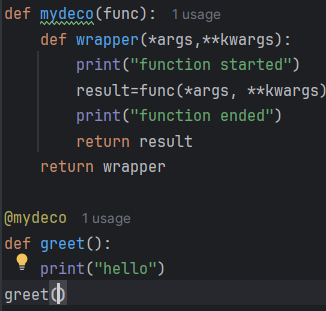
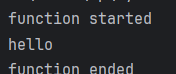
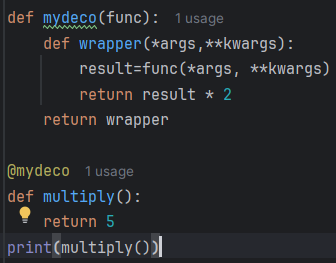
DAY 5: MORNING ASSESSMENT

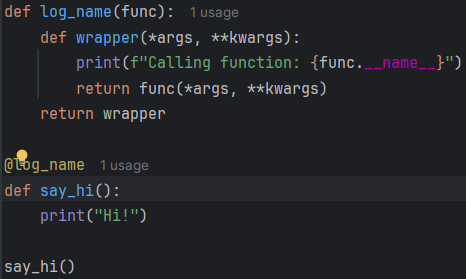
1. Write a decorator to print 'Function started' before a function runs and 'Function ended' after it runs.

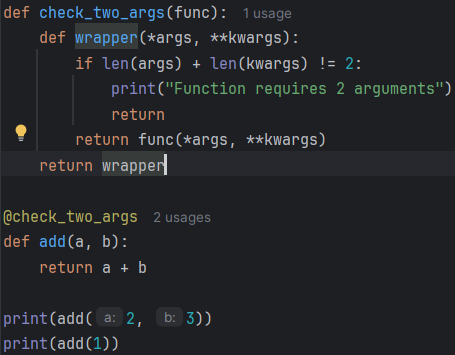
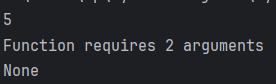
1. Create a decorator that multiplies the return value of a function by 2.

output:10

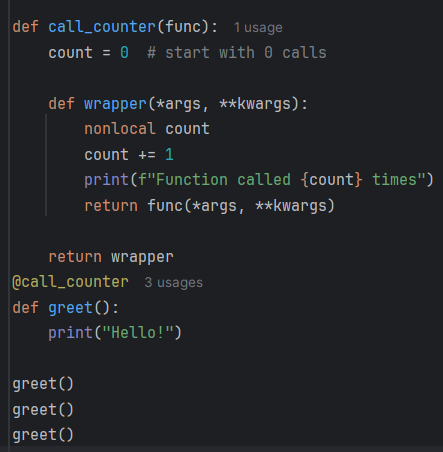
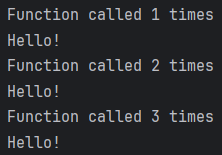
1. Write a decorator that logs the name of the function being called.

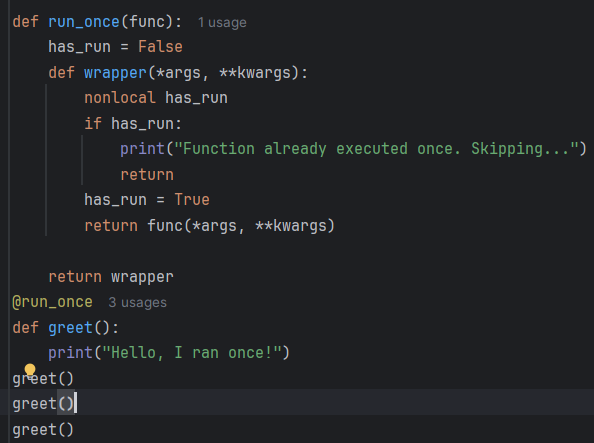
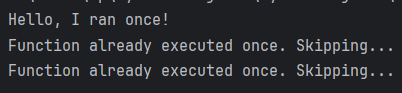
1. Create a decorator to check if the function is called with exactly 2 arguments.

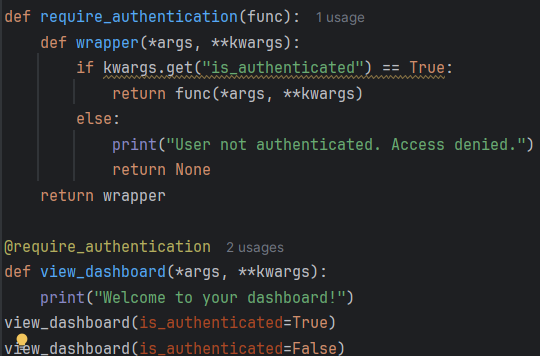
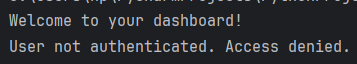
1. Write a decorator that counts and prints how many times the function has been called.

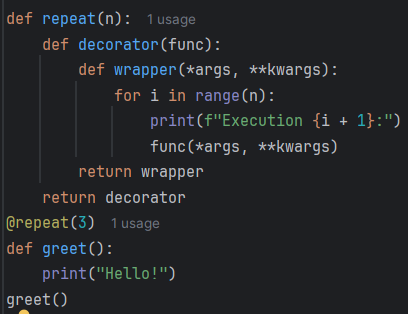
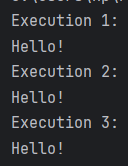
1. Write a decorator that restricts a function from running more than once.

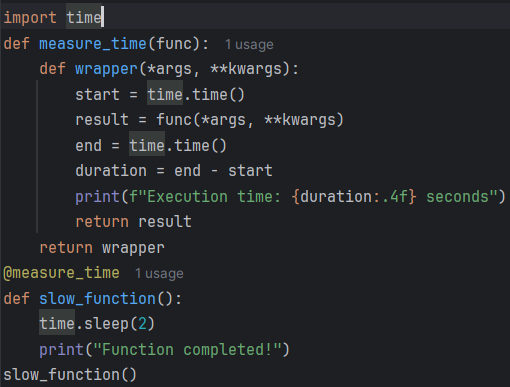
1. Write a decorator to check if a user is authenticated (pass is\_authenticated=True as a keyword argument).

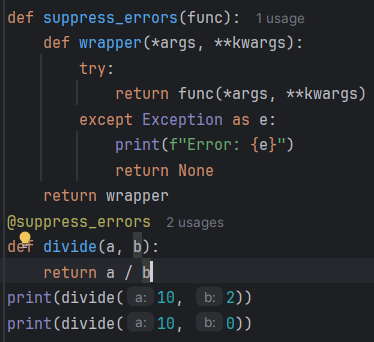
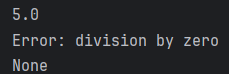
1. Create a decorator with arguments that repeats the function n times.

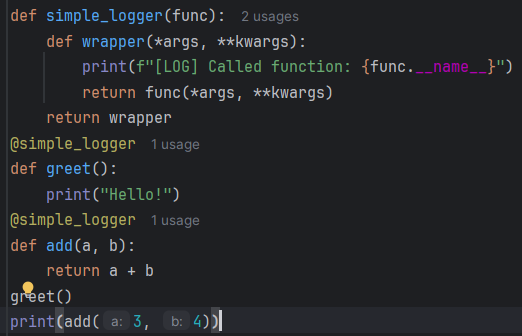
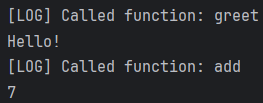
1. Write a decorator that measures the execution time of a function.

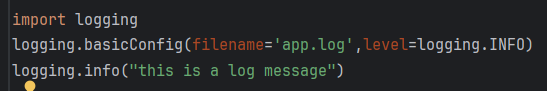
1. Write a decorator that modifies a function to return None if it raises any exception.

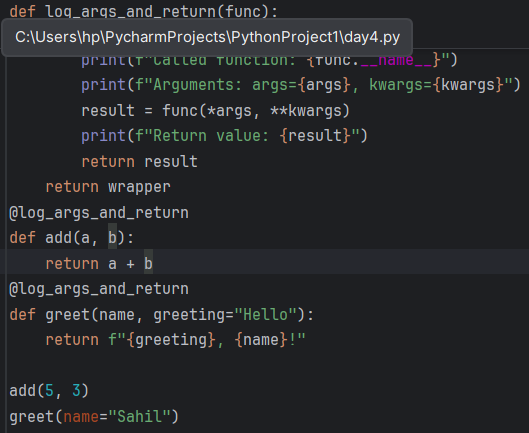
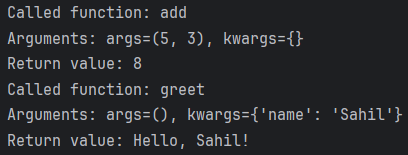
1. Write a simple logging function that logs to the console when a function is called.

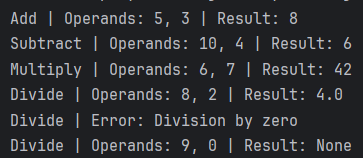
1. Create a logger using the logging module that logs messages to a file.



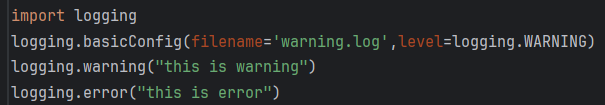
1. Write a function that logs both arguments and return values.

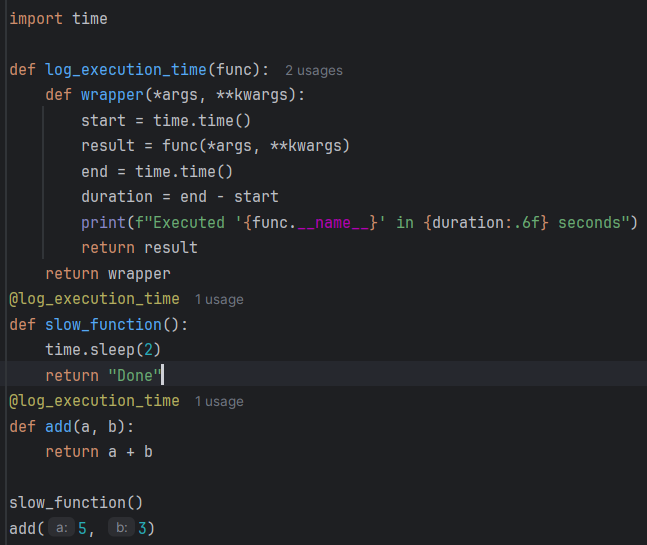
1. Add logging to a calculator function that logs each operation.

1. Configure a logger to log only warnings and errors.

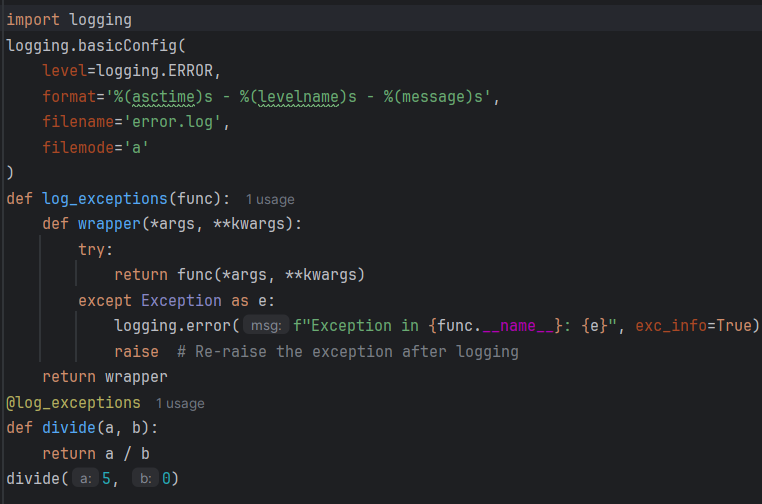


1. Write a decorator that logs the execution time of a function.

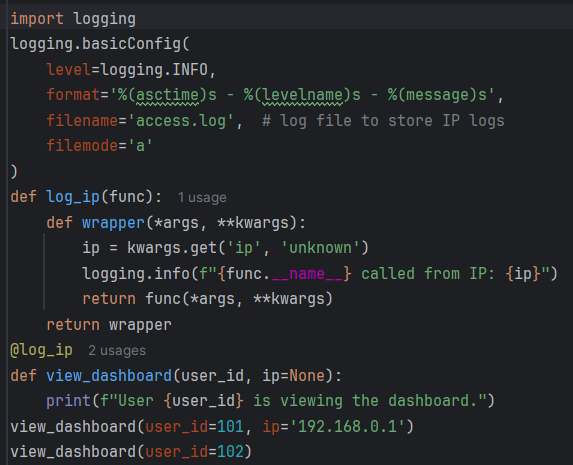
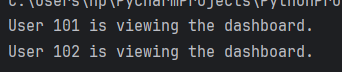




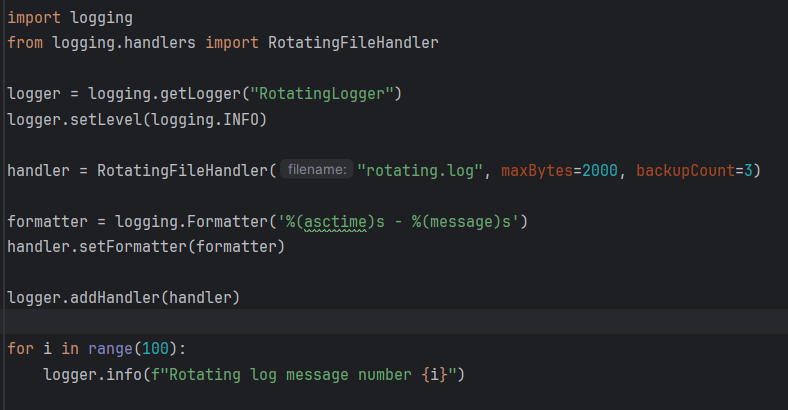
1. Use logging to log uncaught exceptions in a function.



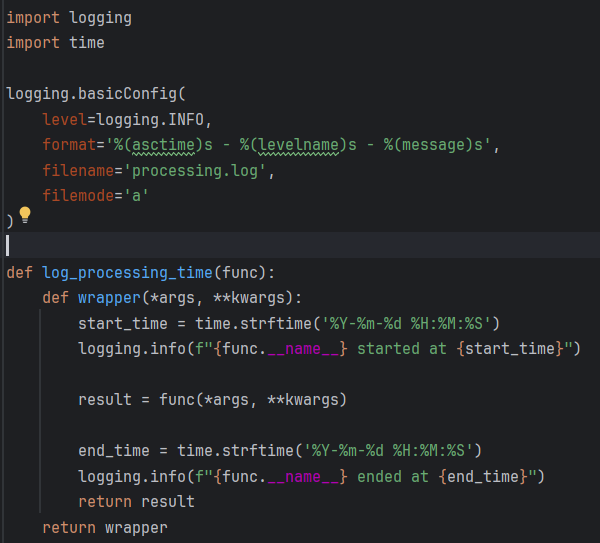
1. Write a logger that logs the user IP address when a function is called.

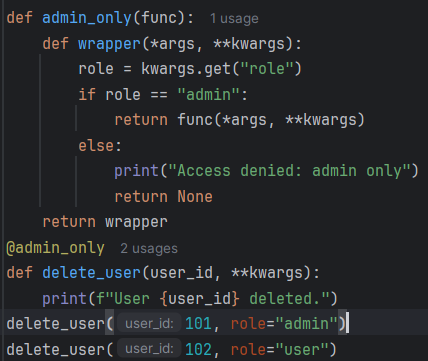
1. Create a rotating file logger using logging.handlers.



1. Write a decorator that logs the start and end time of a data processing function.



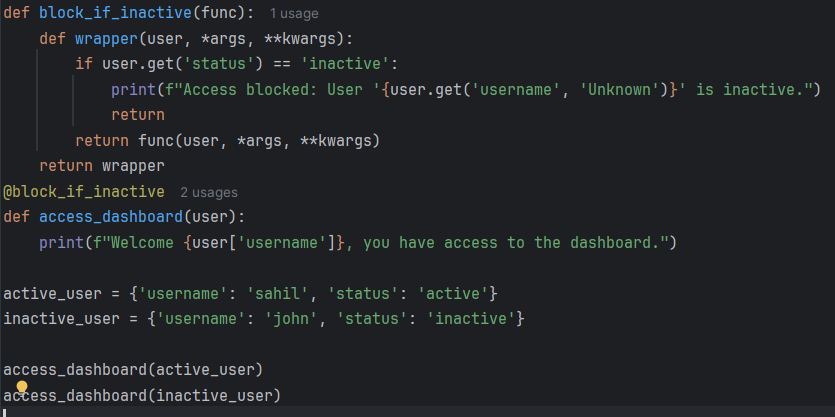
1. Write an authorization decorator that allows only users with the role 'admin'.

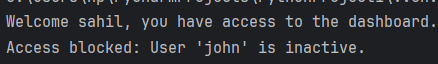
 

1. Create a function that checks if the user has permission 'view\_reports'.

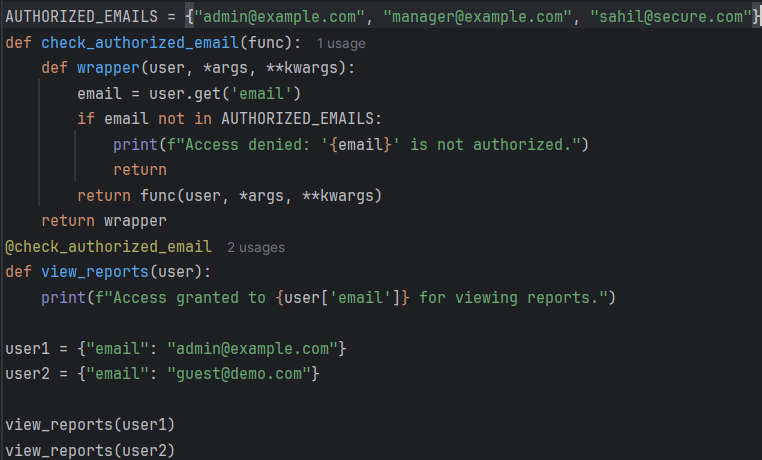
 

1. Write a decorator that blocks a function call if the user’s status is 'inactive'.

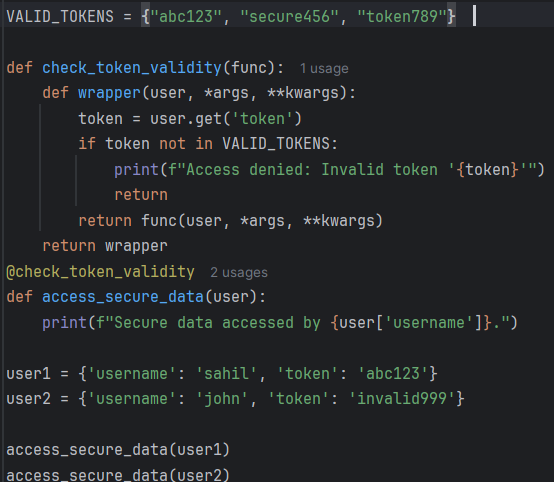




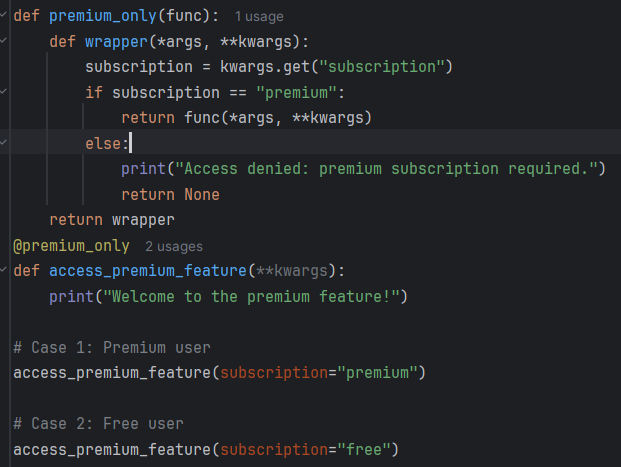
1. Implement a decorator that checks if a user email is in the authorized list.

1. Write a decorator that checks if the user token is valid.

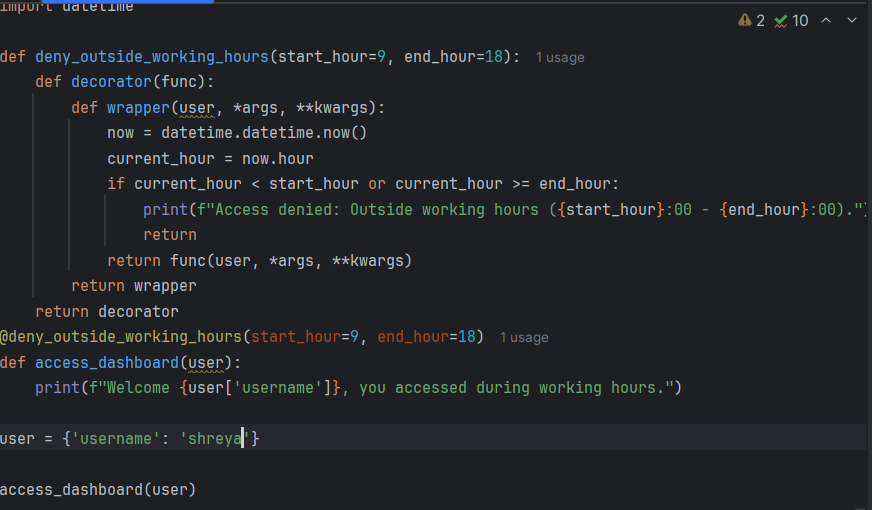
1. Write a decorator that allows only users with subscription 'premium' to access a function.

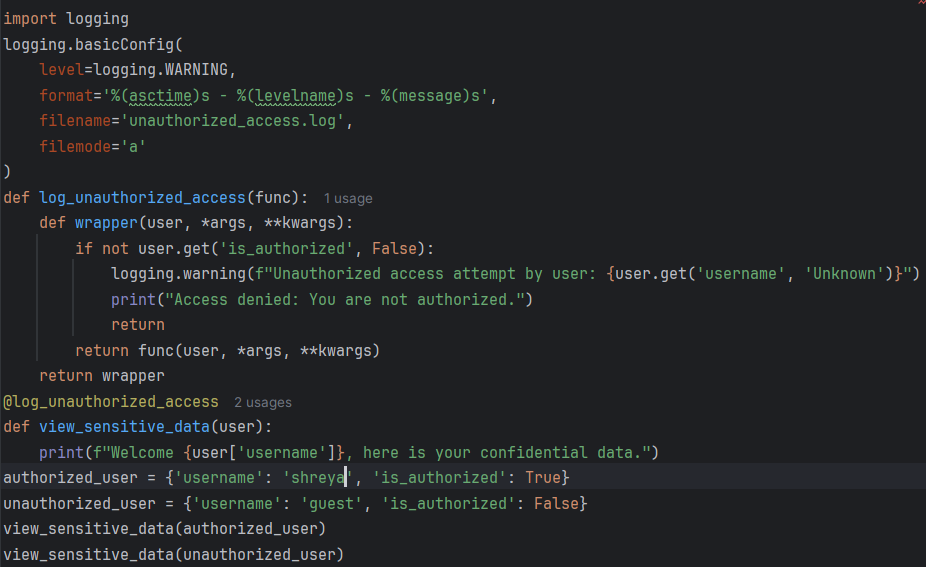
1. Simulate an API call where only authenticated users can access data using a decorator.

1. Write a decorator that denies access if the user tries to access outside working hours.

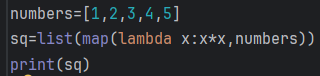
1. Write a decorator that logs unauthorized access attempts.

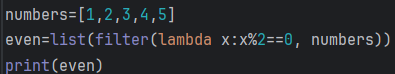
1. Create a decorator that restricts access to functions based on country code.

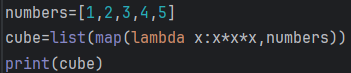

1. Write a lambda function to square a number.

1. Use filter() with lambda to get all even numbers from a list.

output: [2,4]

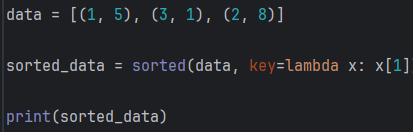
1. Use map() with lambda to get the cube of all numbers in a list.

output: [1, 8, 27, 64, 125]

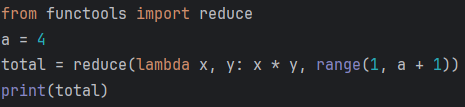
1. Write a lambda function to check if a string is a palindrome.

ouput: true

1. Sort a list of tuples based on the second element using lambda.

output: [(3, 1), (1, 5), (2, 8)]

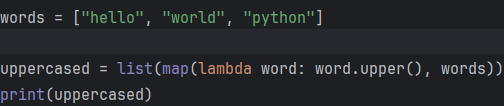
1. Use reduce() with lambda to calculate the factorial of a number.

output: 24

1. Write a lambda function to check if a number is divisible by both 3 and 5.

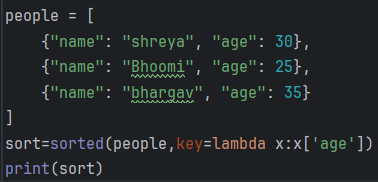
output: divisible by both

1. Use map() and lambda to convert a list of strings to uppercase.



Ouput: ['HELLO', 'WORLD', 'PYTHON']

1. Use lambda inside sorted() to sort a list of dictionaries by the 'age' key.

output:  
[{'name': 'Bhoomi', 'age': 25}, {'name': 'shreya', 'age': 30}, {'name': 'bhargav', 'age': 35}]

1. Write a lambda function that returns the maximum of two numbers.

output: 20