |  |
|  | Қорытынды | 0-100% |  |

Оқытушы: Нургисаева У.М.

Студент:Бегжан Бексұлтан

Мамандығы:3БҚ-1-22

Тараз 2025

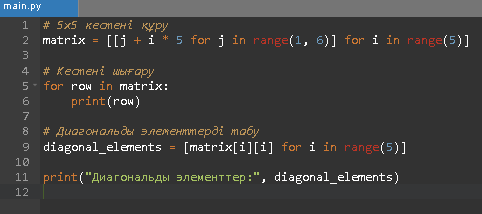
**Зертханалықжұмыс: Кіріктірілгенциклдерменжұмыс**

**Мақсаты:**Кіріктірілгенциклдердіңқұрылымынтүсіну.Кіріктірілген циклдерді дұрыс ұйымдастыруды үйрену.Ішкі және сыртқы циклдерді пайдалана отырып, әртүрлі есептерді шешу.

**Жұмыстың тапсырмалары:**

**1. 5x5 кестенің диагоналін табу**

5x5 өлшеміндегі кесте құрып, оның диагональды элементтерін табуыңыз керек. Диагональды элементтер деп әр қатар мен бағанның индексі тең болатын элементтерді айтамыз. Мысалы, (1,1), (2,2), (3,3), (4,4), (5,5).



Разберём код построчно:

**1. Создание 5x5 матрицы:**

matrix = [[j + i \* 5 for j in range(1, 6)] for i in range(5)]

* Это **генератор списка**, который создаёт двумерный список (матрицу) размером 5x5.
* i — индекс строки (от 0 до 4).
* j — индекс столбца (от 1 до 5).
* j + i \* 5 создаёт элементы таблицы, где первая строка содержит числа от 1 до 5, вторая — от 6 до 10 и так далее.

**Пример полученной матрицы:**

[1, 2, 3, 4, 5]

[6, 7, 8, 9, 10]

[11, 12, 13, 14, 15]

[16, 17, 18, 19, 20]

[21, 22, 23, 24, 25]

**2. Вывод матрицы:**

for row in matrix:

print(row)

* for row in matrix проходит по каждой строке матрицы и выводит её на экран.

**3. Нахождение диагональных элементов:**

diagonal\_elements = [matrix[i][i] for i in range(5)]

* Используется **генератор списка** для извлечения элементов, где индекс строки совпадает с индексом столбца (т.е. [i][i]).
* В нашем примере это:
* [1, 7, 13, 19, 25]

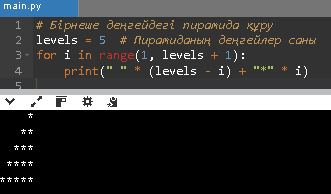
**4. Вывод диагонали:**

print("Диагональды элементтер:", diagonal\_elements)

* Выводит найденные диагональные элементы.

**2. Бірнеше деңгейдегі пирамида құру**

Жұлдызшалардан бірнеше деңгейлі пирамида құру керек. Пирамиданың әр деңгейінде жұлдызшалар саны артатын болады. Алғашқы деңгейде 1 жұлдызша, екінші деңгейде 2 жұлдызша, үшінші деңгейде 3 жұлдызша және солай жалғасады.



Разберём код построчно:

**1. Определение количества уровней пирамиды:**

levels = 5 # Пирамиданың деңгейлер саны

* Здесь переменная levels задаёт количество уровней пирамиды (в данном случае 5).

**2. Цикл для создания уровней:**

for i in range(1, levels + 1):

* range(1, levels + 1) создаёт диапазон от 1 до 5 (включительно), где i — текущий уровень пирамиды.
* Например, если levels = 5, то i принимает значения: 1, 2, 3, 4, 5.

**3. Формирование и вывод строки:**

print(" " \* (levels - i) + "\*" \* i)

* " " \* (levels - i) — создаёт отступы (пробелы) слева, чтобы пирамида была выровнена по центру.
  + Когда i = 1, пробелов будет 5 - 1 = 4, когда i = 2, пробелов 5 - 2 = 3 и так далее.
* "\*" \* i — создаёт строку из i звёздочек.
  + Когда i = 1, звёздочек 1, когда i = 2, звёздочек 2 и так далее.

**Как это работает?**

При levels = 5 программа выведет:

\* # (4 пробела + 1 звёздочка)

\*\* # (3 пробела + 2 звёздочки)

\*\*\* # (2 пробела + 3 звёздочки)

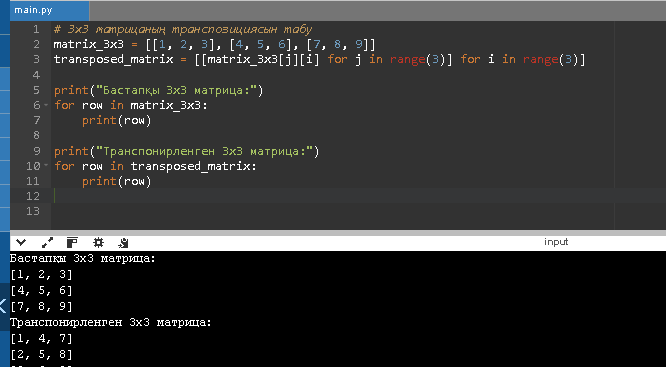
\*\*\*\* # (1 пробел + 4 звёздочки)

\*\*\*\*\* # (0 пробелов + 5 звёздочек)

Таким образом, программа строит пирамиду, увеличивая количество звёздочек на каждом уровне.

**3. 3x3 матрицаның транспозициясын табу**

3x3 өлшеміндегі матрицаның транспозициясын табуыңыз керек. Транспозиция дегеніміз матрицаның жолдары мен бағандарын ауыстыру. Яғни, (1,2) элементі (2,1) элементіне айналады.



**Разбор кода по шагам:**

**1. Исходная 3x3 матрица:**

matrix\_3x3 = [[1, 2, 3],

[4, 5, 6],

[7, 8, 9]]

* Это список списков, представляющий 3x3 матрицу:
* 1 2 3
* 4 5 6
* 7 8 9

**2. Транспонирование матрицы:**

transposed\_matrix = [[matrix\_3x3[j][i] for j in range(3)] for i in range(3)]

* Здесь используется **вложенный генератор списка**:
  + Внешний цикл for i in range(3) отвечает за создание новых строк в транспонированной матрице.
  + Внутренний цикл for j in range(3) проходит по столбцам исходной матрицы и собирает элементы в новую строку.
  + matrix\_3x3[j][i] меняет местами строки и столбцы.

Как это работает:

* i = 0: Берём все элементы из **первого столбца** исходной матрицы → [1, 4, 7]
* i = 1: Берём все элементы из **второго столбца** → [2, 5, 8]
* i = 2: Берём все элементы из **третьего столбца** → [3, 6, 9]

В результате новая матрица будет:

1 4 7

2 5 8

3 6 9

**3. Вывод исходной матрицы:**

print("Бастапқы 3x3 матрица:")

for row in matrix\_3x3:

print(row)

* Проходит по каждой строке matrix\_3x3 и выводит её.

**4. Вывод транспонированной матрицы:**

print("Транспонирленген 3x3 матрица:")

for row in transposed\_matrix:

print(row)

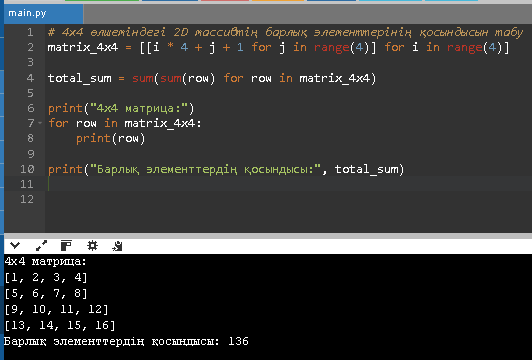
* Аналогично, выводит строки транспонированной матрицы.

**Итог:**

Этот код эффективно транспонирует 3x3 матрицу, заменяя строки на столбцы.

**4. 2D массивтің барлық элементтерінің қосындысын табу**

4x4 өлшеміндегі 2D массивтің барлық элементтерінің қосындысын табу қажет. Мысалы, 4x4 массивтің әрбір элементі қосылып, оның жалпы қосындысын шығару керек.



**Разбор кода:**

**1. Создание 4x4 матрицы:**

matrix\_4x4 = [[i \* 4 + j + 1 for j in range(4)] for i in range(4)]

* Здесь используется **вложенный генератор списка**:
  + i — индекс строки (меняется от 0 до 3).
  + j — индекс столбца (меняется от 0 до 3).
  + (i \* 4 + j + 1) — формула, по которой вычисляются элементы матрицы.

**Как работает?**

* i = 0: [0 \* 4 + 0 + 1, 0 \* 4 + 1 + 1, 0 \* 4 + 2 + 1, 0 \* 4 + 3 + 1] → [1, 2, 3, 4]
* i = 1: [1 \* 4 + 0 + 1, 1 \* 4 + 1 + 1, 1 \* 4 + 2 + 1, 1 \* 4 + 3 + 1] → [5, 6, 7, 8]
* i = 2: [9, 10, 11, 12]
* i = 3: [13, 14, 15, 16]

В итоге формируется 4x4 матрица:

1 2 3 4

5 6 7 8

9 10 11 12

13 14 15 16

**2. Подсчет суммы всех элементов:**

total\_sum = sum(sum(row) for row in matrix\_4x4)

* sum(row) for row in matrix\_4x4 — вычисляет сумму элементов в каждой строке.
* sum(...) — суммирует полученные значения и выводит общий результат.

**Как работает?**

* sum([1, 2, 3, 4]) = 10
* sum([5, 6, 7, 8]) = 26
* sum([9, 10, 11, 12]) = 42
* sum([13, 14, 15, 16]) = 58
* Общая сумма: 10 + 26 + 42 + 58 = 136

**3. Вывод матрицы:**

print("4x4 матрица:")

for row in matrix\_4x4:

print(row)

* Этот цикл проходит по строкам матрицы и выводит их.

print("Барлық элементтердің қосындысы:", total\_sum)

* Выводит сумму всех элементов.

**Итог:**

Код создает матрицу 4x4, вычисляет сумму всех ее элементов и выводит результат.  
Вывод программы:

4x4 матрица:

[1, 2, 3, 4]

[5, 6, 7, 8]

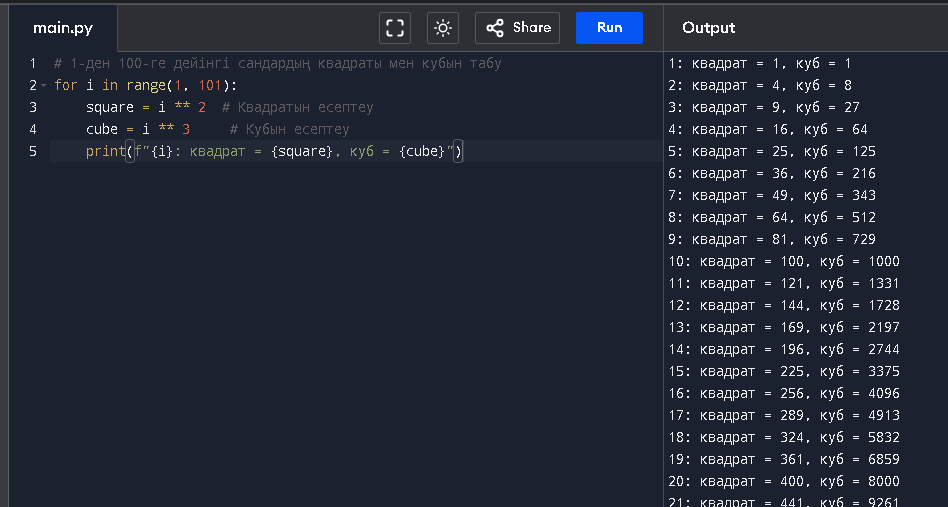
[9, 10, 11, 12]

[13, 14, 15, 16]

Барлық элементтердің қосындысы: 136

**5. 1-ден 100-ге дейінгісандардыңкубтарын табу**

1-ден 100-ге дейінгісандардың квадраты мен кубын табу қажет. Әр сан үшіноныңквадратынжәнекубынесептепшығарукерек.



Этот код на языке Python вычисляет квадраты и кубы чисел от 1 до 100 и выводит их на экран в определённом формате. Давайте разберем его по частям:

for i in range(1, 101): # Цикл от 1 до 100 включительно

square = i \*\* 2 # Возведение числа i в квадрат

cube = i \*\* 3 # Возведение числа i в куб

print(f"{i}: квадрат = {square}, куб = {cube}") # Вывод числа i, его квадрата и куба

1. **for i in range(1, 101):** — цикл, который перебирает значения переменной i от 1 до 100 включительно (поскольку range(1, 101) генерирует числа от 1 до 100).
2. **square = i \*\* 2** — вычисляется квадрат числа i (то есть число возводится в степень 2).
3. **cube = i \*\* 3** — вычисляется куб числа i (то есть число возводится в степень 3).
4. **print(f"{i}: квадрат = {square}, куб = {cube}")** — выводится строка, содержащая текущее значение числа i, его квадрат и куб в формате: "i: квадрат = square, куб = cube".

Этот код будет выполнять вычисления для всех чисел от 1 до 100 и выводить результат на экран. Например, для числа 1 вывод будет таким:

1: квадрат = 1, куб = 1

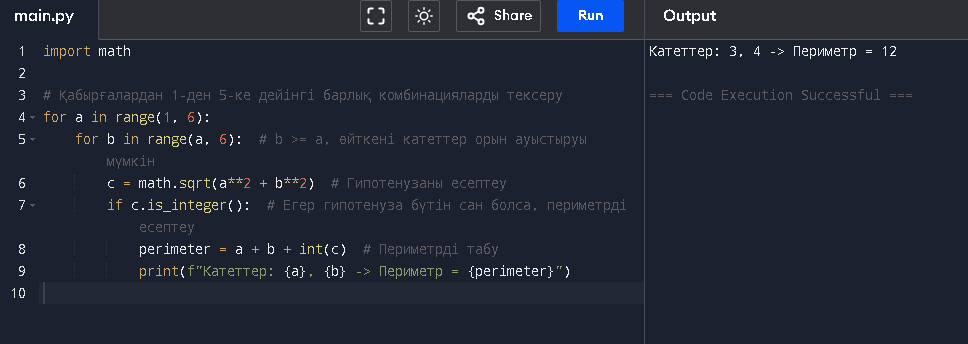
Для числа 2:

2: квадрат = 4, куб = 8

И так далее для всех чисел от 1 до 100.

**6. Тікбұрыштыүшбұрыштыңпериметрінесептеу**

1-ден 5-ке дейінгіқабырғалардантікбұрыштыүшбұрыштарқұрып, олардыңпериметрін табу керек. Тікбұрыштыүшбұрыштыңпериметрінесептеуүшінқабырғалардыңұзындықтарыарқылыформуланықолданасыз.



**Пояснение:**

1. **for a in range(1, 6):** — перебираются все возможные значения для первого катета aaa от 1 до 5.
2. **for b in range(a, 6):** — для второго катета bbb перебираются значения от aaa до 5. Это условие позволяет избежать дублирования пар, например, если a=3a = 3a=3 и b=4b = 4b=4, то пара b=3b = 3b=3 и a=4a = 4a=4 уже не будет проверяться.
3. **c = math.sqrt(a\*\*2 + b\*\*2)** — по теореме Пифагора находим гипотенузу ccc, которая равна квадратному корню из суммы квадратов катетов aaa и bbb.
4. **if c.is\_integer():** — проверяем, является ли гипотенуза целым числом, чтобы треугольник был правильным (с целыми сторонами).
5. **perimeter = a + b + int(c)** — если гипотенуза целое число, то вычисляем периметр, который равен сумме всех сторон.
6. **print(f"Катеты: {a}, {b} -> Периметр = {perimeter}")** — выводим результат.

**Пример результата:**

Для значений катетов от 1 до 5, программа выведет периметр правильных прямоугольных треугольников. Например:

makefile

КопироватьРедактировать

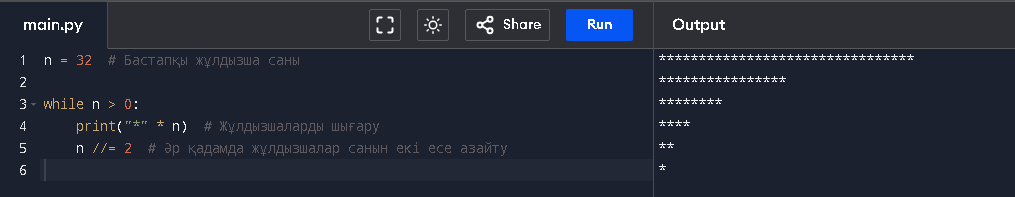
Катеты: 3, 4 -> Периметр = 12

Катеты: 4, 3 -> Периметр = 12

В этом примере прямоугольный треугольник с катетами 3 и 4 имеет гипотенузу, равную 5 (по теореме Пифагора), а периметр будет равен 3+4+5=123 + 4 + 5 = 123+4+5=12.

**7. Қарама-қарсыбағыттағыжұлдызшалардан пирамида құру**

Жұлдызшаларданкерібағытта пирамида құрукерек. Яғни, біріншідеңгейдееңкөпжұлдызшаболады, ал әрдеңгейдежұлдызшалар саны екіесеазаяды.

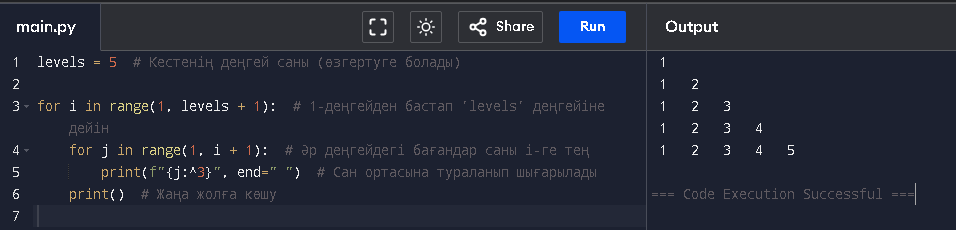


**Кодтың түсіндірмесі:**

1. **n = 32** — бастапқы деңгейдегі жұлдызшалар саны (сіз оны өз қалауыңыз бойынша өзгерте аласыз).
2. **while n > 0:** — цикл жұлдызшалар саны 0-ден үлкен болғанша жұмыс істейді.
3. **print("\*" \* n)** — n жұлдызшадан тұратын жолды басып шығарады.
4. **n //= 2** — жұлдызшалар санын екі есе азайтады (//= бүтін бөлу операторын қолданады).

**8. Көпдеңгейлікестеқұру**

Әр деңгейдебағандар саны көбейетінкестеқұрукерек. Әрдеңгейдебағандардың саны деңгейдіңнөмірінесәйкескеледі. Мысалы, біріншідеңгейде 1 баған, екіншідеңгейде 2 бағанжәнет.с.с.

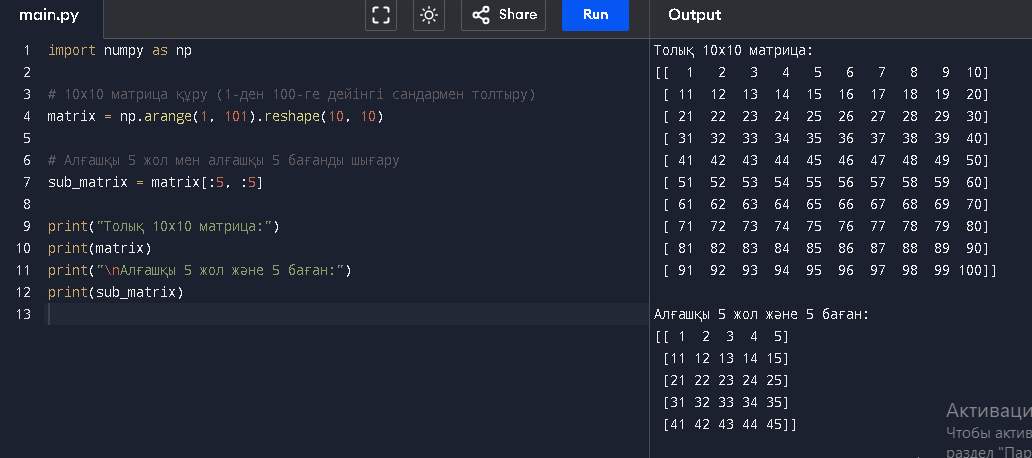


**Кодтың түсіндірмесі:**

1. **levels = 5** — кестенің деңгей санын анықтайды (сіз өзіңізге қажет санды қоя аласыз).
2. **for i in range(1, levels + 1):** — бірінші деңгейден басталып, соңғы деңгейге дейін цикл жұмыс істейді.
3. **for j in range(1, i + 1):** — әр деңгейде баған саны сол деңгей нөміріне тең болады.
4. **print(f"{j:^3}", end=" ")** — сандарды біркелкі шығару үшін оларды ортаға туралаймыз.
5. **print()** — жаңа жолға өту (әр деңгей жаңа қатарда басталады).

**9. 10x10 матрицаныңалғашқы 5 жолы мен бағаныншығару**

10x10 өлшеміндегіматрицаныңалғашқы 5 жолы мен бағаныншығаруқажет. Бұлматрицаныңалғашқы 5 қатарынжәнеалғашқы 5 бағанынбасыпшығарукерек.

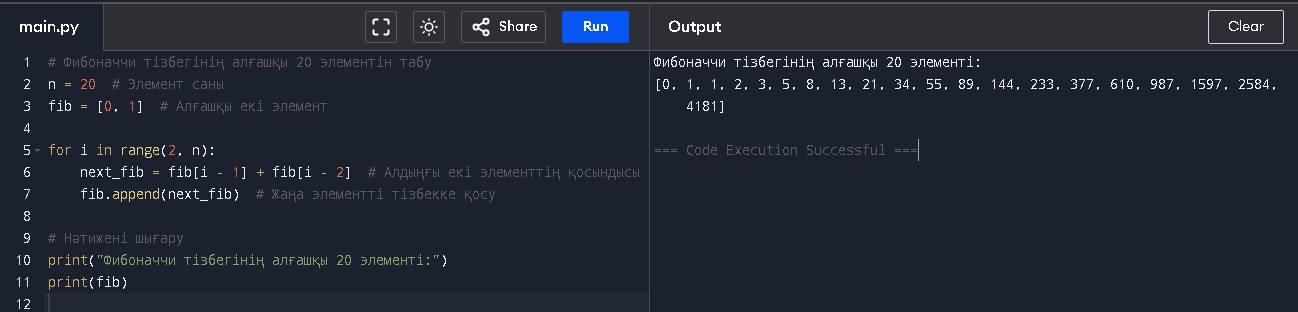


**Кодтың түсіндірмесі:**

1. **np.arange(1, 101).reshape(10, 10)** — 1-ден 100-ге дейінгі сандардан тұратын 10x10 матрица құрады.
2. **matrix[:5, :5]** — матрицаның алғашқы 5 жолы мен алғашқы 5 бағанын бөліп алады.
3. **print(sub\_matrix)** — нәтиже ретінде кіші матрицаны шығарады.

**10. Фибоначчи тізбегініңбірінші 20 элементіншығару**

Фибоначчи тізбегініңалғашқы 20 элементін табу керек. Фибоначчи тізбегіәрбіржаңаэлементіалдыңғыекіэлементтіңқосындысынантұрады. Мысалы, 0, 1, 1, 2, 3, 5, 8, 13, 21, 34 жәнет.б.



**Кодтың түсіндірмесі:**

1. **fib = [0, 1]** — Фибоначчи тізбегінің алғашқы екі элементін (0 және 1) анықтаймыз.
2. **for i in range(2, n):** — тізбекті үшінші элементтен бастап жалғастырамыз.
3. **next\_fib = fib[i - 1] + fib[i - 2]** — әр жаңа элемент алдыңғы екі элементтің қосындысы ретінде есептеледі.
4. **fib.append(next\_fib)** — жаңа есептелген элементті тізбекке қосамыз.
5. **print(fib)** — нәтижені шығарамыз.

**Қорытынды**

**Өзіме жаңа мәлімет жинақтадым**