

statement	Namespace	Python's Memory
x=10		

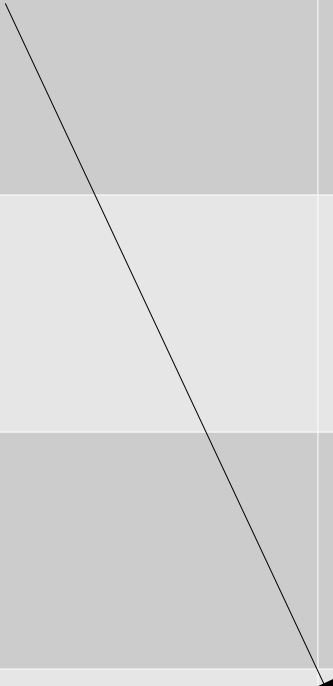
statement	Namespace	Python's Memory
x=10		
		10

statement	Namespace	Python's Memory
x=10	x	
		10

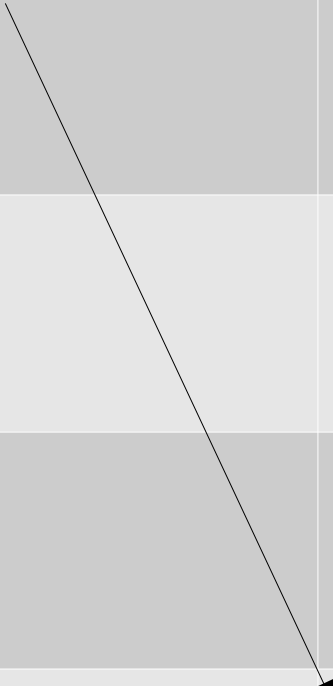
statement	Namespace	Python's Memory
x=10	x	
		10

The diagram illustrates the relationship between a Python statement, its namespace, and its memory representation. The table has three columns: 'statement', 'Namespace', and 'Python's Memory'. In the first row, the statement 'x=10' is shown. In the second row, the namespace 'x' is shown. In the fourth row, the value '10' is shown in the 'Python's Memory' column. A diagonal arrow points from the 'x' in the namespace column to the '10' in the memory column, indicating that the variable 'x' points to the memory address containing the value 10.

statement	Namespace	Python's Memory
x=10	x	
def foo(x): return 2*x		
		10



statement	Namespace	Python's Memory
x=10	x	
def foo(x): return 2*x		
		Function object
		10



statement	Namespace	Python's Memory
x=10	x	
def foo(x): return 2*x	foo	
		Function object
		10

```
graph LR; subgraph Table; direction TB; Row1["statement | Namespace | Python's Memory"]; Row2["x=10 | x | "]; Row3["def foo(x):  
    return 2*x | foo | "]; Row4[" | | Function object"]; Row5[" | | 10"]; Row6[" | | "]; end; Row2 -- "x" --> Row5; Row3 -- "foo" --> Row4;
```

statement	Namespace	Python's Memory
x=10	x	
def foo(x): return 2*x	foo	
foo(20)		Function object
		10

The diagram illustrates the execution of three Python statements and their corresponding namespace and memory representations. The first row shows the statement 'x=10' mapping to a namespace entry 'x' and a memory object '10'. The second row shows the function definition 'def foo(x): return 2\*x' mapping to a namespace entry 'foo' and a memory object 'Function object'. The third row shows the function call 'foo(20)' mapping to an empty namespace entry and a memory object 'Function object'. The fourth row shows an empty statement mapping to an empty namespace entry and a memory object '10'. Arrows indicate the mapping from the namespace entries to the memory objects: 'x' points to '10', 'foo' points to 'Function object', and 'Function object' points to '10'.



statement	Namespace	Python's Memory
x=10	x	
def foo(x): return 2*x	foo	20
foo(20)		Function object
		10

The diagram illustrates the execution of Python code across three rows of a table. The first row shows the assignment `x=10`, which creates a variable `x` in the namespace pointing to the value 10 in memory. The second row shows the definition of a function `foo(x)` that returns `2*x`, creating a `foo` namespace entry and a `Function object` in memory. The third row shows the call `foo(20)`, which evaluates to the value 20 in memory. A final row shows the value 10 in memory, which is the result of the initial assignment.

statement	Namespace	Python's Memory
x=10	x	
def foo(x): return 2*x	foo	20
foo(20)		Function object
		10

statement	Namespace	Python's Memory

statement	Namespace	Python's Memory
x=10	x	
def foo(x): return 2*x	foo	20
foo(20)		Function object
		10

statement	Namespace	Python's Memory
	x	

statement	Namespace	Python's Memory
x=10	x	
def foo(x): return 2*x	foo	20
foo(20)		Function object
		10

statement	Namespace	Python's Memory
	x	

statement	Namespace	Python's Memory
x=10	x	
def foo(x): return 2*x	foo	20
foo(20)		Function object
		10

statement	Namespace	Python's Memory
return 2*x	x	

statement	Namespace	Python's Memory
x=10	x	
def foo(x): return 2*x	foo	20
foo(20)		Function object
		10

statement	Namespace	Python's Memory
return 2*x	x	
		40

statement	Namespace	Python's Memory
x=10	x	
def foo(x): return 2*x	foo	20
foo(20)=> 40 result is returned		Function object
		10

The diagram illustrates the execution of a Python function. It shows the state of variables and objects in memory at different points in time. The table has three columns: 'statement', 'Namespace', and 'Python's Memory'. The rows represent different stages of execution. The first row shows the initial state where x is 10. The second row shows the function foo being defined, which takes x as an argument and returns 2\*x. The third row shows the function being called with 20 as an argument, resulting in a return value of 40. The fourth row shows the function object being created and stored in memory. The fifth row shows the final state where the return value 40 is stored in memory.

statement	Namespace	Python's Memory
return 2*x	x	
		40

This diagram shows the final state of the execution. The return value 40 is stored in the 'Python's Memory' column. The 'Namespace' column is empty, indicating that the function has finished executing and its local namespace has been discarded. The 'statement' column is also empty, indicating that the function has finished executing.

statement	Namespace	Python's Memory
x=10	x	
def foo(x): return 2*x	foo	20
foo(20)=> 40 result is returned		Function object
		10

