

Практическая работа 7 - Функции (def, lambda)

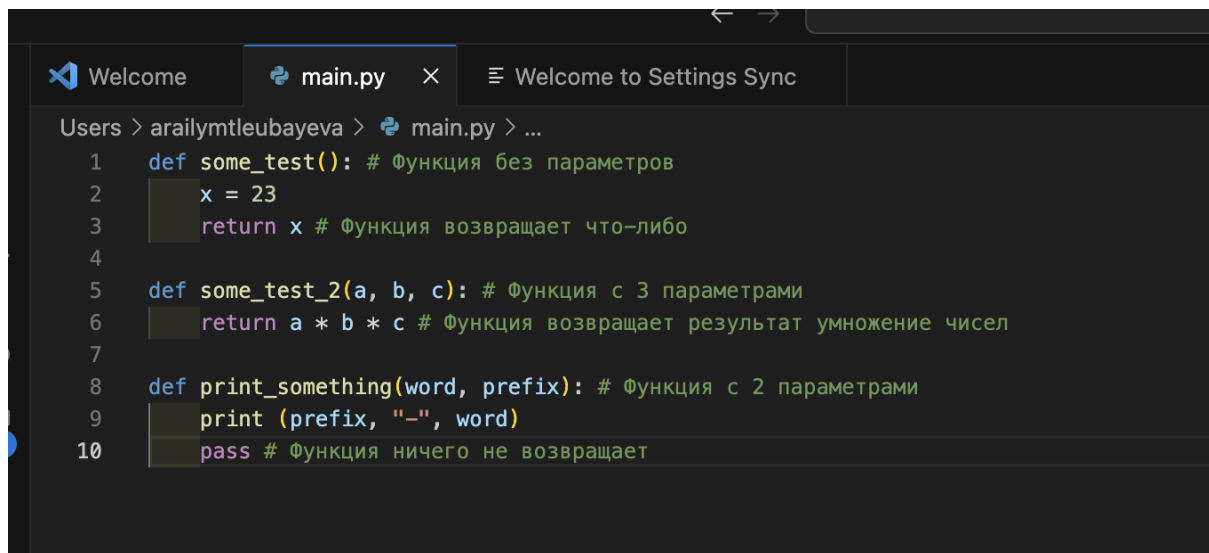
Функции в Python - это блоки кода, которые могут быть вызваны для выполнения конкретных задач. Они являются фундаментальной частью языка Python и позволяют организовывать код в более читаемую и многократно используемую форму. Давайте рассмотрим примеры и практические задачи, связанные с функциями в Python.

Функции можно назвать небольшими подпрограммами, куда можно вынести повторяющийся код и обращаться к нему, когда это будет нужно. Функции значительно облегчают построение программ, так как нам не надо копировать однотипный код множество раз, а можно просто воспользоваться одной общей функцией.

Многие путают функции и методы и не понимают отличий между ними. На самом деле отличий нет, так как что методы, что функции являются одним и тем же. Функции что записаны вне классов называют функциями, а функции что записаны внутри классов называются методами.

Точно такая же ситуация обстоит с переменным. В классах переменные называются полями, а вне классов - переменными.

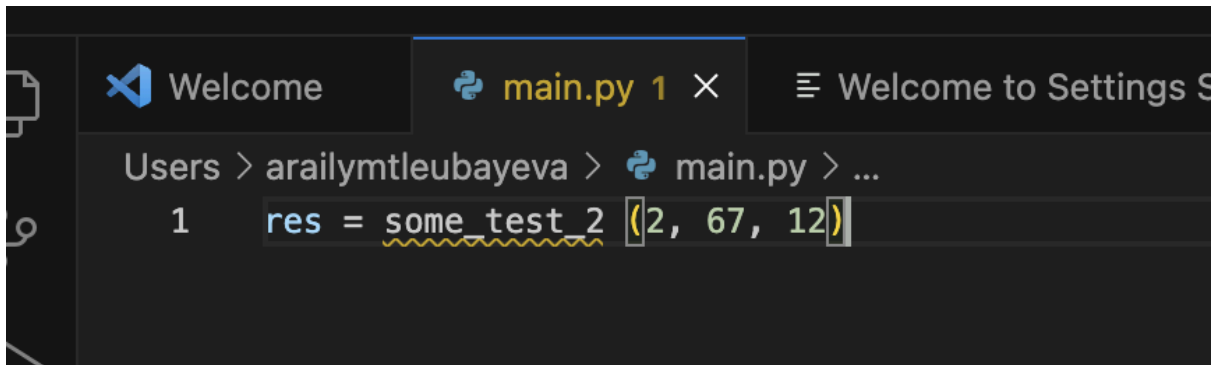
В Python функции создаются при помощи ключевого слова **def**. Каждая функция может иметь какие-либо параметры или же не иметь их вовсе. Функции способны что-либо возвращать в ходе выполнения кода, если это требуется.



```
Users > arailymtleubayeva > main.py > ...
1  def some_test(): # Функция без параметров
2      x = 23
3      return x # Функция возвращает что-либо
4
5  def some_test_2(a, b, c): # Функция с 3 параметрами
6      return a * b * c # Функция возвращает результат умножение чисел
7
8  def print_something(word, prefix): # Функция с 2 параметрами
9      print (prefix, "-", word)
10     pass # Функция ничего не возвращает
```

Когда функция ничего не возвращает, то необходимо прописывать ключевое слово `pass`.

Функции могут возвращать другие функции, тем самым вызывая их. Чтобы обратиться к функции необходимо прописать её название и передать параметры, если таковы имеются:



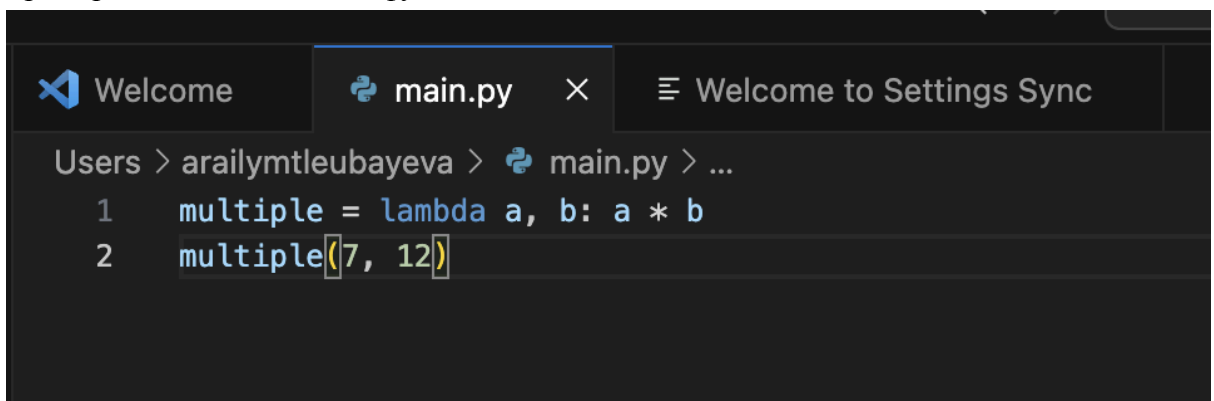
```
Welcome main.py 1 X Welcome to Settings S
Users > arailymtleubayeva > main.py > ...
1 res = some_test_2 (2, 67, 12)
```

В примере выше результат выполнения функции будет помещен в переменную `res`. Далее с переменной можно работать как с обычным значением в программе.

Анонимные функции

Если функция не должна выполнять большой объем кода, то можно воспользоваться анонимной функцией. Для этого потребуется функция `lambda`.

Пример создания «`lambda`» функции:



```
Welcome main.py X Welcome to Settings Sync
Users > arailymtleubayeva > main.py > ...
1 multiple = lambda a, b: a * b
2 multiple(7, 12)
```

Подобная функция не имеет названия, но её можно присвоить к переменной, которую в дальнейшем необходимо вызывать как обычную функцию.

Определение функций

Для определения функции в Python используется ключевое слово `def`, за которым следует имя функции и круглые скобки, содержащие аргументы функции. Тело функции обычно содержит инструкции, которые выполняются при вызове функции.

```
def my_function(arg1, arg2):
    # Тело функции
    result = arg1 + arg2
    return result
```

Вызов функций

Функцию можно вызвать, указав ее имя, аргументы в круглых скобках и, при необходимости, сохранить результат ее выполнения.

```
result = my_function(2, 3) # Вызов функции с аргументами 2 и 3
print(result) # Вывод результата (5)
```

Аргументы функции

Функции могут иметь аргументы, которые передаются при вызове. Аргументы могут быть обязательными или необязательными.

Обязательные аргументы

```
def add(a, b):
    return a + b
```

Аргументы со значениями по умолчанию

```
def greet(name="Гость"):
    return f"Привет, {name}!"
```

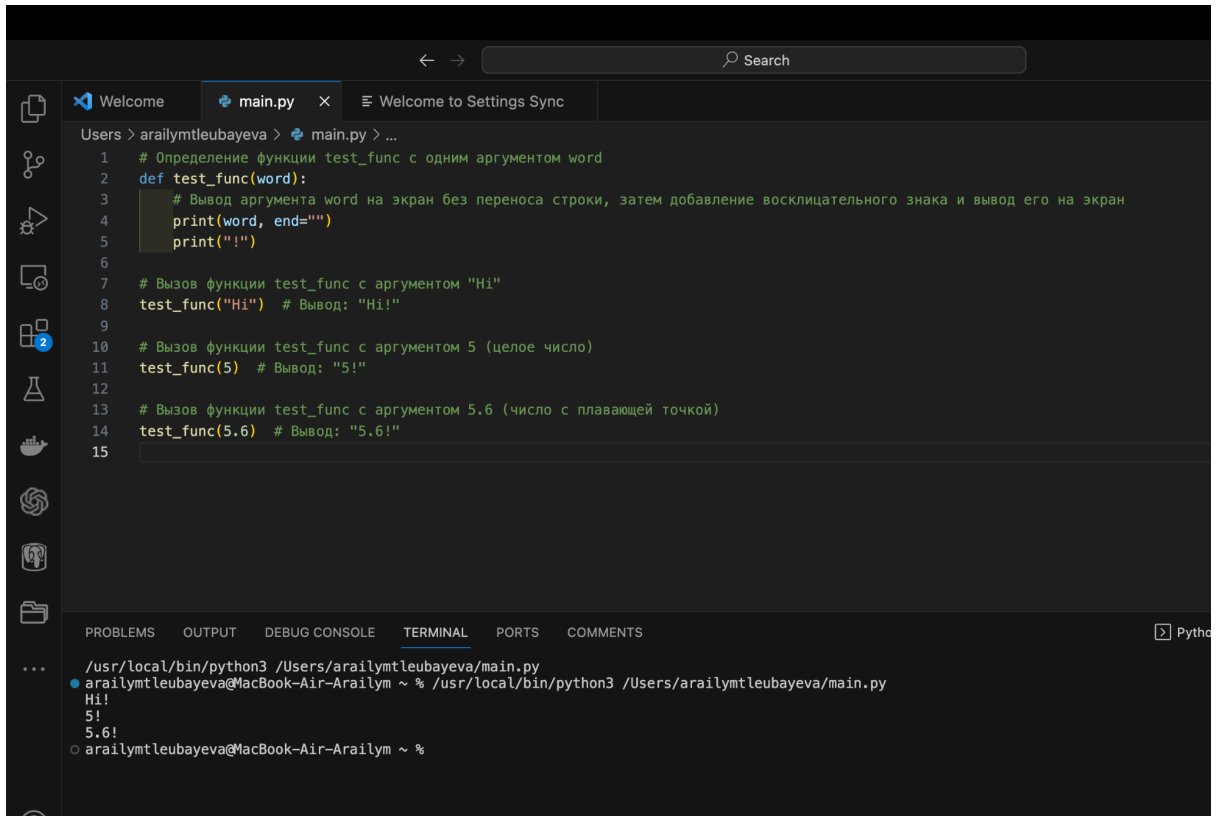
Возвращение значений

Функции могут возвращать результат выполнения с помощью ключевого слова return. Можно вернуть одно или несколько значений.

```
def add_and_multiply(a, b):
    sum_result = a + b
    product_result = a * b
    return sum_result, product_result # Возвращение кортежа
```

Примеры:

Простая функция



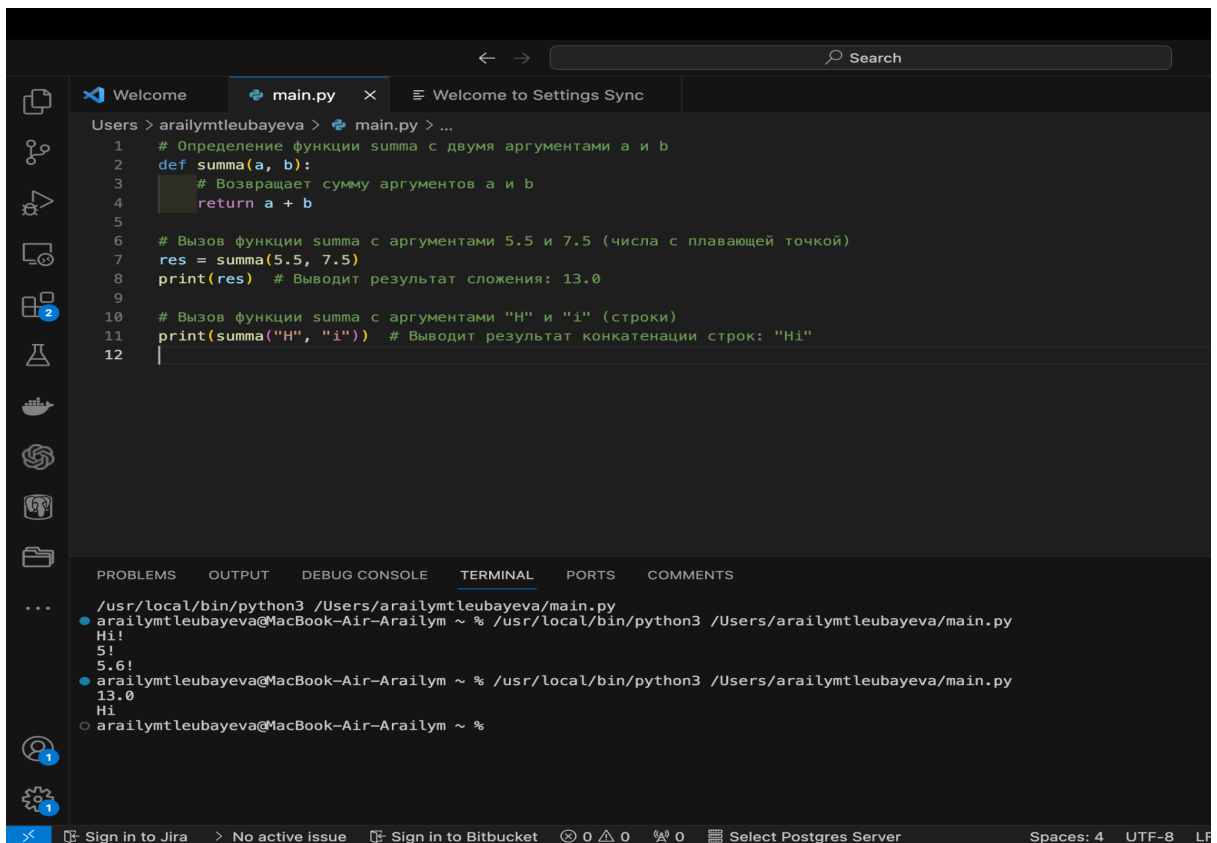
The screenshot shows the VS Code editor with a file named `main.py` open. The code defines a function `test_func` that takes a `word` argument and prints it followed by an exclamation mark. The function is then called with three different arguments: a string, an integer, and a float. The terminal at the bottom shows the execution of the script, which produces the output: `Hi!`, `5!`, and `5.6!`.

```
1 # Определение функции test_func с одним аргументом word
2 def test_func(word):
3     # Вывод аргумента word на экран без переноса строки, затем добавление восклицательного знака и вывод его на экран
4     print(word, end="")
5     print("!")
6
7 # Вызов функции test_func с аргументом "Hi"
8 test_func("Hi") # Вывод: "Hi!"
9
10 # Вызов функции test_func с аргументом 5 (целое число)
11 test_func(5) # Вывод: "5!"
12
13 # Вызов функции test_func с аргументом 5.6 (число с плавающей точкой)
14 test_func(5.6) # Вывод: "5.6!"
15
```

Terminal output:

```
/usr/local/bin/python3 /Users/arailymtleubayeva/main.py
arailymtleubayeva@MacBook-Air-Arilym ~ % /usr/local/bin/python3 /Users/arailymtleubayeva/main.py
Hi!
5!
5.6!
arailymtleubayeva@MacBook-Air-Arilym ~ %
```

Функции с параметрами:



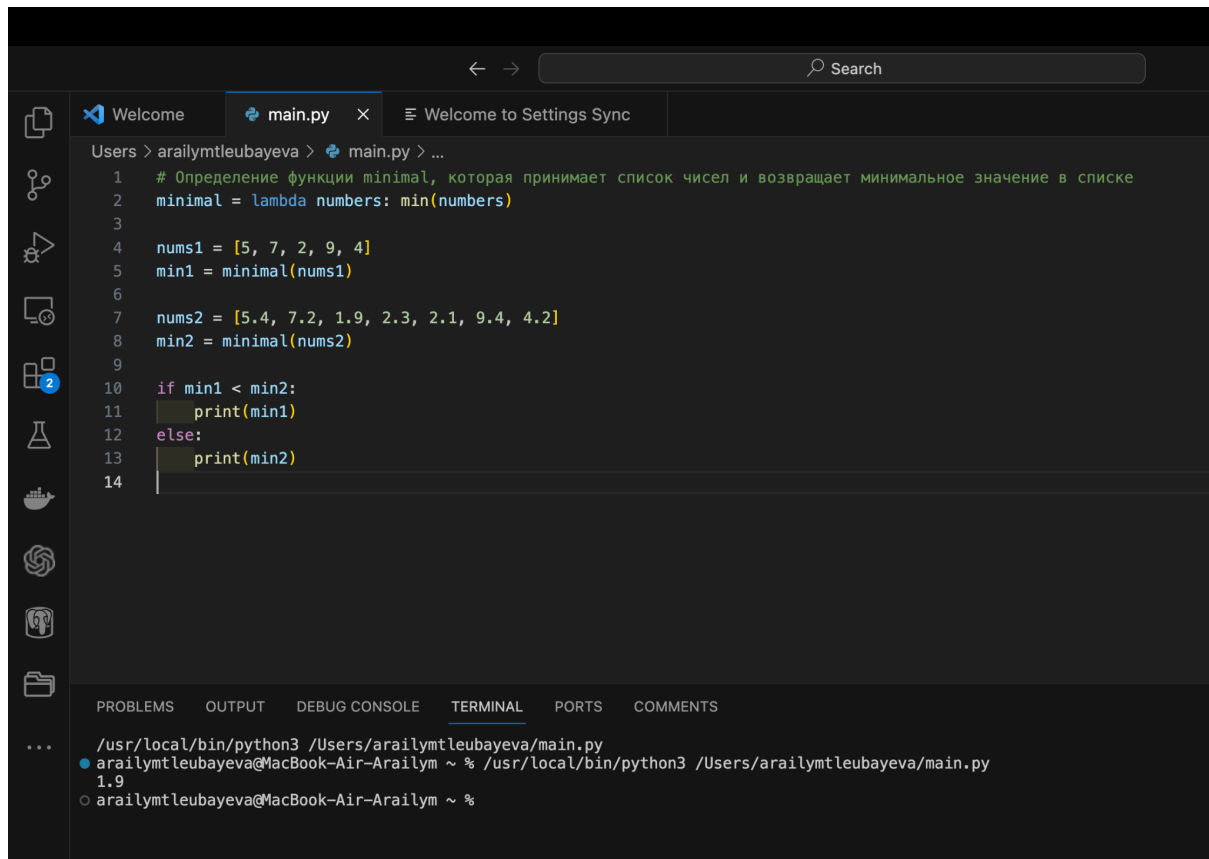
The screenshot shows the VS Code editor with a file named `main.py` open. The code defines a function `summa` that takes two arguments, `a` and `b`, and returns their sum. The function is then called with two different sets of arguments: two floats and two strings. The terminal at the bottom shows the execution of the script, which produces the output: `Hi!`, `5!`, `5.6!`, `13.0`, and `Hi`.

```
1 # Определение функции summa с двумя аргументами a и b
2 def summa(a, b):
3     # Возвращает сумму аргументов a и b
4     return a + b
5
6 # Вызов функции summa с аргументами 5.5 и 7.5 (числа с плавающей точкой)
7 res = summa(5.5, 7.5)
8 print(res) # Выводит результат сложения: 13.0
9
10 # Вызов функции summa с аргументами "H" и "i" (строки)
11 print(summa("H", "i")) # Выводит результат конкатенации строк: "Hi"
12
```

Terminal output:

```
/usr/local/bin/python3 /Users/arailymtleubayeva/main.py
arailymtleubayeva@MacBook-Air-Arilym ~ % /usr/local/bin/python3 /Users/arailymtleubayeva/main.py
Hi!
5!
5.6!
13.0
Hi
arailymtleubayeva@MacBook-Air-Arilym ~ %
```

Функция для нахождения минимума

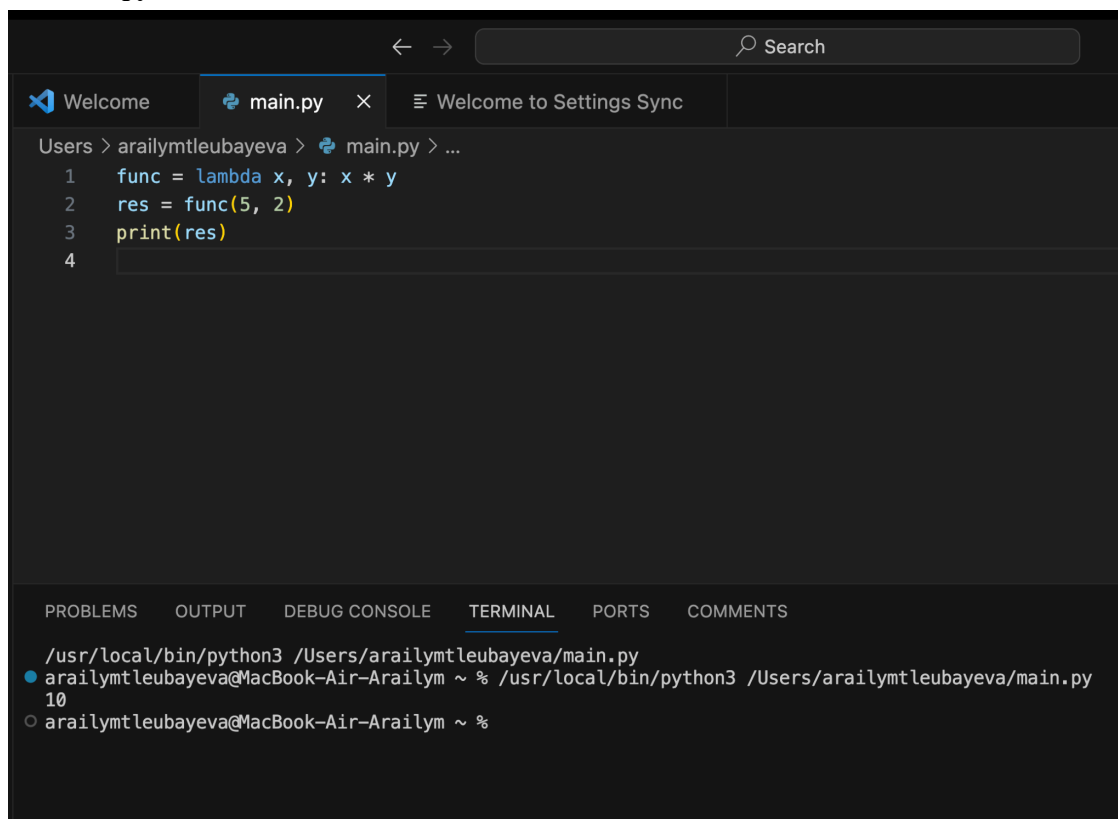


The screenshot shows the Visual Studio Code editor with a file named `main.py` open. The code defines a `minimal` lambda function that takes a list of numbers and returns the minimum value. It then creates two lists, `nums1` and `nums2`, and uses the `minimal` function to find their minimum values. Finally, it prints the minimum of the two results.

```
1 # Определение функции minimal, которая принимает список чисел и возвращает минимальное значение в списке
2 minimal = lambda numbers: min(numbers)
3
4 nums1 = [5, 7, 2, 9, 4]
5 min1 = minimal(nums1)
6
7 nums2 = [5.4, 7.2, 1.9, 2.3, 2.1, 9.4, 4.2]
8 min2 = minimal(nums2)
9
10 if min1 < min2:
11     print(min1)
12 else:
13     print(min2)
14
```

The terminal at the bottom shows the command `/usr/local/bin/python3 /Users/arailymtleubayeva/main.py` being executed, resulting in the output `1.9`.

lambda функция:



The screenshot shows the Visual Studio Code editor with a file named `main.py` open. The code defines a simple lambda function `func` that takes two arguments `x` and `y` and returns their product `x * y`. It then calls `func(5, 2)` and prints the result.

```
1 func = lambda x, y: x * y
2 res = func(5, 2)
3 print(res)
4
```

The terminal at the bottom shows the command `/usr/local/bin/python3 /Users/arailymtleubayeva/main.py` being executed, resulting in the output `10`.

Самостоятельная работа

Простые задачи(50б):

1) Сумма элементов списка:

Напишите функцию, которая принимает список чисел в качестве аргумента и возвращает сумму всех элементов в списке.

2) Поиск максимального элемента:

Напишите функцию, которая принимает список чисел и возвращает наибольший элемент в этом списке.

3) Генерация числовой последовательности:

Напишите функцию, которая принимает начальное и конечное значение и возвращает список чисел в заданном диапазоне.

4) Факториал числа:

Напишите функцию, которая вычисляет факториал целого числа. Факториал числа n - это произведение всех положительных целых чисел от 1 до n .

5) Подсчет количества гласных:

Напишите функцию, которая принимает строку и возвращает количество гласных букв в этой строке.

Средней сложности задачи(50б)

Задача 1: Функция для вычисления среднего(10б)

Напишите функцию `calculate_average`, которая принимает список чисел в качестве аргумента и возвращает среднее значение этих чисел. Затем вызовите эту функцию и выведите результат.

Задача 2: Функция для поиска наибольшей буквы(10б)

Напишите функцию `find_largest_letter`, которая принимает строку и возвращает наибольшую букву (в алфавитном порядке) в этой строке. Затем вызовите эту функцию и выведите результат.

Задача 3: Лямбда-функция для умножения(10б)

Напишите лямбда-функцию, которая принимает два аргумента и возвращает их произведение. Затем вызовите эту лямбда-функцию и выведите результат.

Задача 4: Функция для перевода градусов Цельсия в Фаренгейты(10б)

Напишите функцию `celsius_to_fahrenheit`, которая принимает температуру в градусах Цельсия и возвращает эквивалентную температуру в градусах Фаренгейта. Затем вызовите эту функцию и выведите результат.

Задача 5: Функция для подсчета числа делителей (10б)

Напишите функцию `count_divisors`, которая принимает целое число в качестве аргумента и возвращает количество его делителей. Делитель - это число, на которое заданное число делится без остатка. Затем вызовите эту функцию и выведите результат.