9 Bytecode

Introduction

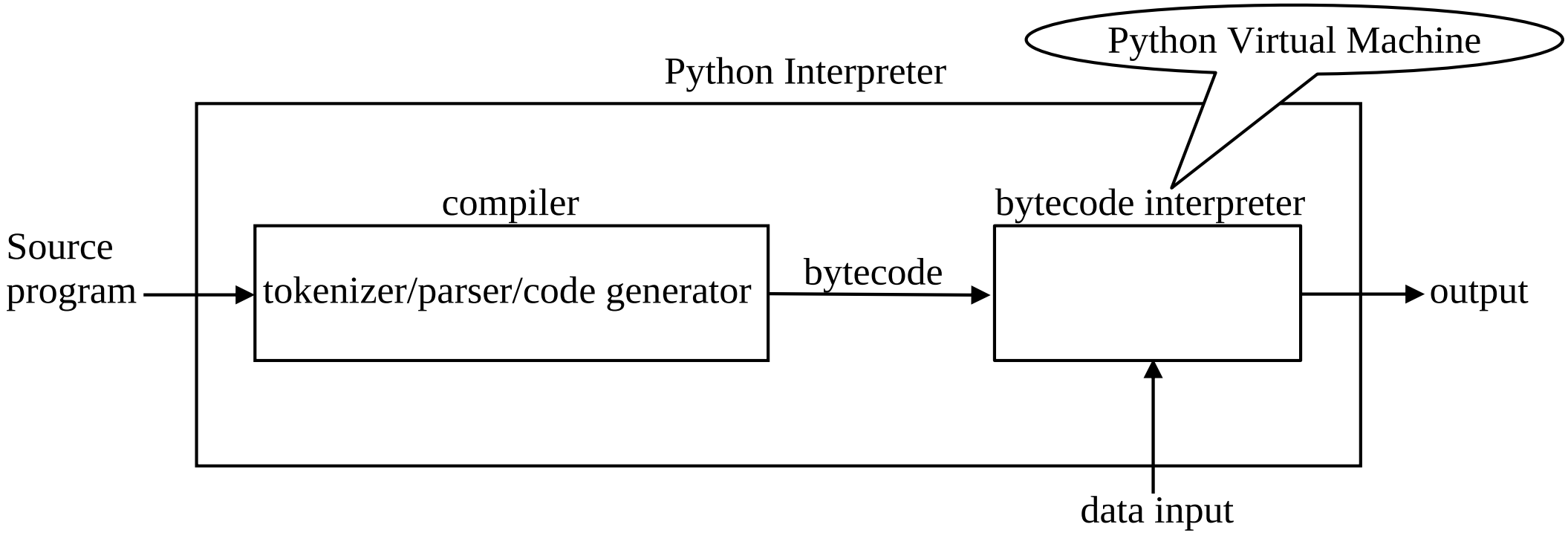


Figure 9.1

Structures Produced by the Parser in a Hybrid Interpreter

1. `co_code`: the table in which the code generator stores the bytecode it generates.
2. `co_names`: the table that holds the names of the global variables in the source code.
3. `co_consts`: the table that holds the values of the constants in the source program.

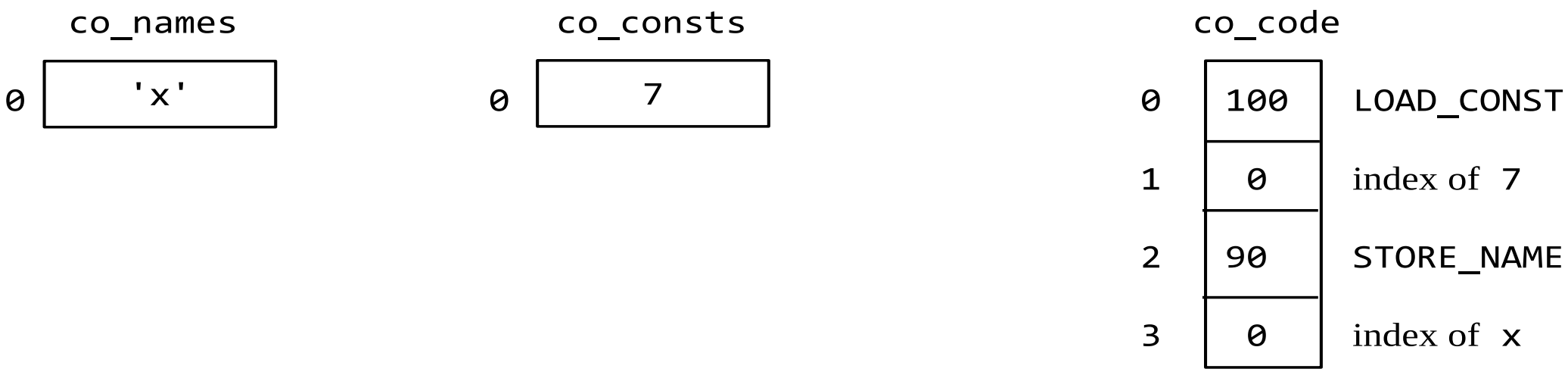
```
co_code = []
co_names = []
co_consts = []
```

Code Generation for a Small Program

sample2.py

```
x = 7
y = x
```

Figure 9.2



co_names

0	'x'
1	'y'

co_consts

0	7

co_code

0	100	LOAD_CONST
1	0	index of 7
2	90	STORE_NAME
3	0	index of x
4	101	LOAD_NAME
5	0	index of x
6	90	STORE_NAME
7	1	index of y

Bytecode Interpretation of a Small Program

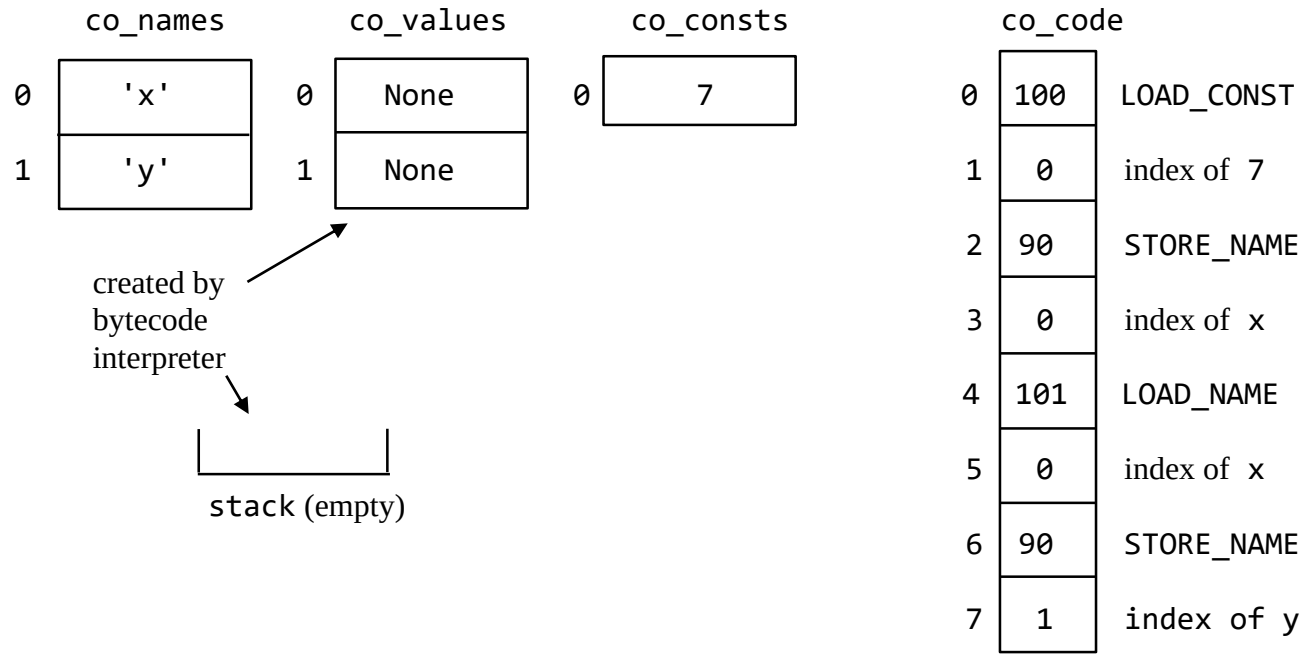
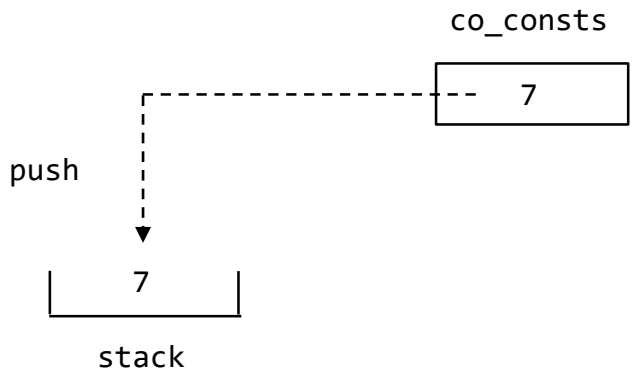
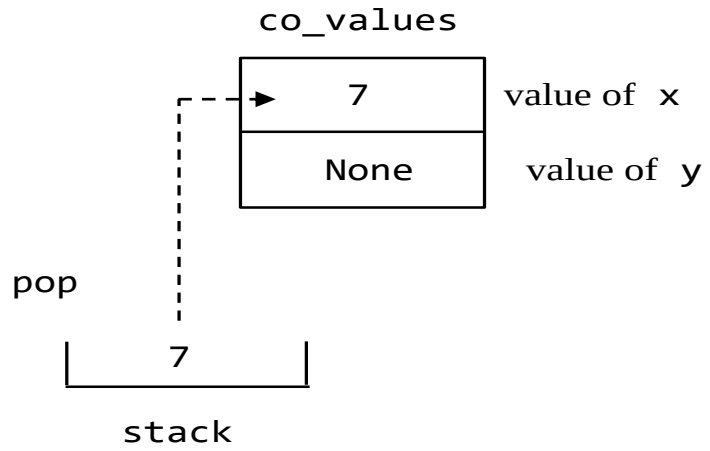
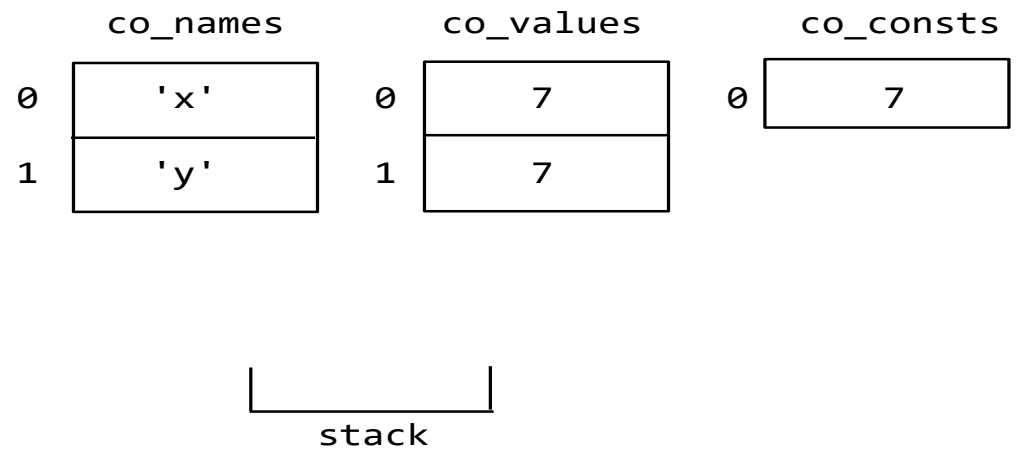


Figure 9.3





After both assignment statements executed:



co_code

0	100	LOAD_CONST
1	0	index of 7
2	90	STORE_NAME
3	0	index of x
4	101	LOAD_NAME
5	0	index of x
6	90	STORE_NAME
7	1	index of y

Displaying Bytecode

```
co_names  =  ['x', 'y']
co_consts =  [7]
co_code   =  [100, 0, 90, 0, 101, 0, 90, 1]
```

Better:

Statement number	co_code index	Opcode name	Index	Item referenced
1	0	LOAD_CONST	0	(7)
	2	STORE_NAME	0	(x)
2	4	LOAD_NAME	0	(x)
	6	STORE_NAME	1	(y)

Figure 9.4

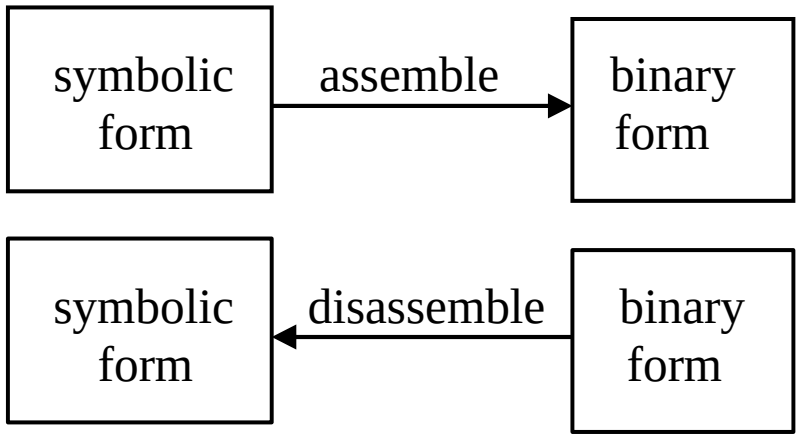


Figure 9.5

```
python -m dis sample2.py
```

Opcodes for Our First Hybrid Interpreter

Opcode	Index	Name	Effect
11		UNARY_NEGATIVE	Negates TOS
20		BINARY_MULTIPLY	Pops TOS, TOS1; pushes TOS1 * TOS
23		BINARY_ADD	Pops TOS, TOS1; pushes TOS1 + TOS
71		PRINT_ITEM	Pops and displays TOS
72		PRINT_NEWLINE	Outputs newline character
90	namei	STORE_NAME	Pops TOS into co_values[namei]
100	consti	LOAD_CONST	Pushes co_consts[consti]
101	namei	LOAD_NAME	Pushes co_values[namei]

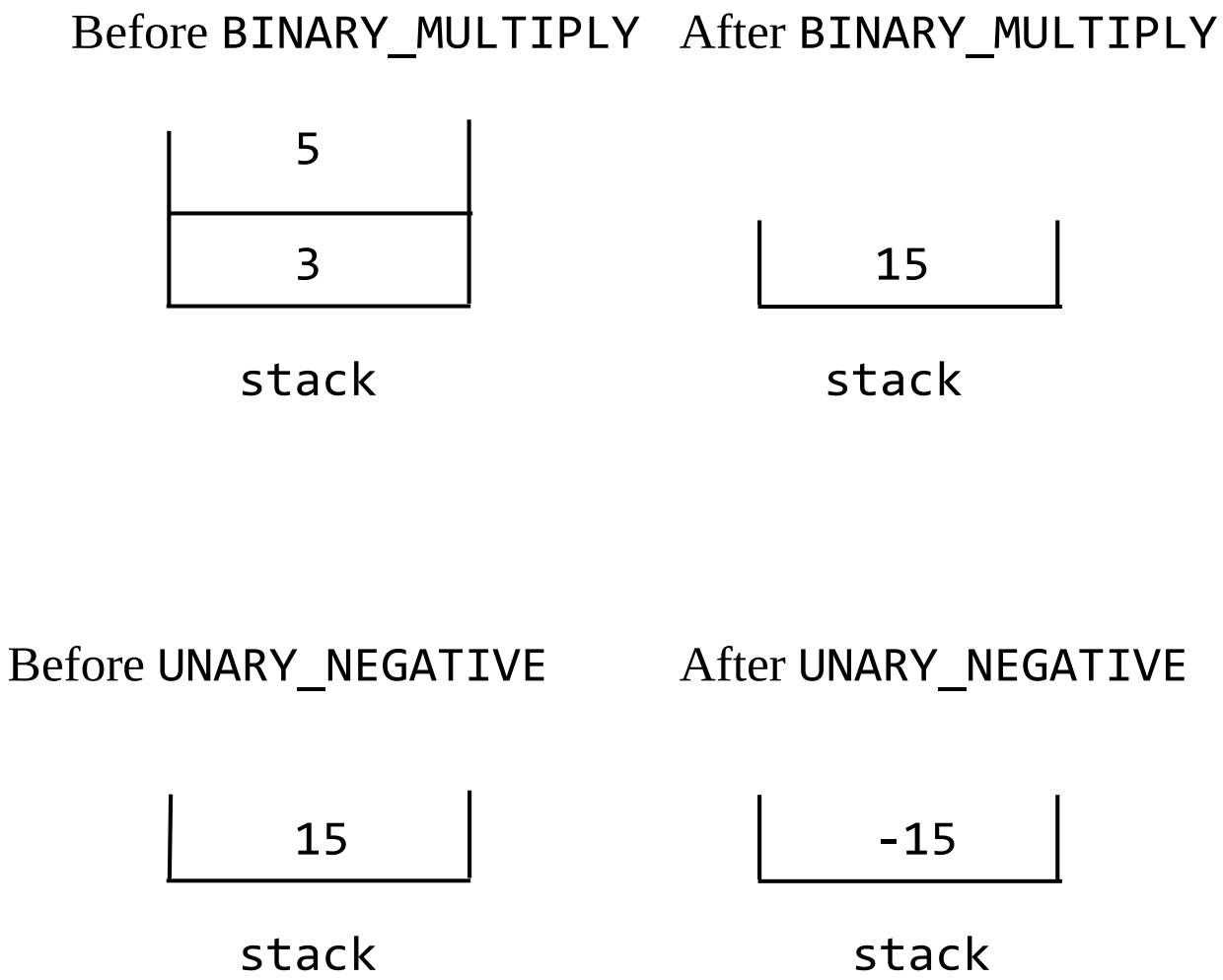

Before BINARY_MULTIPLY
5
3
stack
After BINARY_MULTIPLY
15
stack
Before UNARY_NEGATIVE
15
stack
After UNARY_NEGATIVE
-15
stack

Figure 9.6