# PYTHON PROJECT

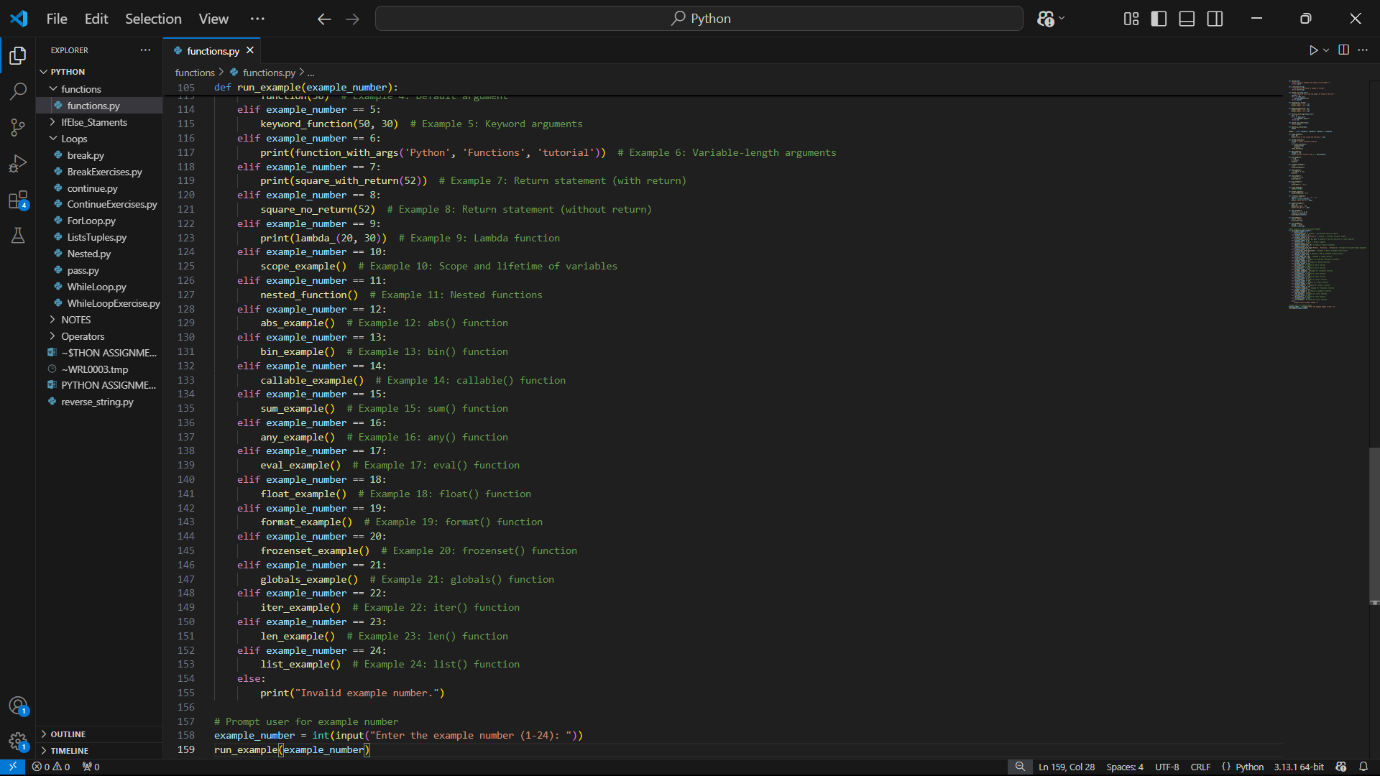
PYTHON FUNCTIONS:

A screenshot of a computer program

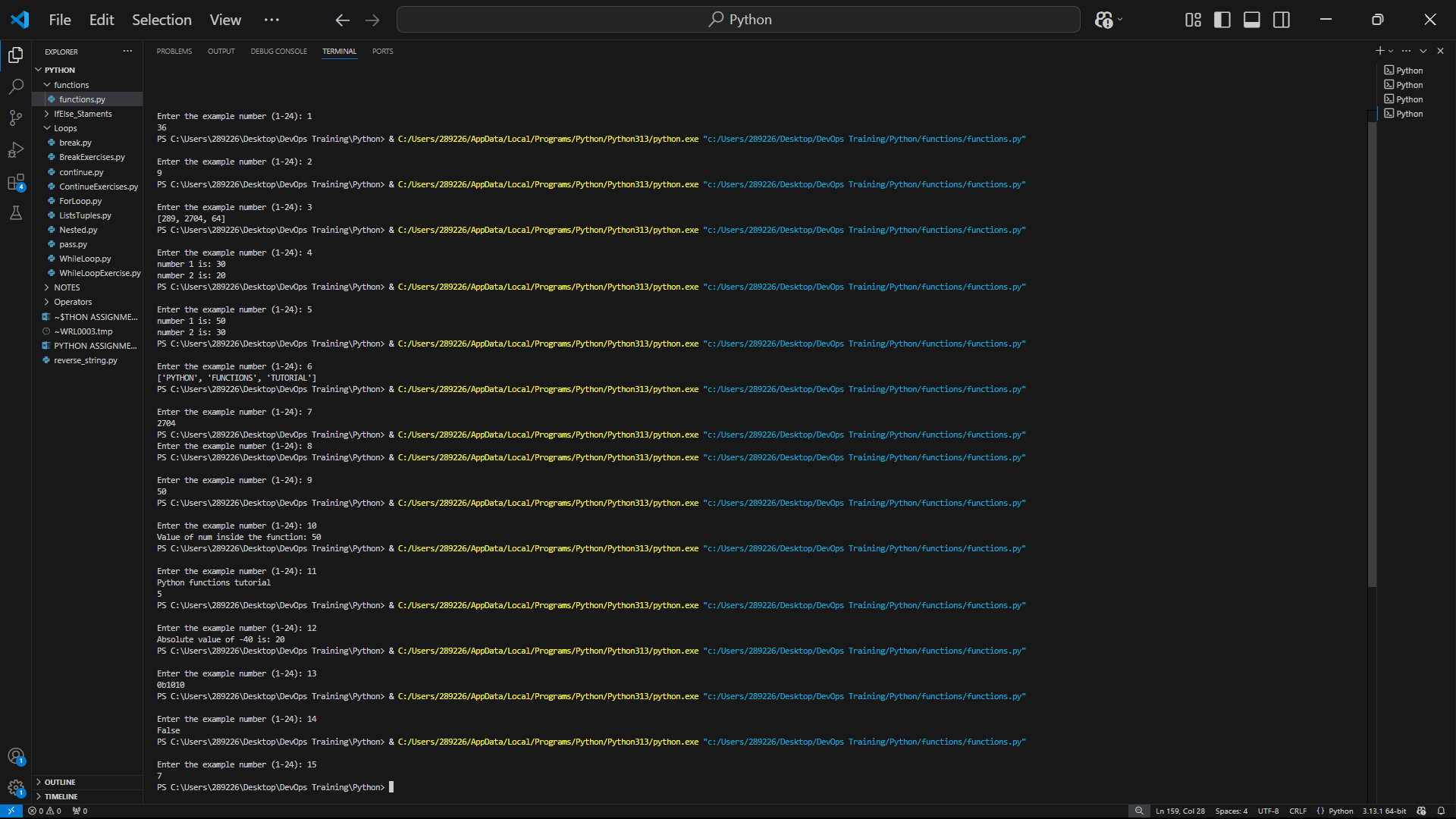
Description automatically generated

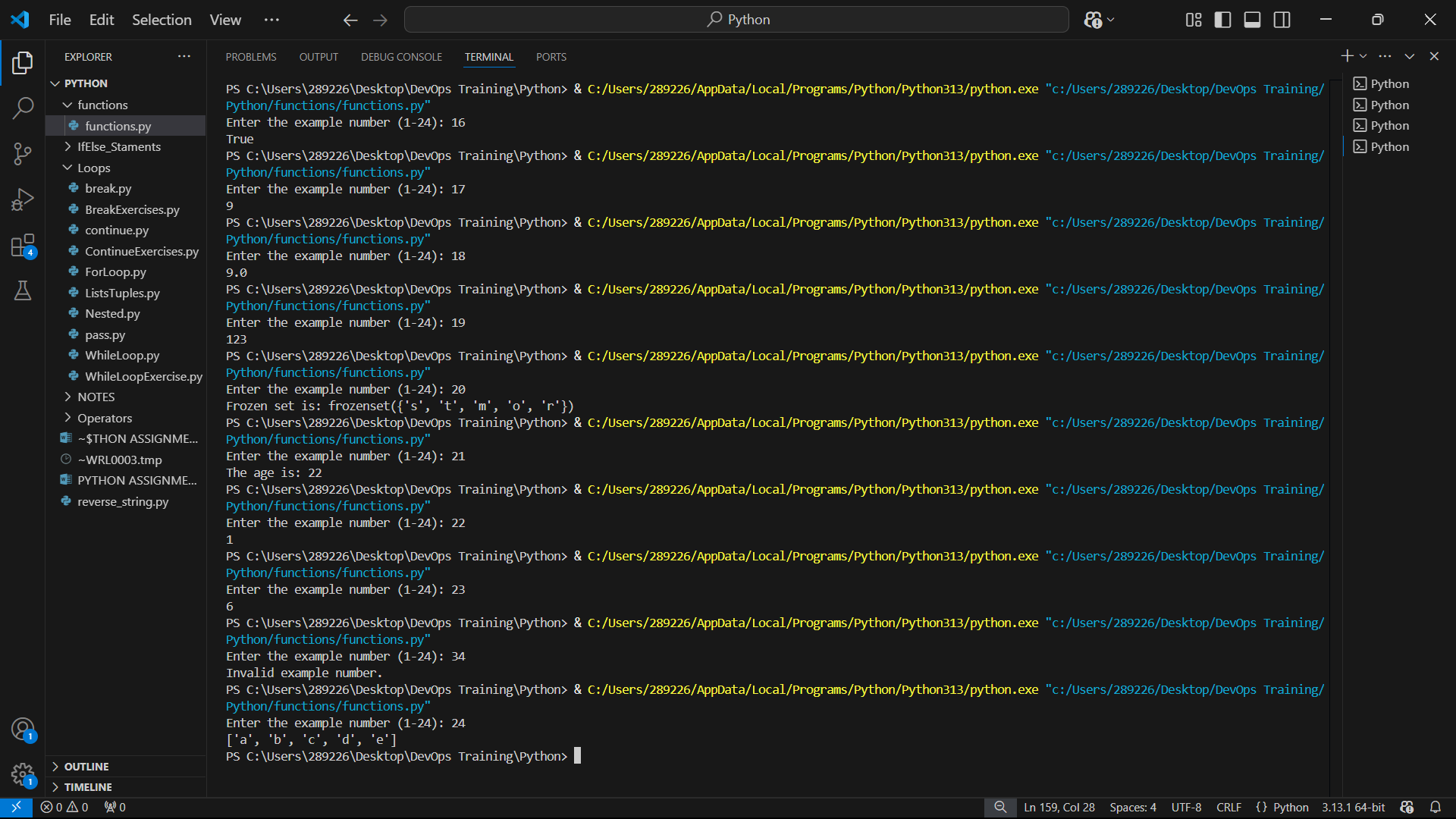
A screenshot of a computer

Description automatically generated



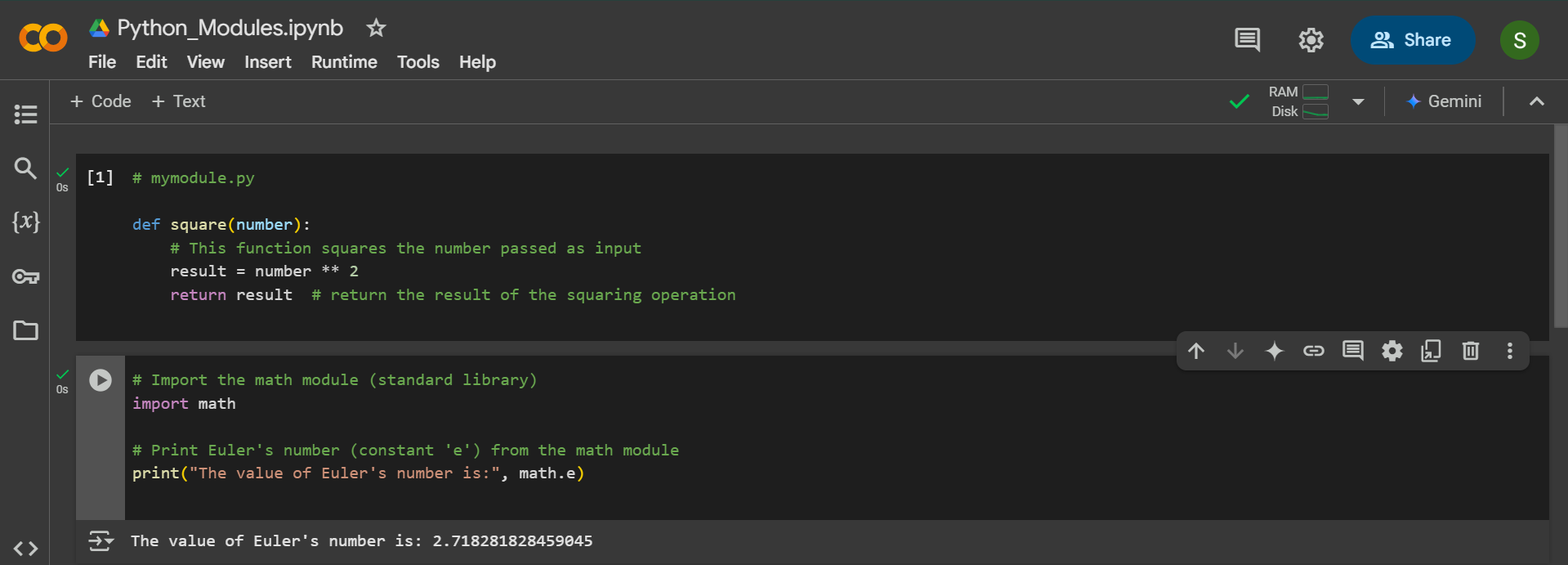
OUTPUT:



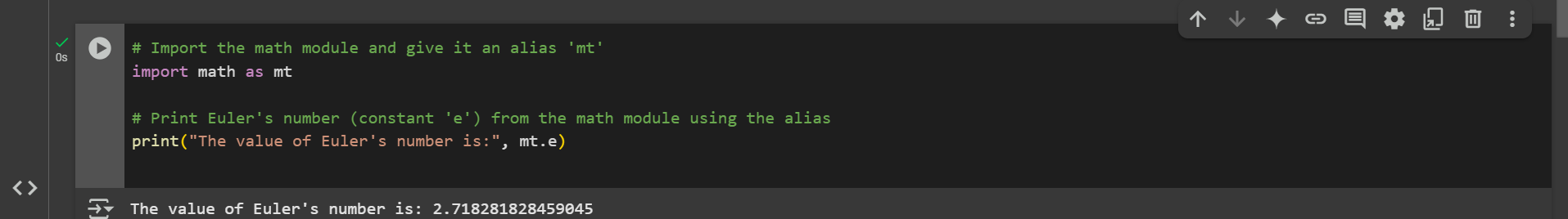


Python Modules

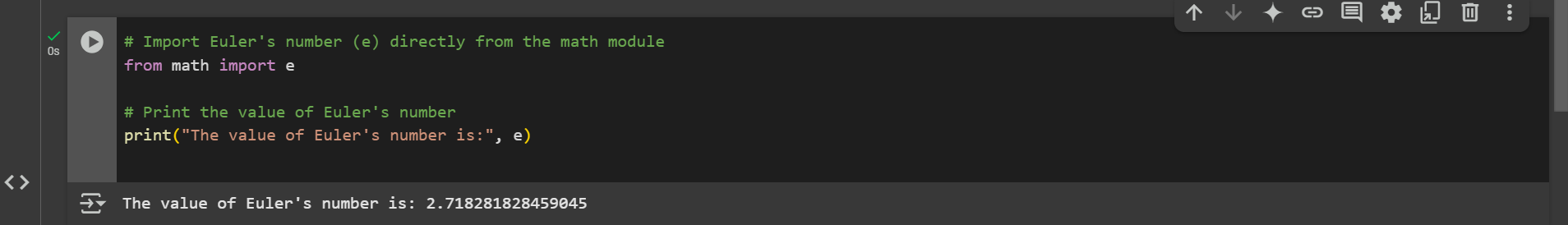
Python import Statement



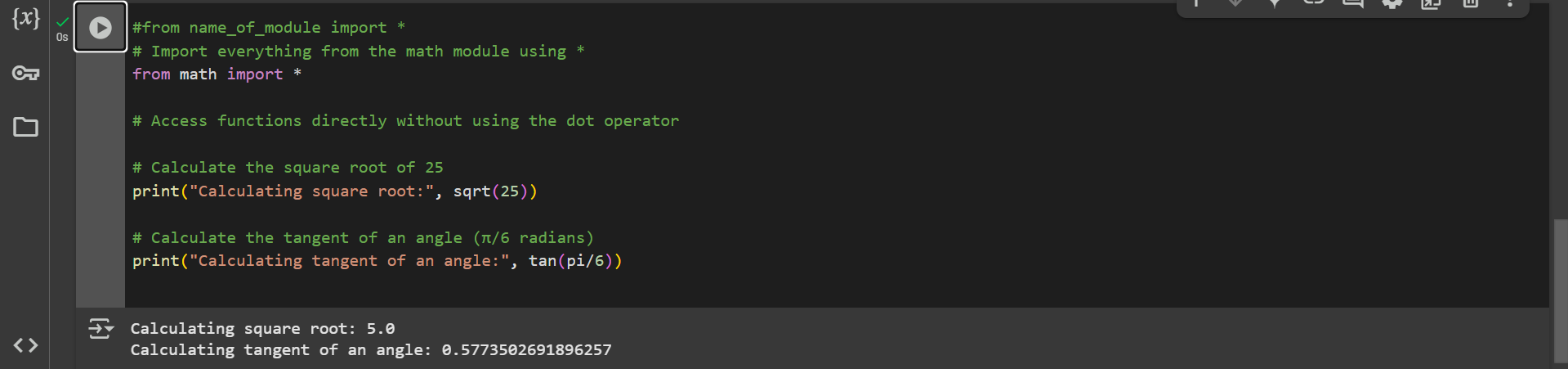
Importing and also Renaming



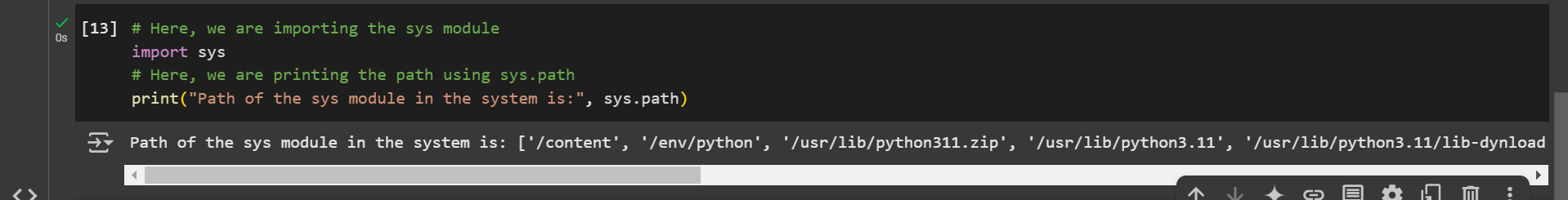
Python from...import Statement



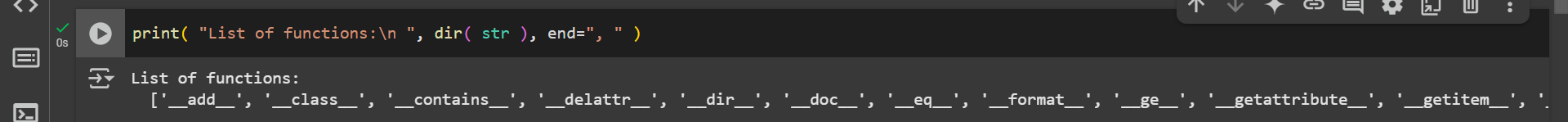
Import all Names - From import \* Statement



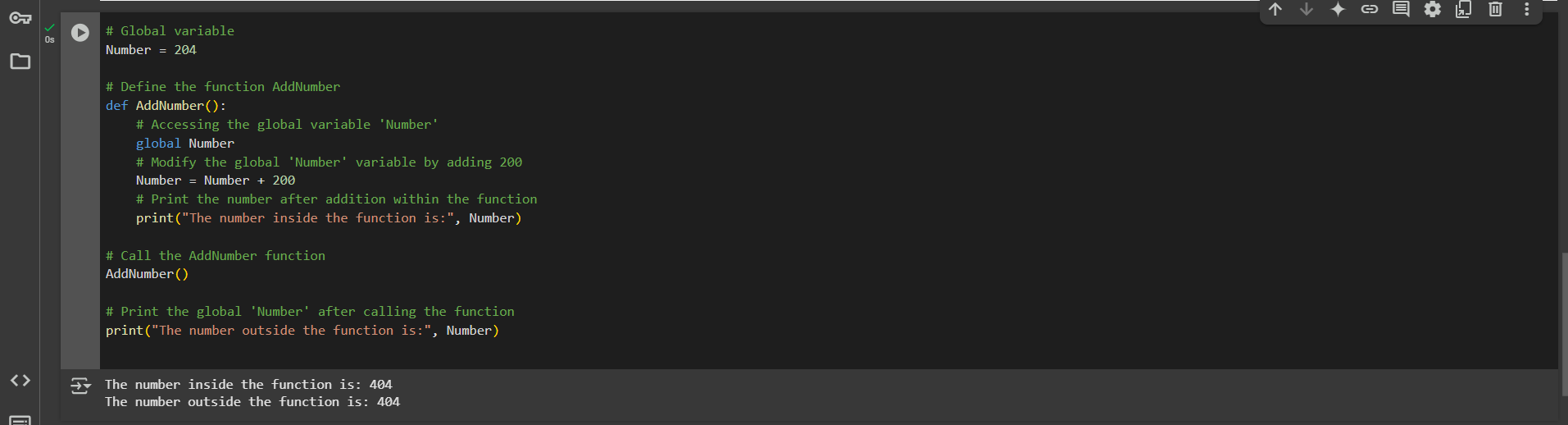
Locating Path of Modules



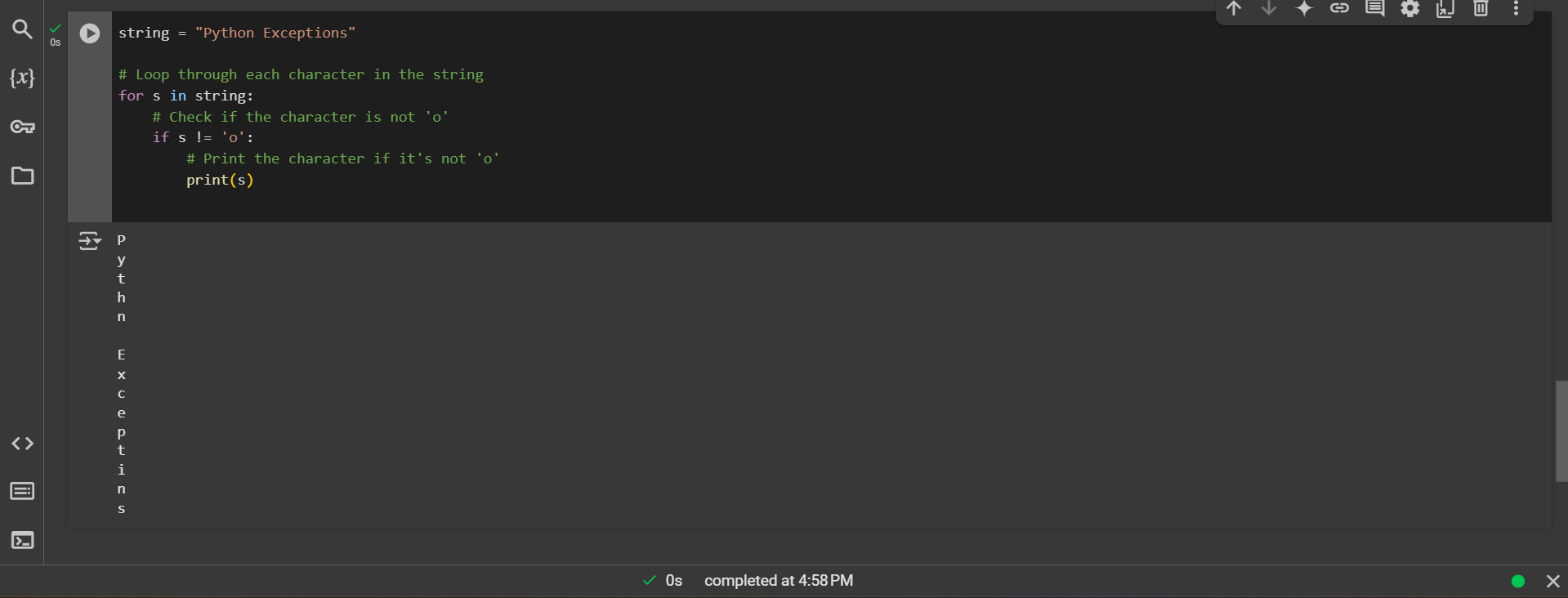
The dir() Built-in Function



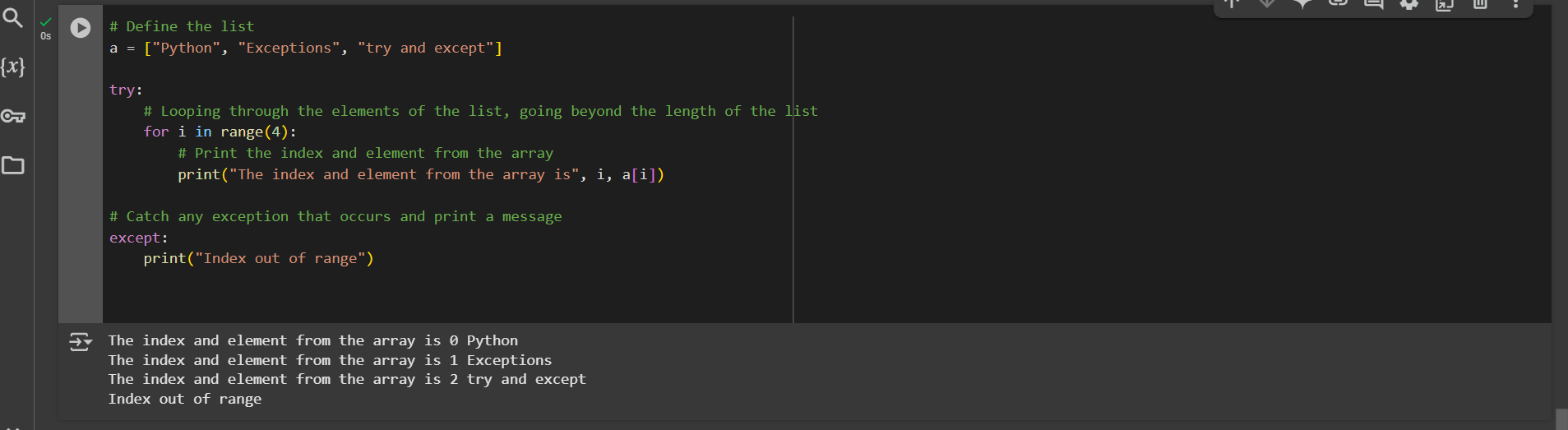
Namespaces and Scoping



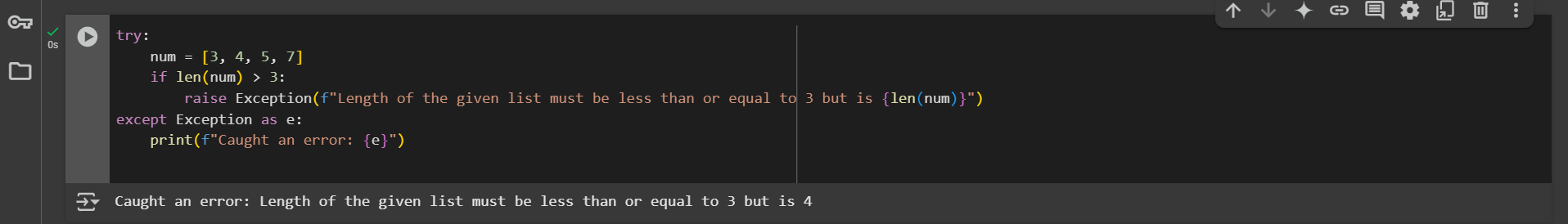
Python Exceptions



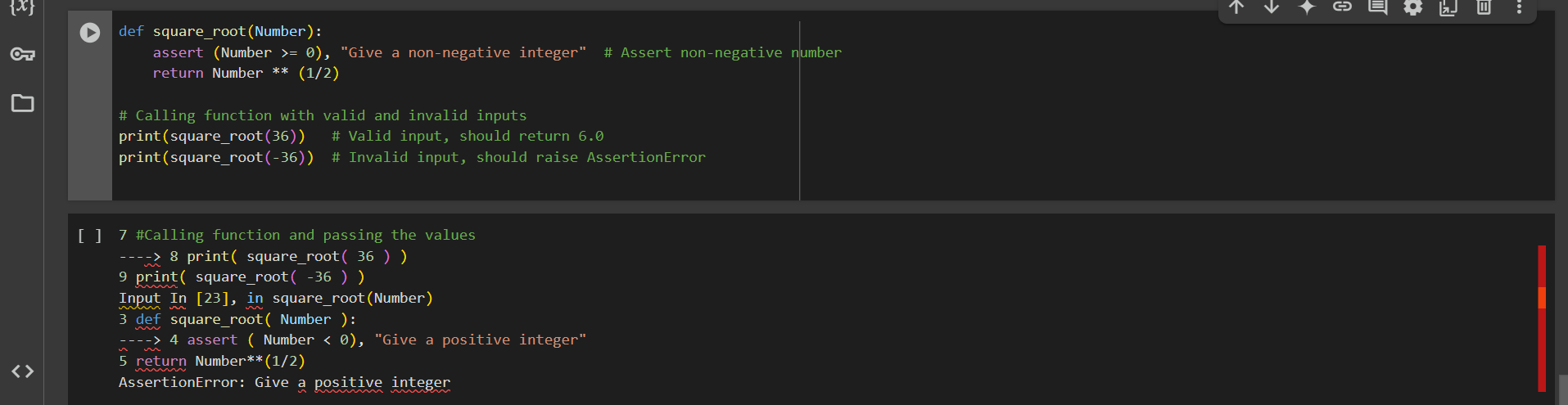
Try and Except Statement - Catching Exceptions



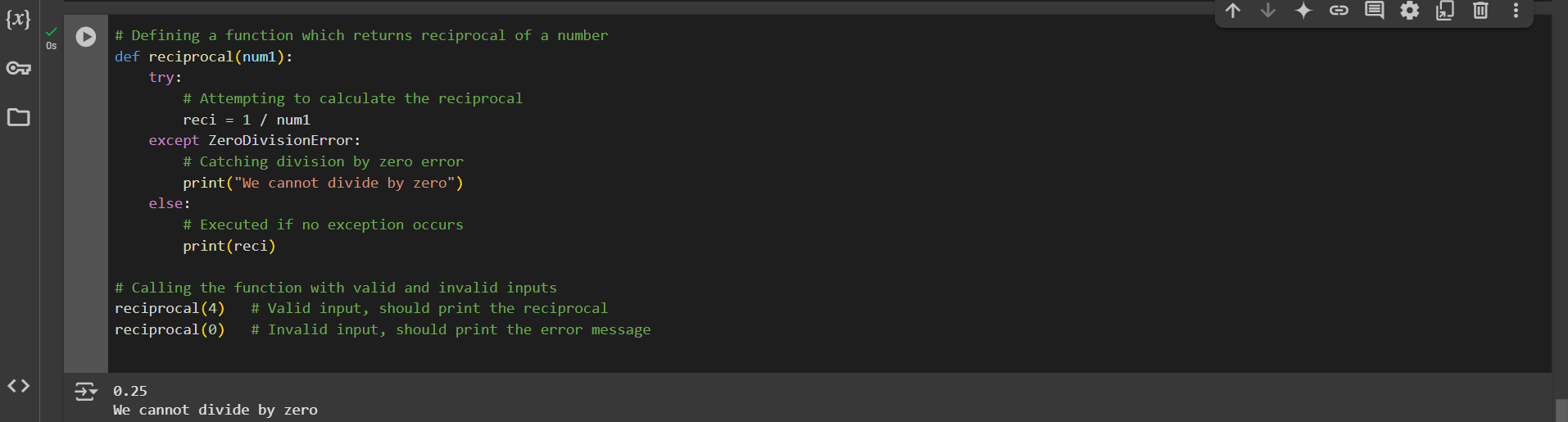
How to Raise an Exception



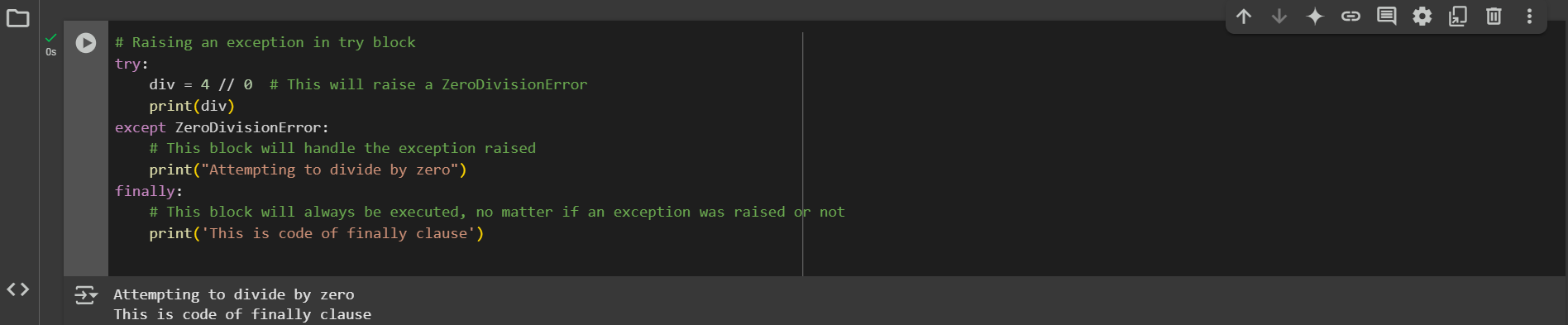
Assertions in Python



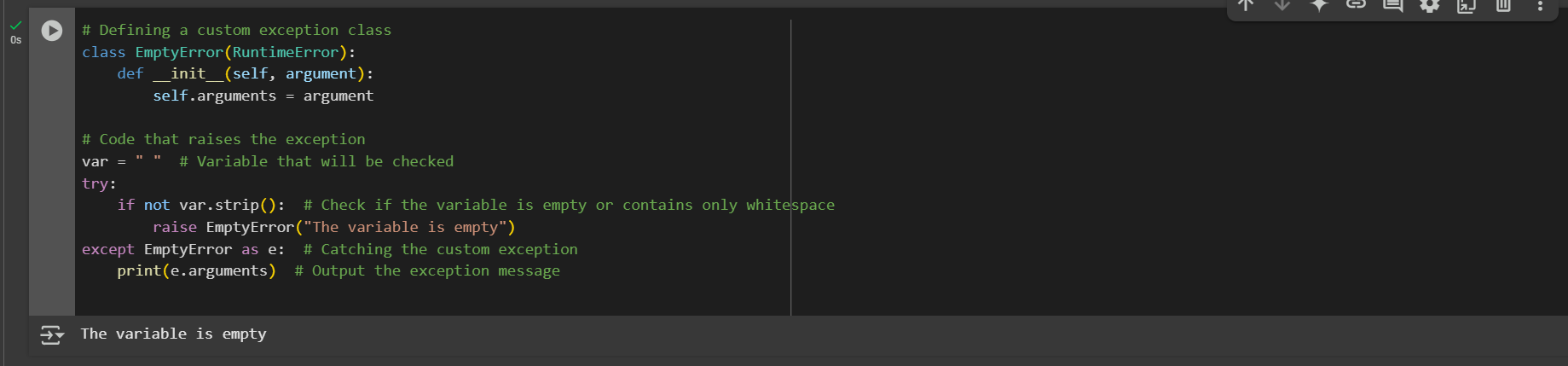
Try with Else Clause



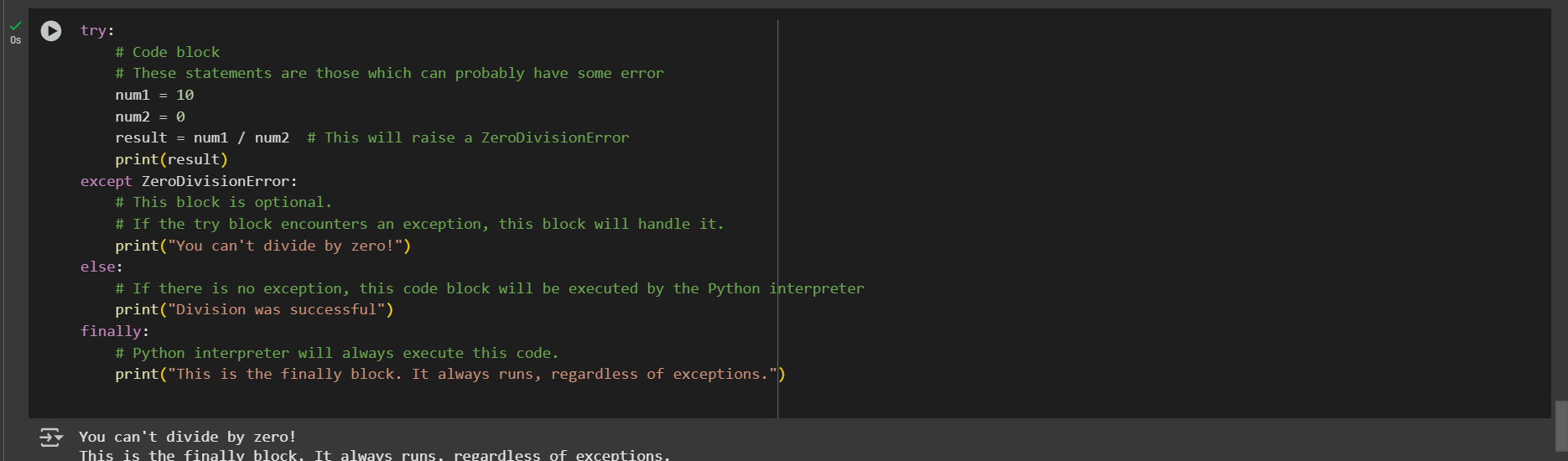
Finally Keyword in Python



User-Defined Exceptions



try, except, else, and finally clauses



Python Arrays

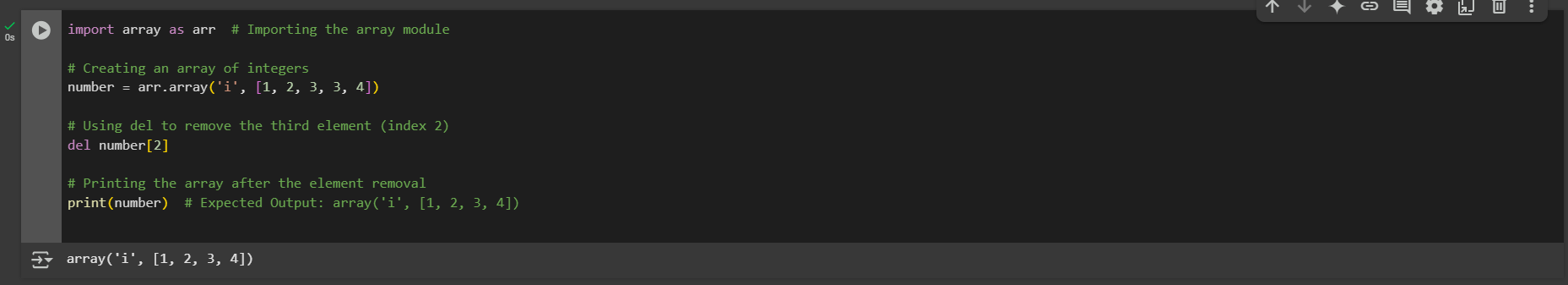
Accessing array element



Arrays are mutable, and their elements can be changed similarly to lists.



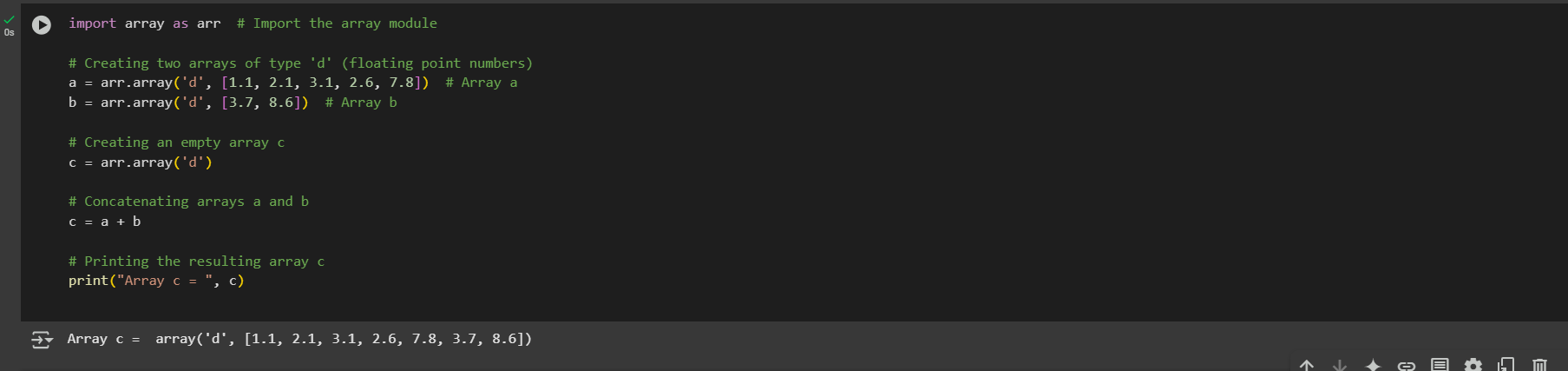
The elements can be deleted from an array using Python's del statement. If we want to delete any value from the Array, we can use the indices of a particular element.



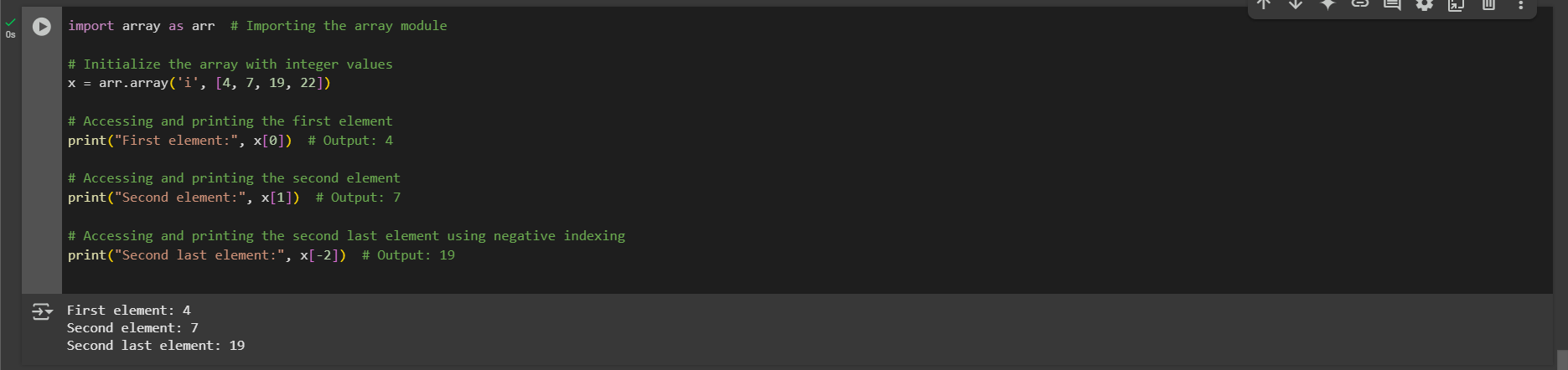
Array Concatenation

We can easily concatenate any two arrays using the + symbol.

Example 1:

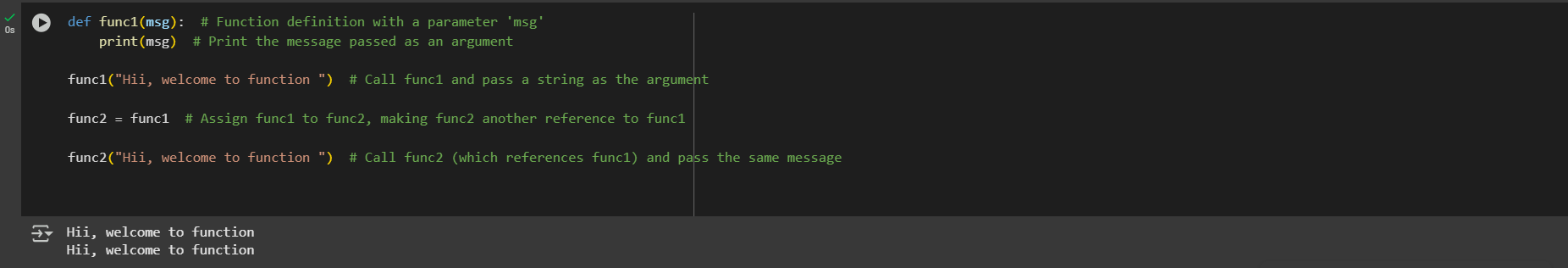


Example 2:

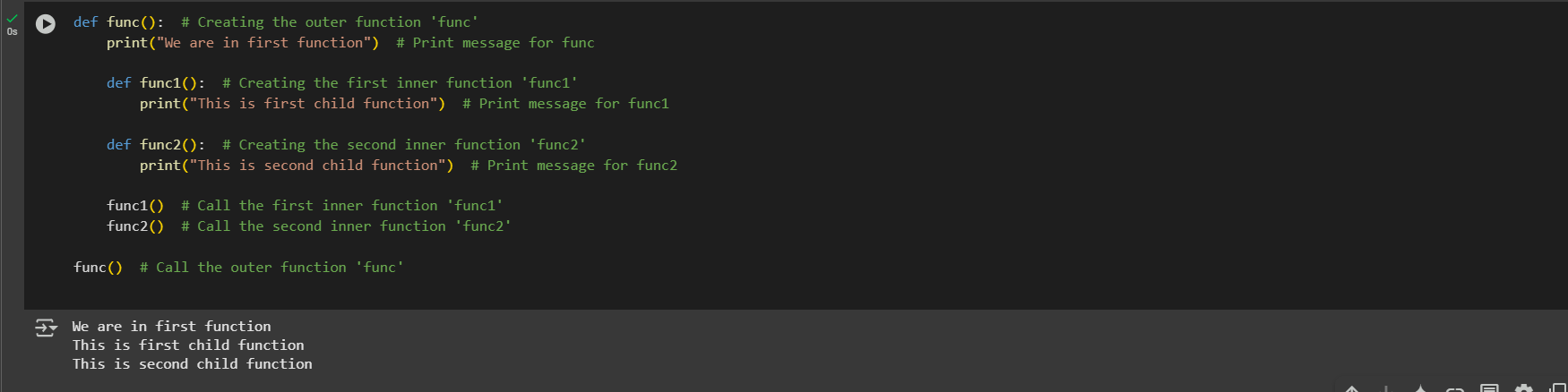


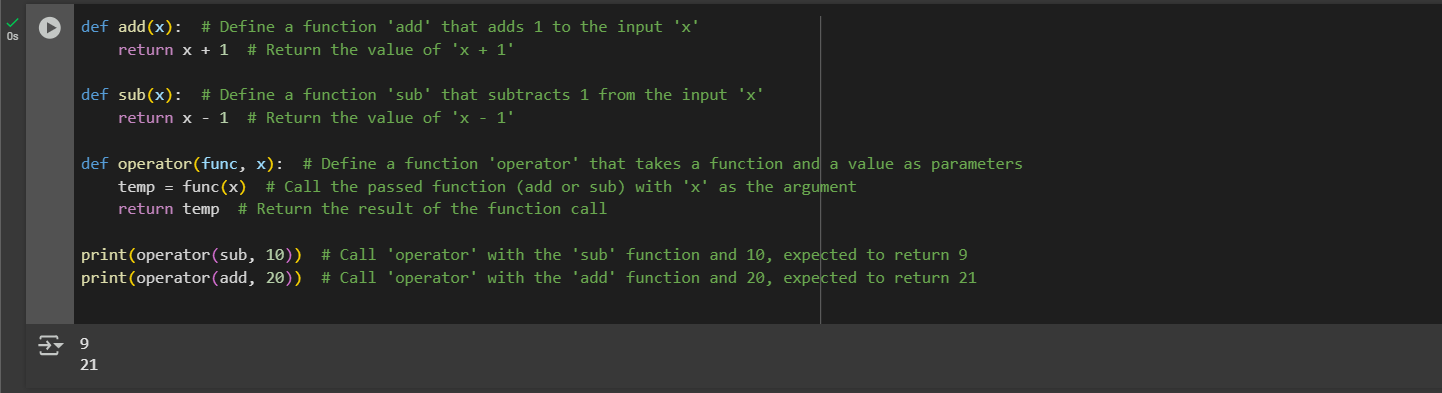
Python Decorator

Example

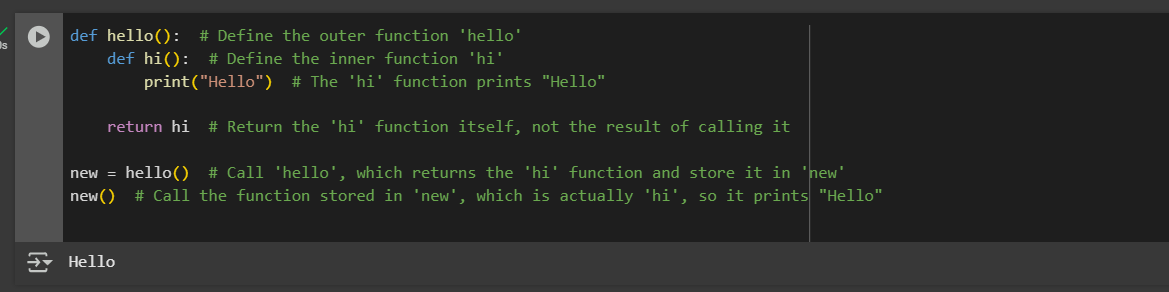


Inner Function

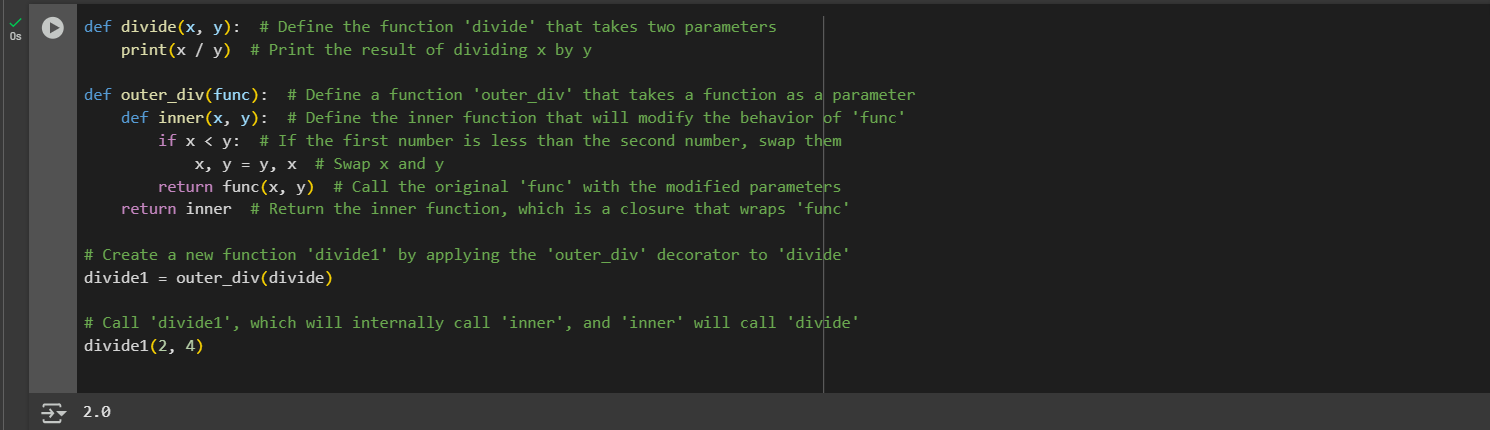




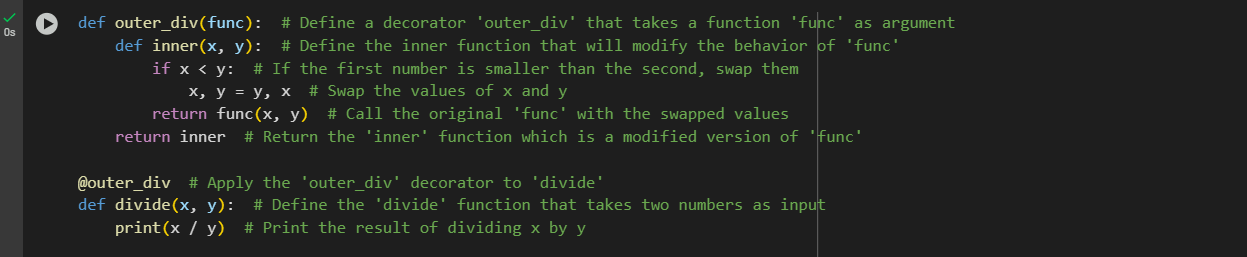
A function can return another function. Consider the below example:



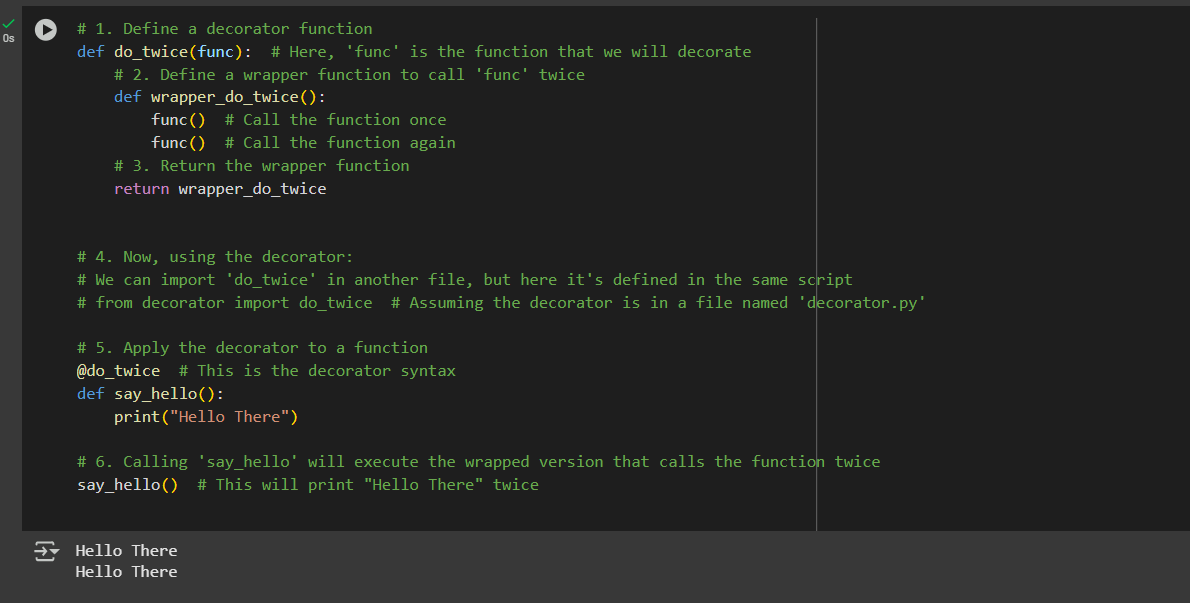
Decorating functions with parameters



Syntactic Decorator

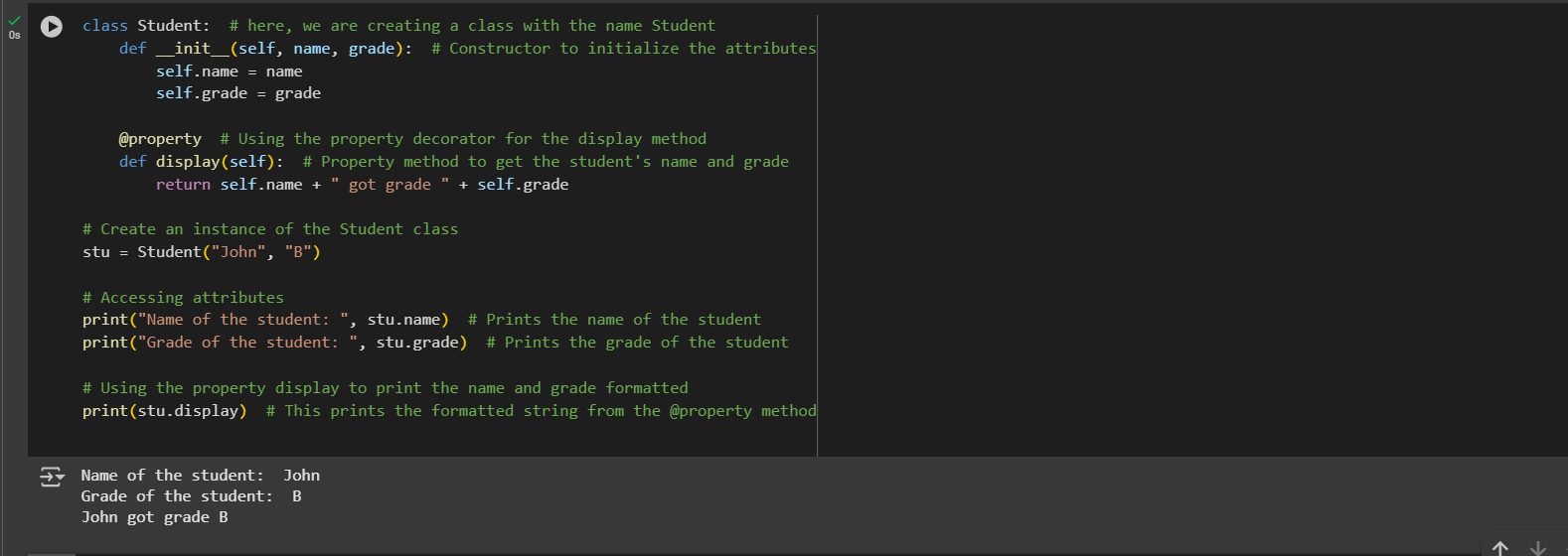


Reusing Decorator

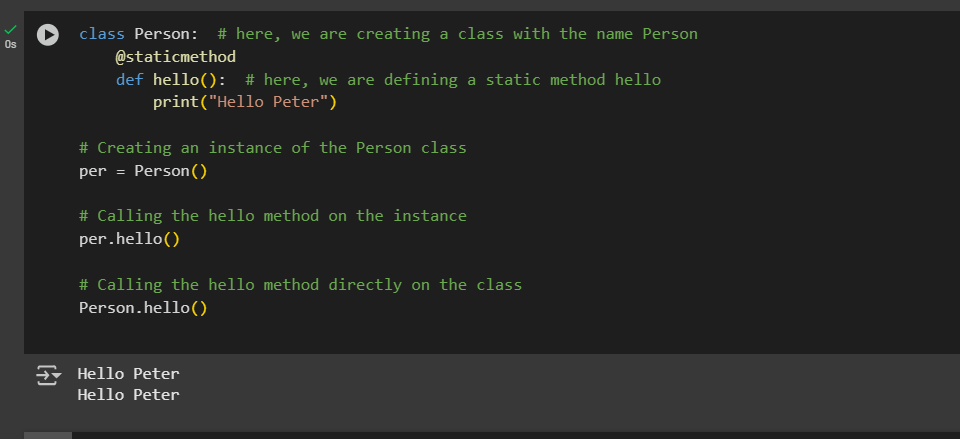


Fancy Decorators

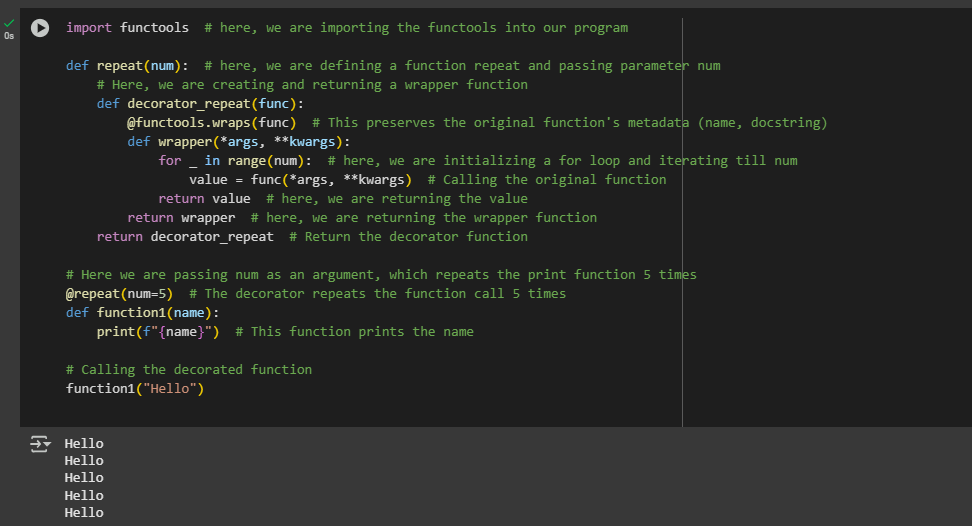
Example: 1



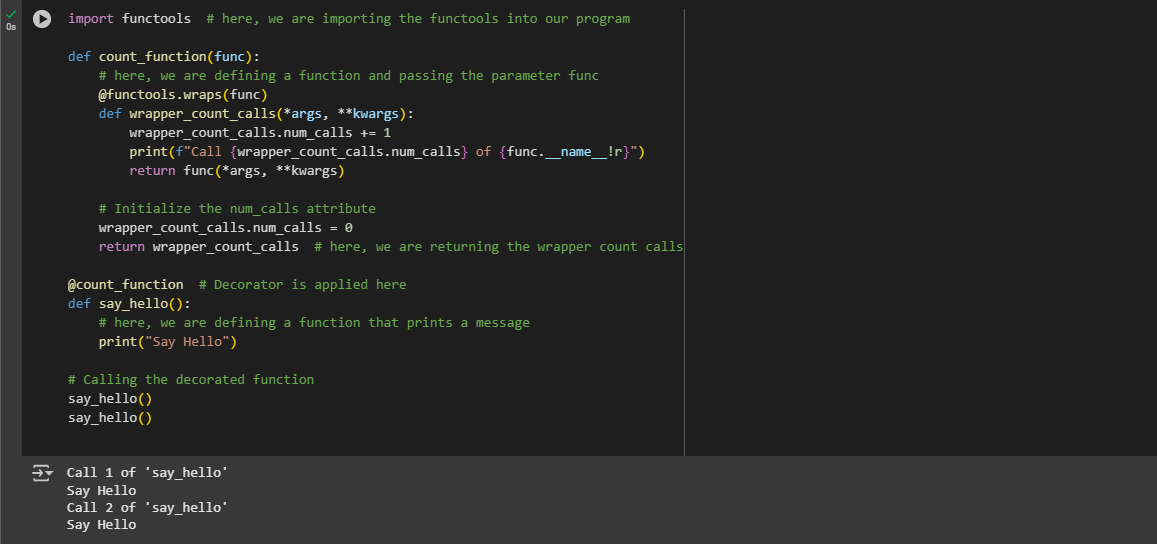
Example: 2-



Decorator with Arguments



Stateful Decorators



Classes as Decorators

