Lab 08:
Inline Assembly

Nick Zoghb

BLOCKCHAIN
AT BERKELEY

LAB OUTLINE

1. STACK REFRESHER
2. SOLIDITY ASSEMBLY REFRESHER
3. ASSIGNMENT

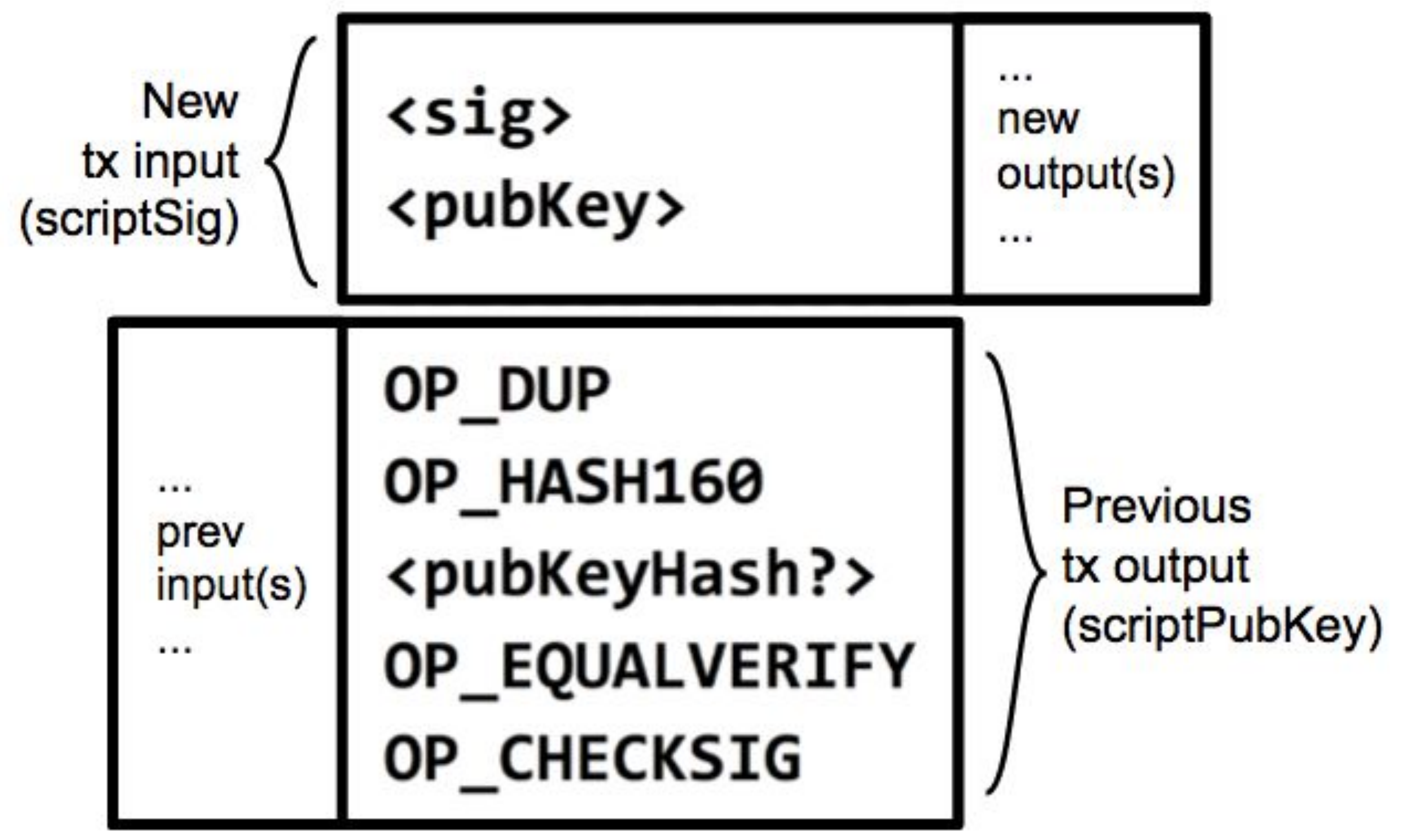
1

STACK REFRESHER

BLOCKCHAIN AT BERKELEY

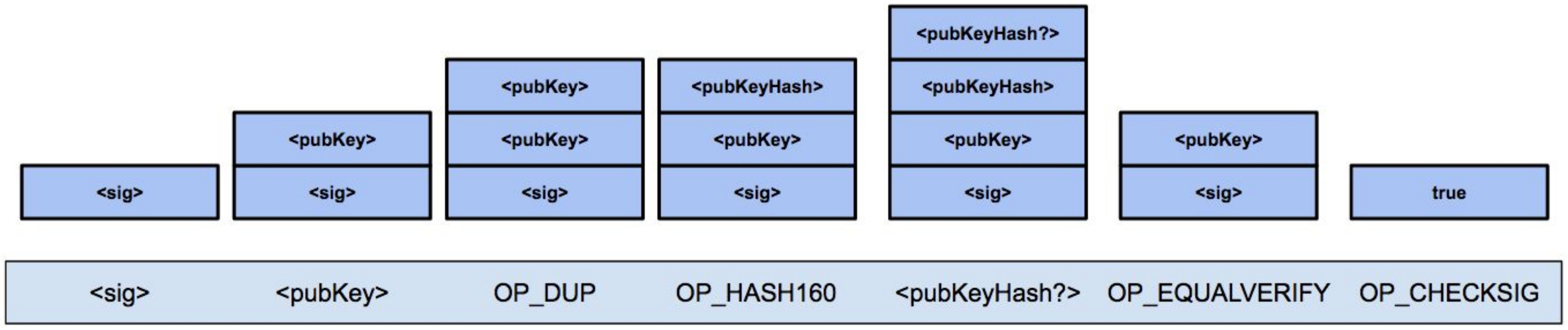
BITCOIN SCRIPT

A REFRESHER



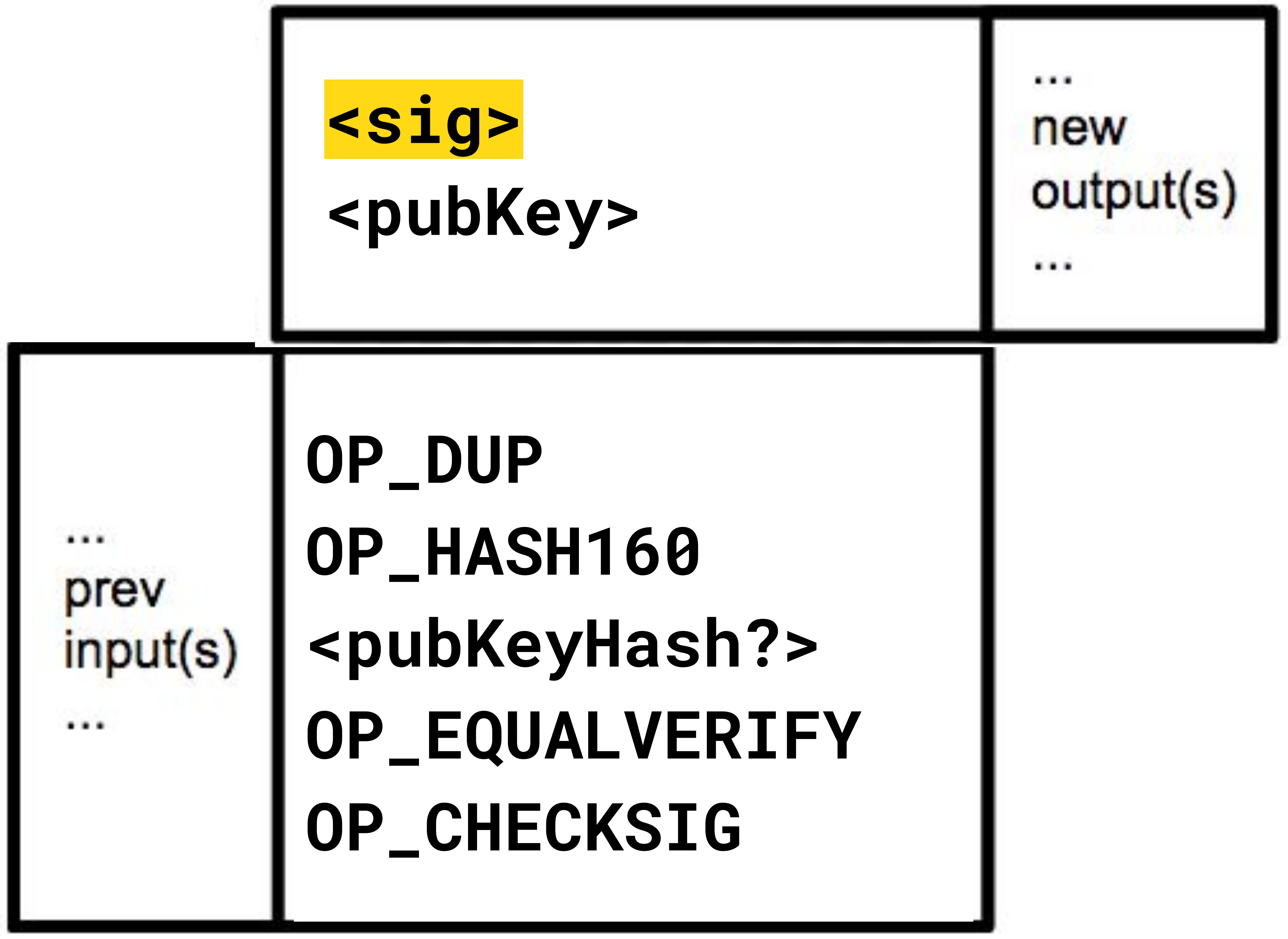
- Output says: "This amount can be redeemed by
- 1) the **`<pubKey>`** that hashes to address **`<pubKeyHash?>`**
- 2) plus a **`<sig>`** from the owner of that **`<pubKey>`**
- ...that will make this script evaluate to **`true`**."

Read the Princeton's Bitcoin and Cryptocurrency Technologies for more information.

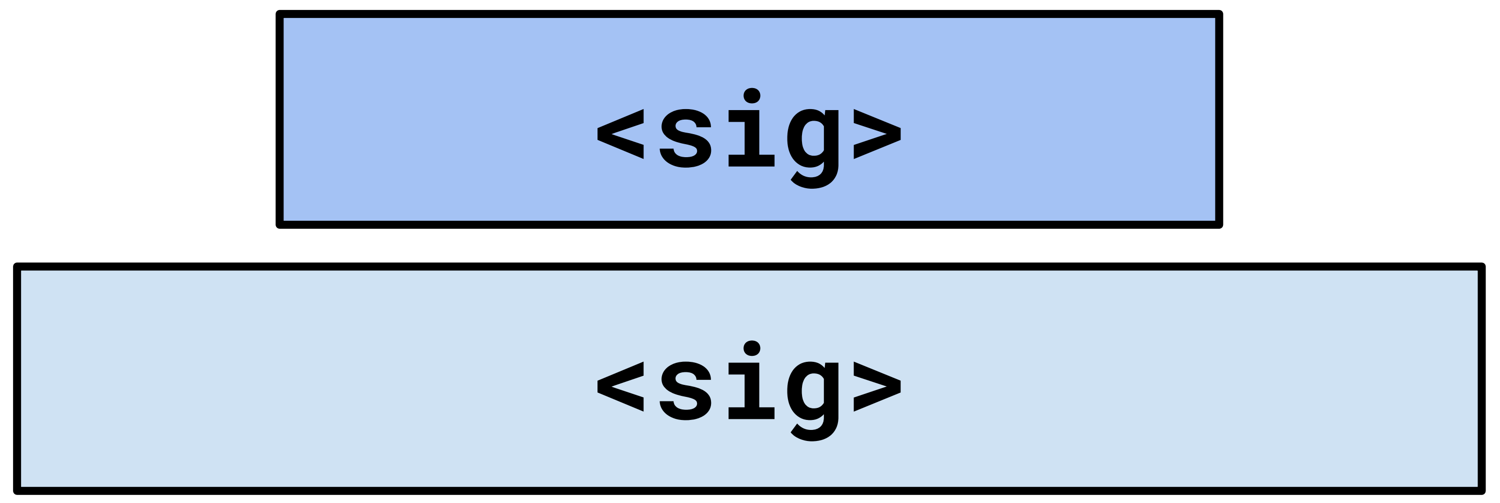


BITCOIN SCRIPT

P2PKH EXAMPLE EXECUTION

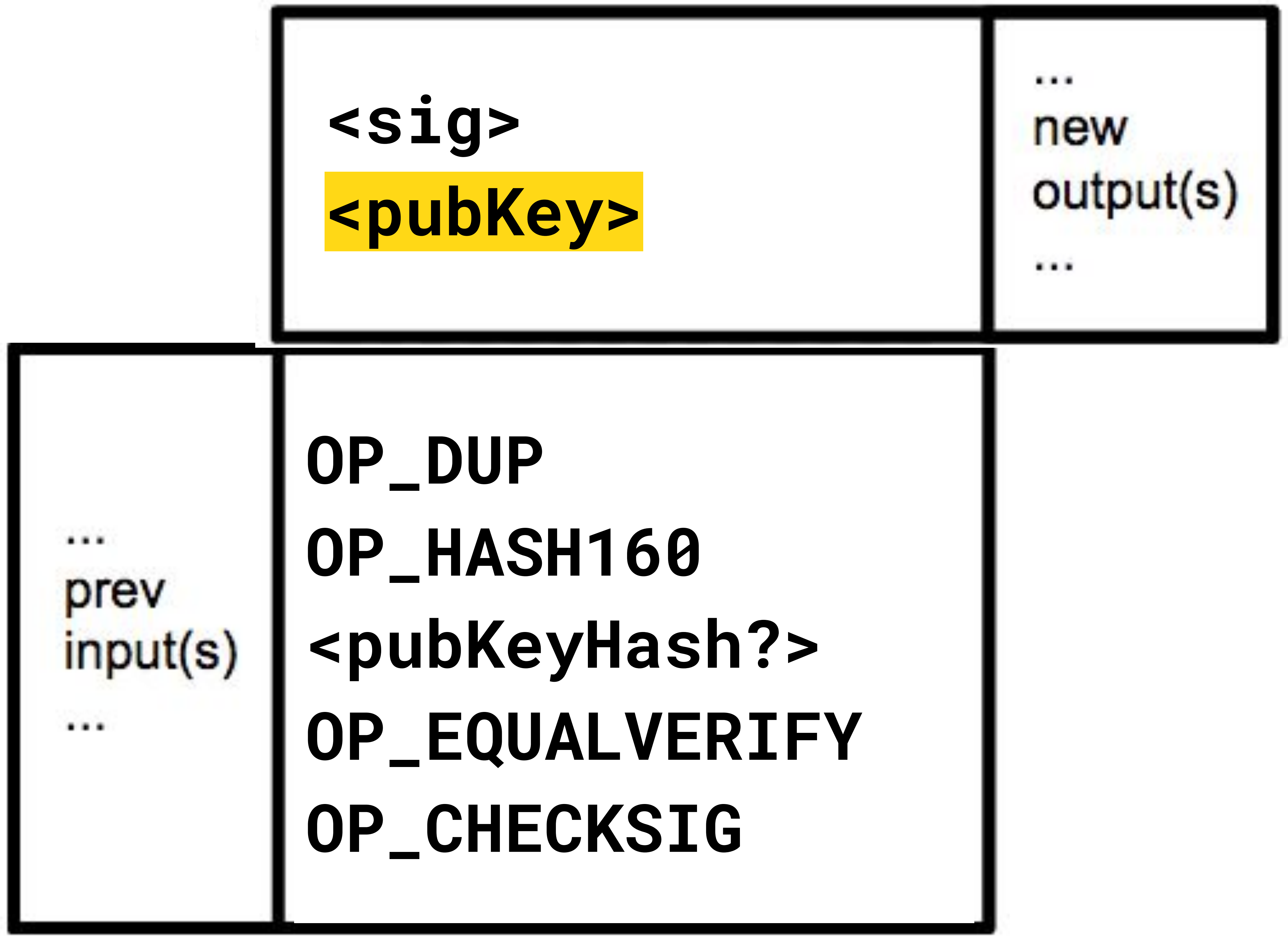


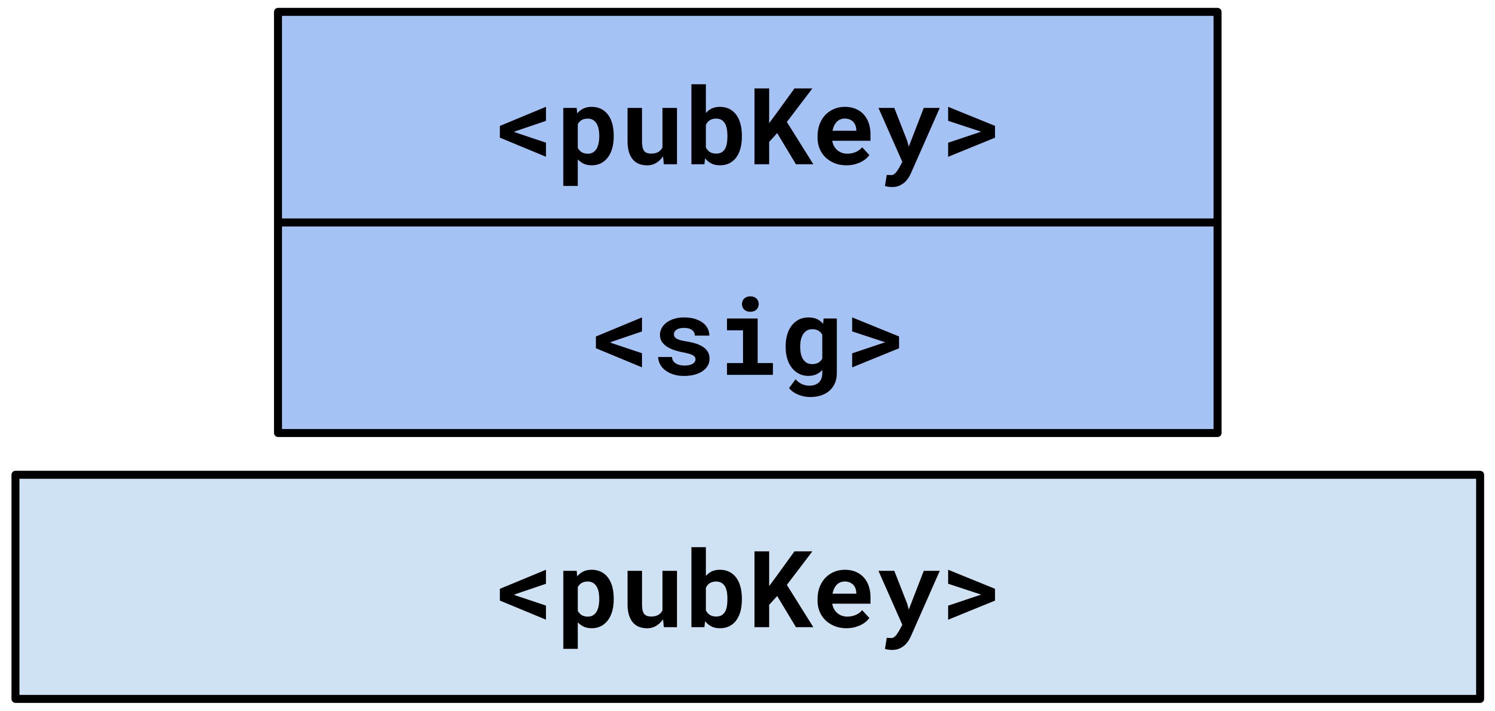
```
OP_DUP
OP_HASH160
<pubKeyHash?>
OP_EQUALVERIFY
OP_CHECKSIG
```



BITCOIN SCRIPT

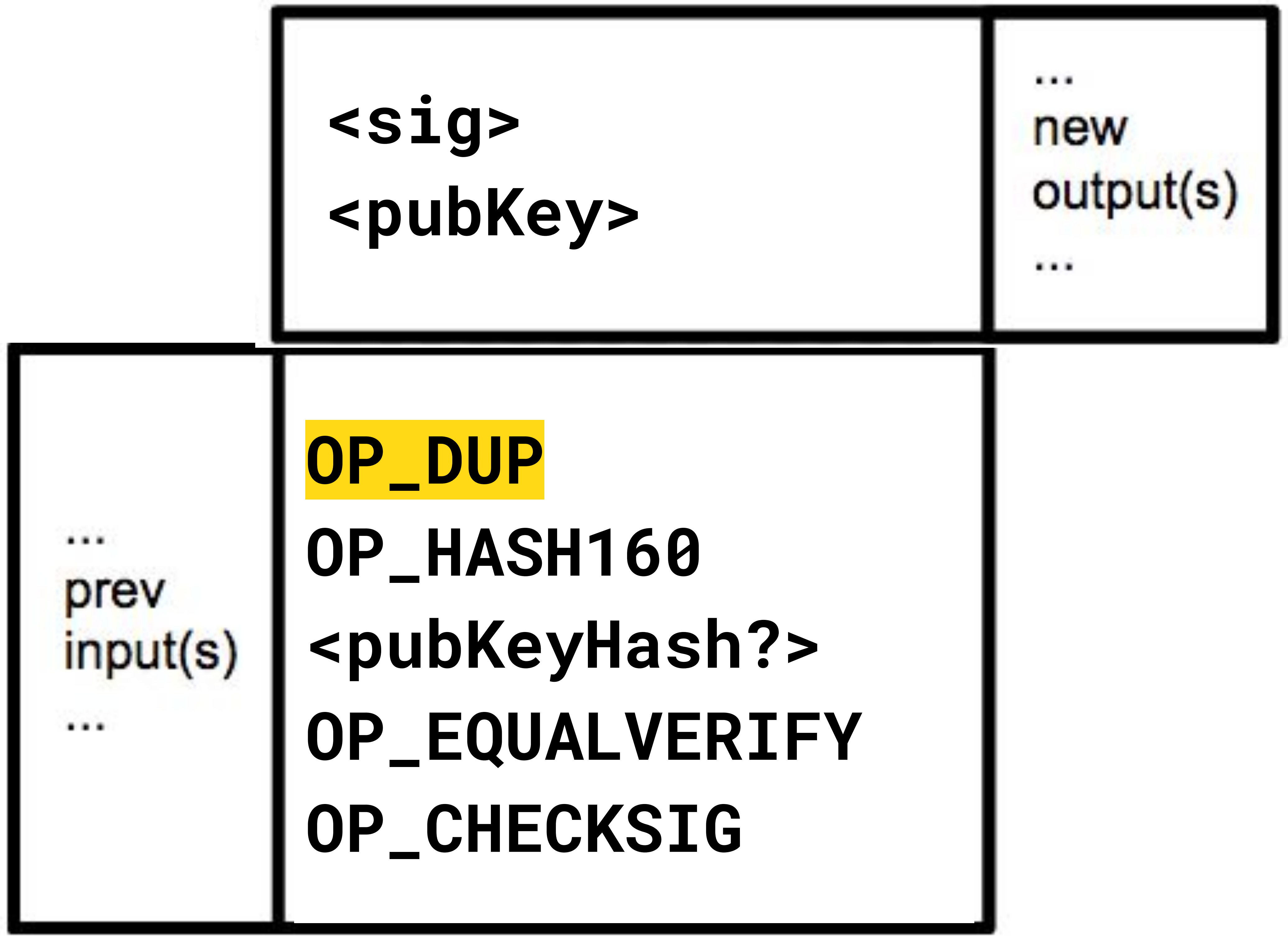
P2PKH EXAMPLE EXECUTION

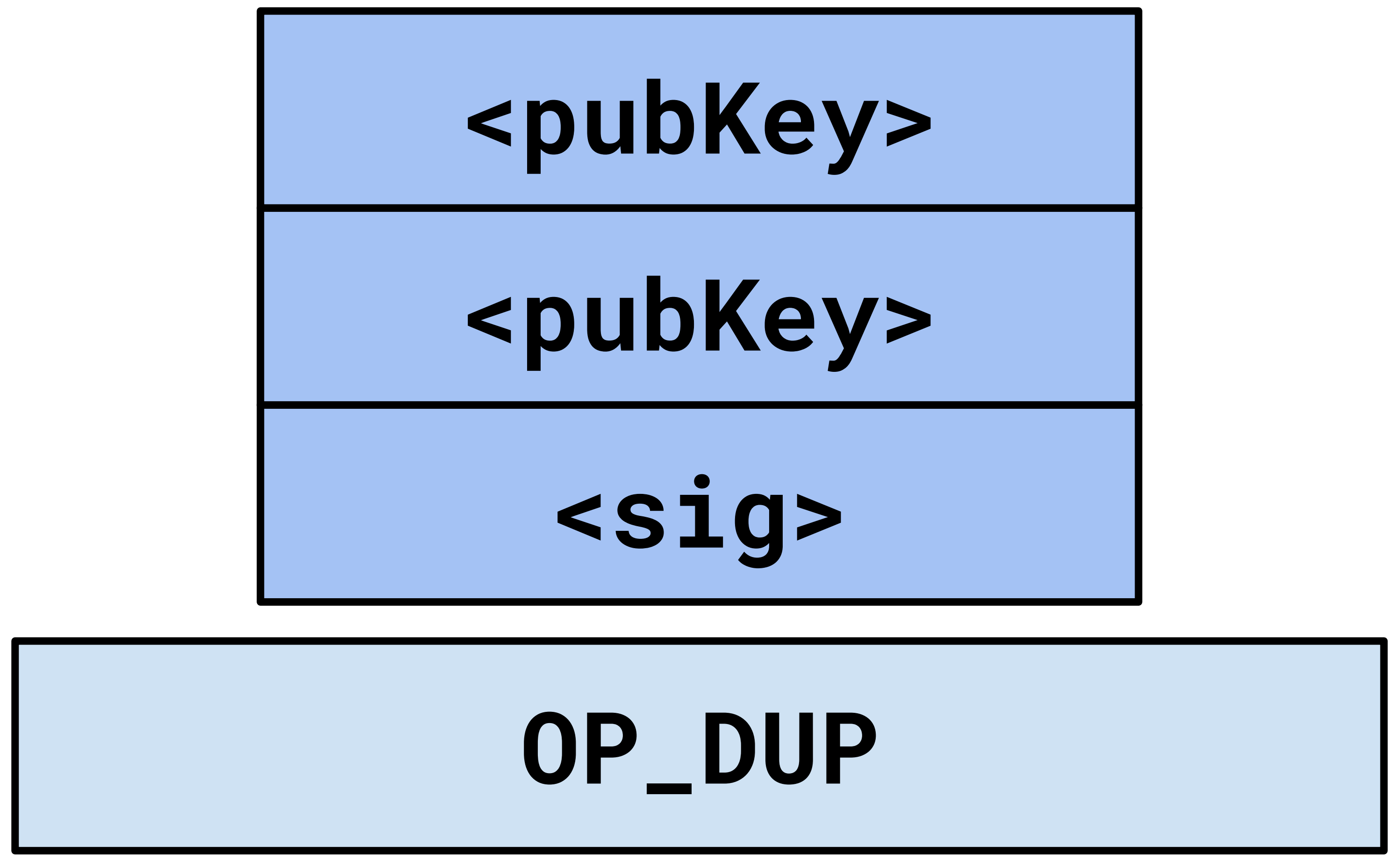




BITCOIN SCRIPT

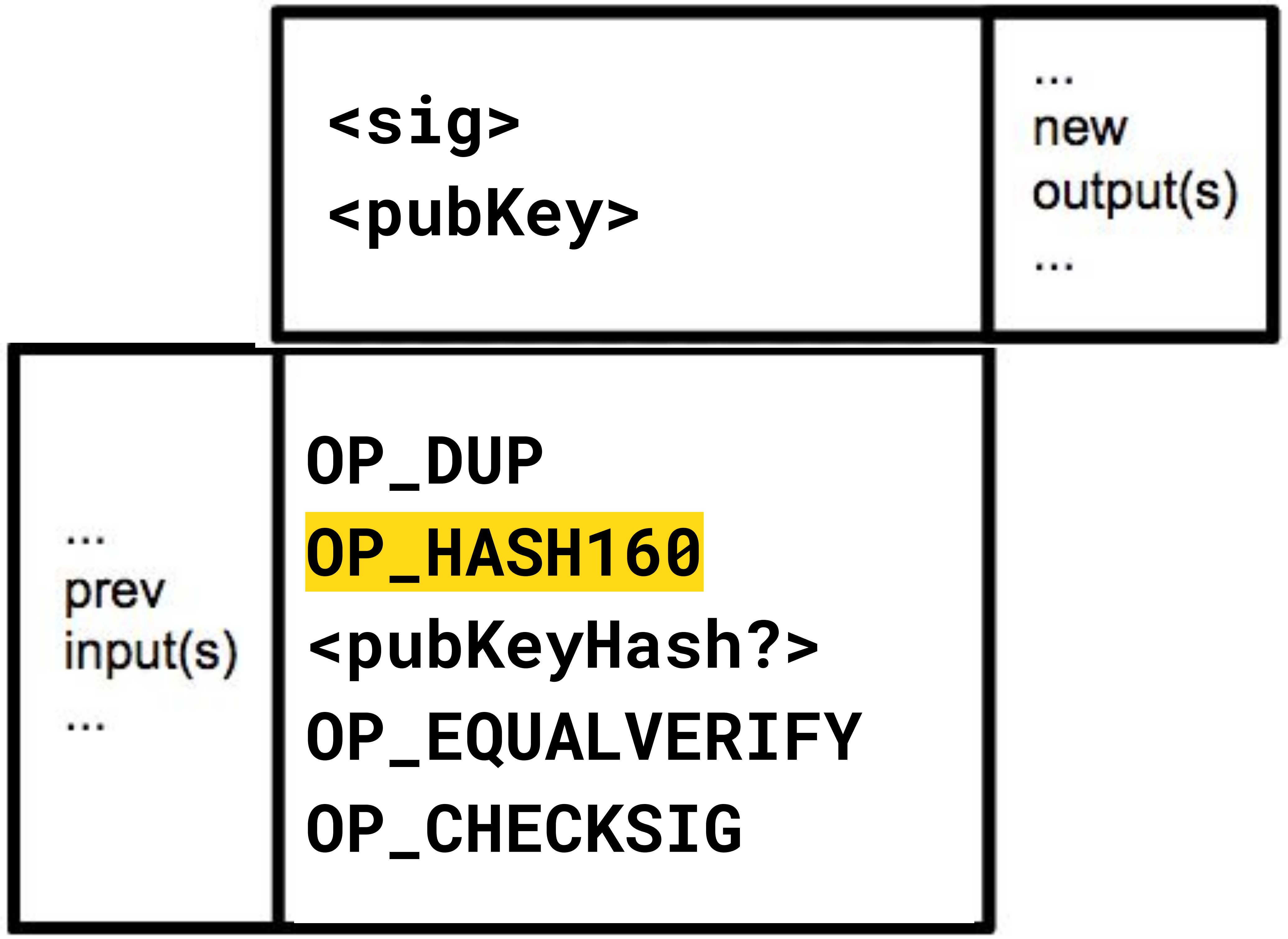
P2PKH EXAMPLE EXECUTION

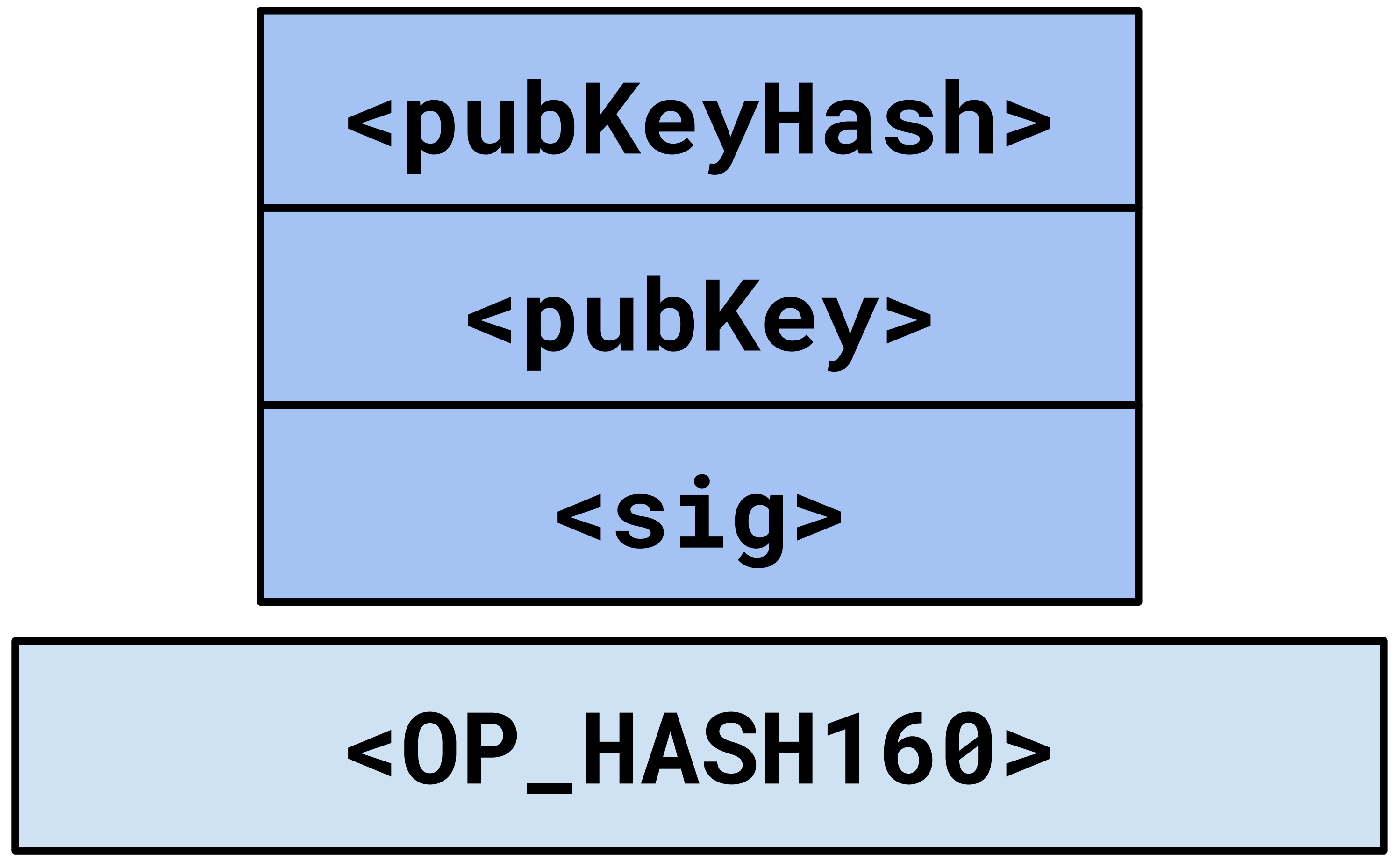




BITCOIN SCRIPT

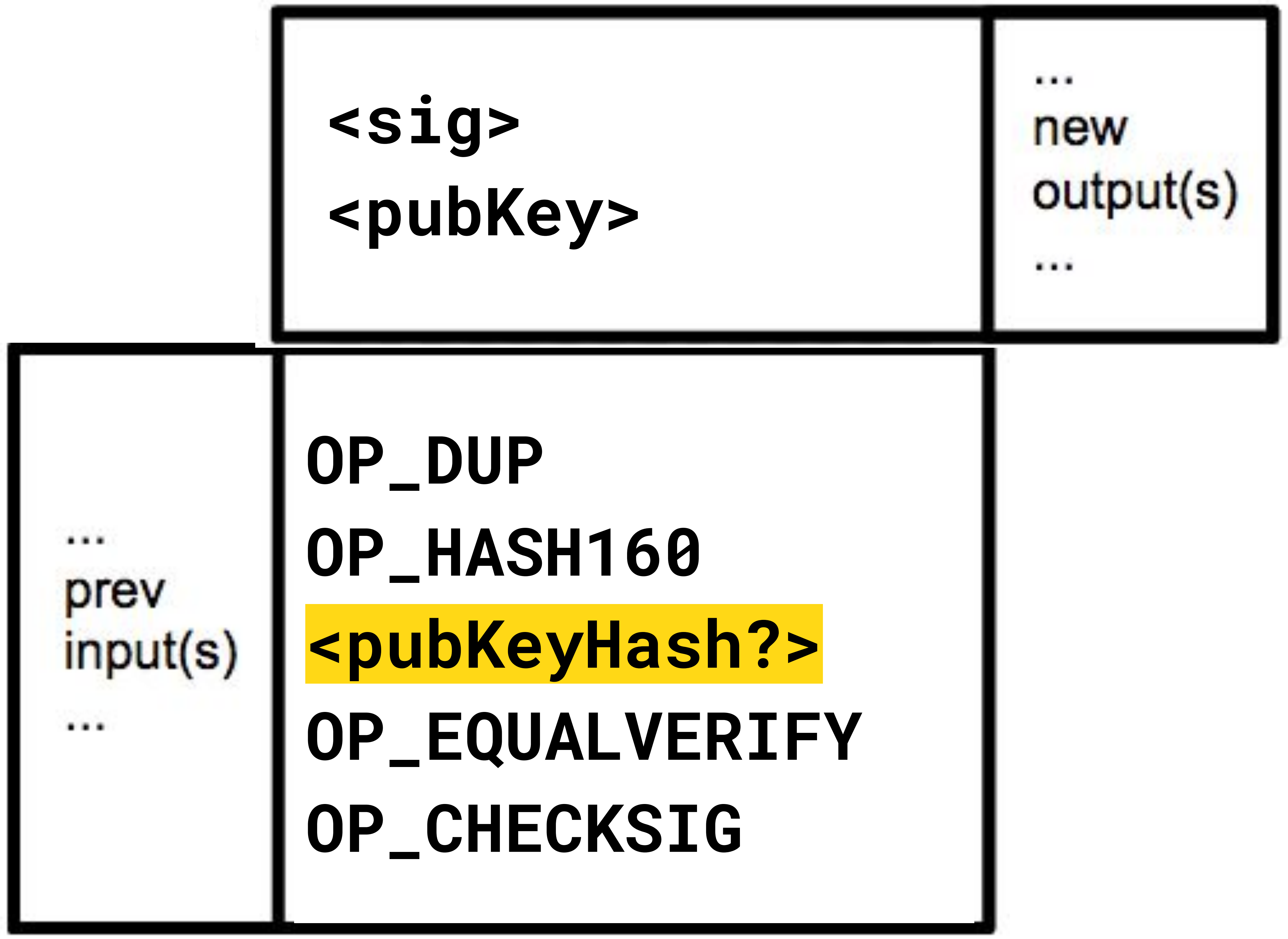
P2PKH EXAMPLE EXECUTION

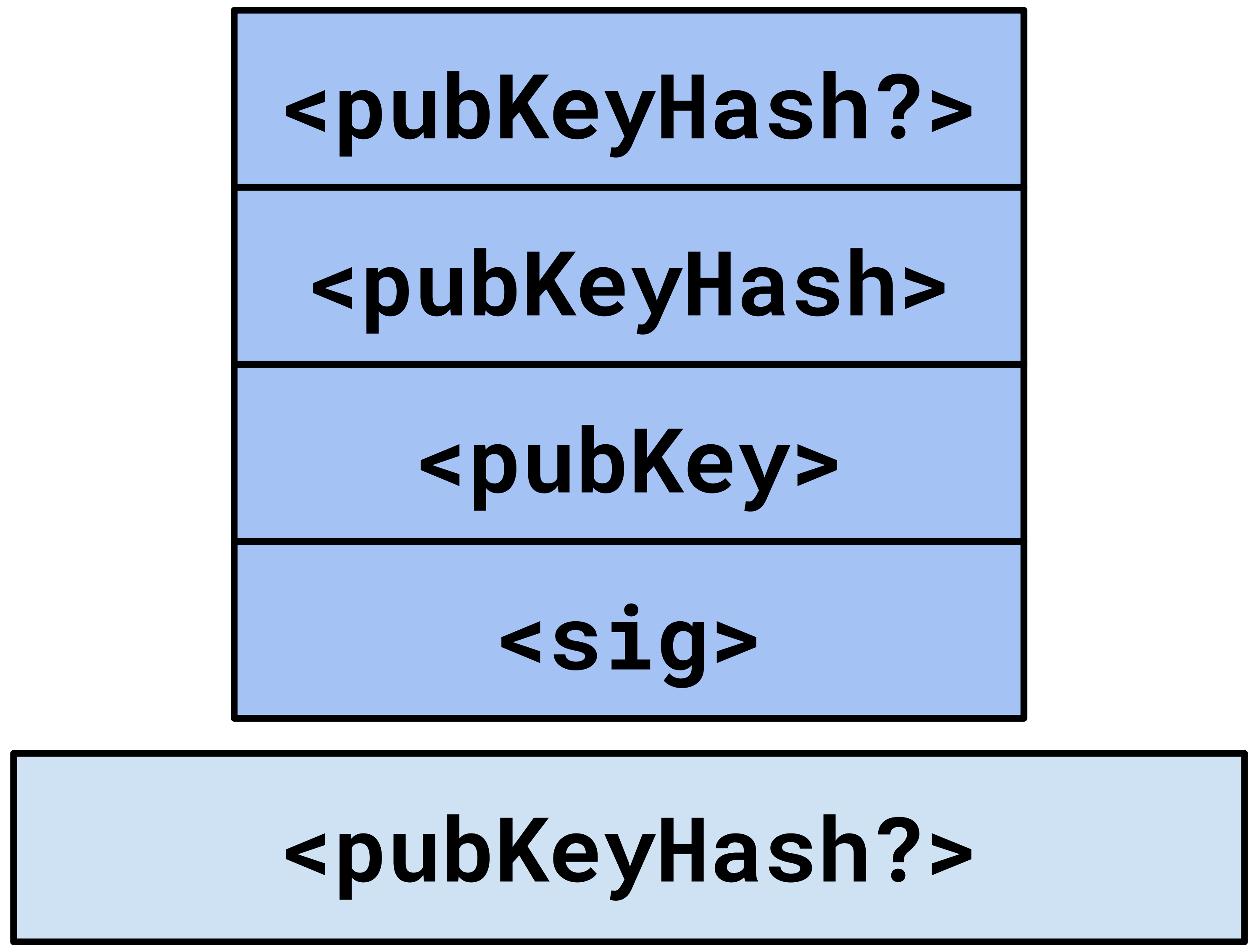




BITCOIN SCRIPT

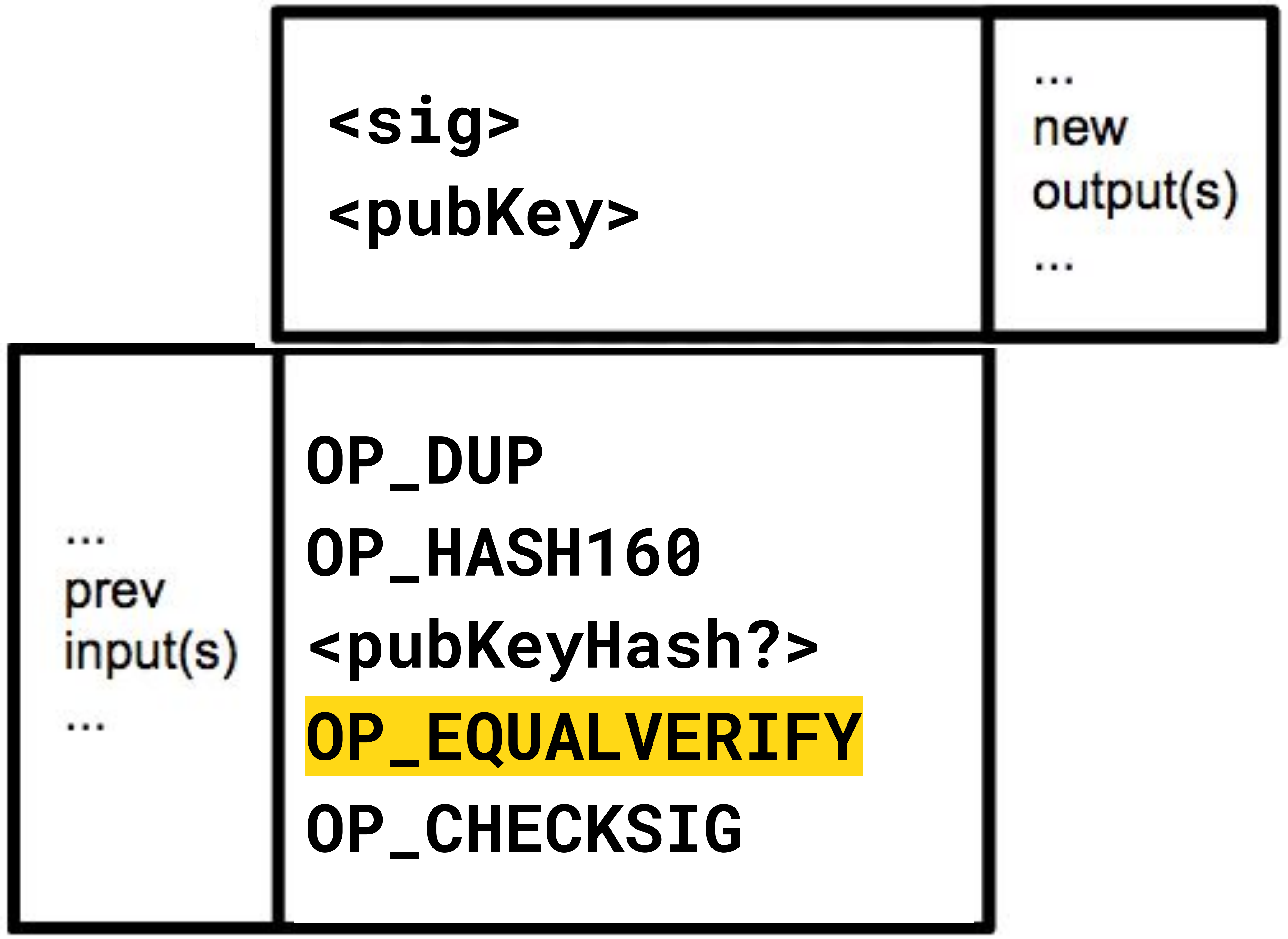
P2PKH EXAMPLE EXECUTION

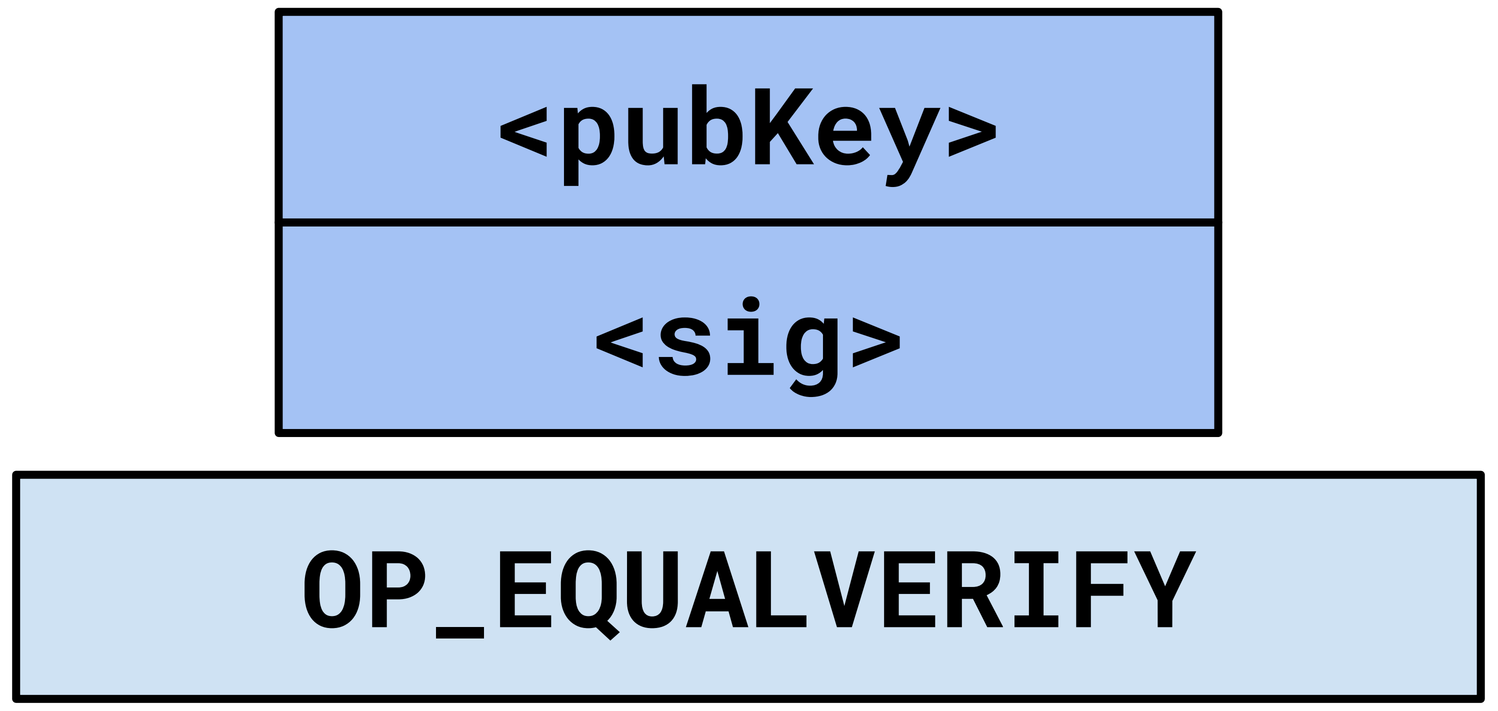




BITCOIN SCRIPT

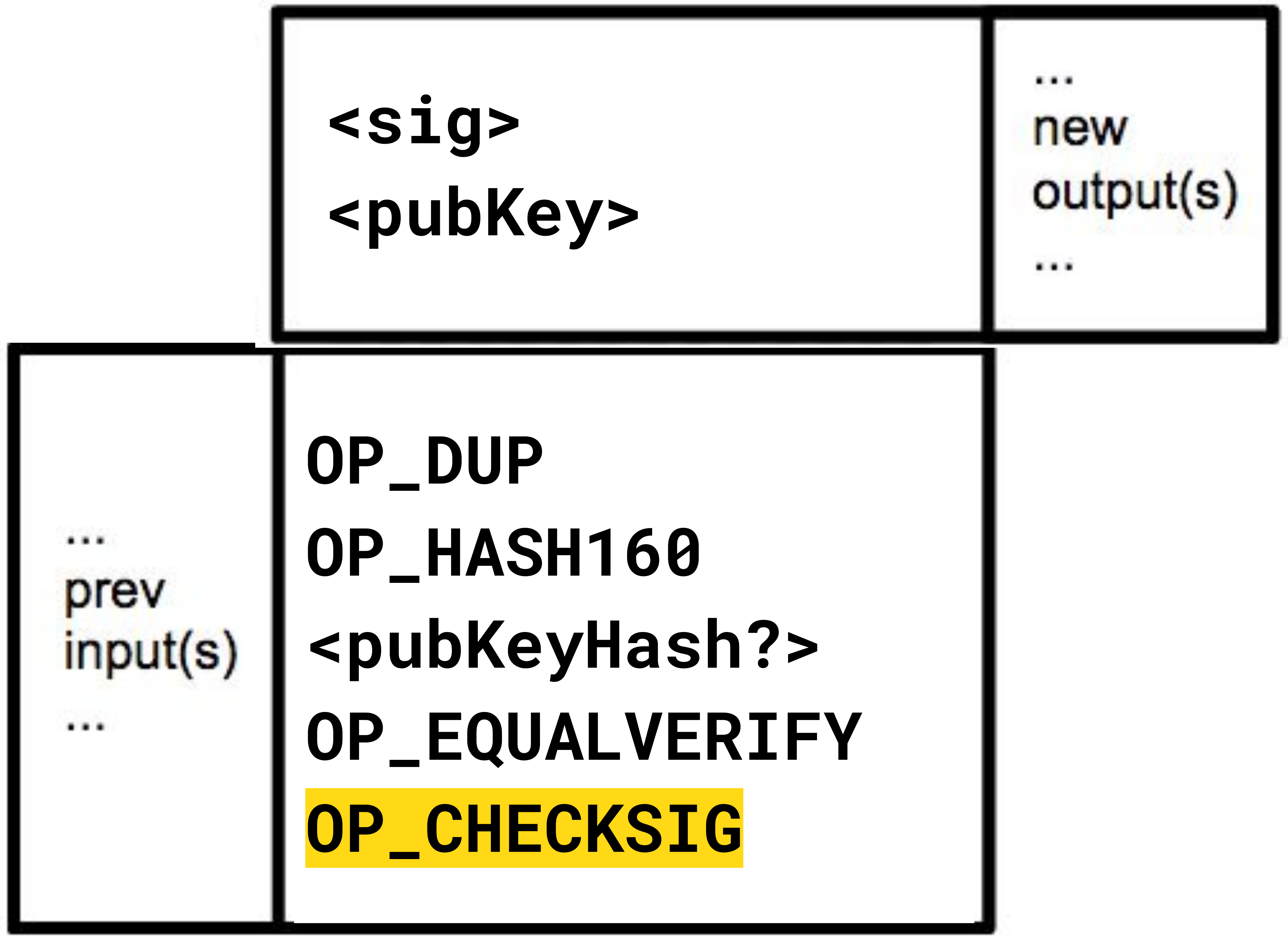
P2PKH EXAMPLE EXECUTION

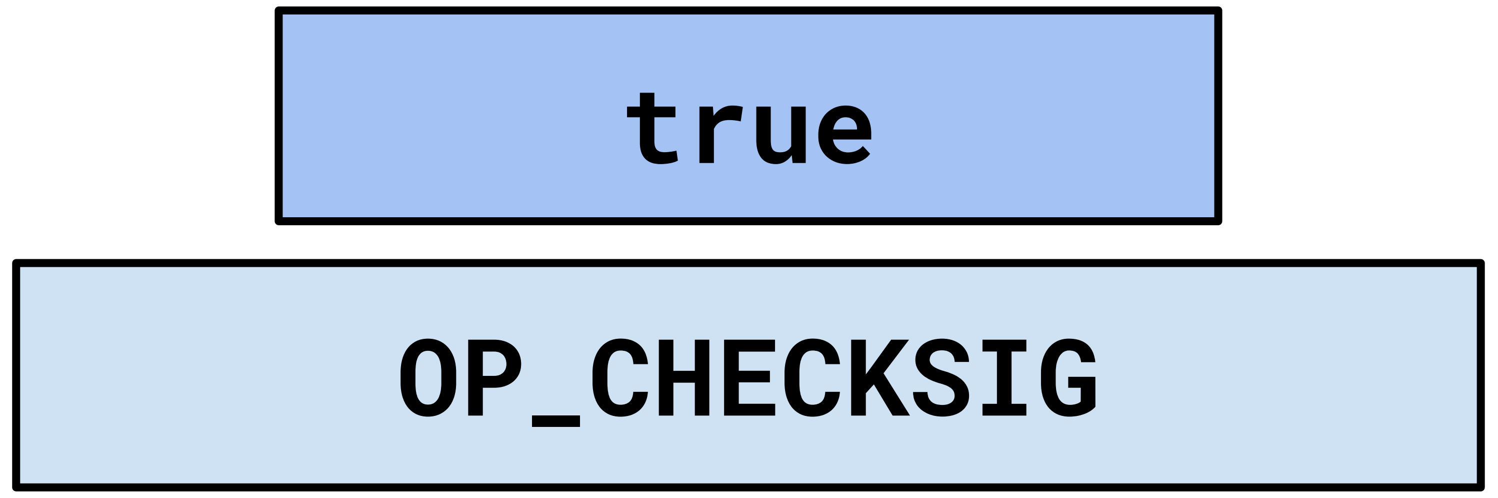




BITCOIN SCRIPT

P2PKH EXAMPLE EXECUTION





2

SOLIDITY ASSEMBLY REFRESHER

BITCOIN SCRIPT

P2PKH EXAMPLE EXECUTION

```
function nativeLoops() public returns (uint _r) {
   for(uint i = 0; i < 10; i++) {
       _r++;
   }
}
```

```
function asmLoops() public returns (uint _r) {
   assembly {
       let i := 0
       loop:
       i := add(i, 1)
       _r := add(_r, 1)
       jumpi(loop, lt(i, 10))
   }
}
```

```
function inlineAsmLoops() public returns (uint _r) {
   assembly {
       0 // i
       10 // max
       loop:
       // i := add(i, 1)
       dup2
       1
       add
       swap2
       pop
      ...
```

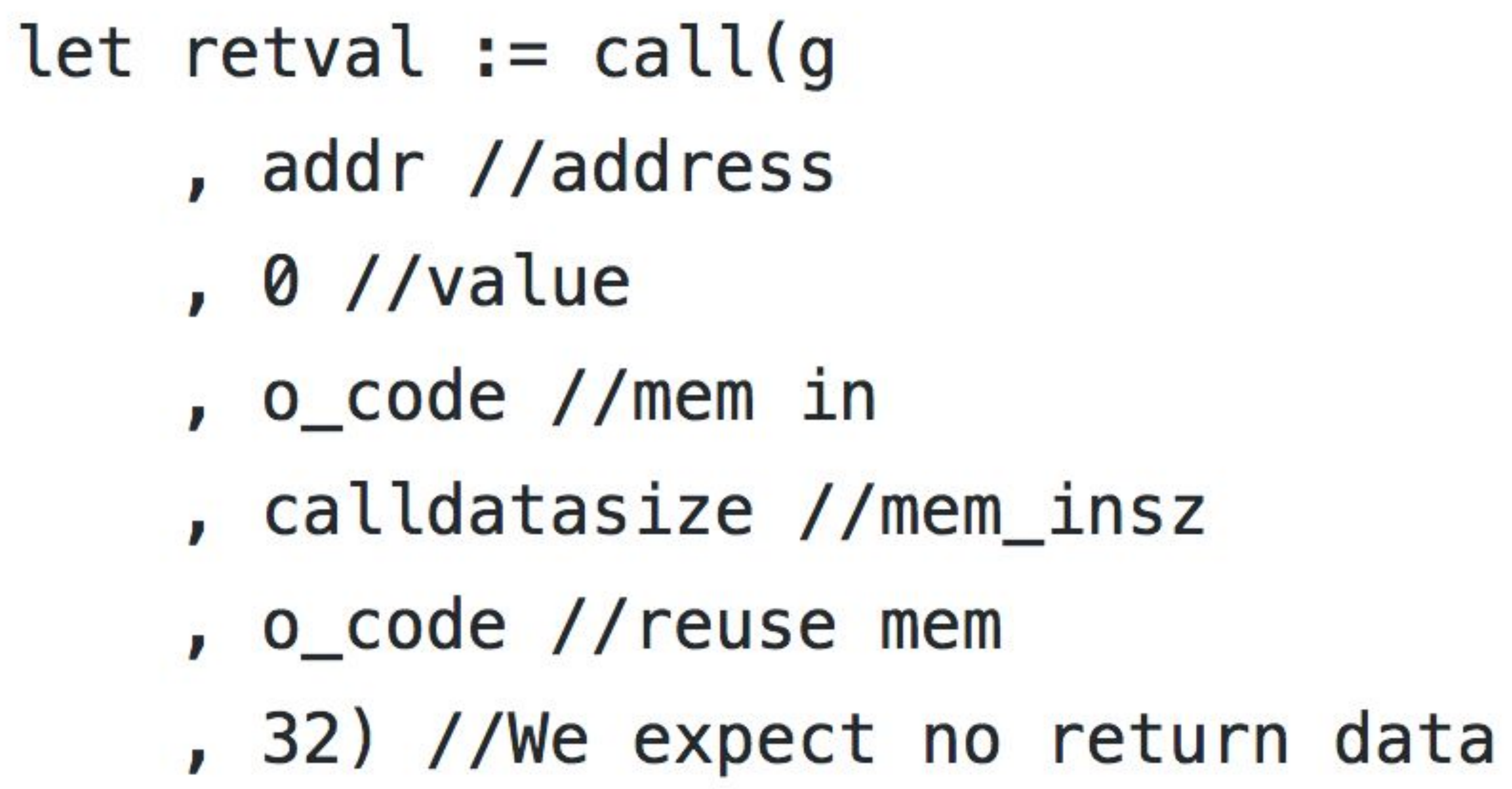
```
    ...
      // _r := add(_r, 1)
       dup3
       1
       add
       swap3
       pop
       // lt(i, 10)
       dup1
       dup3
       lt
       // jumpi(loop, lt(i, 10))
       loop
       jumpi
       pop
       pop
   }
```

3

ASSIGNMENT

ASSIGNMENT: INLINE ASSEMBLY

Explain what's going on in `Mystery.sol`



```
let retval := call(g
    , addr //address
    , 0 //value
    , o_code //mem in
    , calldatasize //mem_insz
    , o_code //reuse mem
    , 32) //We expect no return data
```

SEE YOU NEXT TIME

Scaling

Sharding

Casper

State Channels

Lightning/Plasma

IPFS (Extra)

BLOCKCHAIN AT BERKELEY