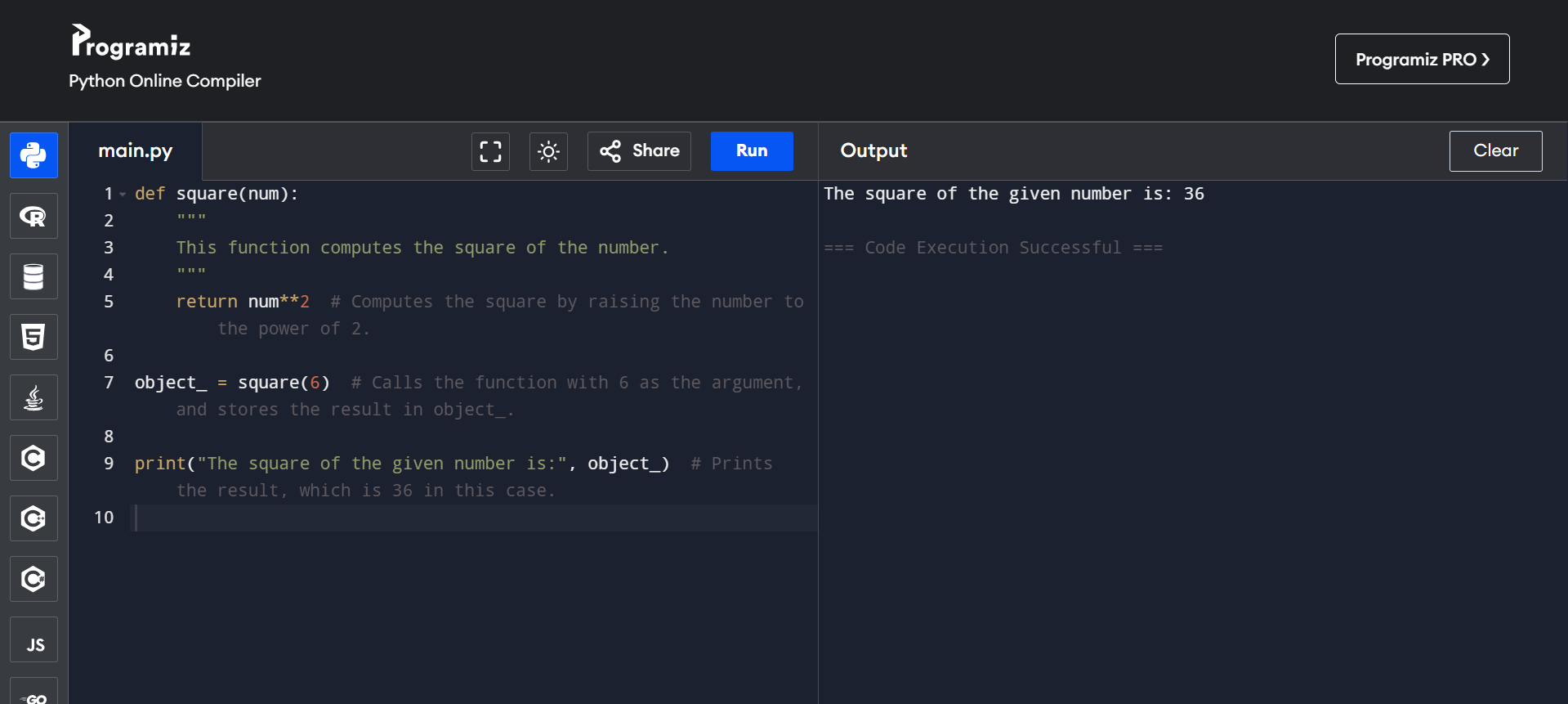
**PYTHON PROJECT**

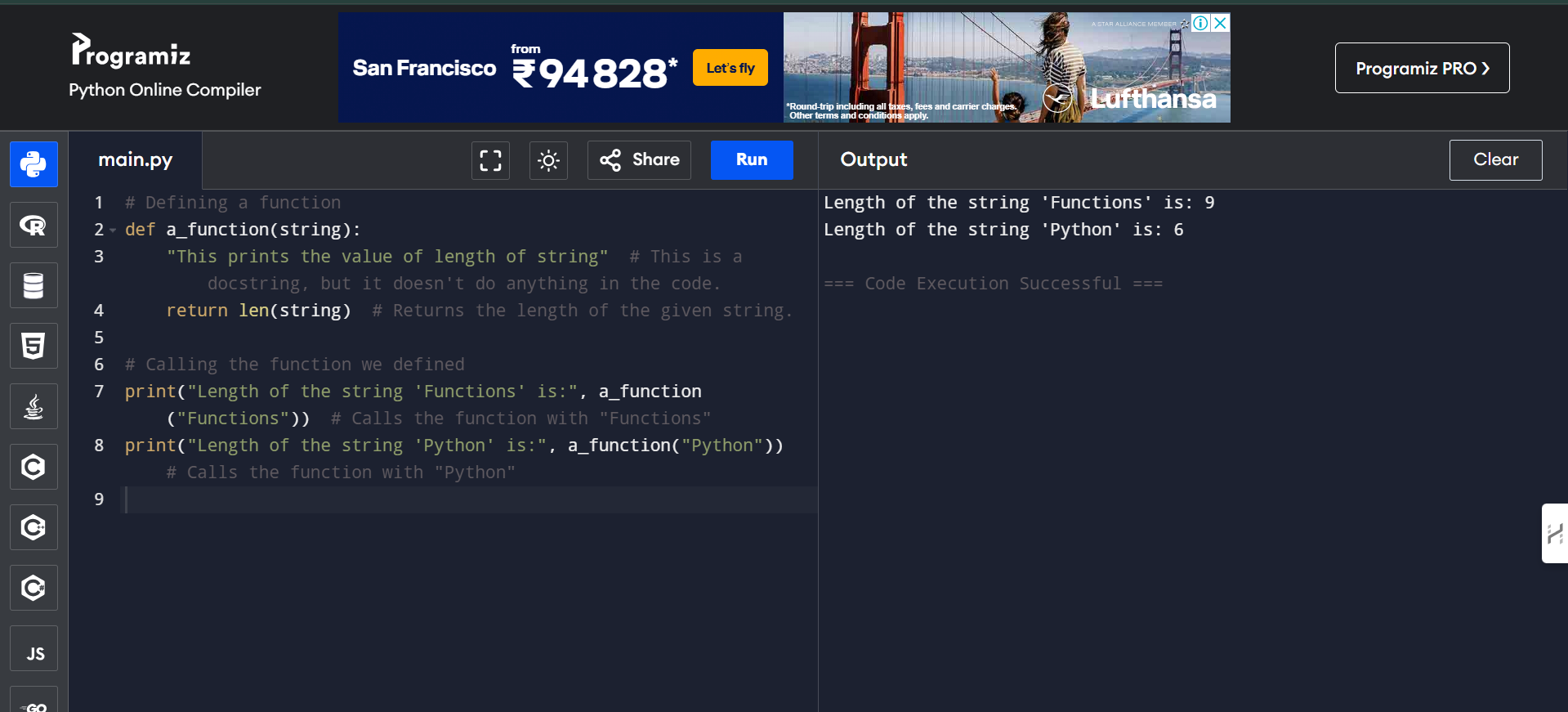
**Python Functions**

A collection of related assertions that carry out a mathematical, analytical, or evaluative operation is known as a function. An assortment of proclamations called Python Capabilities returns the specific errand. Python functions are necessary for intermediate-level programming and are easy to define. Function names meet the same standards as variable names do. The objective is to define a function and group-specific frequently performed actions. Instead of repeatedly creating the same code block for various input variables, we can call the function and reuse the code it contains with different variables.

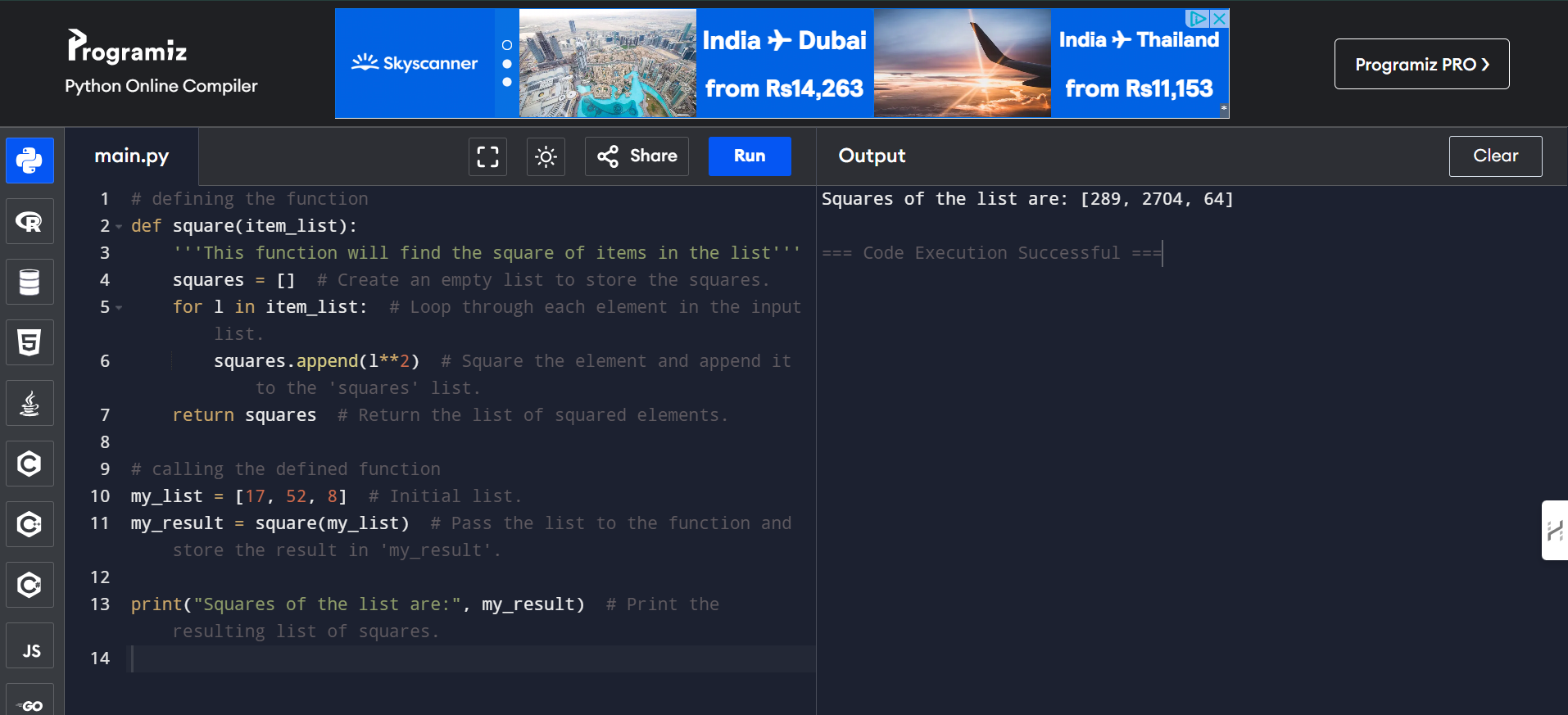
**Illustration of a User-Defined Function**

****

**Calling a Function**

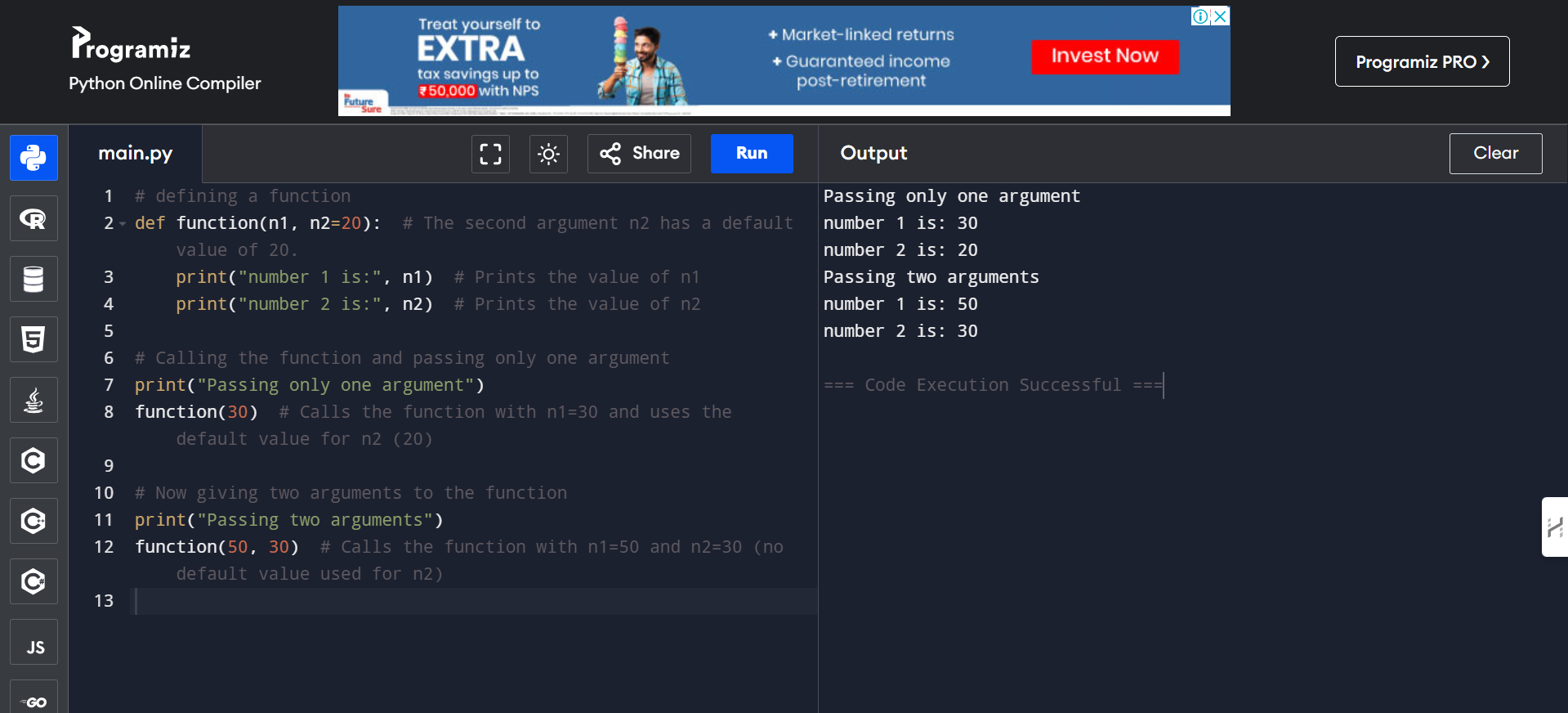
****

**Pass by Reference vs. Pass by Value**

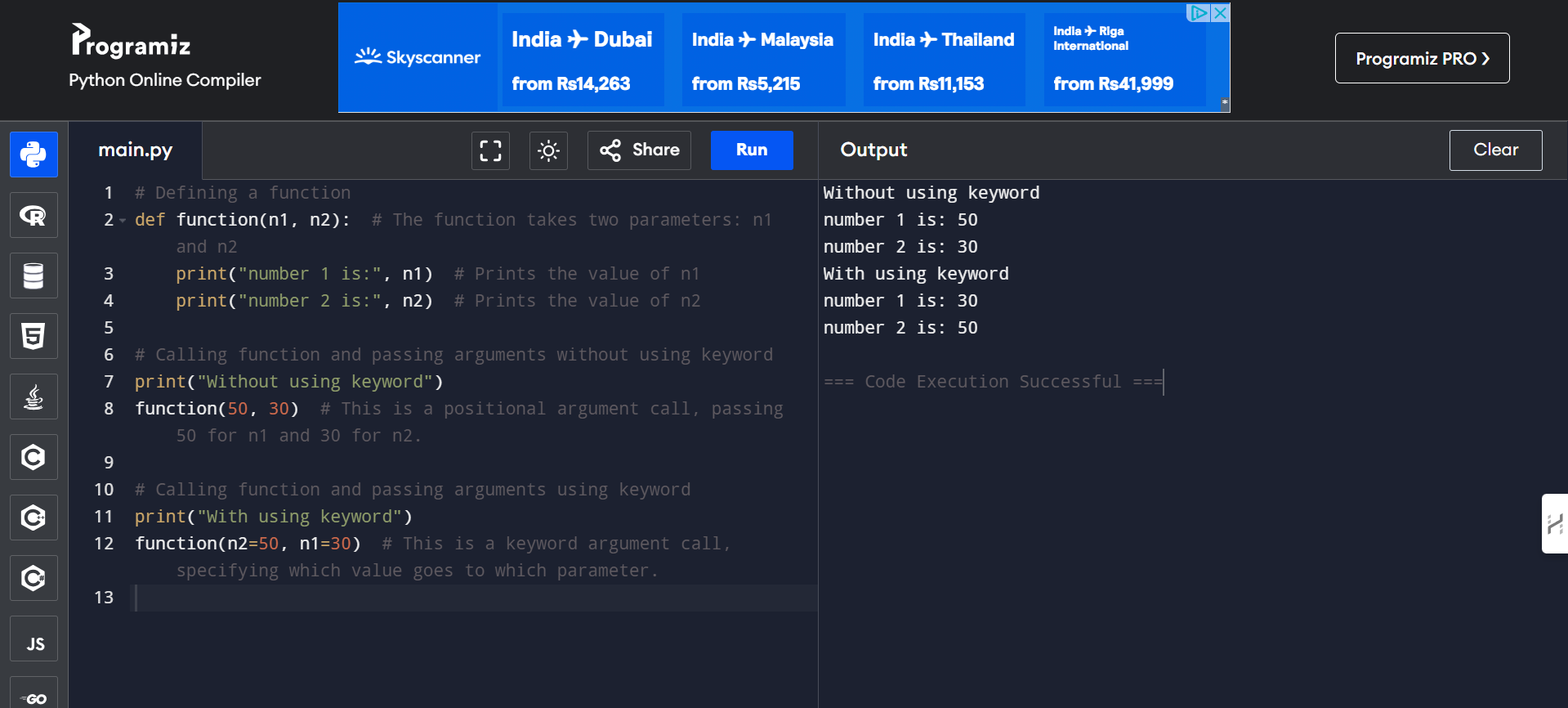
****

**Function Arguments**

1. **Default Arguments**

****

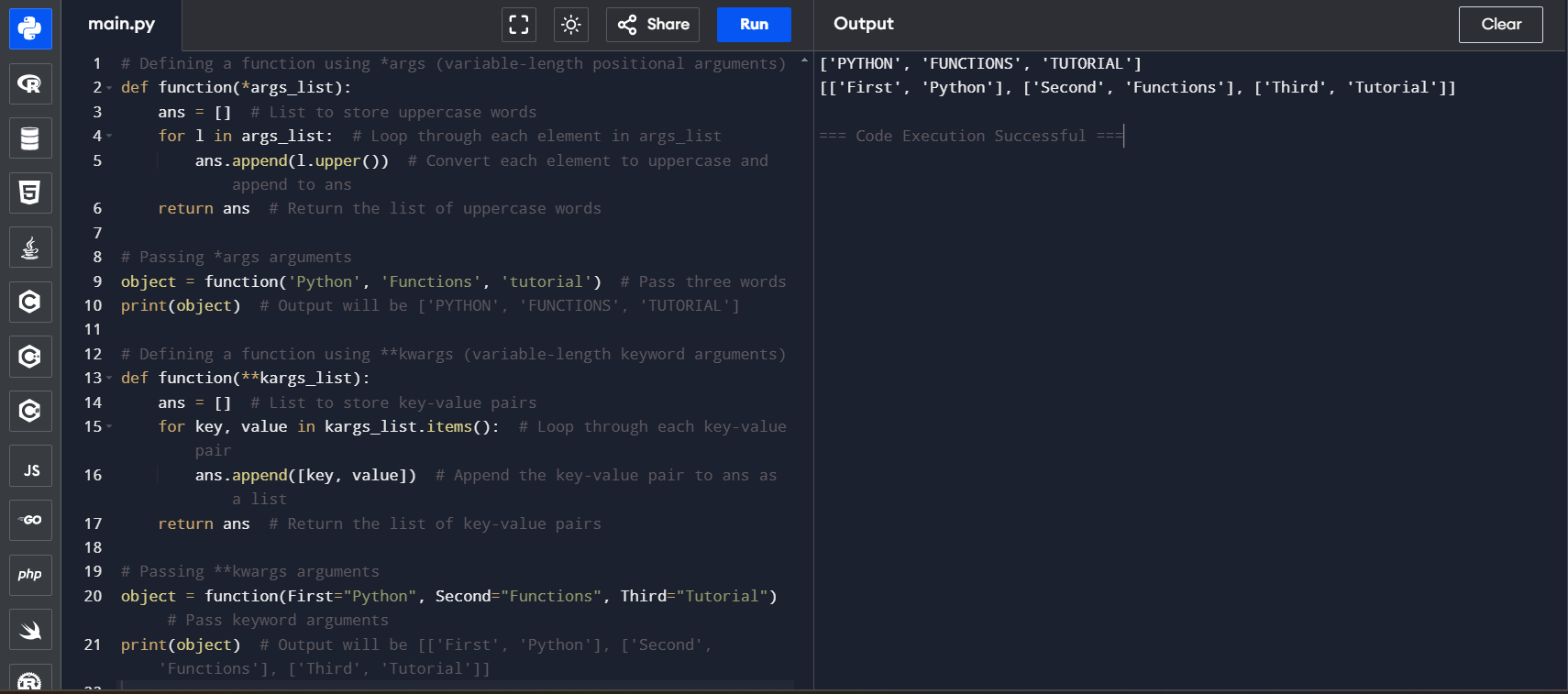
1. **Keyword Arguments**

****

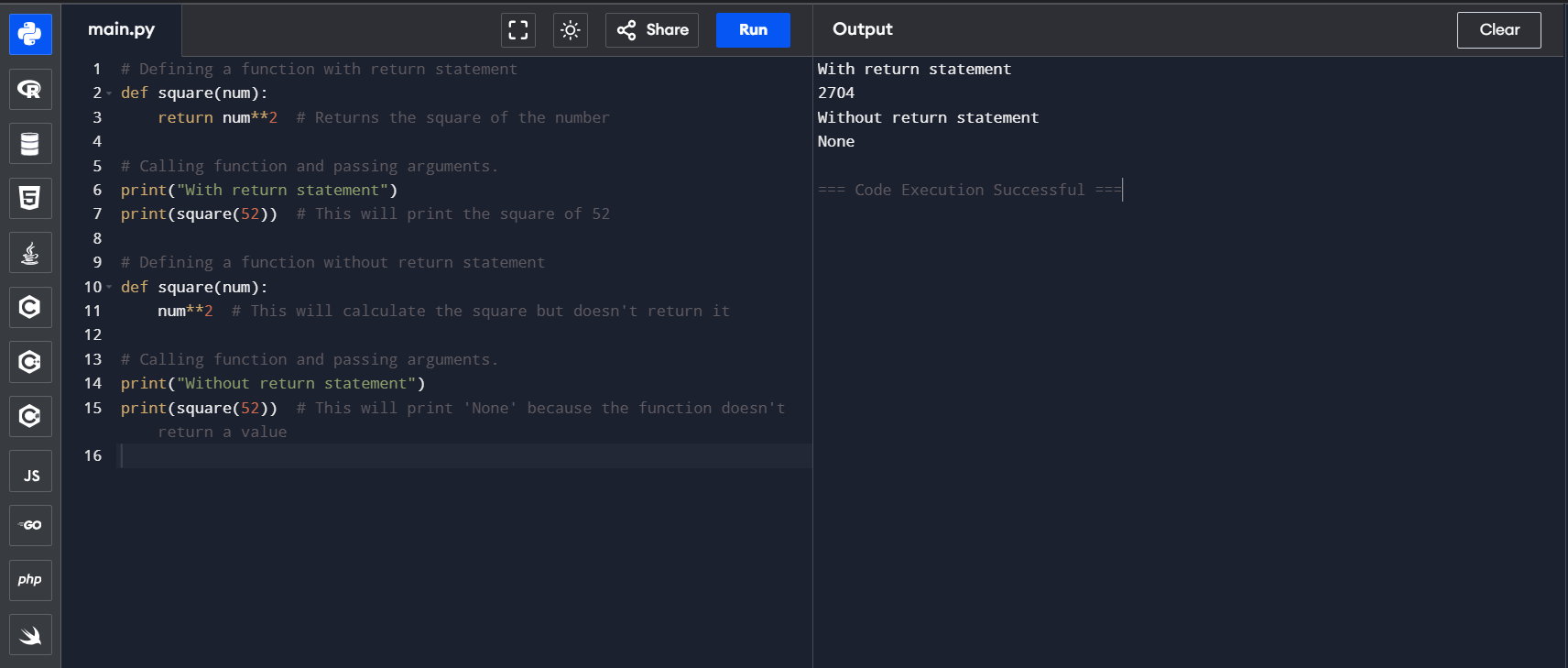
1. **Required Arguments**

****

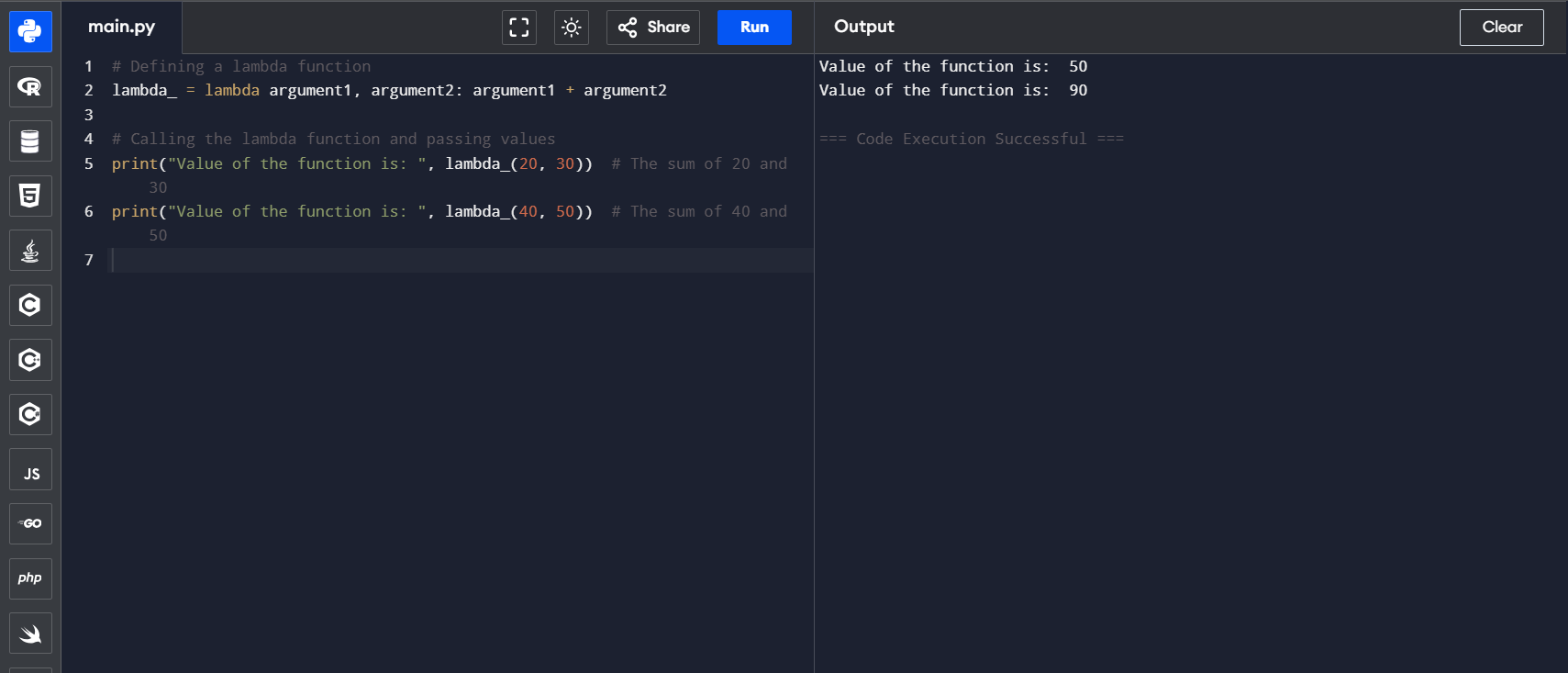
1. **Variable-Length Arguments**

****

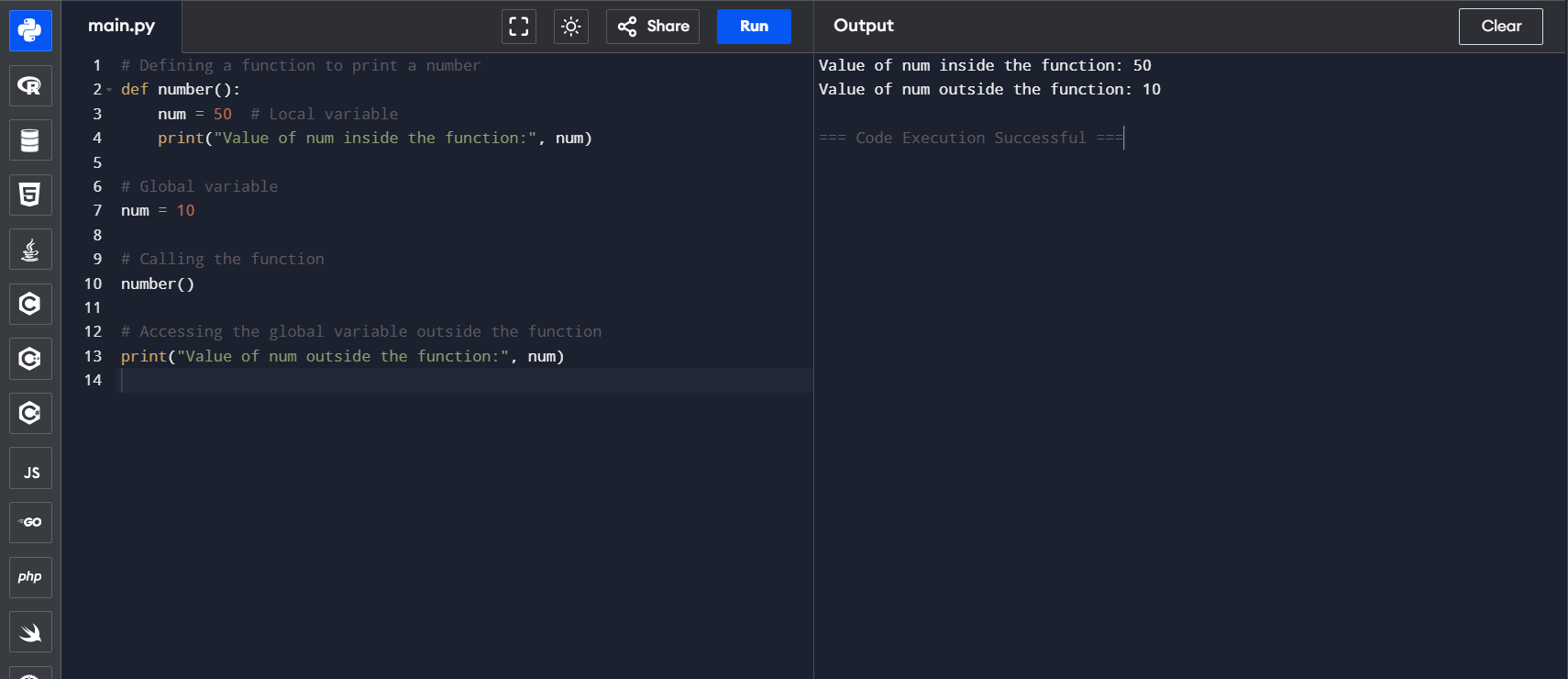
**return Statement**

****

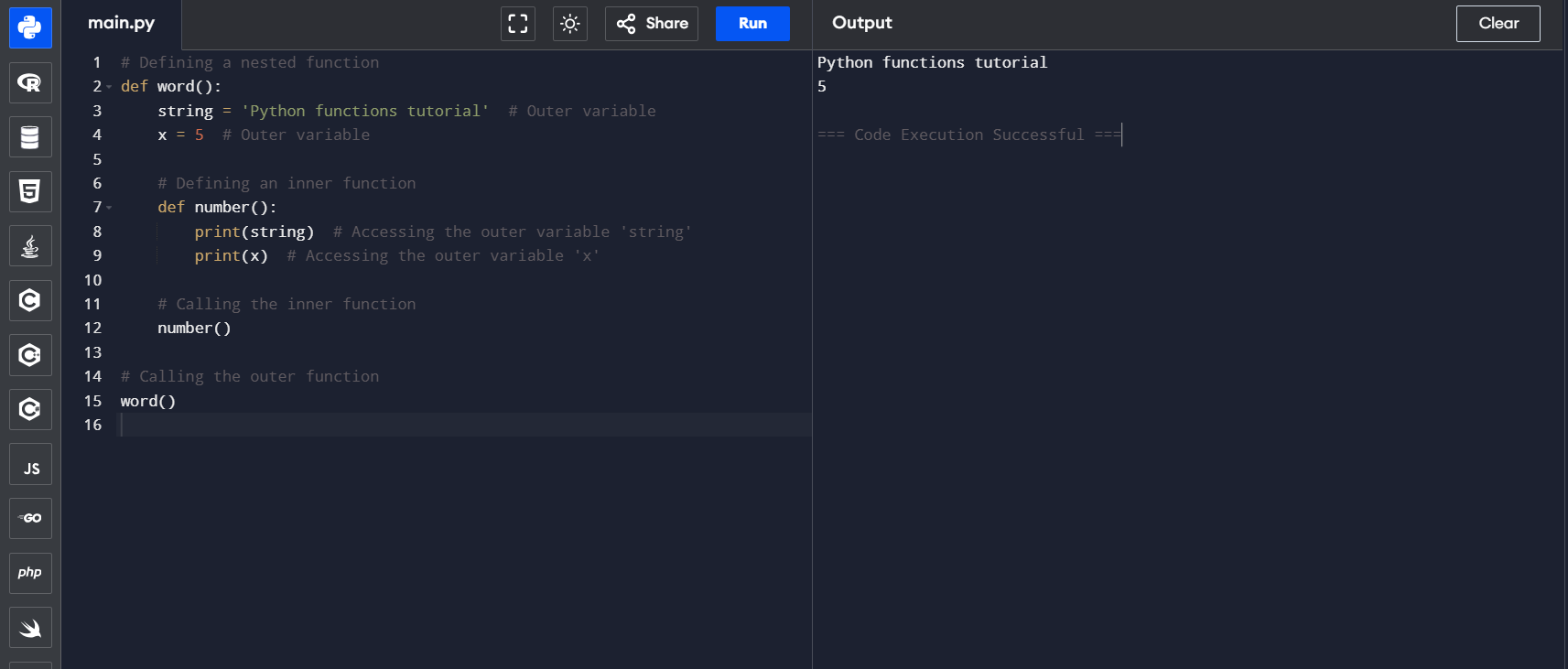
**The Anonymous Functions**

****

**Scope and Lifetime of Variables**

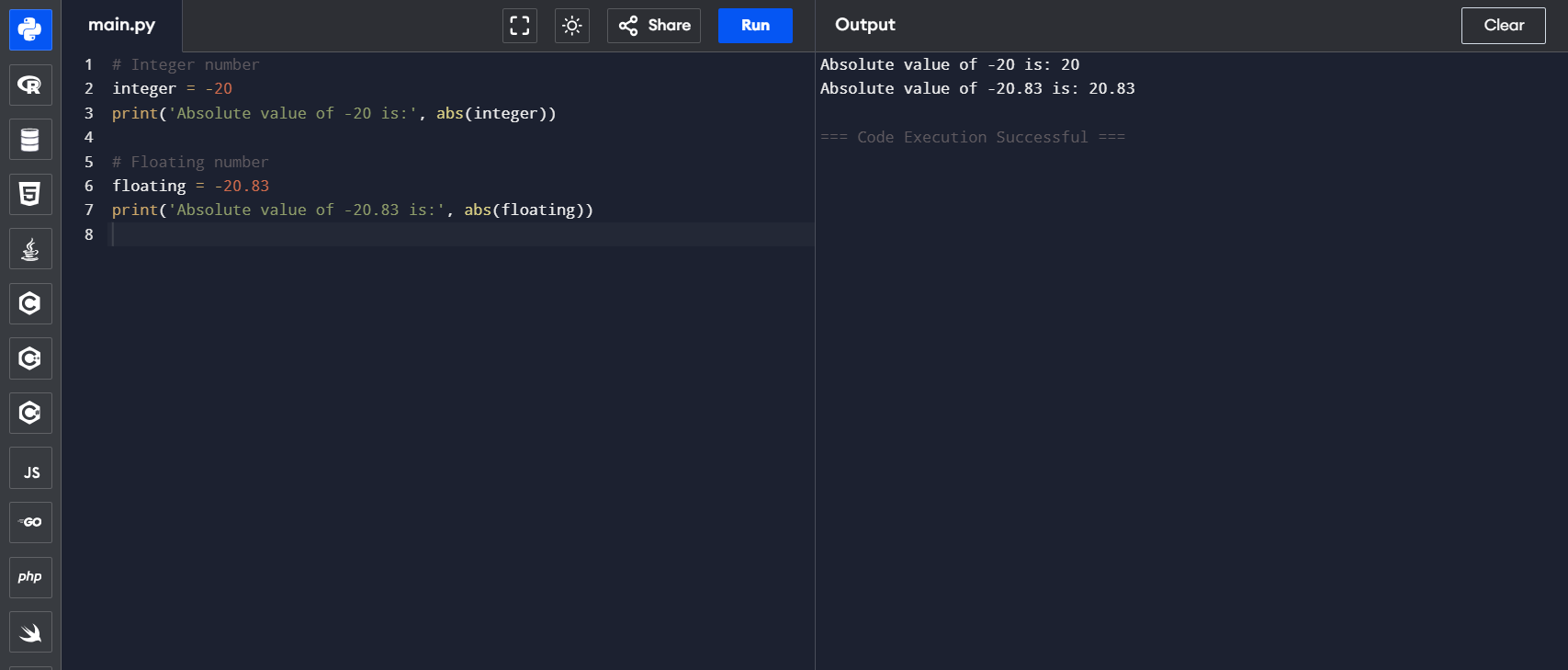
****

**Python Capability inside Another Capability**

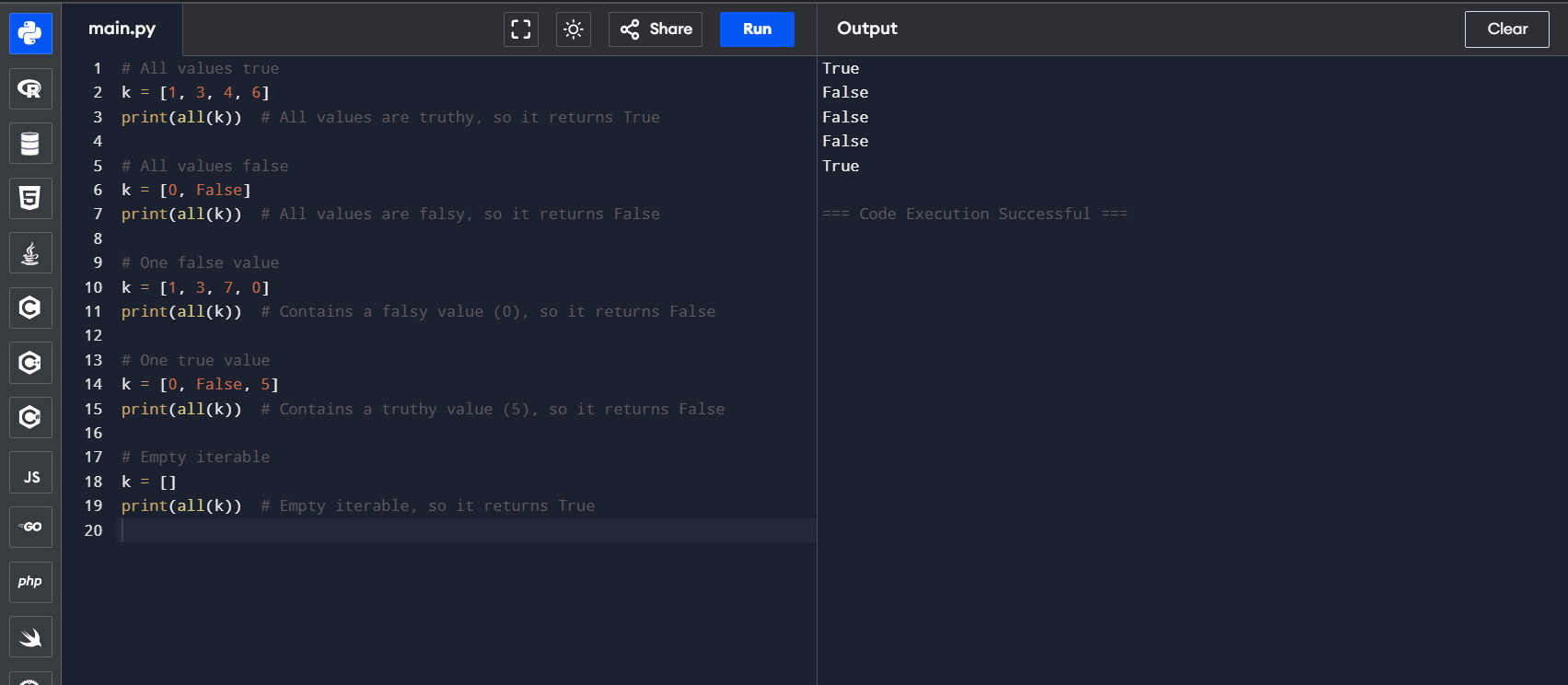
****

**Python Built-in Functions**

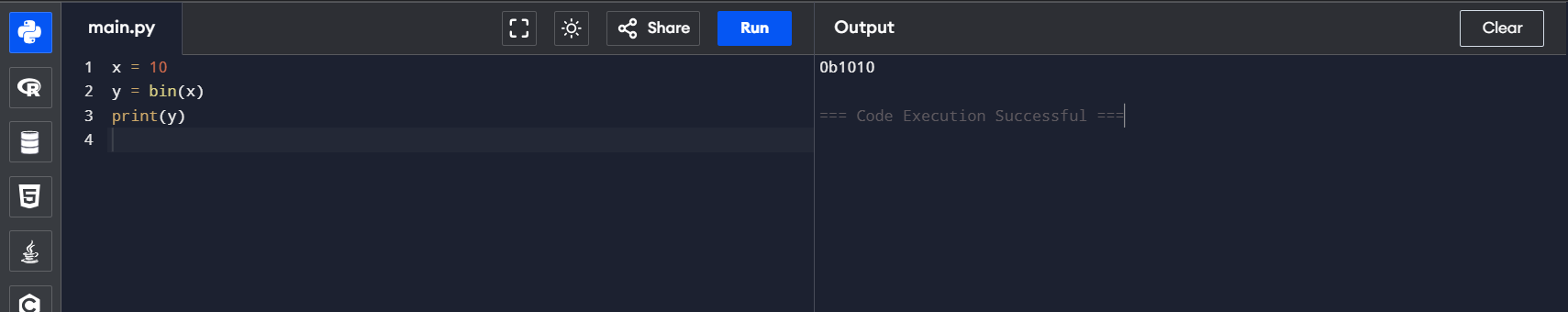
**Python abs() Function Example**

****

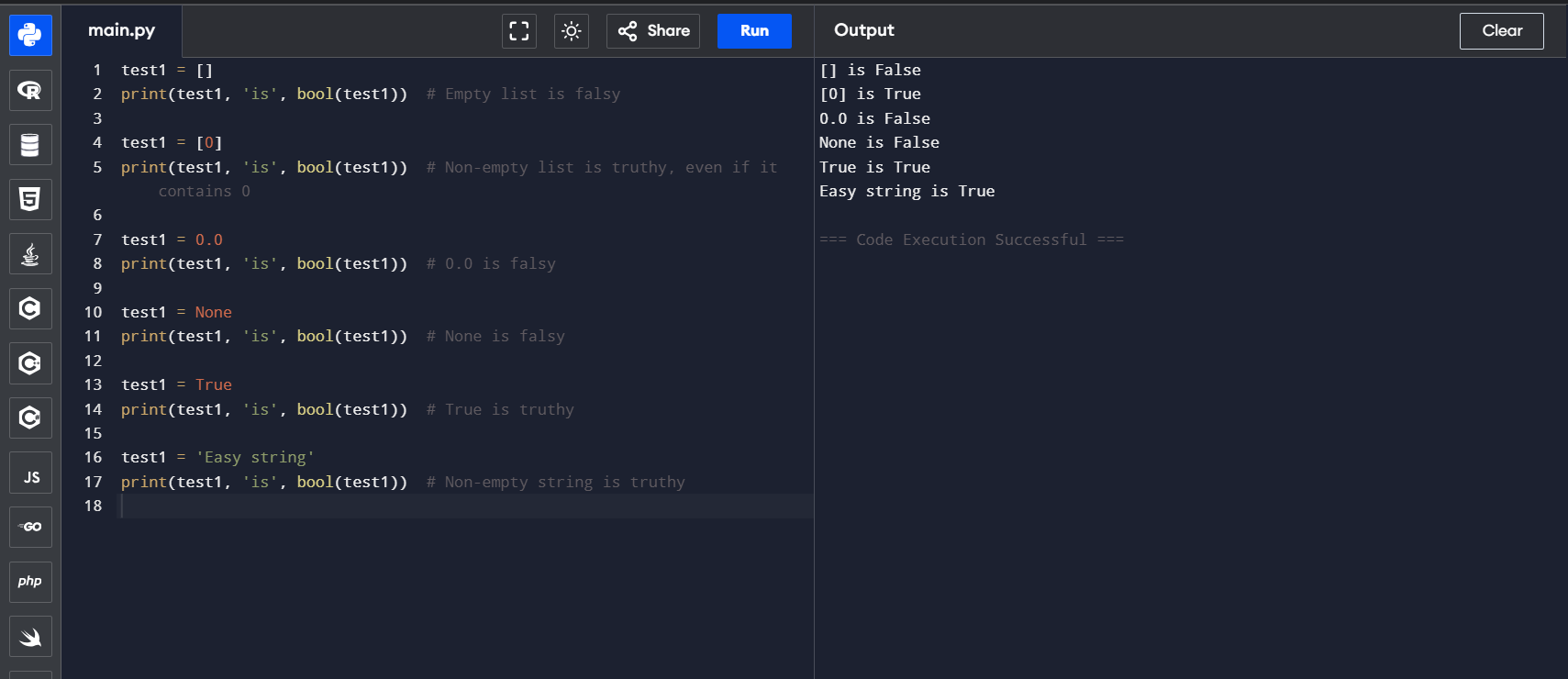
**Python all() Function**

****

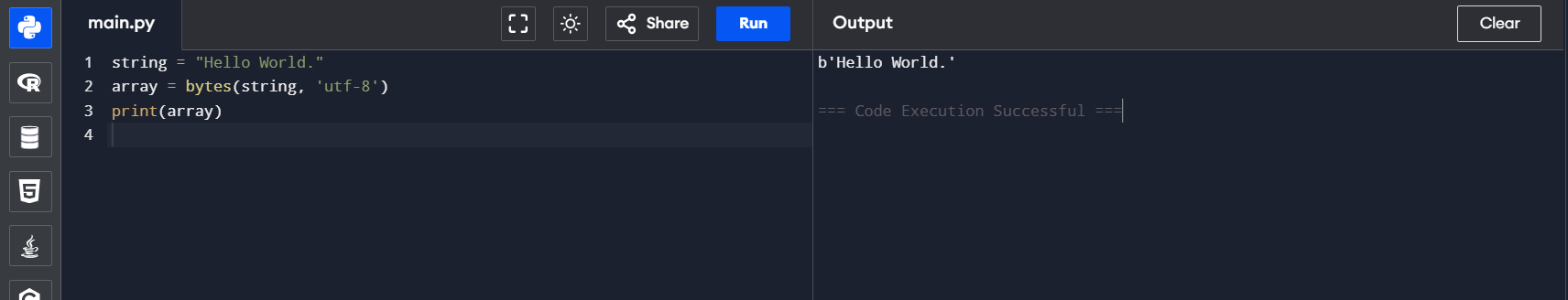
**Python bin() Function**

****

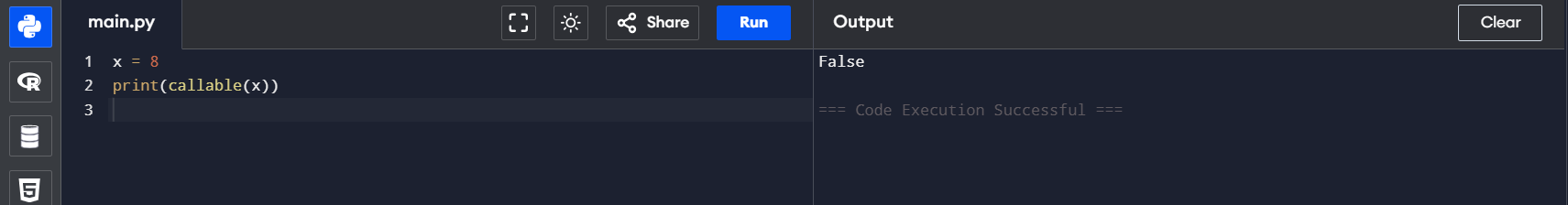
**Python bool()**

****

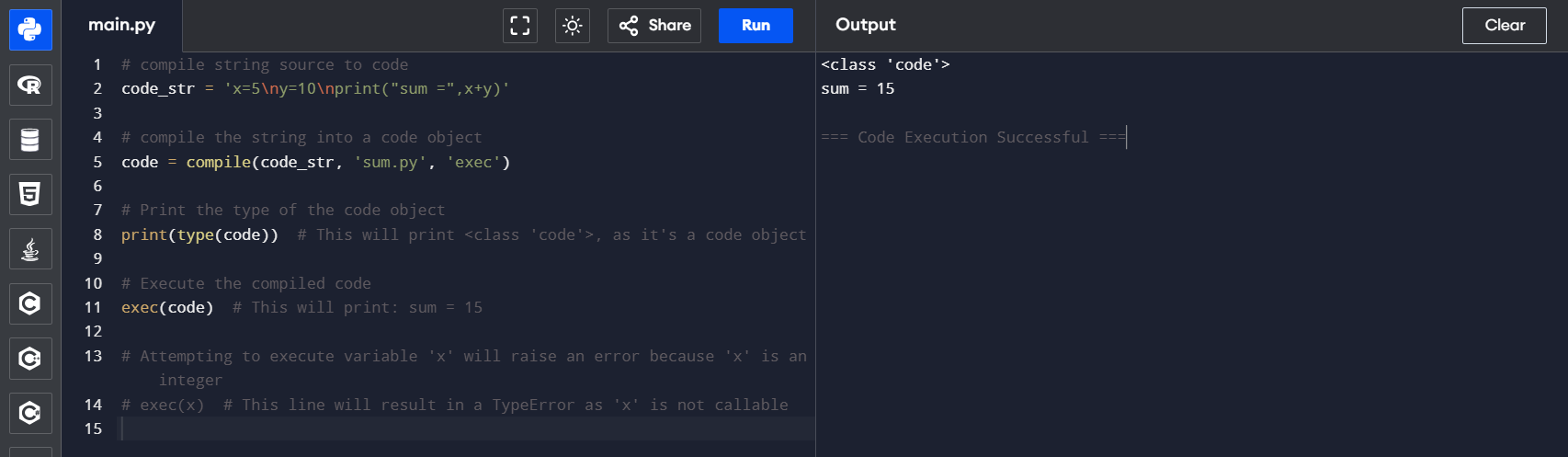
**Python bytes()**

****

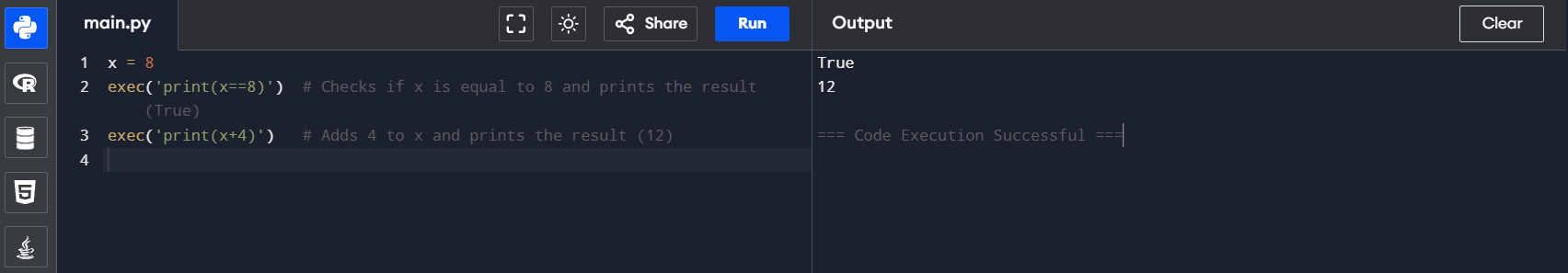
**Python callable() Function**

****

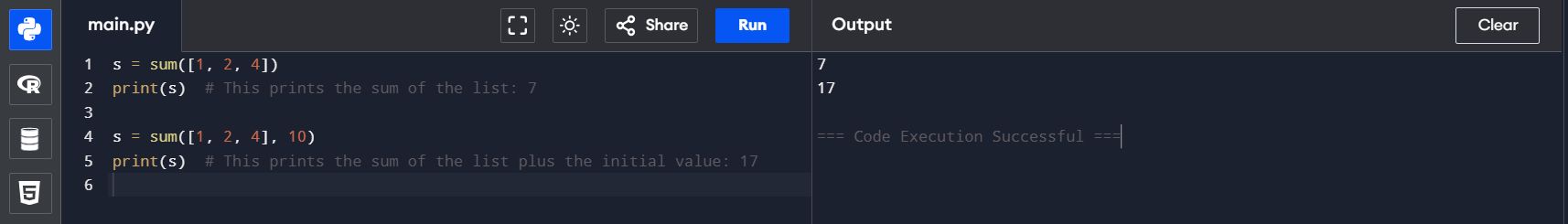
**Python compile() Function**

****

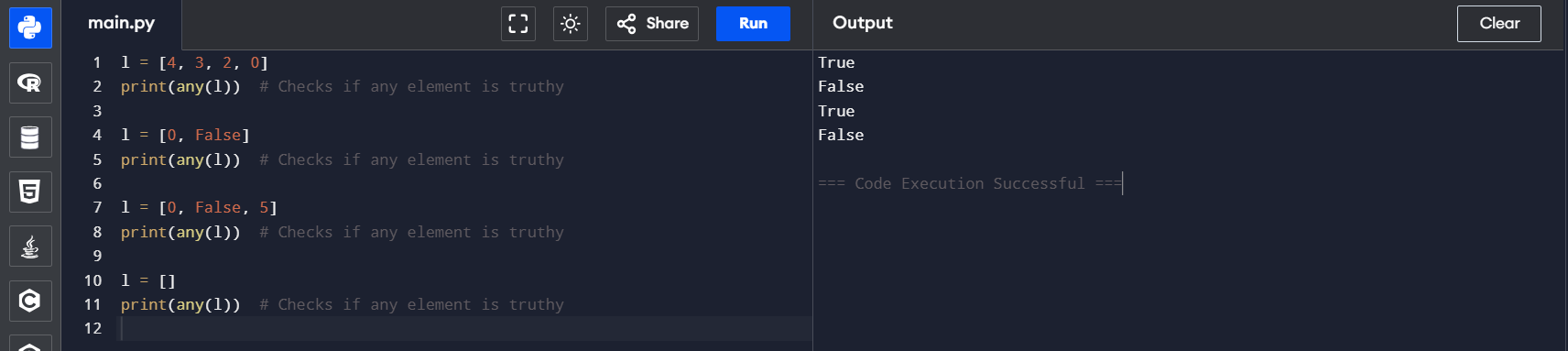
**Python exec() Function**

****

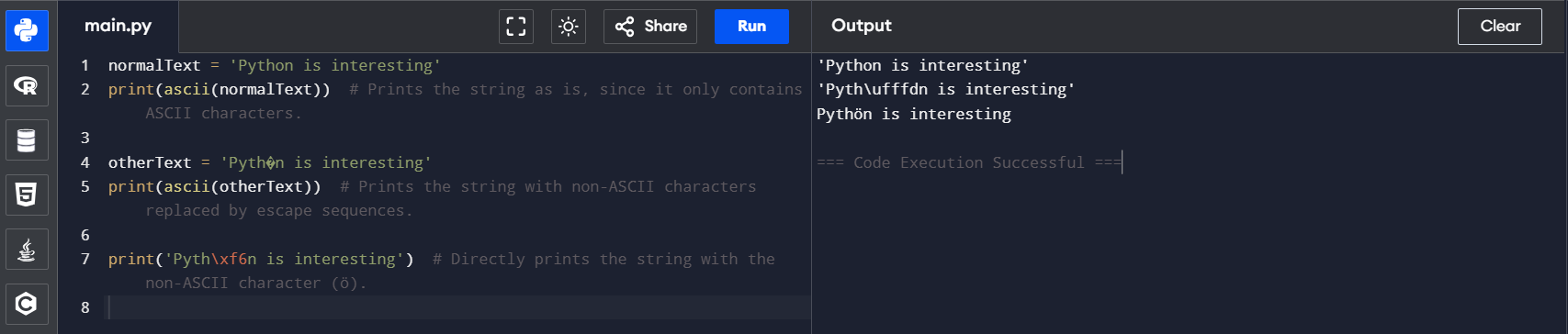
**Python sum() Function**

****

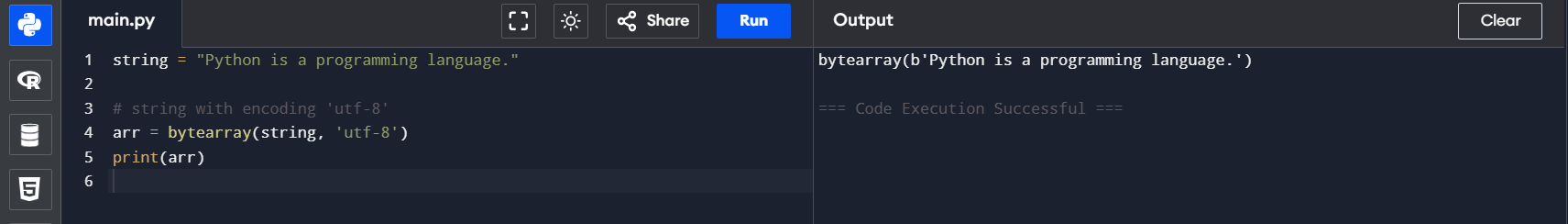
**Python any() Function**

****

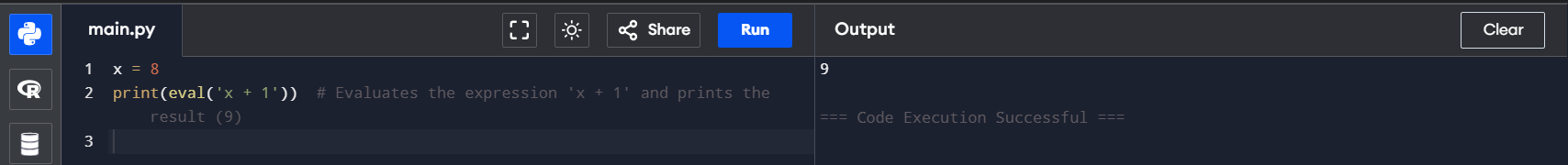
**Python ascii() Function**

****

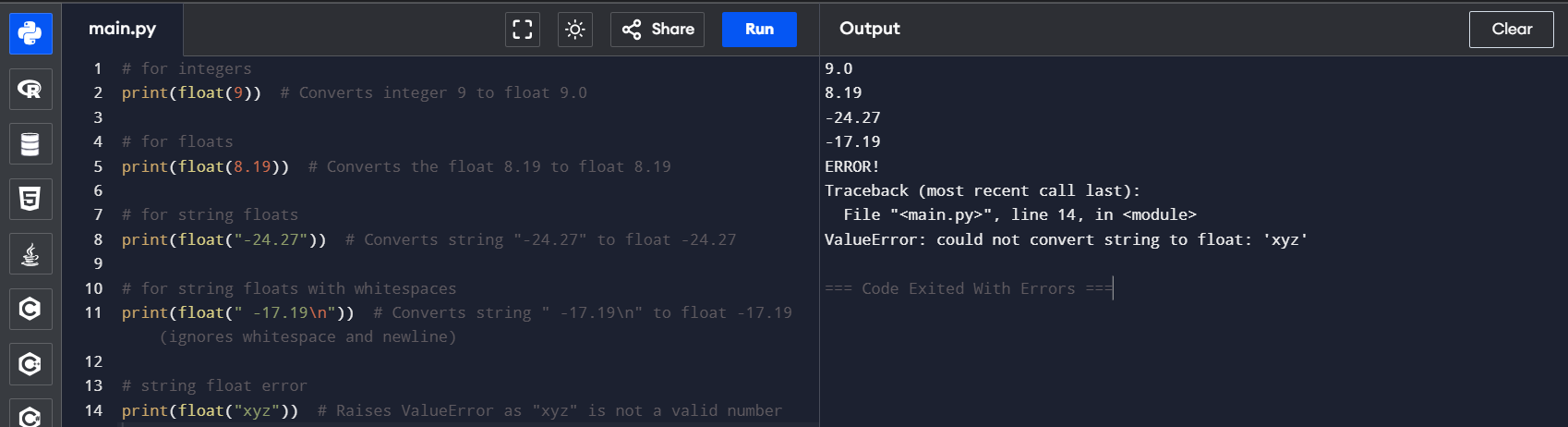
**Python bytearray()**

****

**Python eval() Function**

****

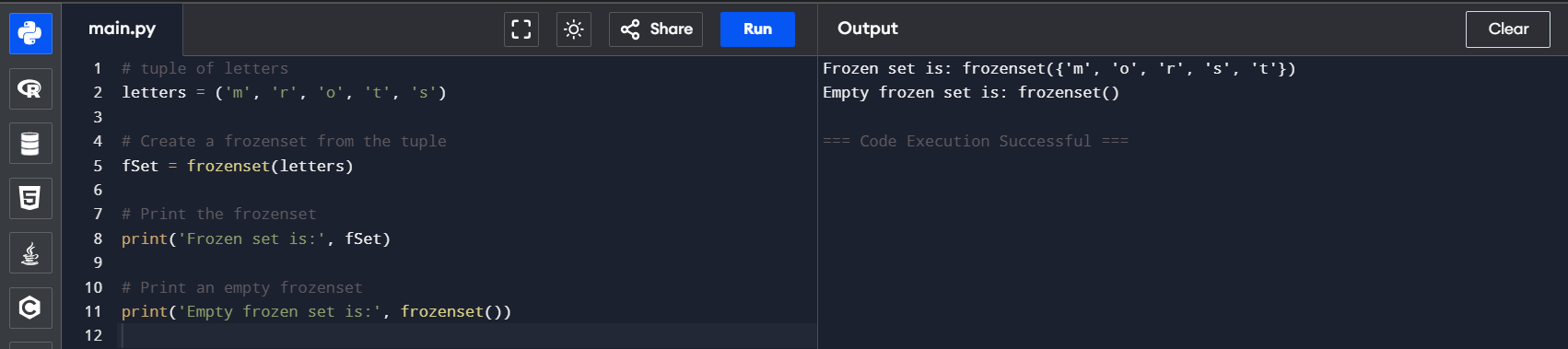
**Python float()**

****

**Python format() Function**

****

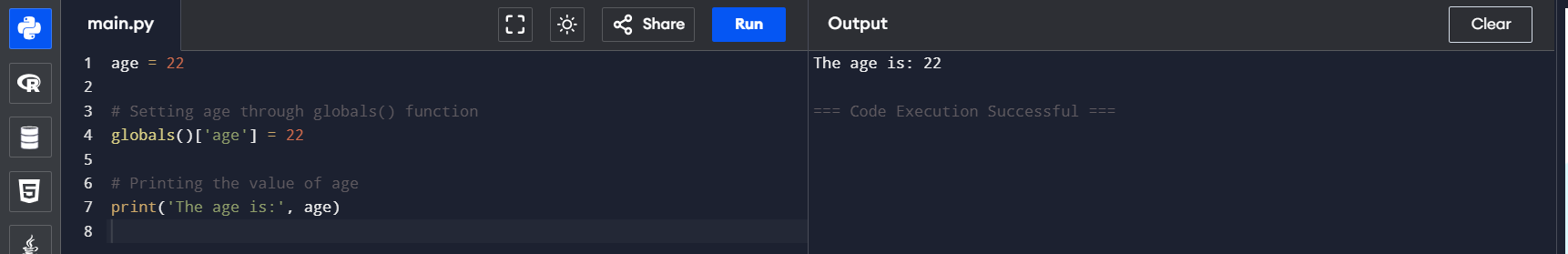
**Python frozenset()**

****

**Python getattr() Function**

****

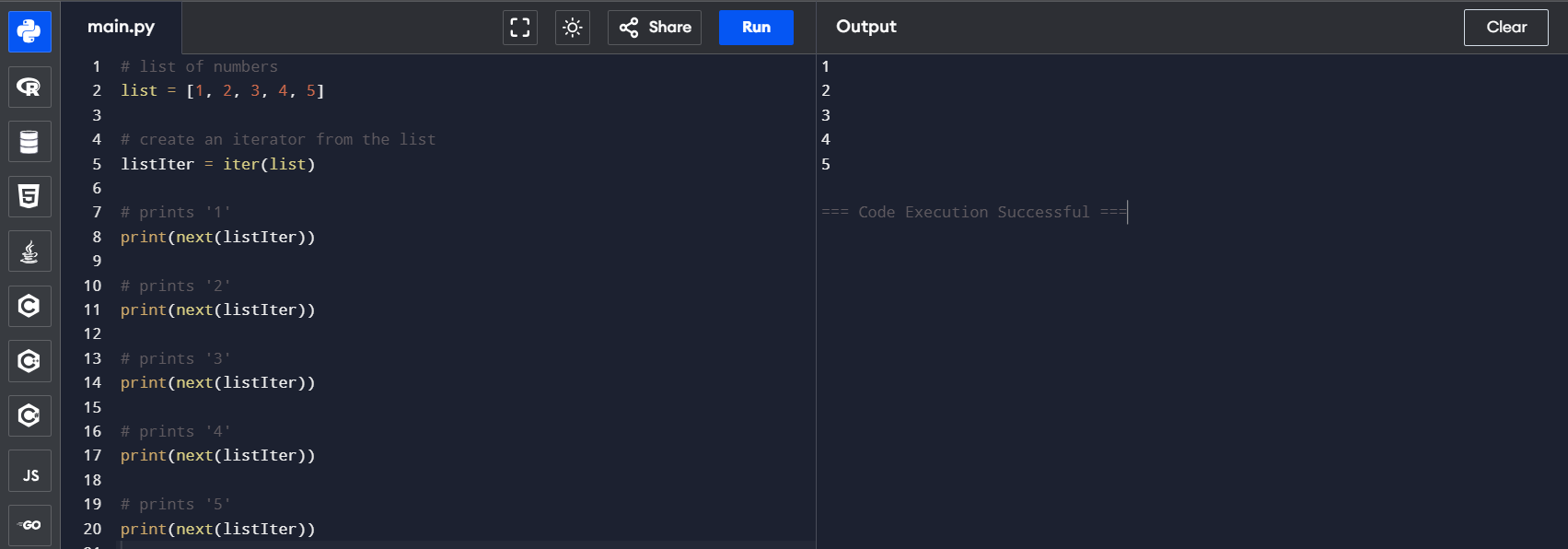
**Python globals() Function**

****

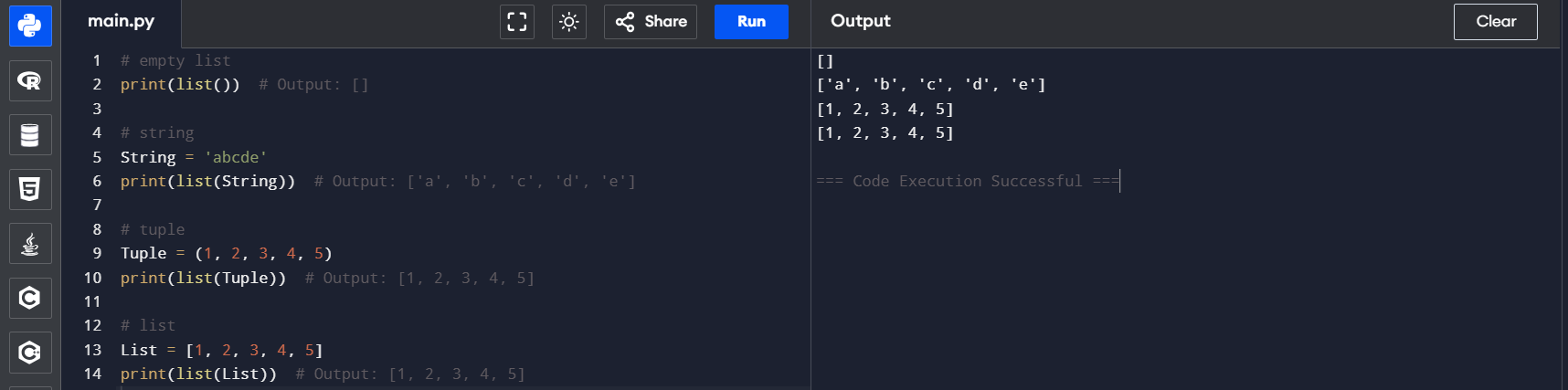
**Python hasattr() Function**

****

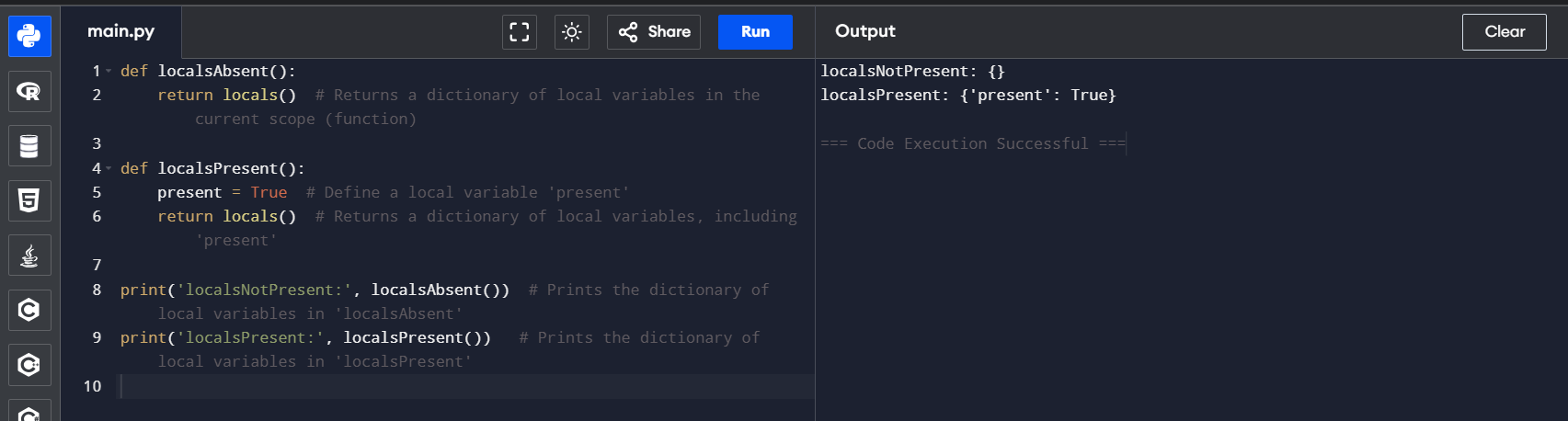
**Python iter() Function**

****

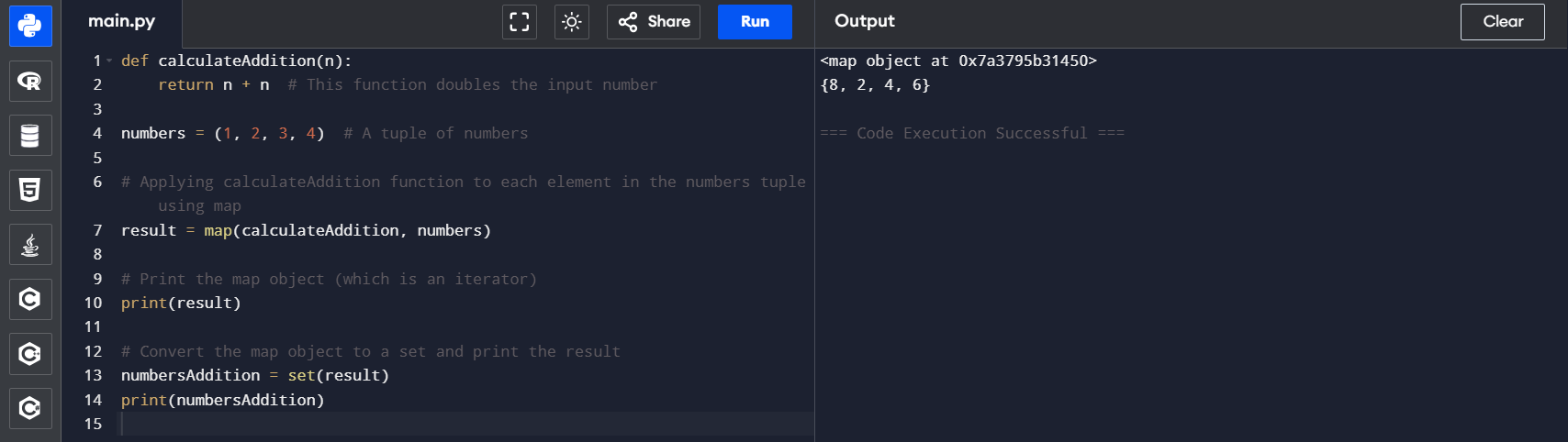
**Python list()**

****

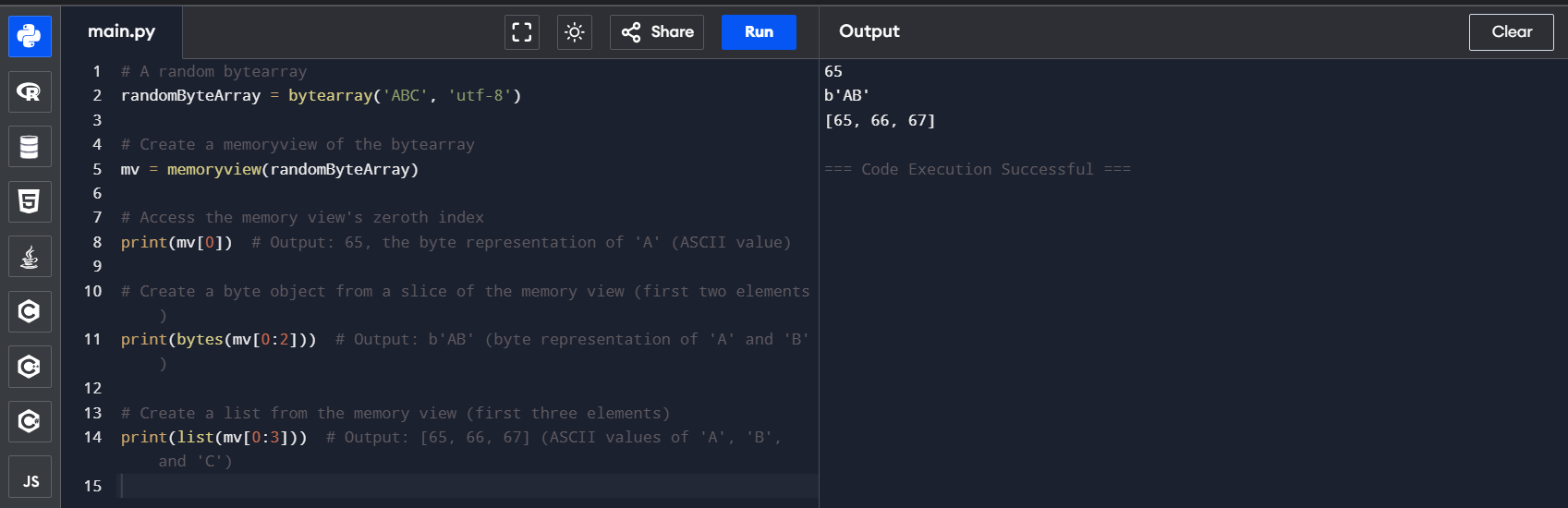
**Python locals() Function**

****

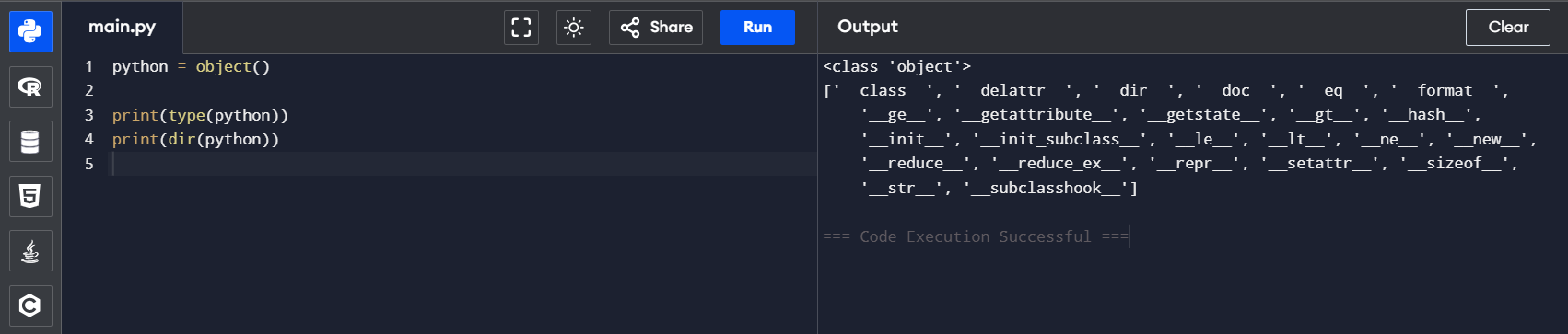
**Python map() Function**

****

**Python memoryview() Function**

****

**Python object()**

****

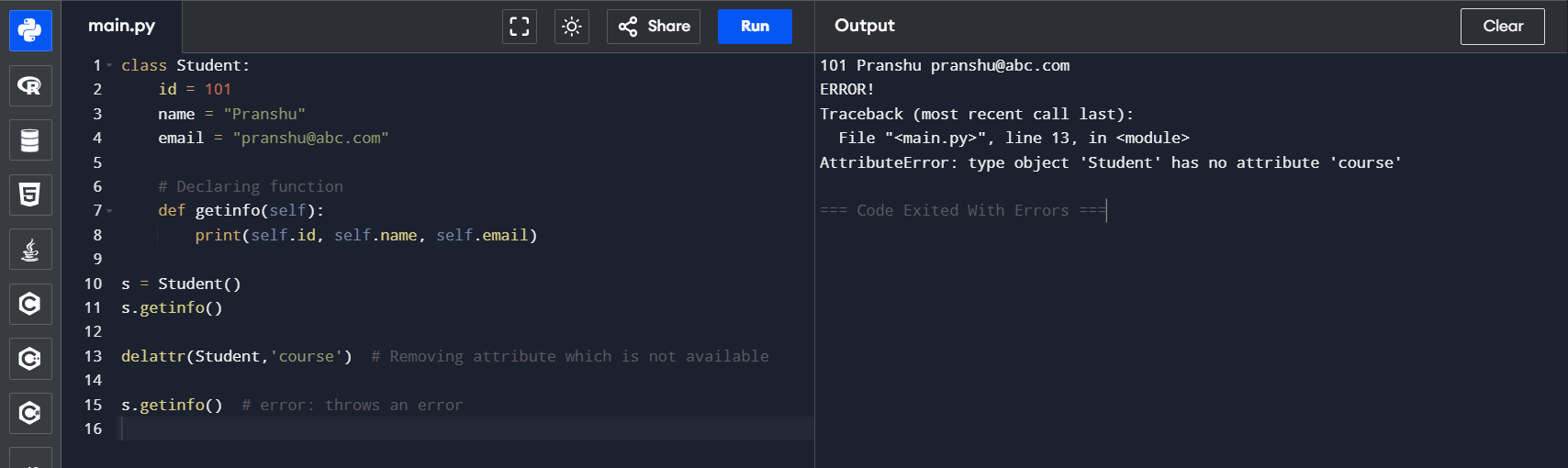
**Python chr() Function**

****

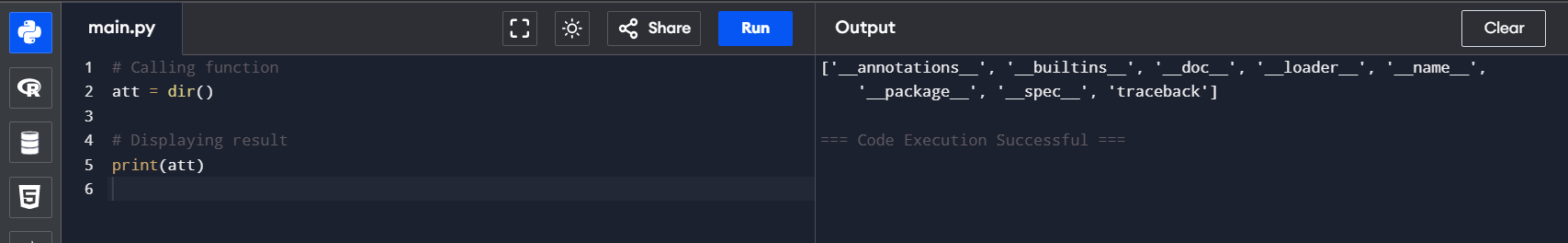
**Python complex()**

****

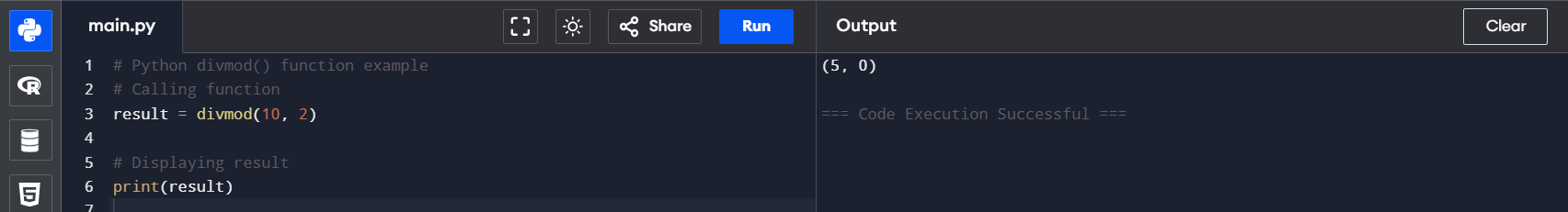
**Python delattr() Function**

****

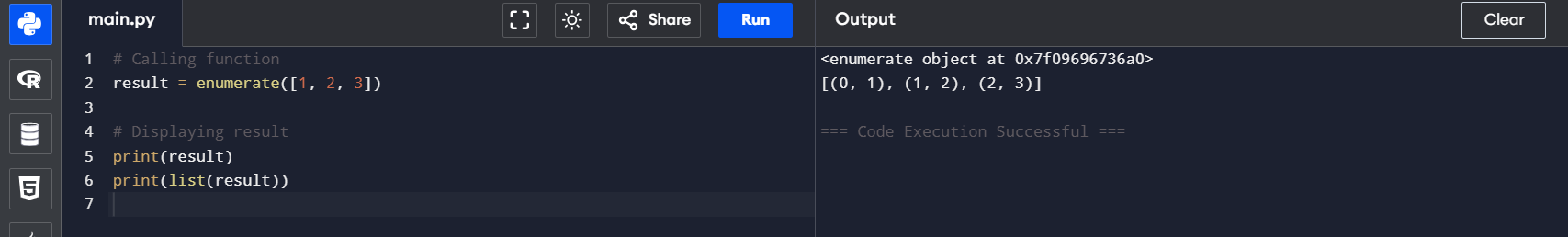
**Python dir() Function**

****

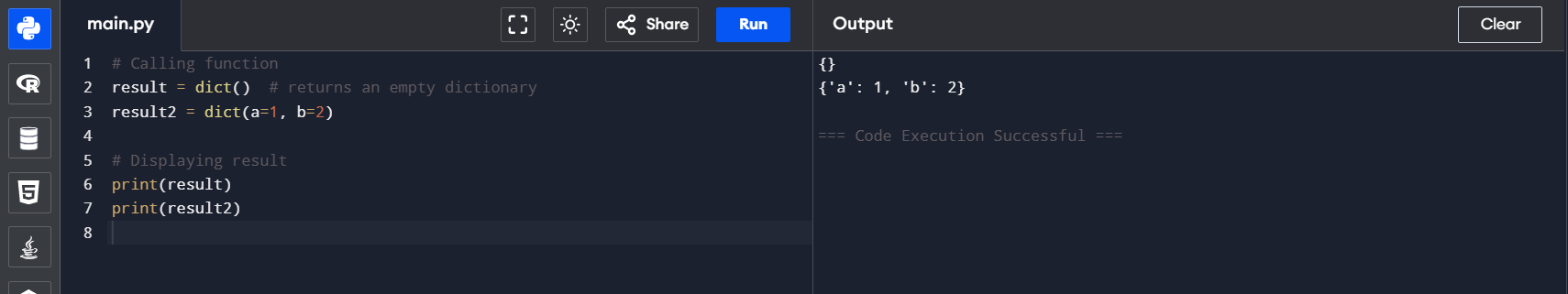
**Python divmod() Function**

****

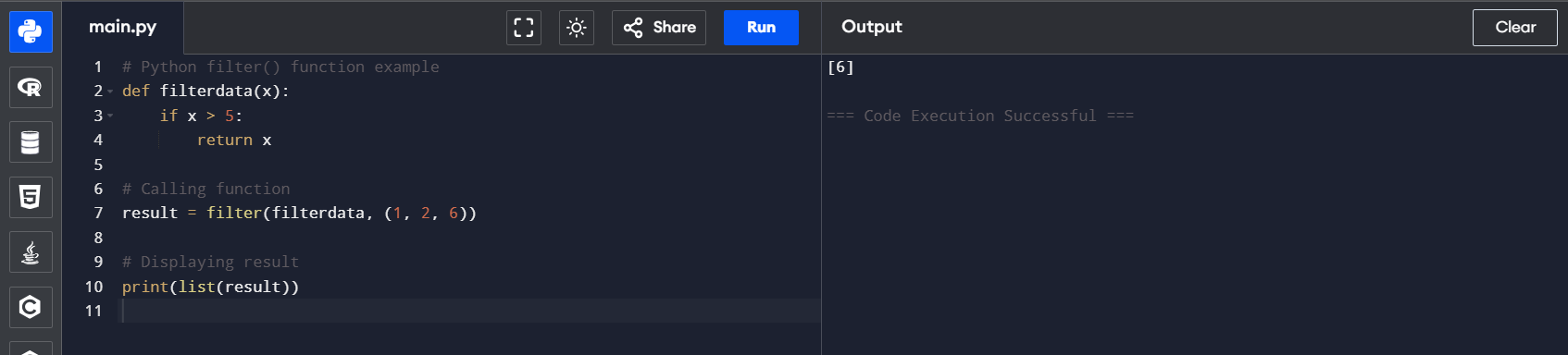
**Python enumerate() Function**

****

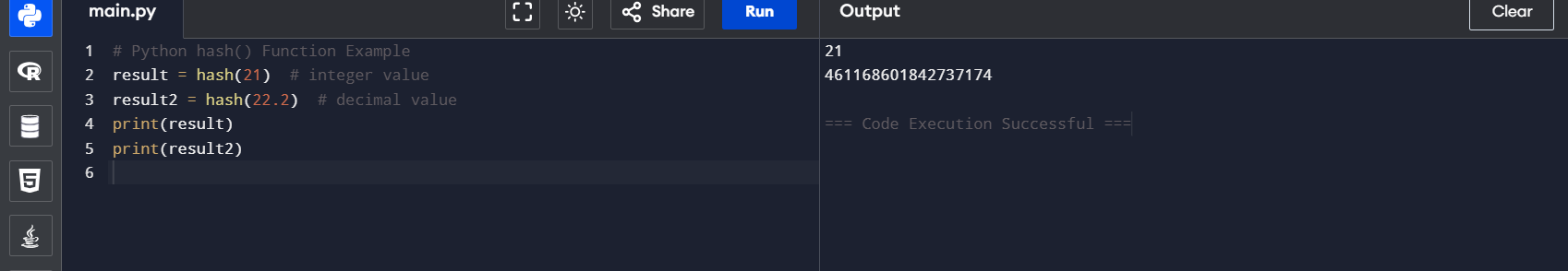
**Python dict()**

****

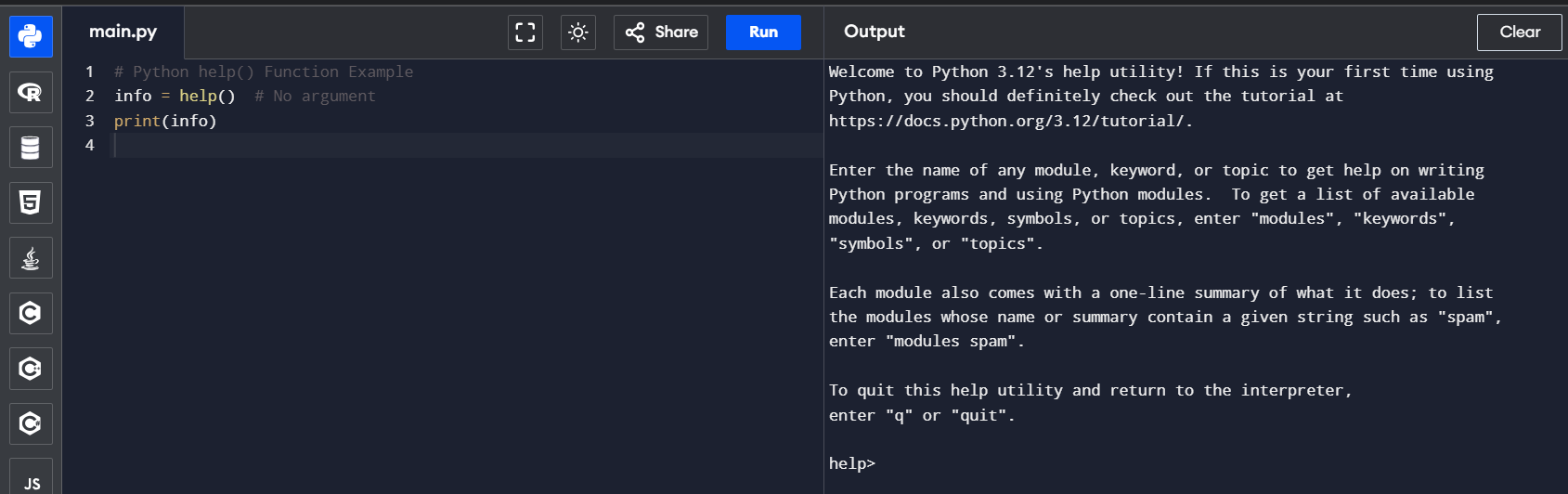
**Python filter() Function**

****

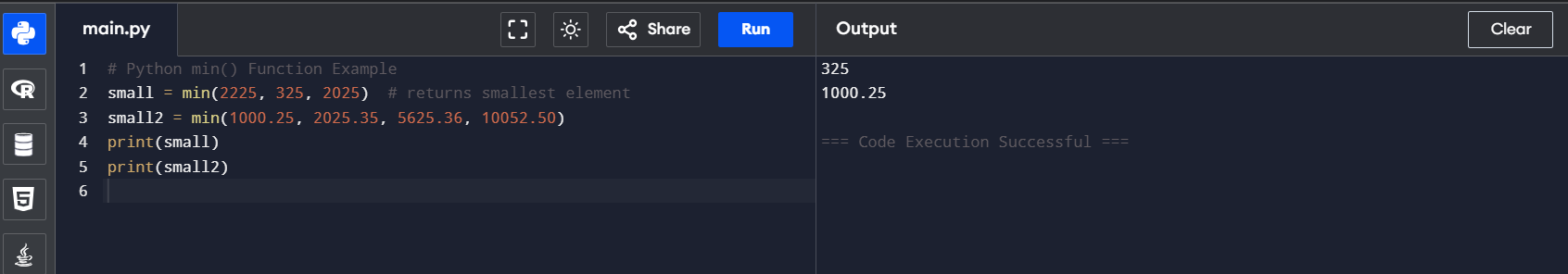
**Python hash() Function**

****

**Python help() Function**

****

**Python min() Function**

****

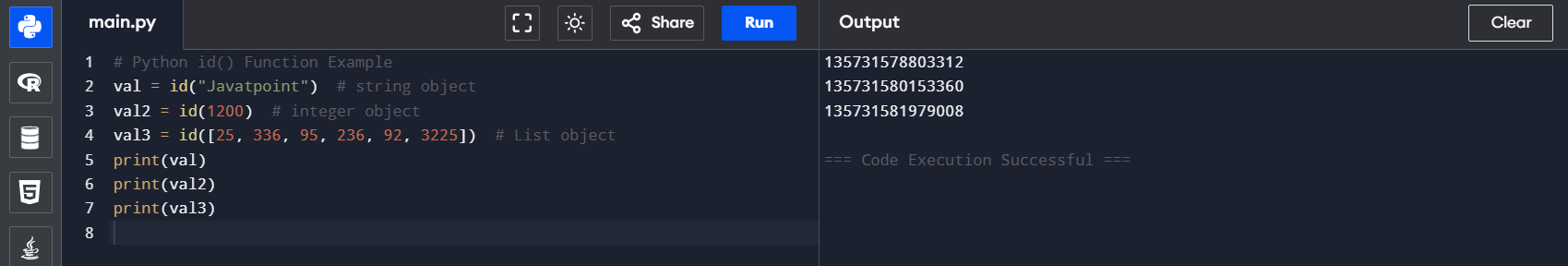
**Python set() Function**

****

**Python hex() Function**

****

**Python id() Function**

****

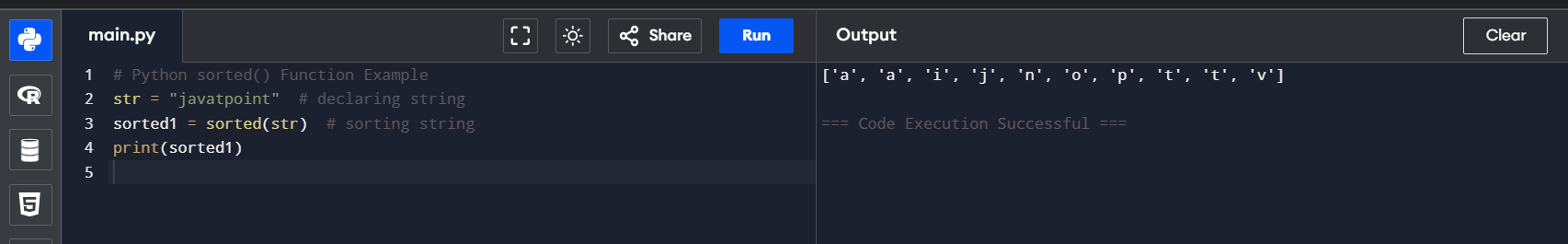
**Python setattr() Function**

****

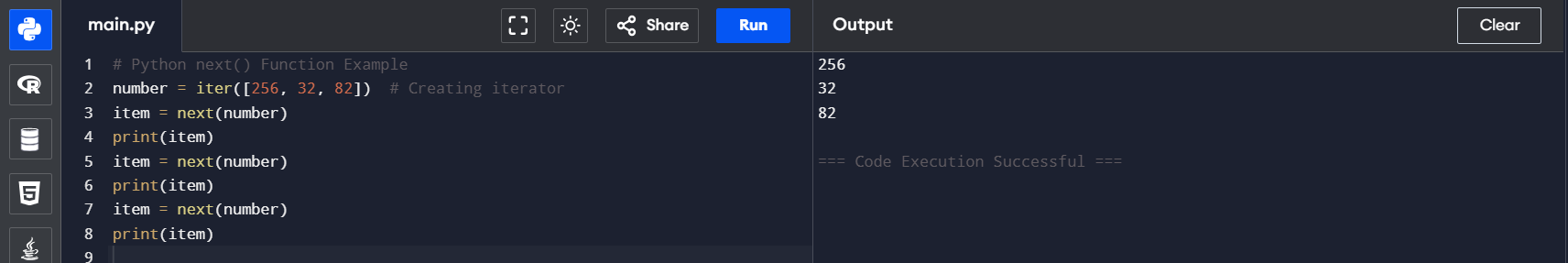
**Python slice() Function**

****

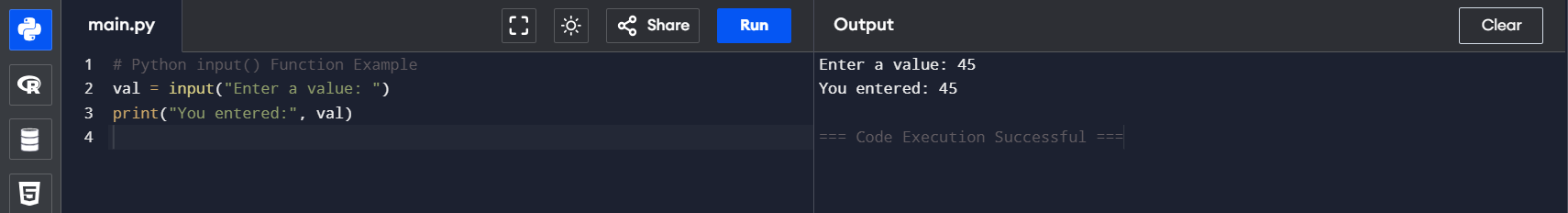
**Python sorted() Function**

****

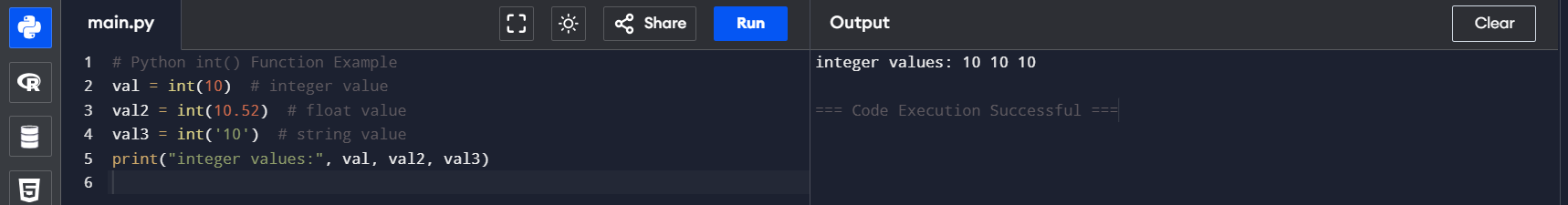
**Python next() Function**

****

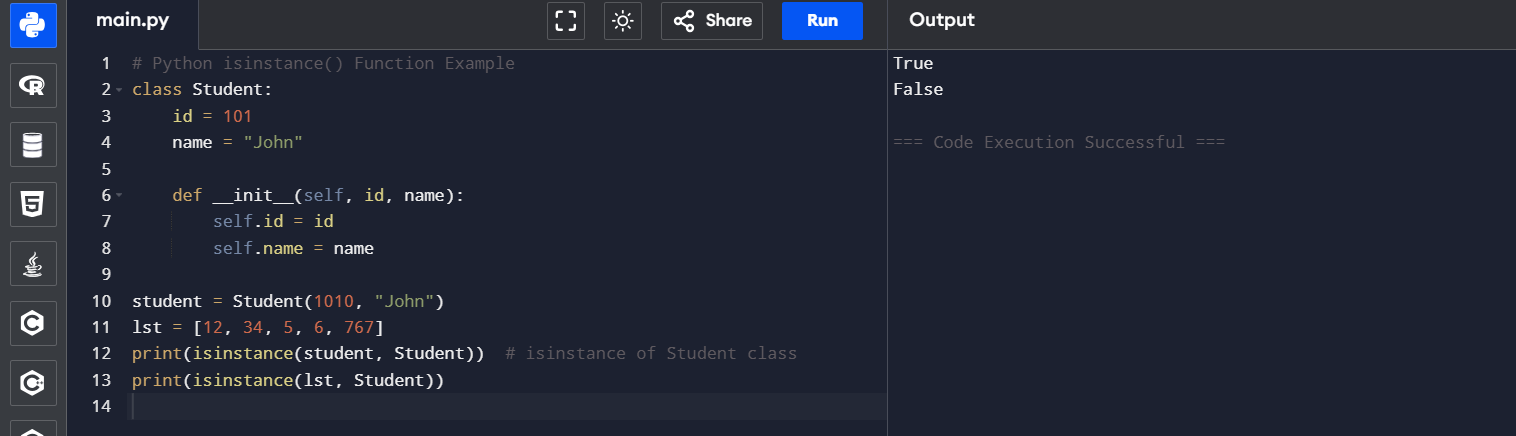
**Python input() Function**

****

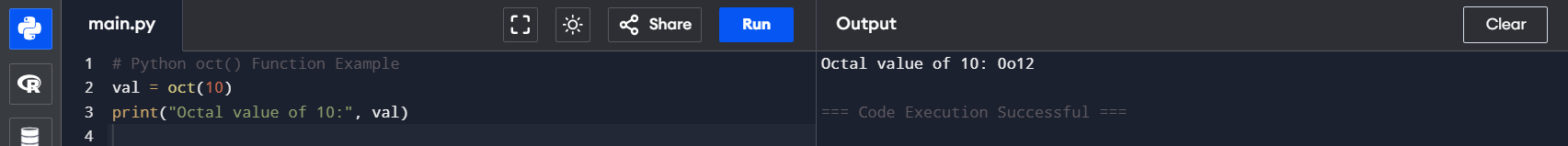
**Python int() Function**

****

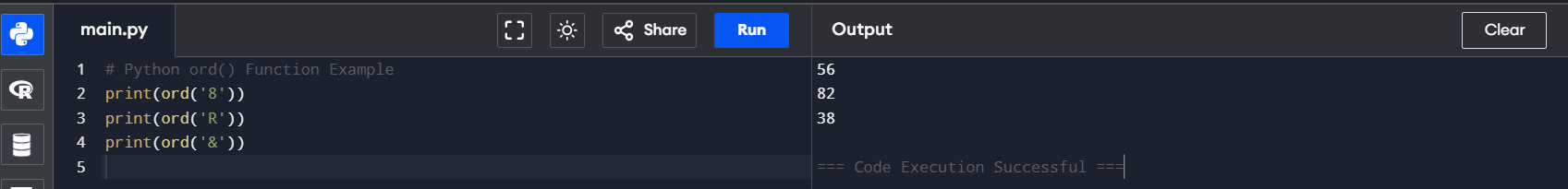
**Python isinstance() Function**

****

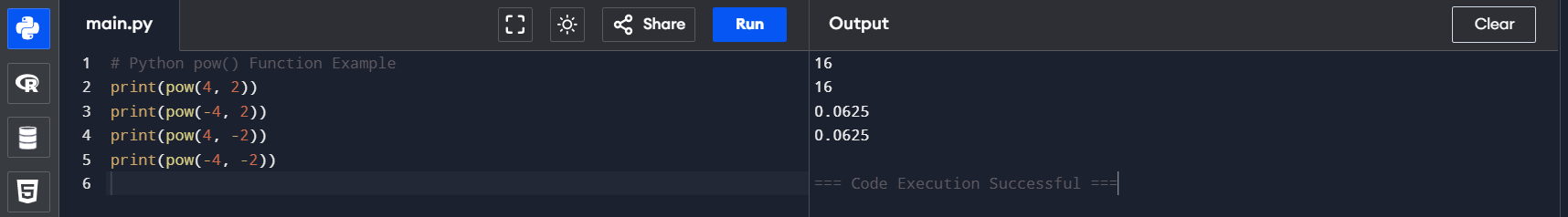
**Python oct() Function**

****

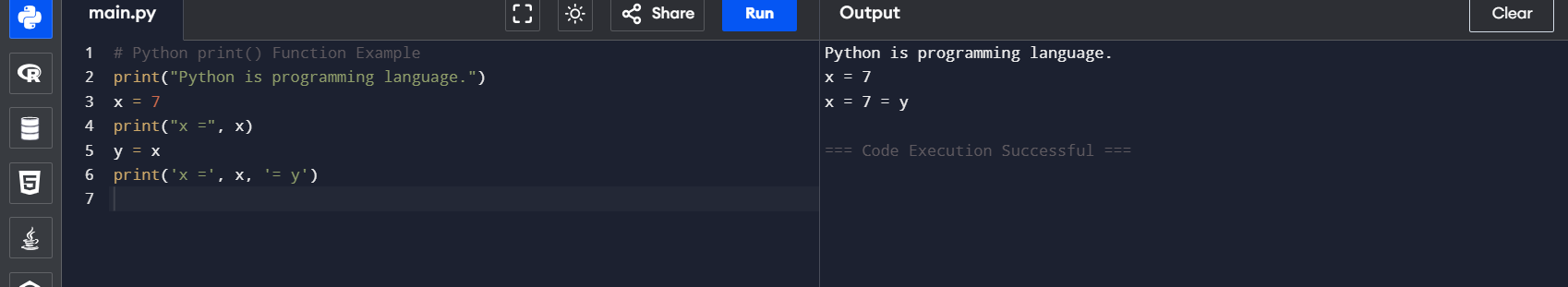
**Python ord() Function**

****

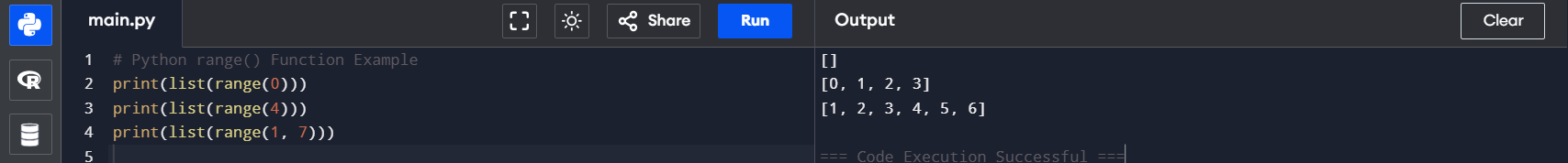
**Python pow() Function**

****

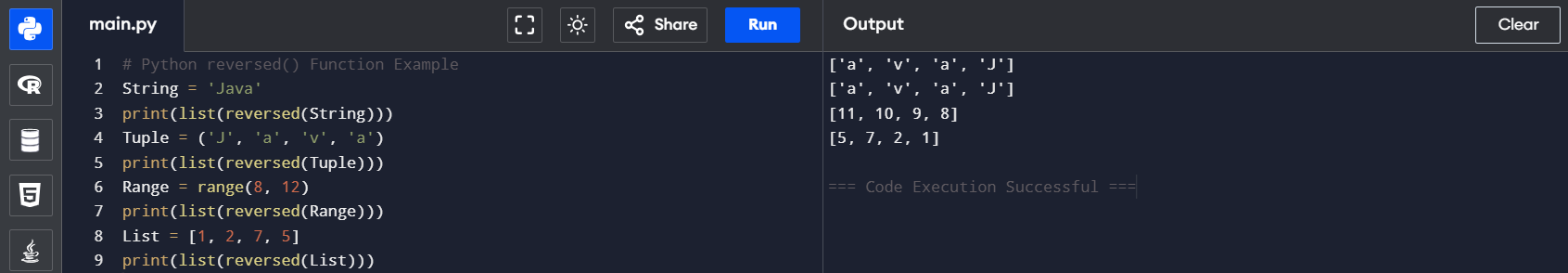
**Python print() Function**

****

**Python range() Function**

****

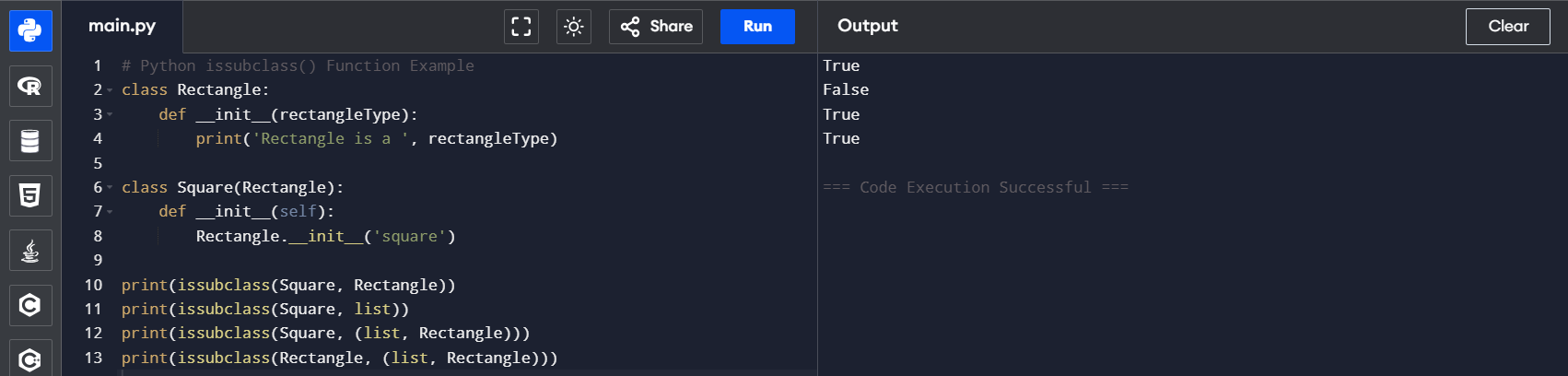
**Python reversed() Function**

****

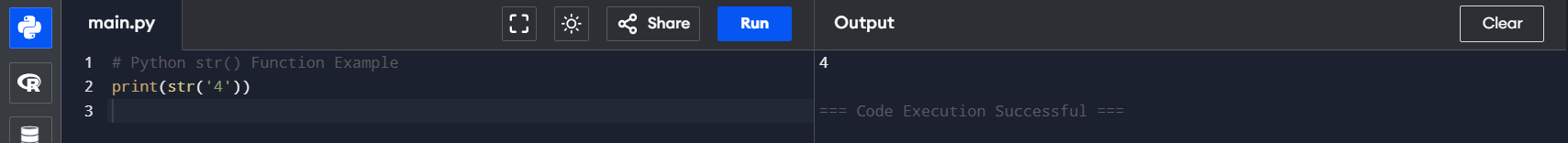
**Python round() Function**

****

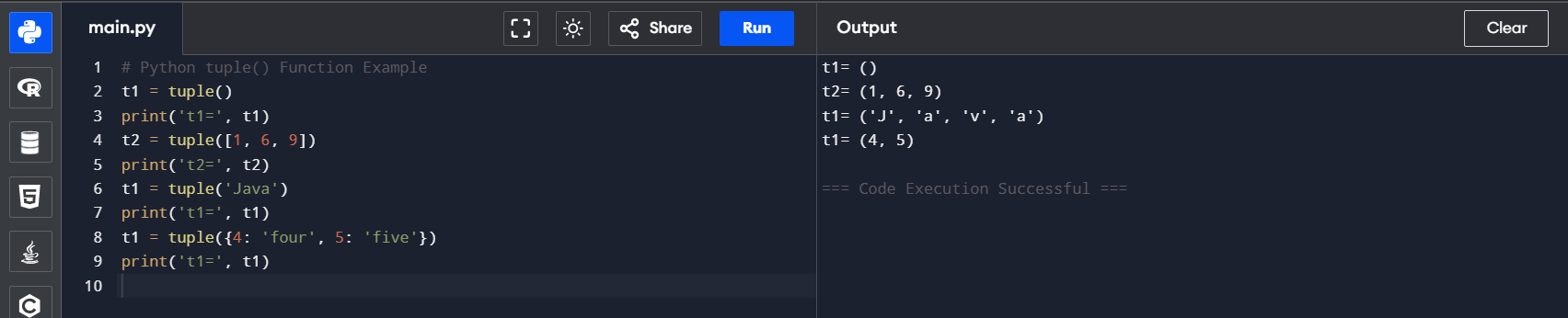
**Python issubclass() Function**

****

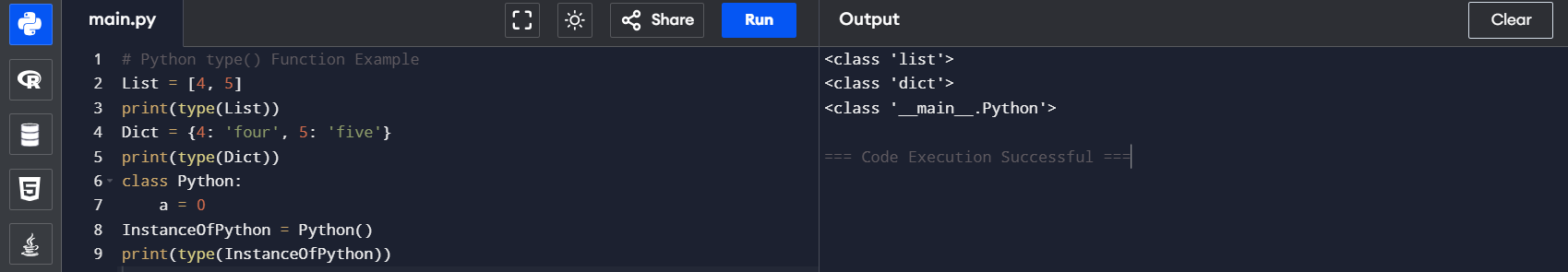
**Python str**

****

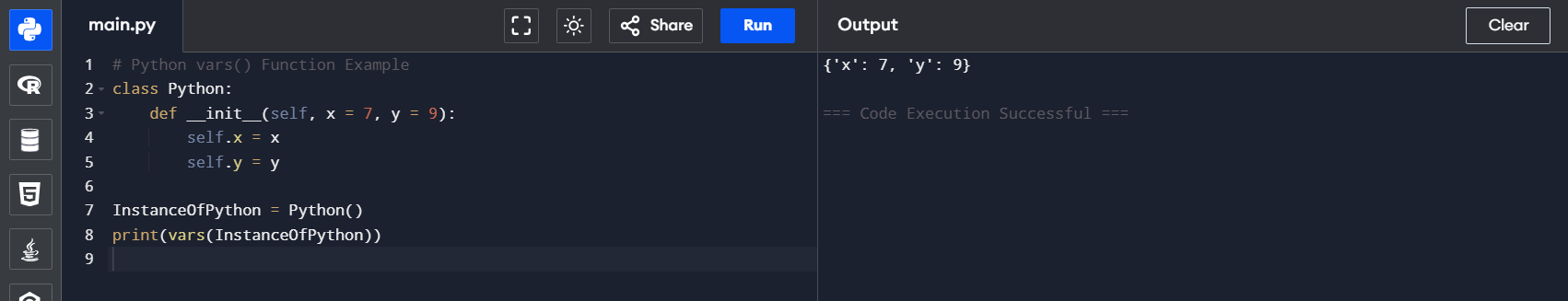
**Python tuple() Function**

****

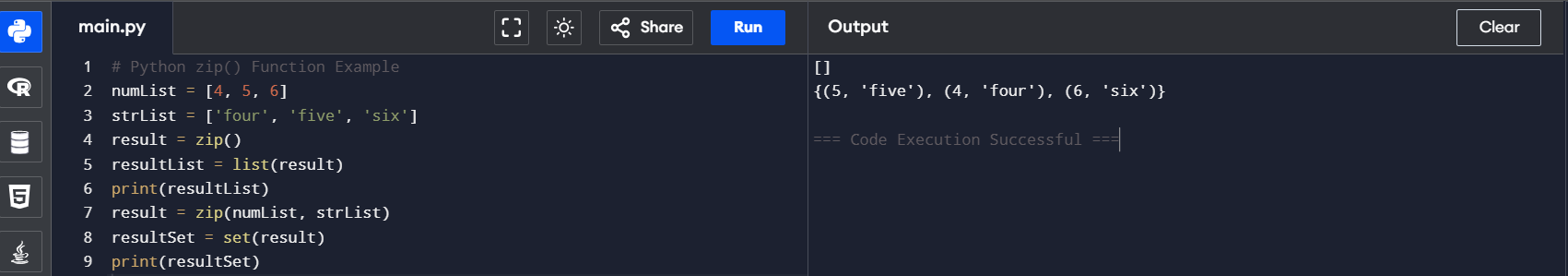
**Python type()**

****

**Python vars() function**

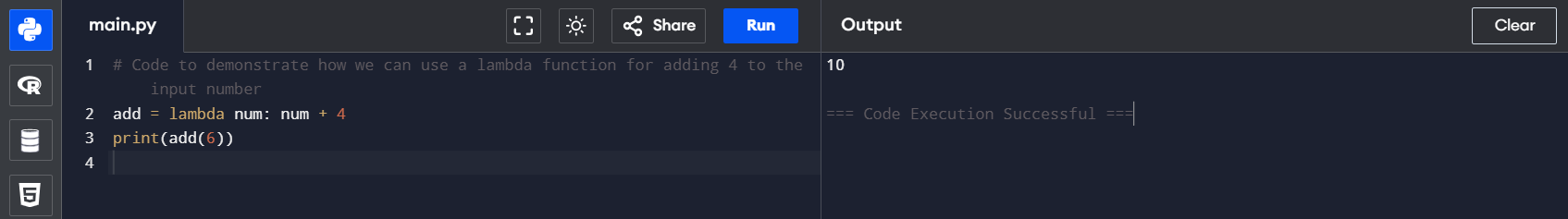
****

**Python zip() Function**

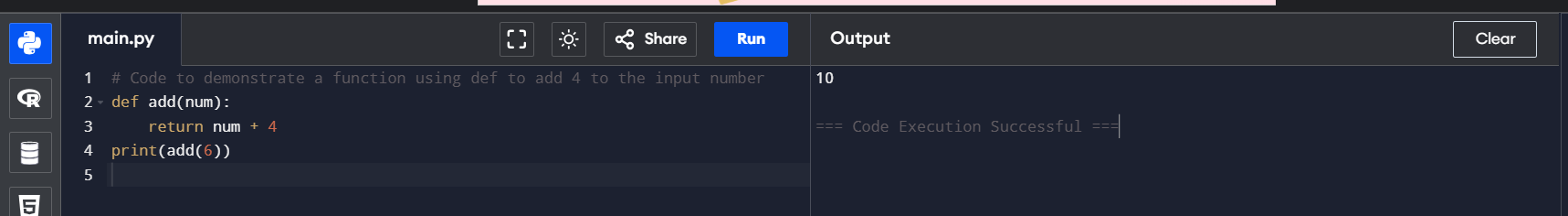
****

**Python Lambda Functions**

**Code 1:**

****

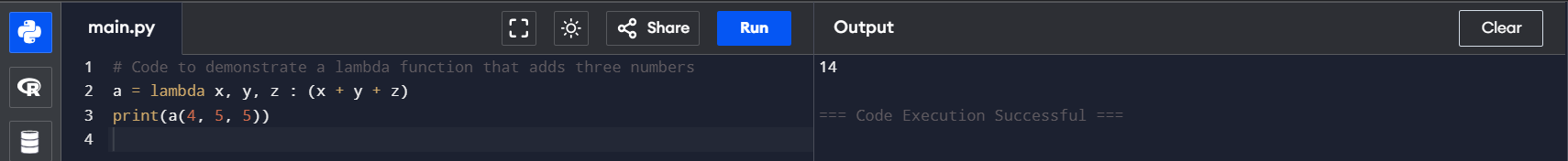
**Program Code 2:**

****

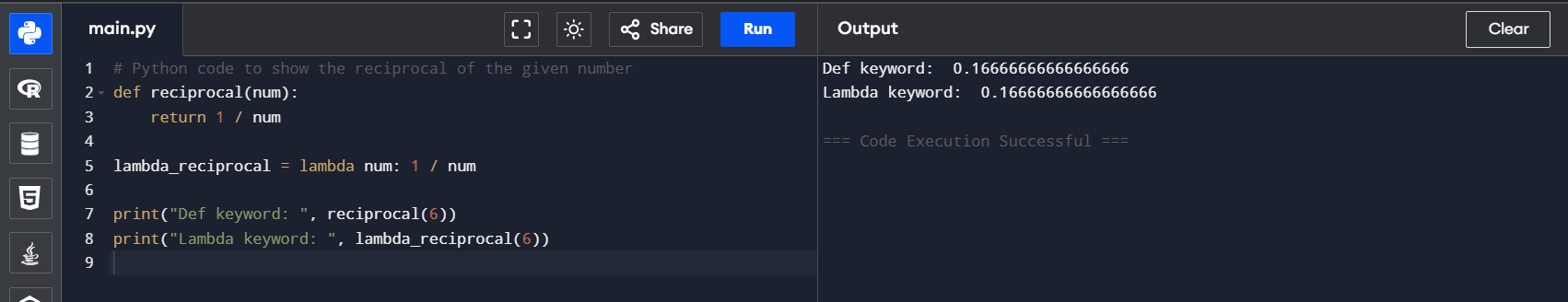
**Program Code 3:**

****

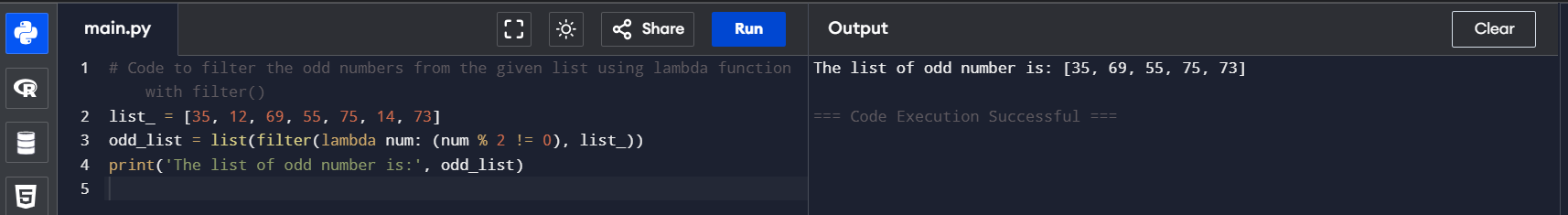
**Program Code 4:**

****

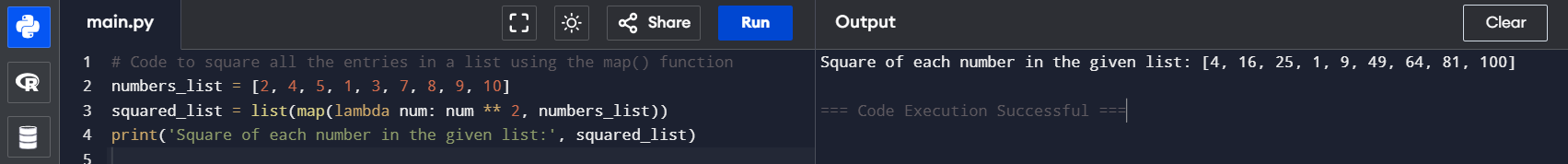
**What's the Distinction Between Lambda and Def Functions?**

****

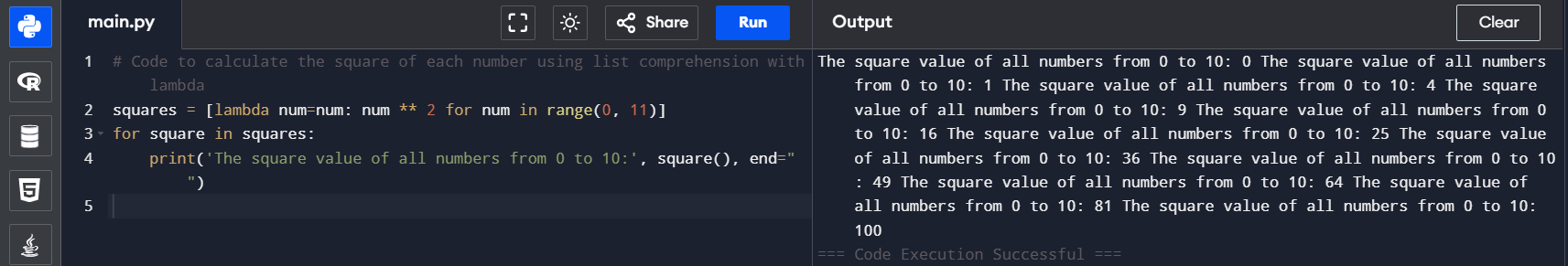
**Using Lambda Function with filter()**

****

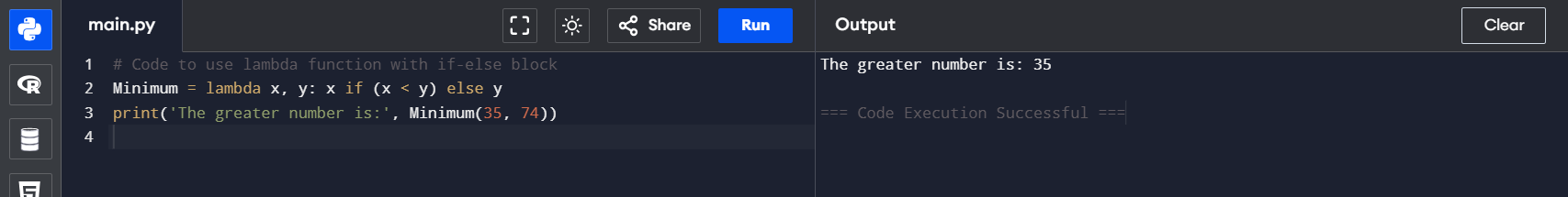
**Using Lambda Function with map()**

****

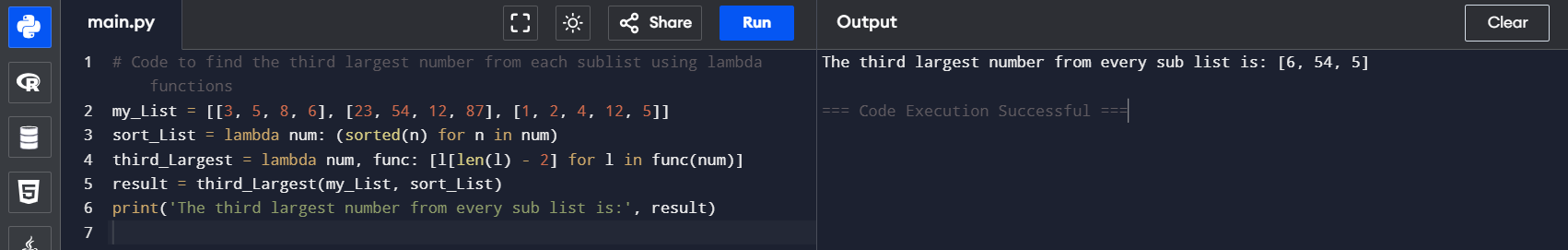
**Using Lambda Function with List Comprehension**

****

**Using Lambda Function with if-else**

****

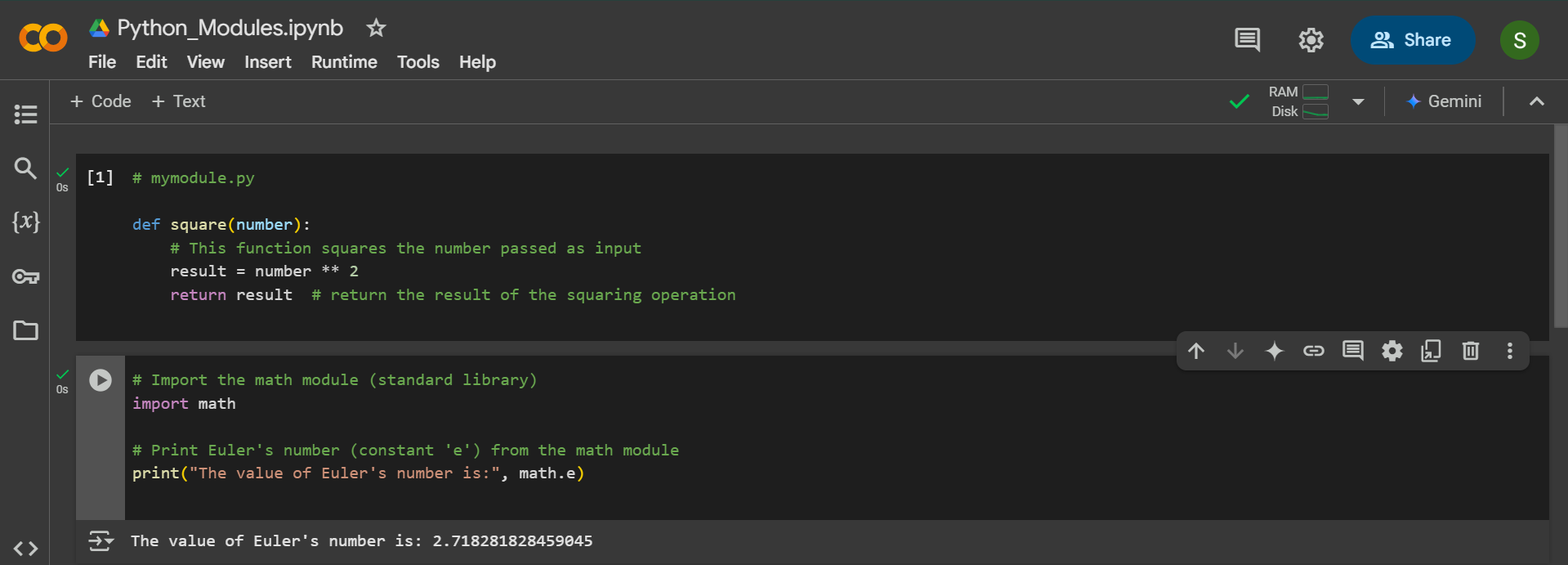
**Using Lambda with Multiple Statements**

****

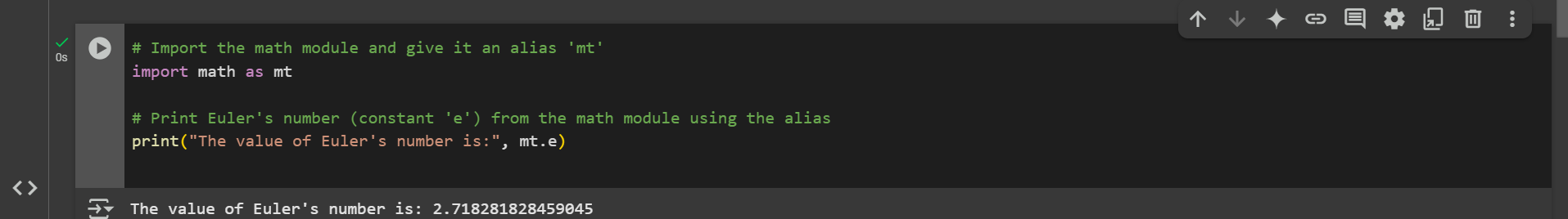
**Python Modules**

Modular programming is the practice of segmenting a single, complicated coding task into multiple, simpler, easier-to-manage sub-tasks. We call these subtasks modules. Therefore, we can build a bigger program by assembling different modules that act like building blocks.

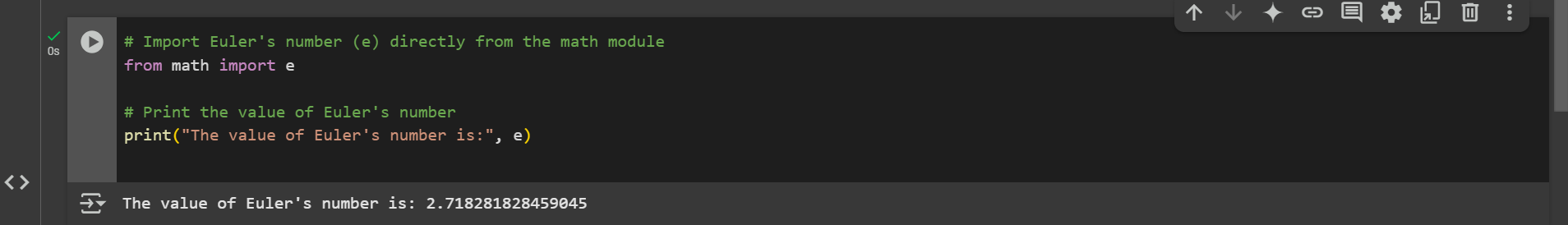
**Python import Statement**

****

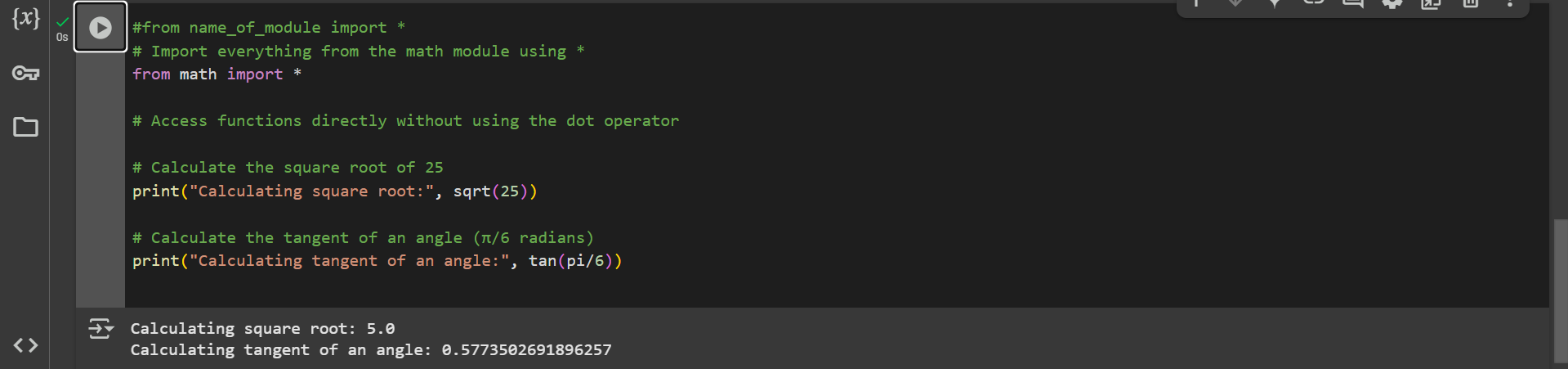
**Importing and also Renaming**

****

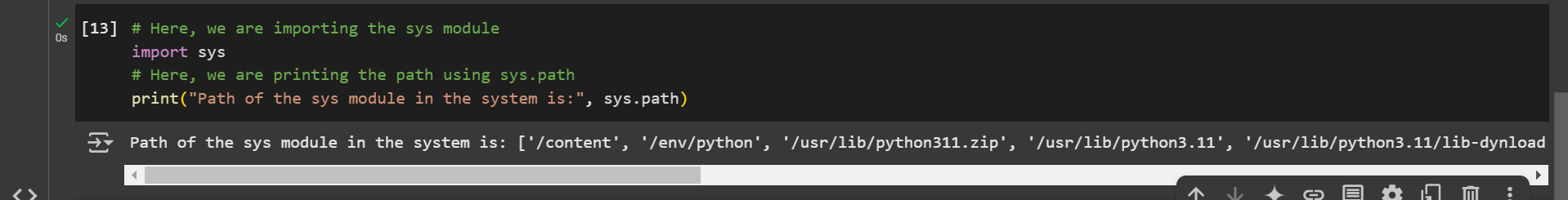
**Python from...import Statement**

****

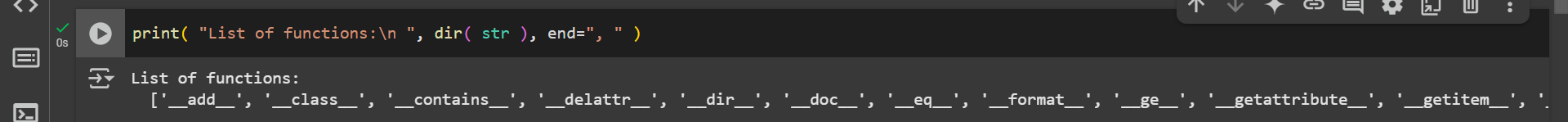
**Import all Names - From import \* Statement**

****

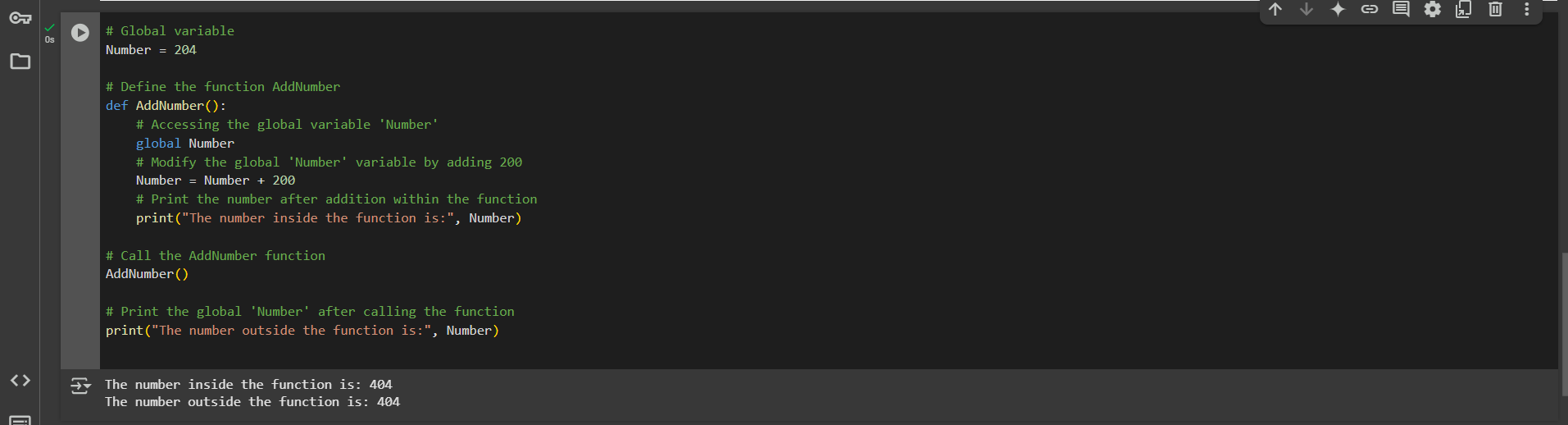
**Locating Path of Modules**

****

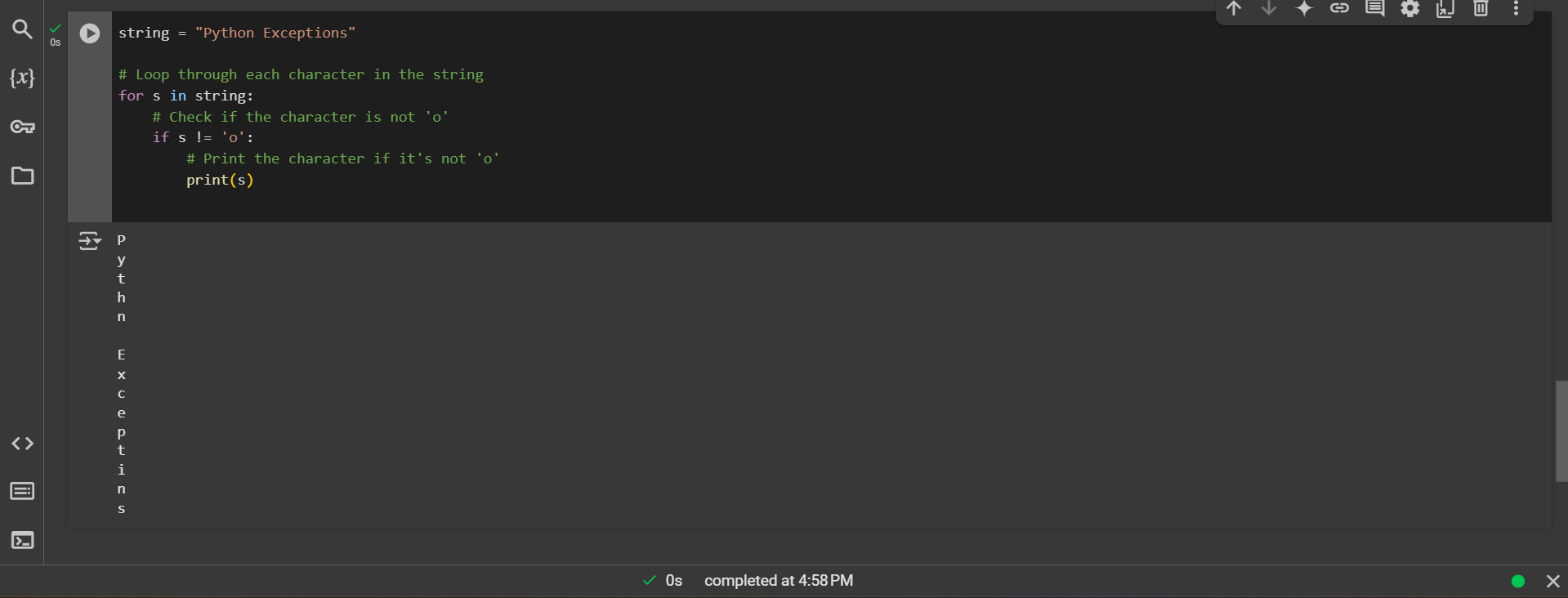
**The dir() Built-in Function**

****

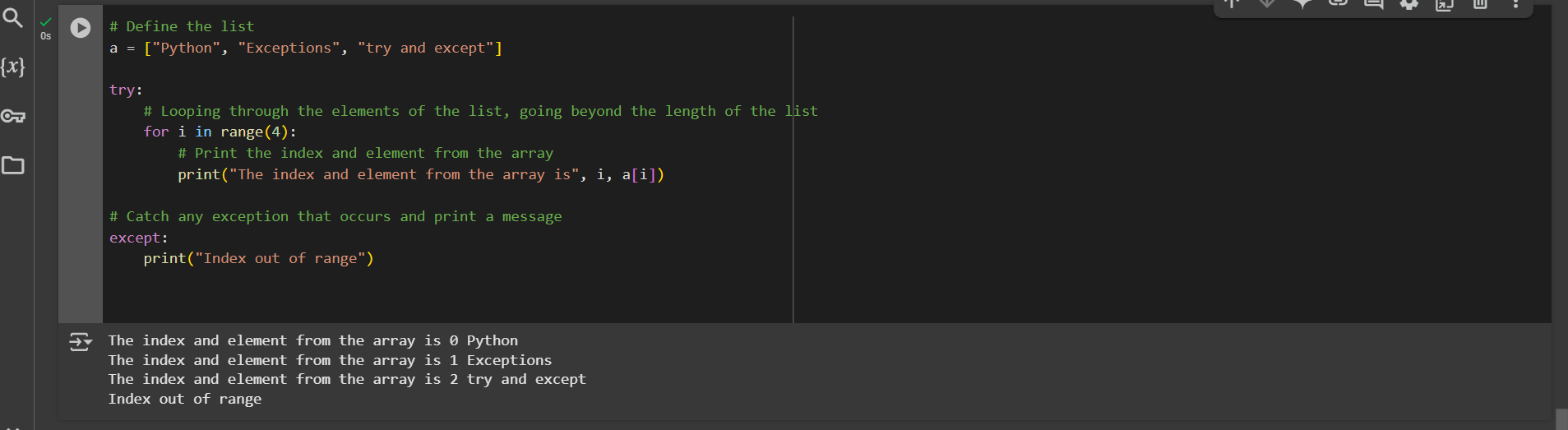
**Namespaces and Scoping**

****

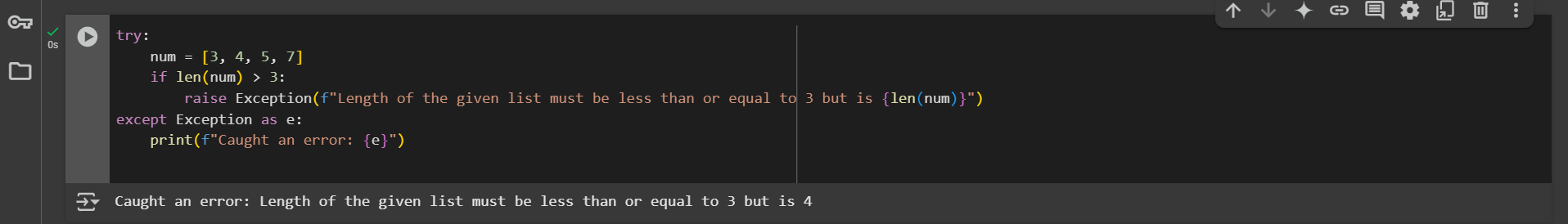
**Python Exceptions**

****

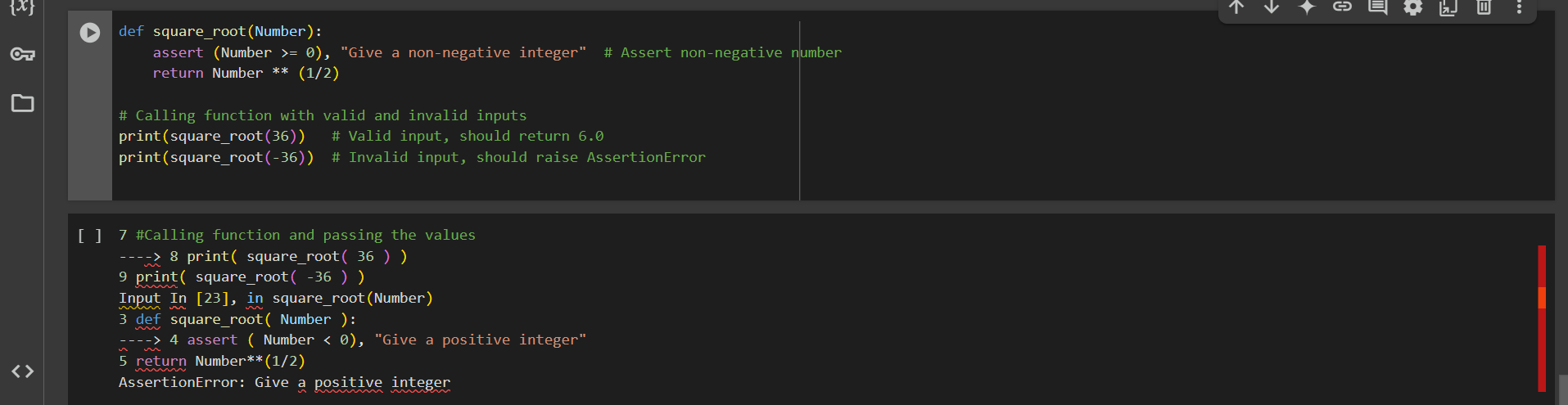
**Try and Except Statement - Catching Exceptions**

****

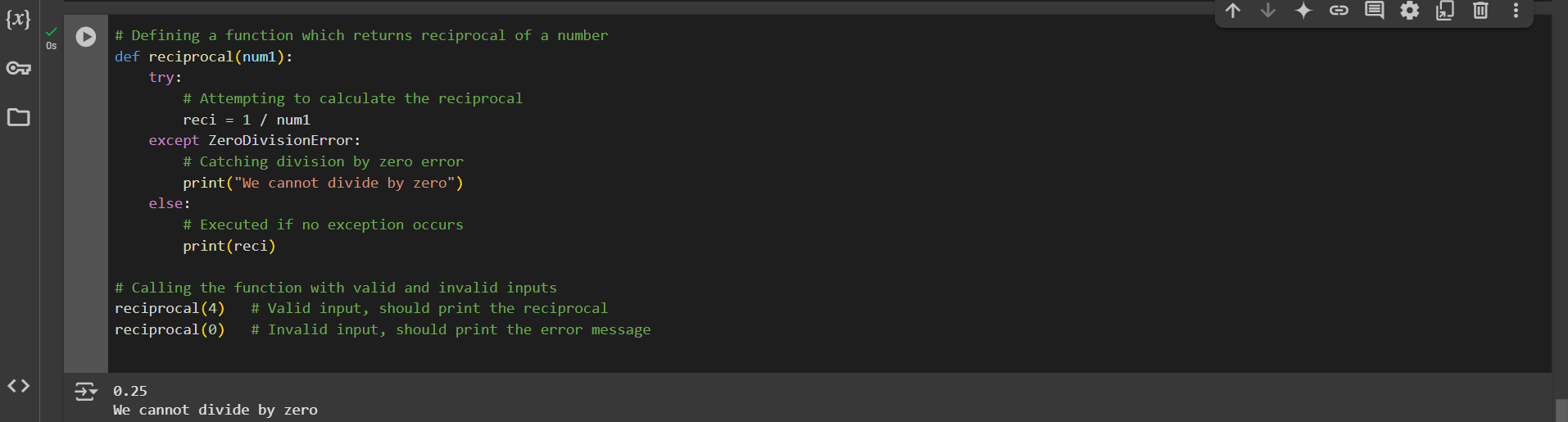
**How to Raise an Exception**

****

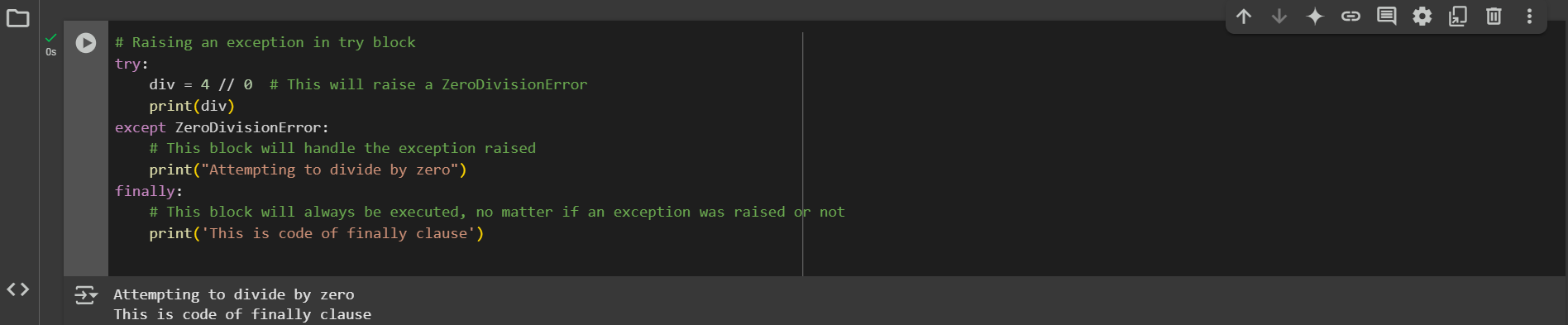
**Assertions in Python**

****

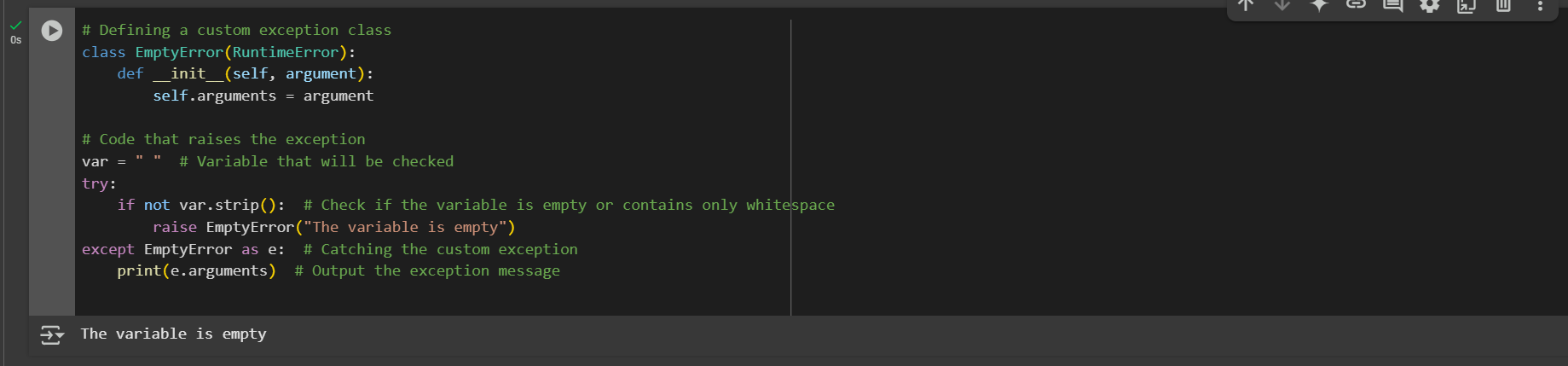
**Try with Else Clause**

****

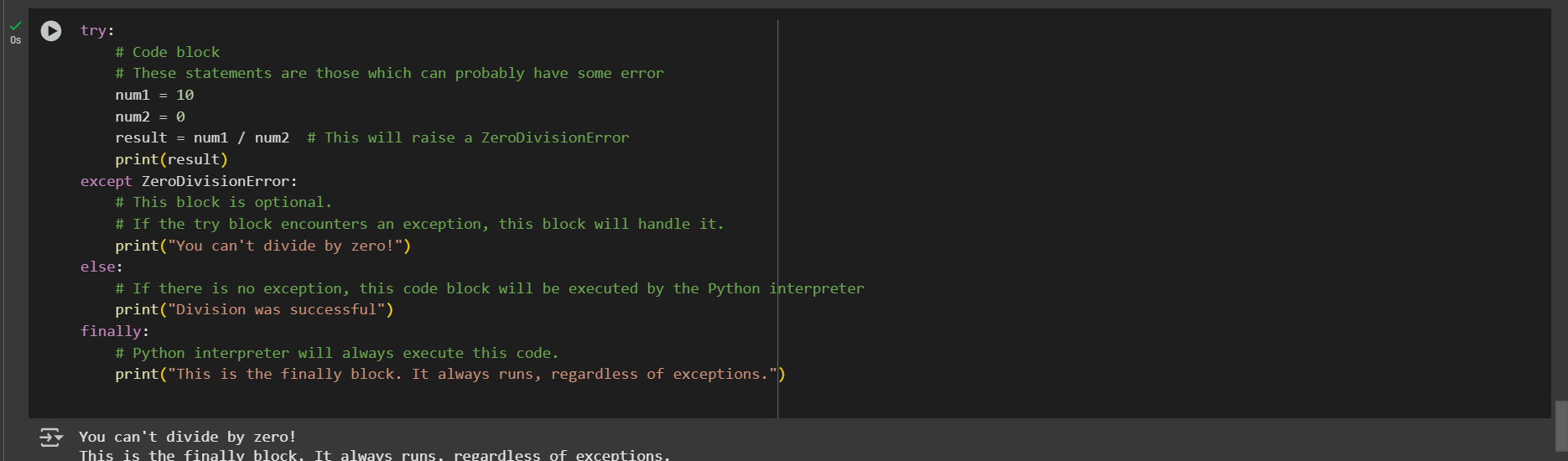
**Finally Keyword in Python**

****

**User-Defined Exceptions**

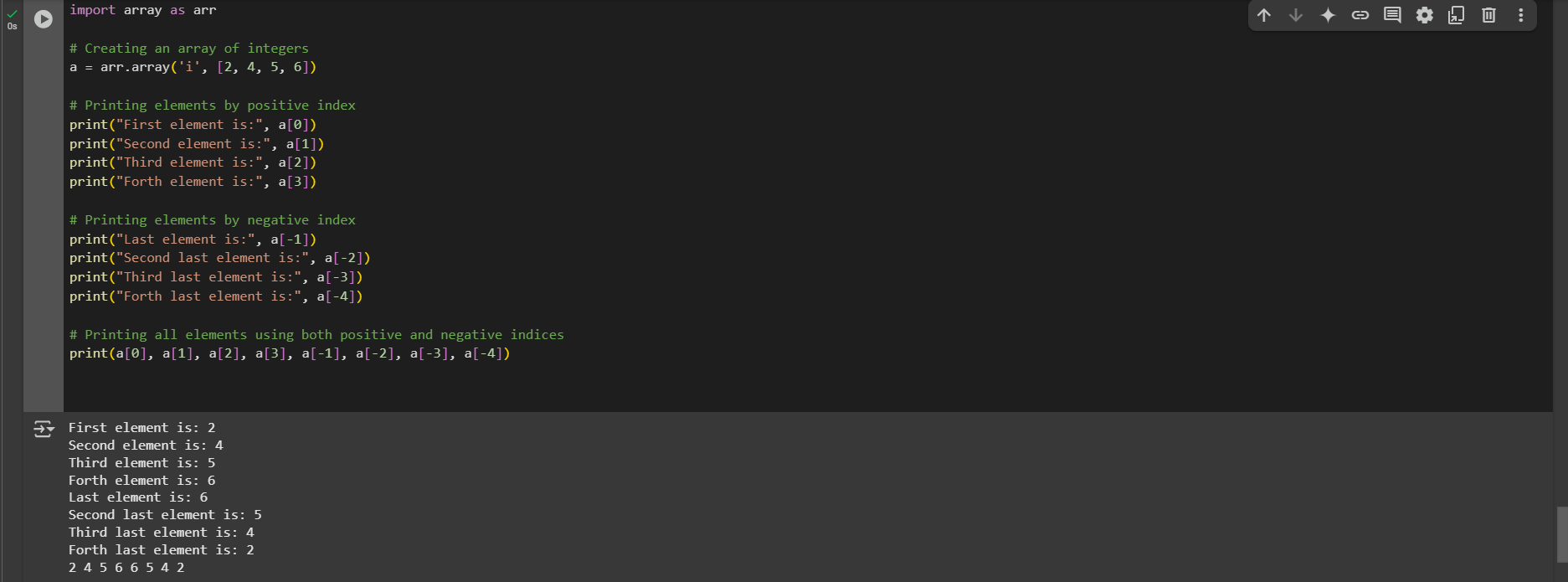
****

**try, except, else, and finally clauses**

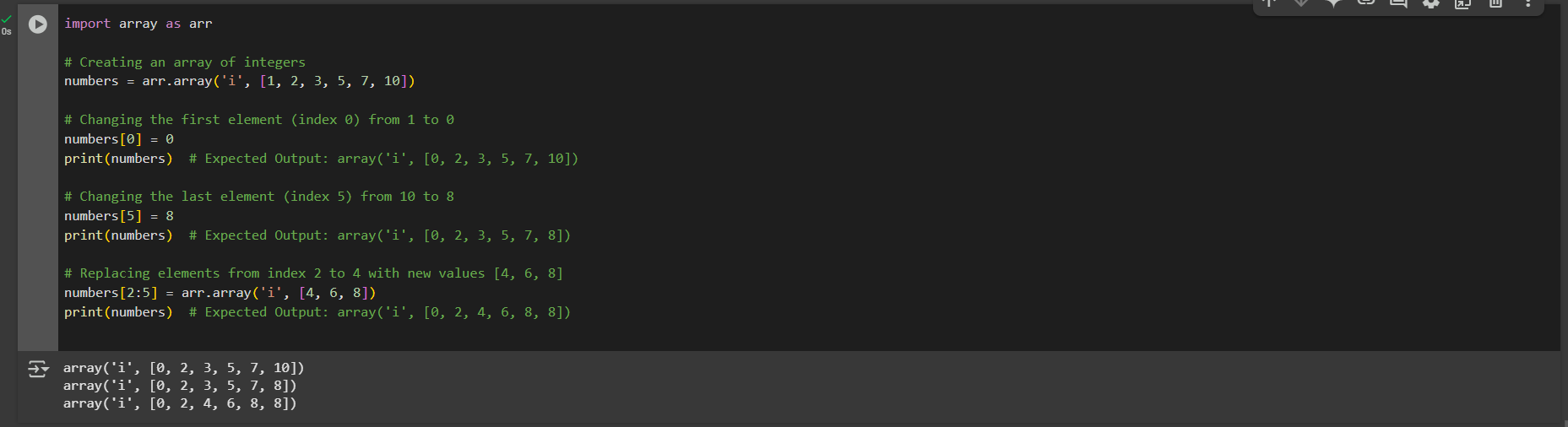
****

**Python Arrays**

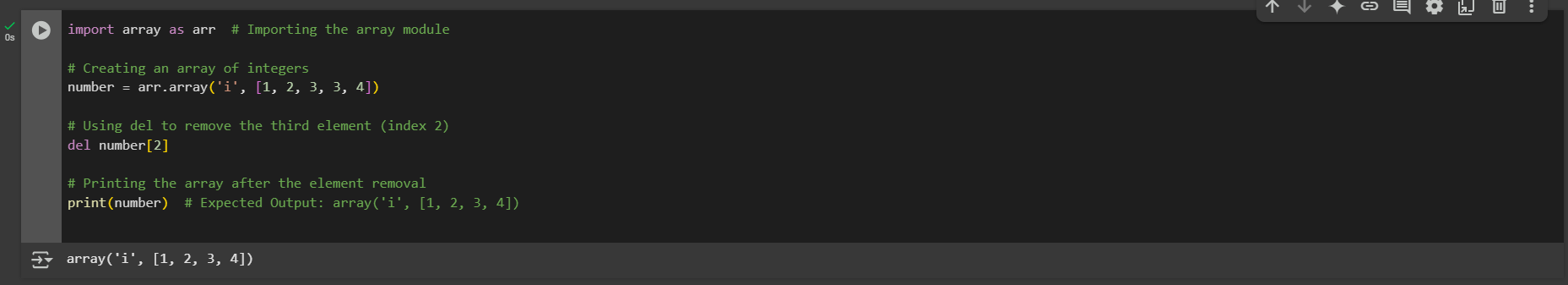
**Accessing array element**

****

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

****

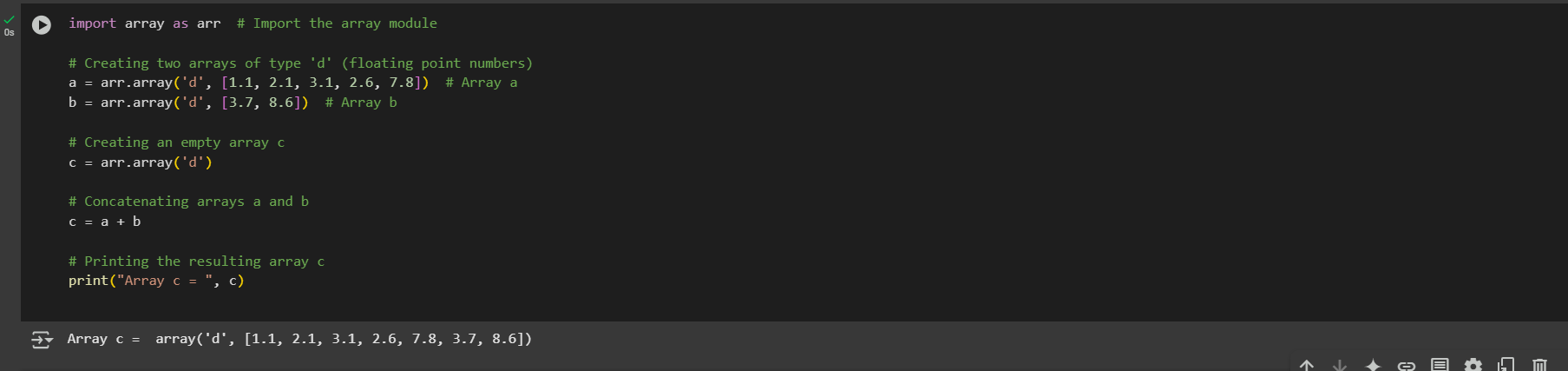
The elements can be deleted from an array using Python's del statement. If we want to delete any value from the Array, we can use the indices of a particular element.

****

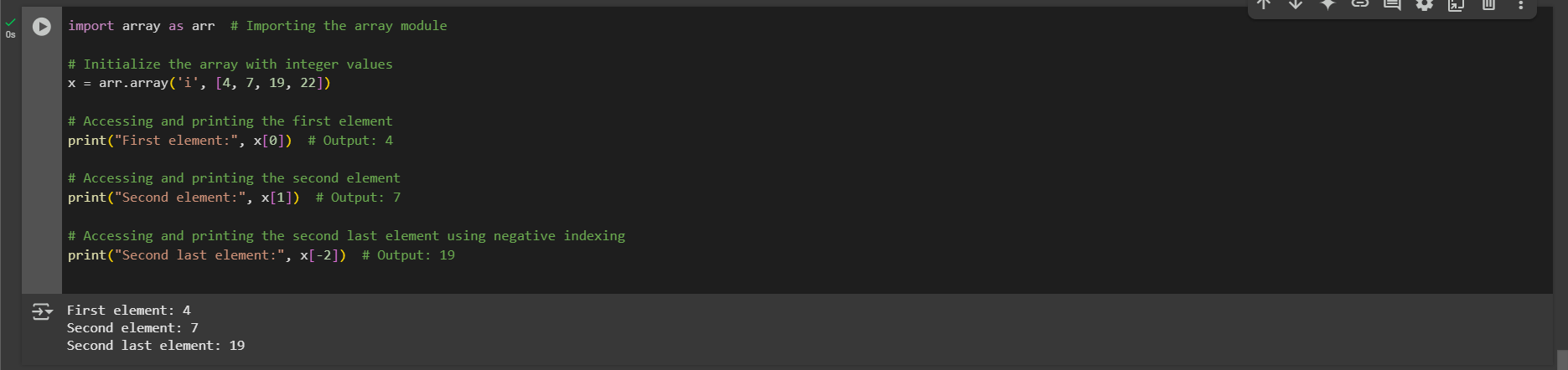
**Array Concatenation**

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

Example 1:

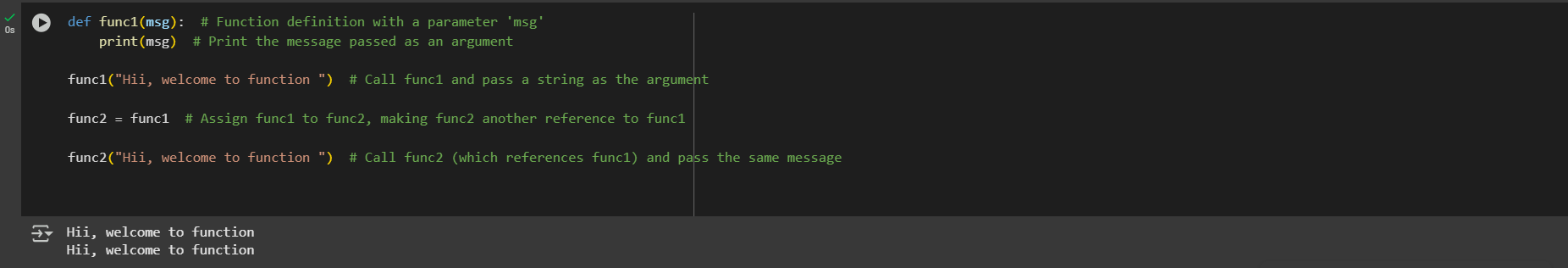


**Example 2:**

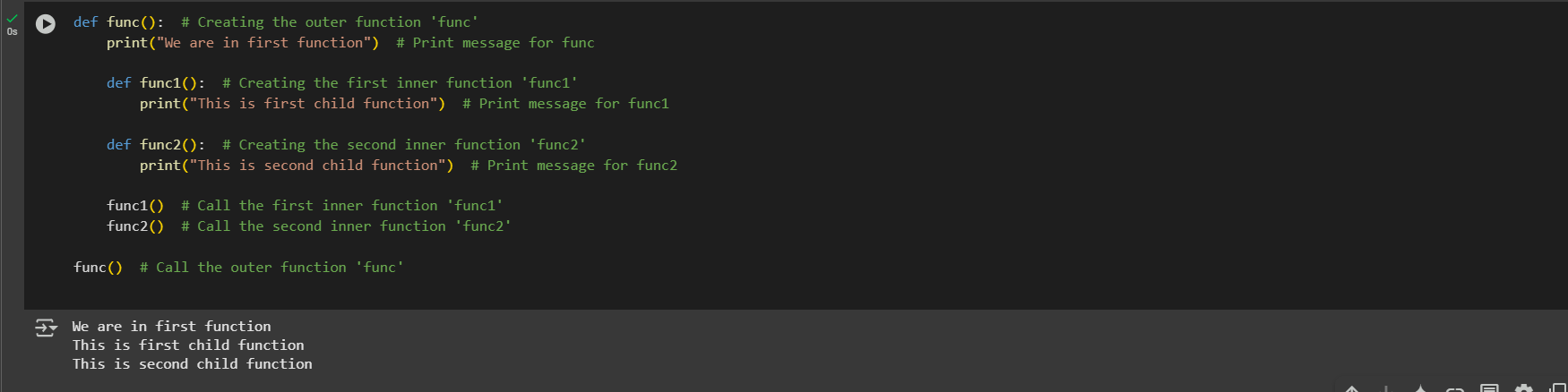
****

**Python Decorator**

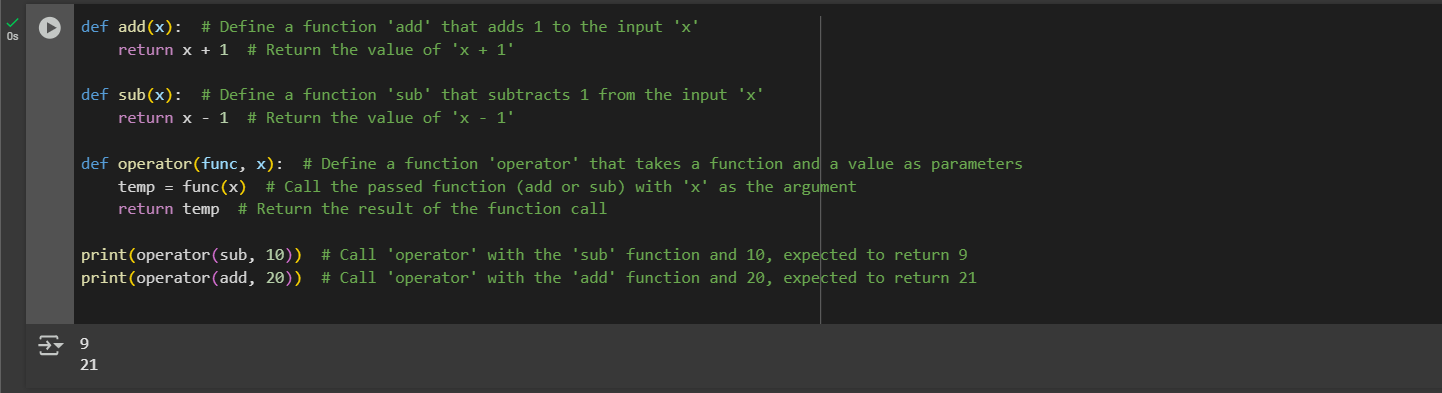
**Example**

****

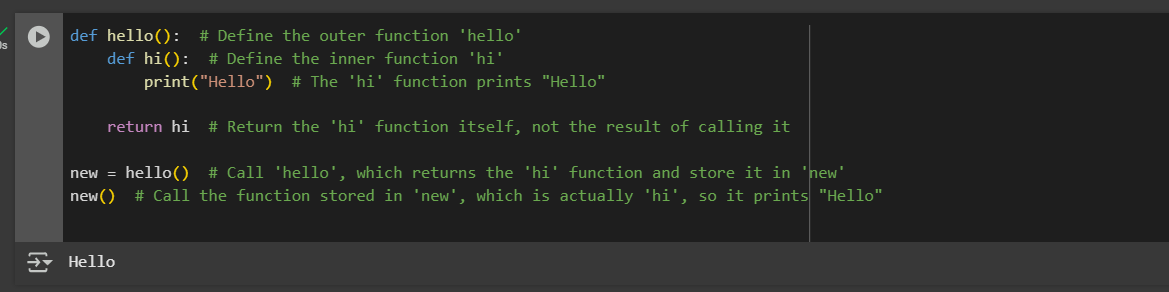
**Inner Function**

****

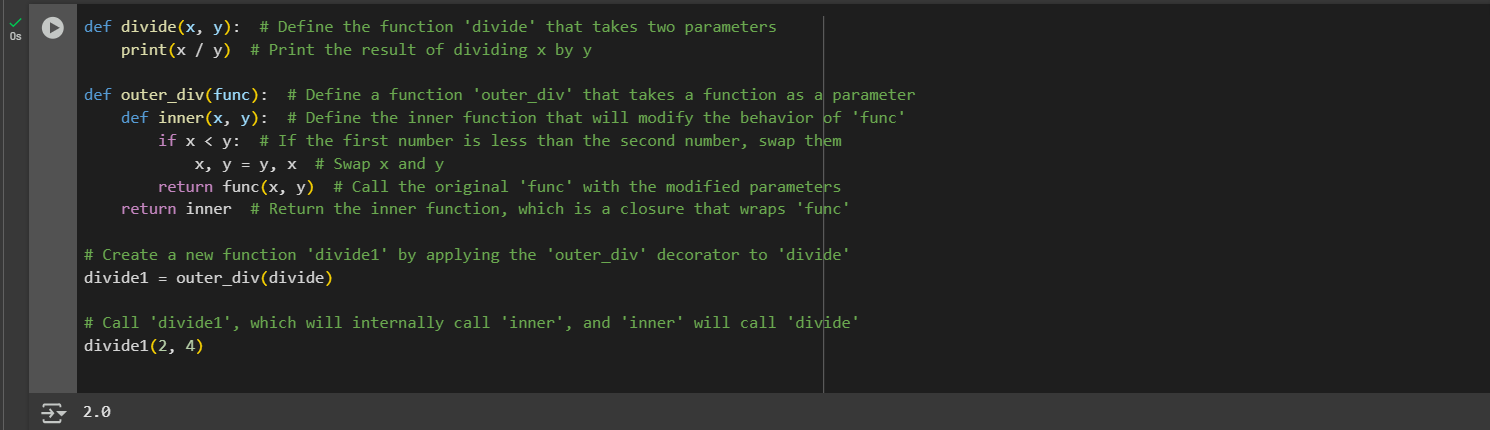
A function that accepts other function as an argument is also called higher order function. Consider the following example:

****

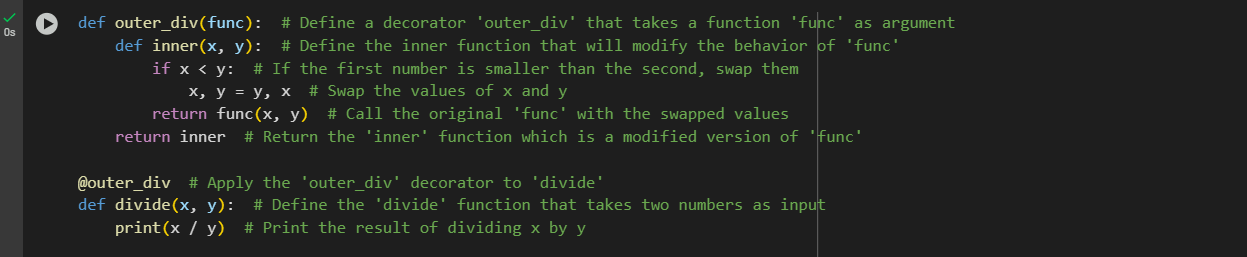
**A function can return another function. Consider the below example:**

****

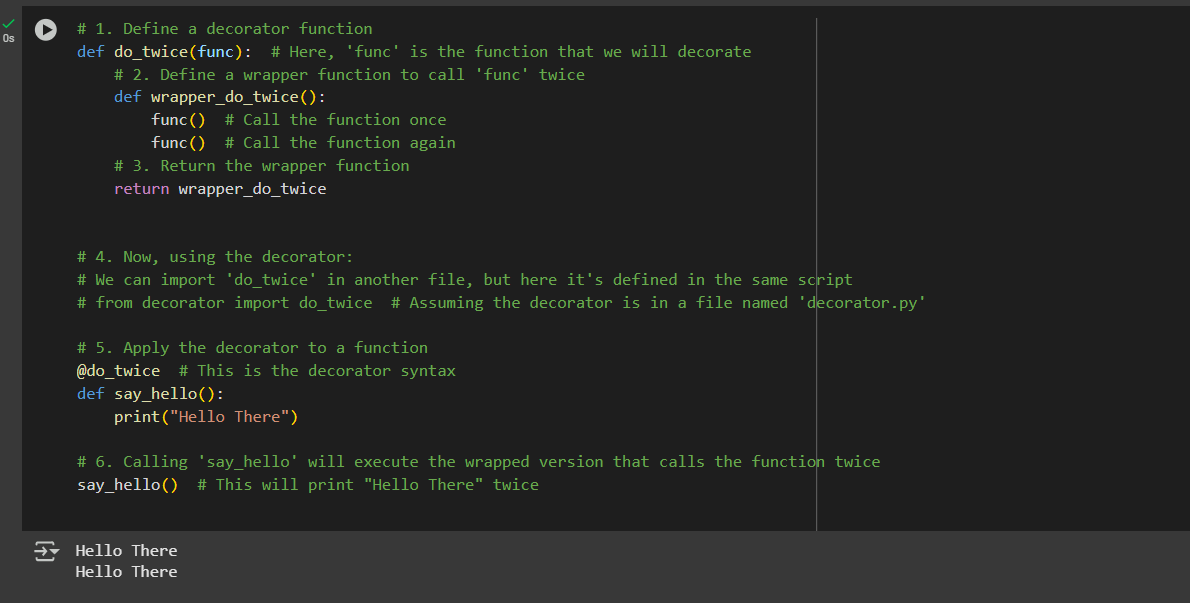
**Decorating functions with parameters**

****

**Syntactic Decorator**

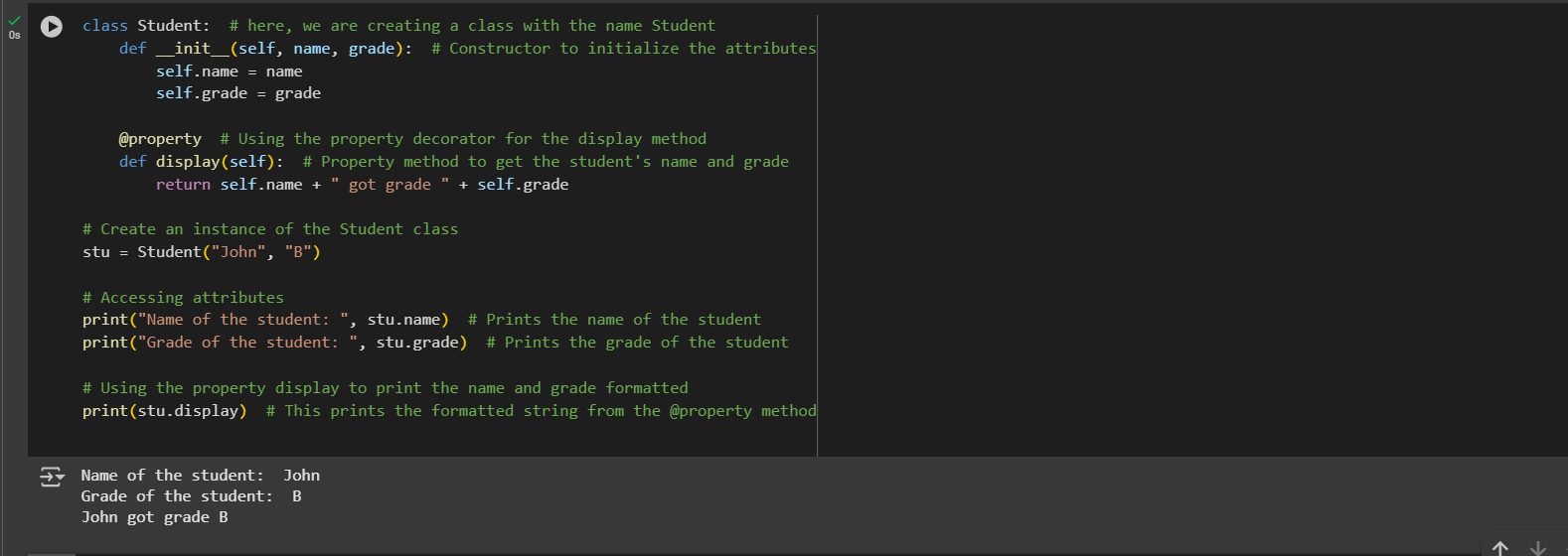
****

**Reusing Decorator**

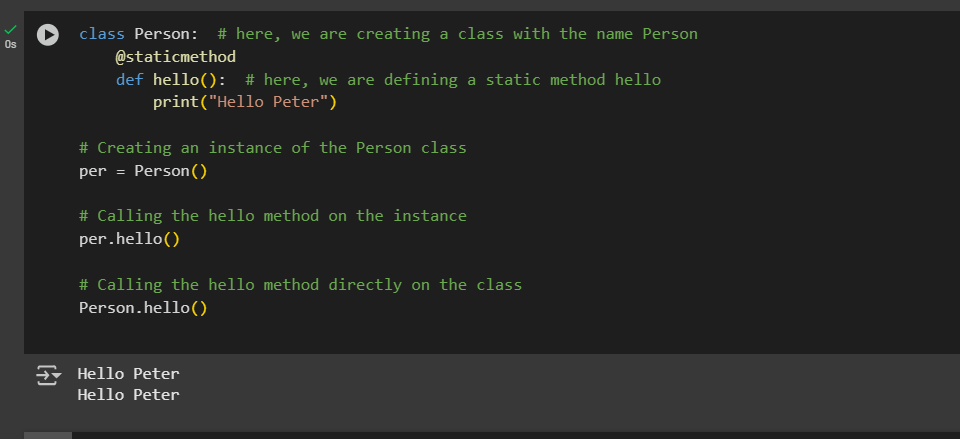
****

**Fancy Decorators**

**Example: 1-** @property decorator - By using it, we can use the class function as an attribute. Consider the following code:

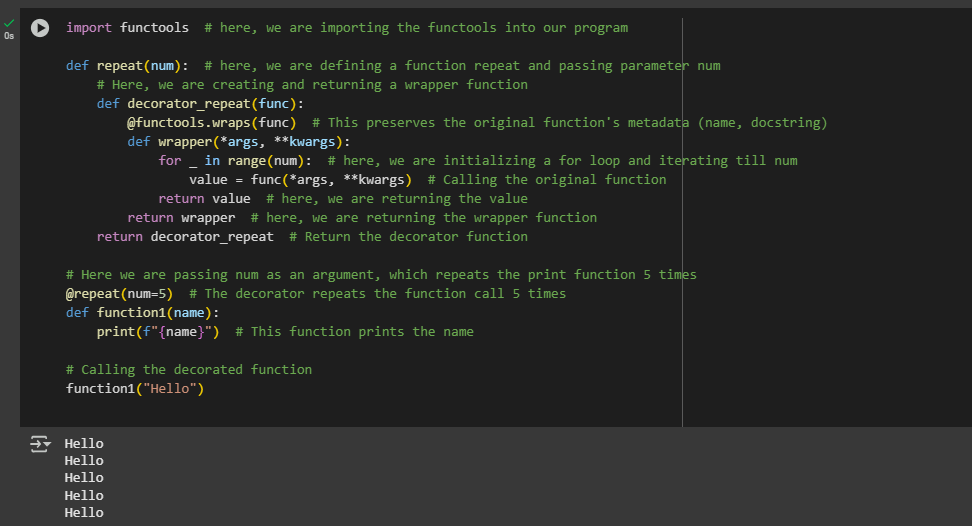
****

**Example: 2-**

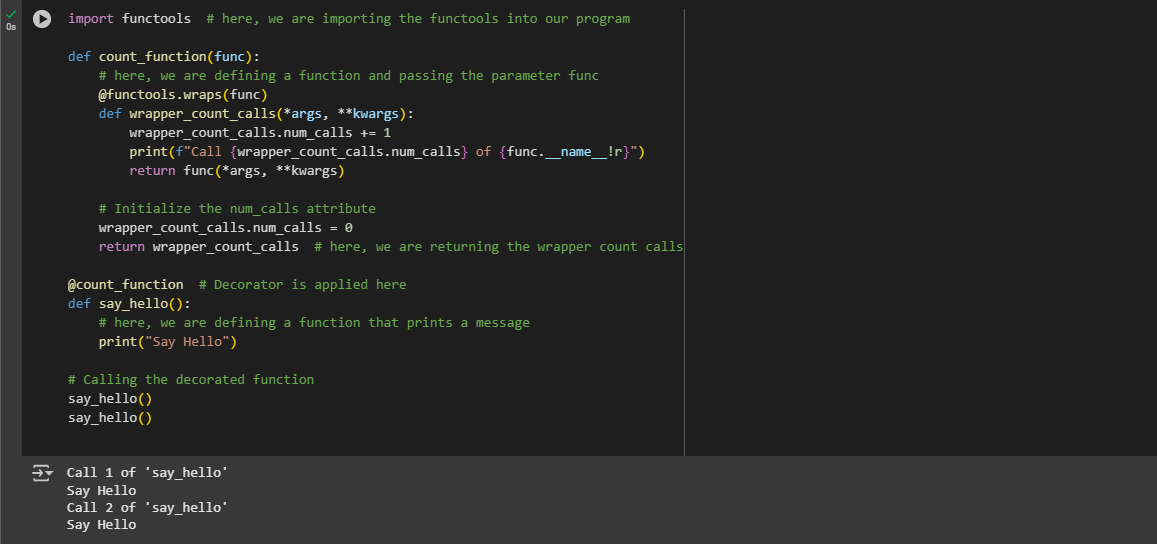
****

**Decorator with Arguments**

It is always useful to pass arguments in a decorator. The decorator can be executed several times according to the given value of the argument. Let us consider the following example:



**Stateful Decorators**



**Classes as Decorators**

