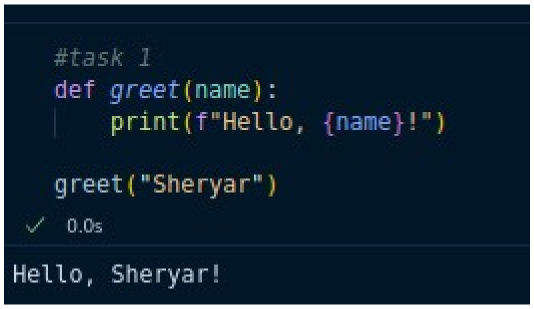
Python Programming Tasks: Functions, Scope, Lambda, Map/Filter, Nested Functions, File & Exception Handling

Tasks

1. Write a simple function `greet` that takes a name as input and prints 'Hello, <name>!'

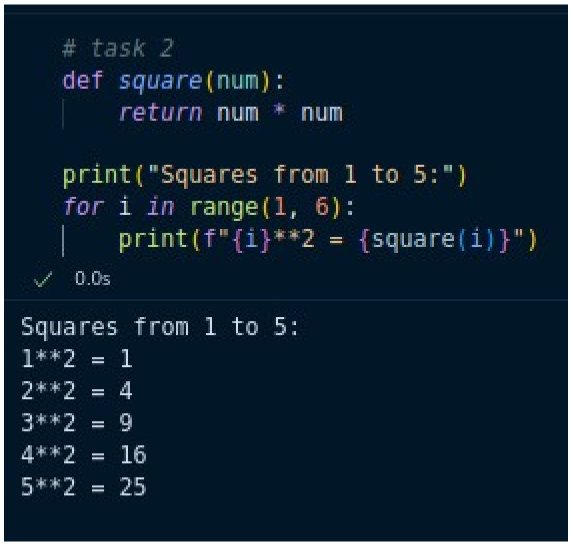


```
#task 1
def greet(name):
    print(f"Hello, {name}!")

greet("Sheryar")
```

```
✓ 0.0s
```

```
Hello, Sheryar!
```

2. Create a function that calculates the square of a number. Demonstrate it with a loop for numbers 1 to 5.

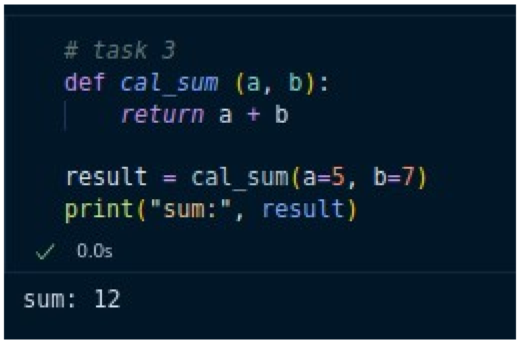


```
# task 2
def square(num):
    return num * num

print("Squares from 1 to 5:")
for i in range(1, 6):
    print(f"{i}**2 = {square(i)}")
```

```
✓ 0.0s
```

```
Squares from 1 to 5:
1**2 = 1
2**2 = 4
3**2 = 9
4**2 = 16
5**2 = 25
```

3. Write a function `add_numbers(a, b)` that returns the sum of two numbers. Call it using keyword arguments.

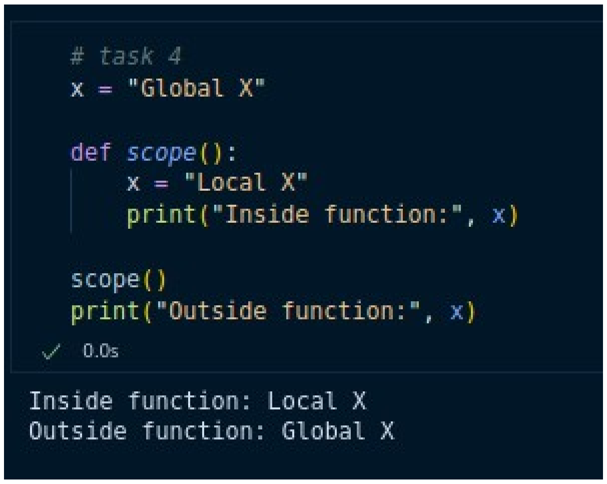


```
# task 3
def cal_sum (a, b):
    return a + b

result = cal_sum(a=5, b=7)
print("sum:", result)
```

✓ 0.0s

```
sum: 12
```

4. Demonstrate variable scope by creating a global variable and a local variable with the same name inside a function. Print both values.

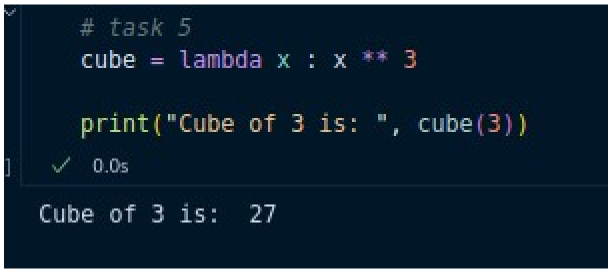


```
# task 4
x = "Global X"

def scope():
    x = "Local X"
    print("Inside function:", x)

scope()
print("Outside function:", x)
```

✓ 0.0s

```
Inside function: Local X
Outside function: Global X
```

5. Use a lambda function to return the cube of a number. Show its result for the number 4.

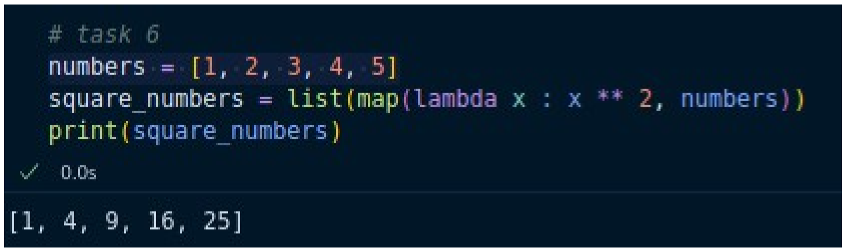


```
# task 5
cube = lambda x : x ** 3

print("Cube of 3 is: ", cube(3))
```

✓ 0.0s

```
Cube of 3 is:  27
```

6. Given a list of numbers, use `map` with a lambda to return their squares.

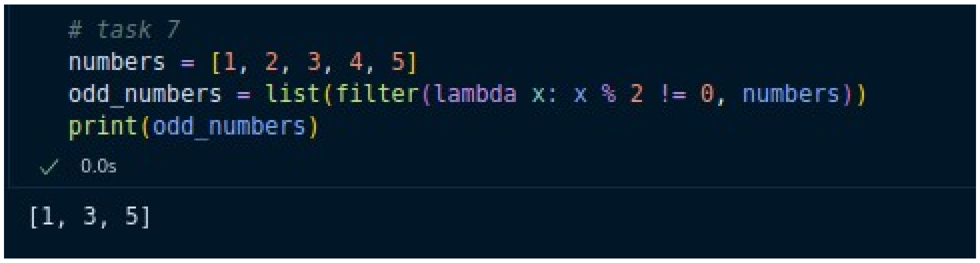


```
# task 6
numbers = [1, 2, 3, 4, 5]
square_numbers = list(map(lambda x : x ** 2, numbers))
print(square_numbers)
```

✓ 0.0s

```
[1, 4, 9, 16, 25]
```

7. Given a list of numbers, use `filter` with a lambda to return only odd numbers.

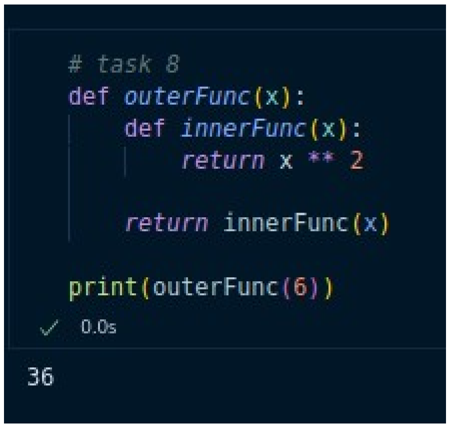


```
# task 7
numbers = [1, 2, 3, 4, 5]
odd_numbers = list(filter(lambda x: x % 2 != 0, numbers))
print(odd_numbers)
```

✓ 0.0s

```
[1, 3, 5]
```

8. Write a nested function `outer` with an inner function `inner` that returns the square of a number. Call the inner function through `outer`.



```
# task 8
def outerFunc(x):
    def innerFunc(x):
        return x ** 2

    return innerFunc(x)

print(outerFunc(6))
```

✓ 0.0s

```
36
```

9. Write a program that opens a text file, writes three lines to it, and then reads it back to print its contents.

10. (Advanced) Implement a safe division function `safe_divide(a, b)` that uses exception handling. It should return the result if division is possible, otherwise print a meaningful error message (e.g., 'Cannot divide by zero').