SET UP

- Program factoid.py is concerned with shades of primality
- Module primal.py provides three functions to assist

PYTHON FILES

Program factoid.py

```
import primal

a1, b1 = 6, 2
a2, b2 = 7, 3
a3, b3 = 4, 5

r1 = primal.is_factor( a1, b1 )
r2 = primal.is_factor( a2, b2 )
r3 = primal.is_factor( a3, b3 )


print( r1 )
print( r2 )
print( r3 )
```

PYTHON FILES

Program factoid.py

```
import primal

a1, b1 = 6, 2
a2, b2 = 7, 3
a3, b3 = 4, 5

r1 = primal.is_factor( a1, b1 )
r2 = primal.is_factor( a2, b2 )
r3 = primal.is_factor( a3, b3 )


print( r1 )
print( r2 )
print( r3 )
```

Module primal.py

```
def is_factor( x, y ) :

    rem = x % y
    if ( rem == 0 ) :
        result = True
    else:
        result = False

    return result


def are_relative_primes( x, y ) :

    ...
```

Program Trace – begins with the program code, traditionally called the main

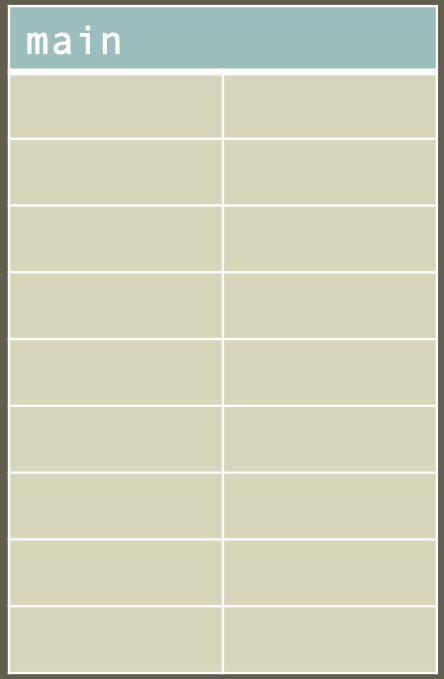
```
Program factoid.py

import primal

a1, b1 = 6, 2
a2, b2 = 7, 3
a3, b3 = 4, 5

r1 = primal.is_factor( a1, b1 )
r2 = primal.is_factor( a2, b2 )
r3 = primal.is_factor( a3, b3 )


print( r1 )
print( r2 )
print( r3 )
```



Program Trace – *execution steps through program*

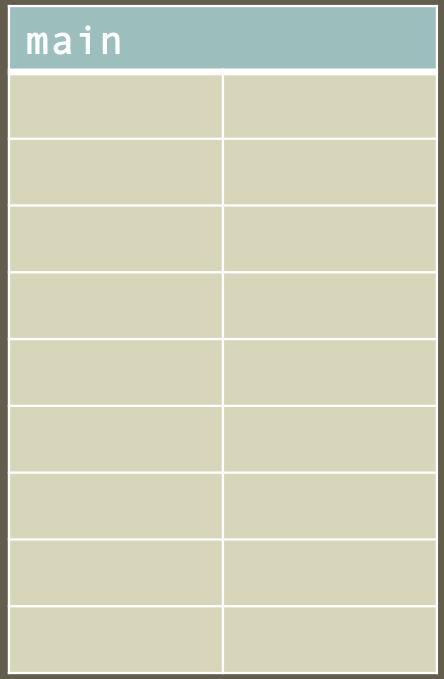
Program factoid.py

```
import primal

a1, b1 = 6, 2
a2, b2 = 7, 3
a3, b3 = 4, 5

r1 = primal.is_factor( a1, b1 )
r2 = primal.is_factor( a2, b2 )
r3 = primal.is_factor( a3, b3 )


print( r1 )
print( r2 )
print( r3 )
```



Module primal.py

```
def is_factor( x, y ) :

    rem = x % y
    if ( rem == 0 ) :
        result = True
    else:
        result = False

    return result


def are_relative_primes( x, y ) :

    ...
```

Program Trace – *execution steps through program*

Program factoid.py

```
import primal

a1, b1 = 6, 2
a2, b2 = 7, 3
a3, b3 = 4, 5

r1 = primal.is_factor( a1, b1 )
r2 = primal.is_factor( a2, b2 )
r3 = primal.is_factor( a3, b3 )


print( r1 )
print( r2 )
print( r3 )
```

main

a1	6
b1	2

Module primal.py

```
def is_factor( x, y ) :

    rem = x % y
    if ( rem == 0 ) :
        result = True
    else:
        result = False

    return result


def are_relative_primes( x, y ) :

    ...
```

Program Trace – *execution steps through program*

Program factoid.py

```
import primal

a1, b1 = 6, 2
a2, b2 = 7, 3
a3, b3 = 4, 5

r1 = primal.is_factor( a1, b1 )
r2 = primal.is_factor( a2, b2 )
r3 = primal.is_factor( a3, b3 )


print( r1 )
print( r2 )
print( r3 )
```

main	
a1	6
b1	2
a2	7
b2	3

Module primal.py

```
def is_factor( x, y ) :

    rem = x % y
    if ( rem == 0 ) :
        result = True
    else:
        result = False

    return result


def are_relative_primes( x, y ) :

    ...
```

Program Trace – *execution steps through program*

Program factoid.py

```
import primal

a1, b1 = 6, 2
a2, b2 = 7, 3
a3, b3 = 4, 5

r1 = primal.is_factor( a1, b1 )
r2 = primal.is_factor( a2, b2 )
r3 = primal.is_factor( a3, b3 )

print( r1 )
print( r2 )
print( r3 )
```

main	
a1	6
b1	2
a2	7
b2	3
a3	4
b3	5

Module primal.py

```
def is_factor( x, y ) :

    rem = x % y
    if ( rem == 0 ) :
        result = True
    else:
        result = False

    return result


def are_relative_primes( x, y ) :

    ...
```

Program Trace – *function invocation causes transfer of control*

Program factoid.py

```
import primal

a1, b1 = 6, 2
a2, b2 = 7, 3
a3, b3 = 4, 5

r1 = primal.is_factor( a1, b1 )
r2 = primal.is_factor( a2, b2 )
r3 = primal.is_factor( a3, b3 )

print( r1 )
print( r2 )
print( r3 )
```

main	
a1	6
b1	2
a2	7
b2	3
a3	4
b3	5

Module primal.py

```
def is_factor( x, y ) :

    rem = x % y
    if ( rem == 0 ) :
        result = True
    else:
        result = False

    return result


def are_relative_primes( x, y ) :

    ...
```

Program Trace – *copies of arguments are passed (assigned) to parameters*

Program factoid.py

```
import primal

a1, b1 = 6, 2
a2, b2 = 7, 3
a3, b3 = 4, 5

r1 = primal.is_factor( a1, b1 )
r2 = primal.is_factor( a2, b2 )
r3 = primal.is_factor( a3, b3 )

print( r1 )
print( r2 )
print( r3 )
```

main

a1	6
b1	2
a2	7
b2	3
a3	4
b3	5
r1	???

Module primal.py

```
def is_factor( x, y ) :

    rem = x % y
    if
                    x
    el              y

    return result


def are_relative_primes( x, y ) :

    ...
```

is_factor()

x	
y	

Program Trace – *copies of arguments are passed (assigned) to parameters*

Program factoid.py

```
import primal

a1, b1 = 6, 2
a2, b2 = 7, 3
a3, b3 = 4, 5

r1 = primal.is_factor( a1, b1 )
r2 = primal.is_factor( a2, b2 )
r3 = primal.is_factor( a3, b3 )

print( r1 )
print( r2 )
print( r3 )
```

main

a1	6
b1	2
a2	7
b2	3
a3	4
b3	5
r1	???

Module primal.py

```
def is_factor( x, y ) :

    rem = x % y
    if
    el
    return result


def are_relative_primes( x, y ) :

    ...
```

is_factor()

x	6
y	

Program Trace – *copies of arguments are passed (assigned) to parameters*

Program factoid.py

```
import primal

a1, b1 = 6, 2
a2, b2 = 7, 3
a3, b3 = 4, 5

r1 = primal.is_factor( a1, b1 )
r2 = primal.is_factor( a2, b2 )
r3 = primal.is_factor( a3, b3 )

print( r1 )
print( r2 )
print( r3 )
```

main

a1	6
b1	2
a2	7
b2	3
a3	4
b3	5
r1	???

Module primal.py

```
def is_factor( x, y ) :

    rem = x % y
    if
    el

    return result


def are_relative_primes( x, y ) :

    ...
```

is_factor()

x	6
y	

Program Trace – *copies of arguments are passed (assigned) to parameters*

Program factoid.py

```
import primal

a1, b1 = 6, 2
a2, b2 = 7, 3
a3, b3 = 4, 5

r1 = primal.is_factor( a1, b1 )
r2 = primal.is_factor( a2, b2 )
r3 = primal.is_factor( a3, b3 )

print( r1 )
print( r2 )
print( r3 )
```

main

a1	6
b1	2
a2	7
b2	3
a3	4
b3	5
r1	???

Module primal.py

```
def is_factor( x, y ) :

    rem = x % y
    if
    el

    return result


def are_relative_primes( x, y ) :

    ...
```

is_factor()

x	6
y	2

Program Trace – *cpu control is passed to function*

Program factoid.py

```
import primal

a1, b1 = 6, 2
a2, b2 = 7, 3
a3, b3 = 4, 5

r1 = primal.is_factor( a1, b1 )
r2 = primal.is_factor( a2, b2 )
r3 = primal.is_factor( a3, b3 )

print( r1 )
print( r2 )
print( r3 )
```

main

a1	6
b1	2
a2	7
b2	3
a3	4
b3	5
r1	???

Module primal.py

```
def is_factor( x, y ) :

    rem = x % y
    if
    el

    return result


def are_relative_primes( x, y ) :

    ...
```

is_factor()

x	6
y	2

Program Trace – *cpu control is passed to function*

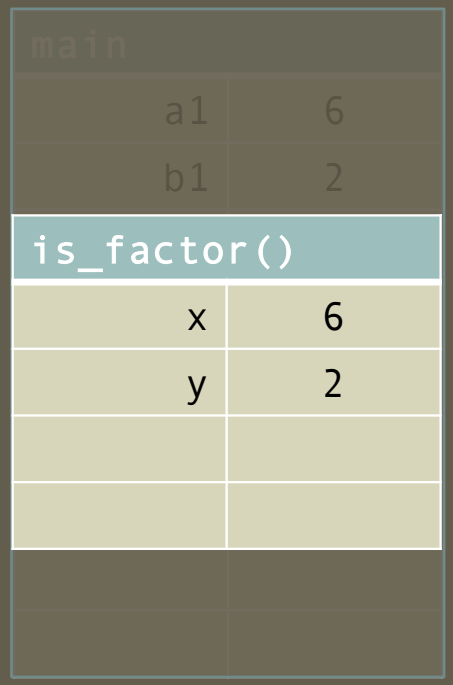
Program factoid.py

```
import primal

a1, b1 = 6, 2
a2, b2 = 7, 3
a3, b3 = 4, 5

r1 = primal.is_factor( a1, b1 )
r2 = primal.is_factor( a2, b2 )
r3 = primal.is_factor( a3, b3 )

print( r1 )
print( r2 )
print( r3 )
```



Module primal.py

```
def is_factor( x, y ) :

    rem = x % y
    if ( rem == 0 ) :
        result = True
    else:
        result = False

    return result


def are_relative_primes( x, y ) :

    ...
```

Program Trace – *function executes*

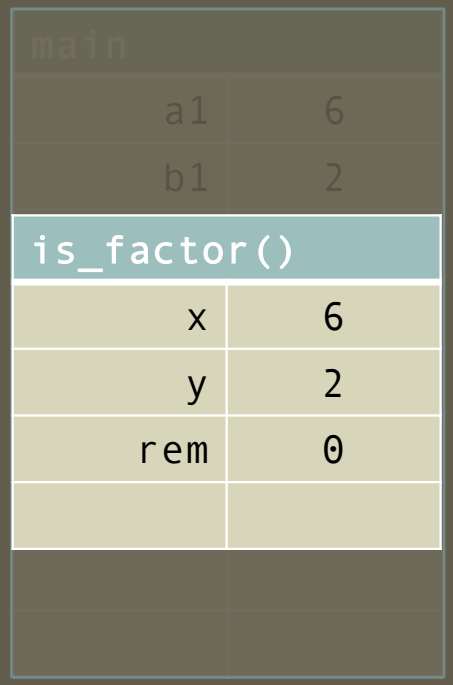
Program factoid.py

```
import primal

a1, b1 = 6, 2
a2, b2 = 7, 3
a3, b3 = 4, 5

r1 = primal.is_factor( a1, b1 )
r2 = primal.is_factor( a2, b2 )
r3 = primal.is_factor( a3, b3 )

print( r1 )
print( r2 )
print( r3 )
```



Module primal.py

```
def is_factor( x, y ) :

    rem = x % y
    if ( rem == 0 ) :
        result = True
    else:
        result = False

    return result


def are_relative_primes( x, y ) :

    ...
```

Program Trace – *function executes*

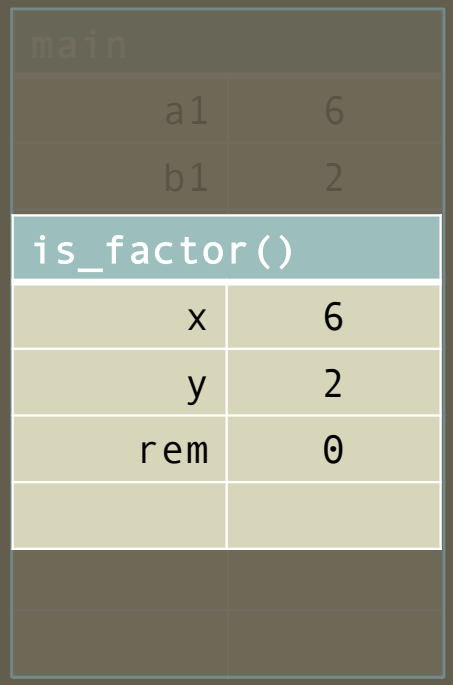
Program factoid.py

```
import primal

a1, b1 = 6, 2
a2, b2 = 7, 3
a3, b3 = 4, 5

r1 = primal.is_factor( a1, b1 )
r2 = primal.is_factor( a2, b2 )
r3 = primal.is_factor( a3, b3 )

print( r1 )
print( r2 )
print( r3 )
```



Module primal.py

```
def is_factor( x, y ) :

    rem = x % y
    if ( rem == 0 ) :
        result = True
    else:
        result = False

    return result


def are_relative_primes( x, y ) :

    ...
```

Program Trace – *function executes*

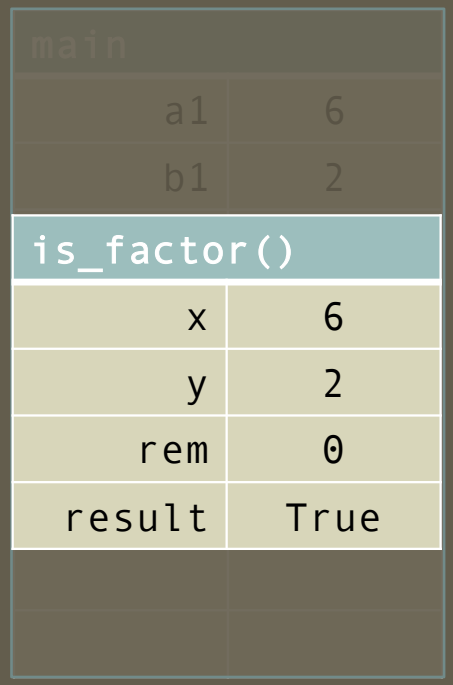
Program factoid.py

```
import primal

a1, b1 = 6, 2
a2, b2 = 7, 3
a3, b3 = 4, 5

r1 = primal.is_factor( a1, b1 )
r2 = primal.is_factor( a2, b2 )
r3 = primal.is_factor( a3, b3 )

print( r1 )
print( r2 )
print( r3 )
```



Module primal.py

```
def is_factor( x, y ) :

    rem = x % y
    if ( rem == 0 ) :
        result = True
    else:
        result = False

    return result


def are_relative_primes( x, y ) :

    ...
```

Program Trace – *function executes*

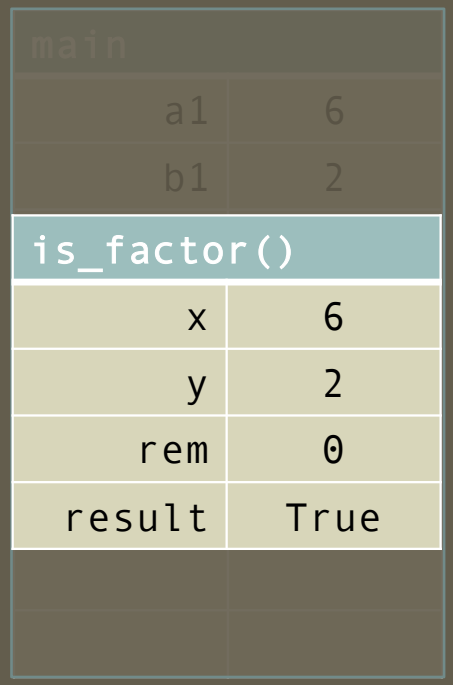
Program factoid.py

```
import primal

a1, b1 = 6, 2
a2, b2 = 7, 3
a3, b3 = 4, 5

r1 = primal.is_factor( a1, b1 )
r2 = primal.is_factor( a2, b2 )
r3 = primal.is_factor( a3, b3 )

print( r1 )
print( r2 )
print( r3 )
```



Module primal.py

```
def is_factor( x, y ) :

    rem = x % y
    if ( rem == 0 ) :
        result = True
    else:
        result = False

    return result


def are_relative_primes( x, y ) :
    ...
```

Program Trace – *return expression is the value of the invocation*

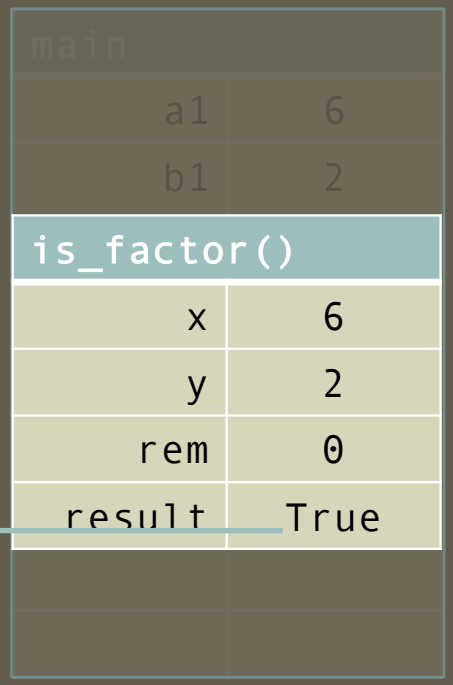
Program factoid.py

```
import primal

a1, b1 = 6, 2
a2, b2 = 7, 3
a3, b3 = 4, 5

r1 = primal.is_factor( a1, b1 )
r2 = primal.is_factor( a2, b2 )
r3 = primal.is_factor( a3, b3 )

print( r1 )
print( r2 )
print( r3 )
```



Module primal.py

```
def is_factor( x, y ) :

    rem = x % y
    if ( rem == 0 ) :
        result = True
    else:
        result = False

    return result


def are_relative_primes( x, y ) :

    ...
```

Program Trace – *return expression is the value of the invocation*

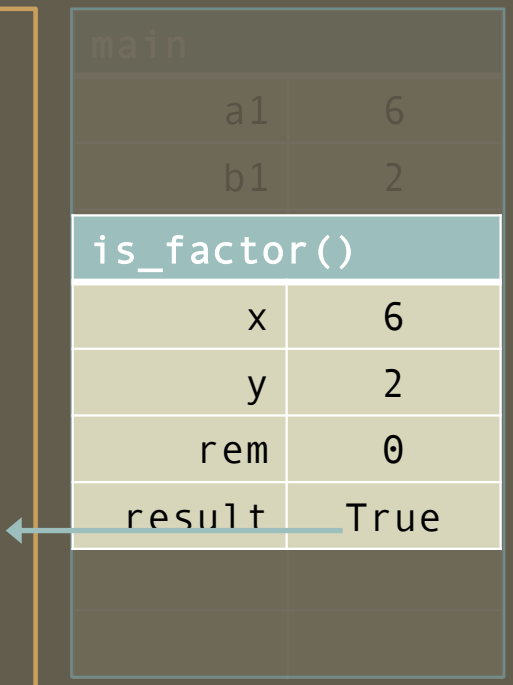
Program factoid.py

```
import primal

a1, b1 = 6, 2
a2, b2 = 7, 3
a3, b3 = 4, 5

r1 = True
r2 = primal.is_factor( a2, b2 )
r3 = primal.is_factor( a3, b3 )

print( r1 )
print( r2 )
print( r3 )
```



Module primal.py

```
def is_factor( x, y ) :

    rem = x % y
    if ( rem == 0 ) :
        result = True
    else:
        result = False

    return result


def are_relative_primes( x, y ) :

    ...
```

Program Trace – *cpu control is passed back to program*

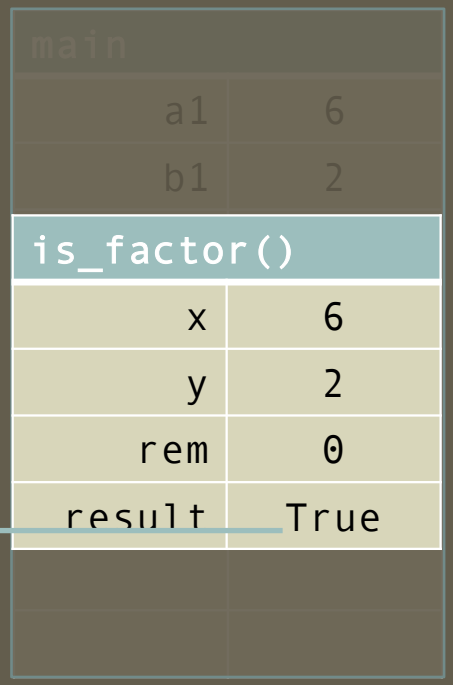
Program factoid.py

```
import primal

a1, b1 = 6, 2
a2, b2 = 7, 3
a3, b3 = 4, 5

r1 = primal.is_factor( a1, b1 )
r2 = primal.is_factor( a2, b2 )
r3 = primal.is_factor( a3, b3 )

print( r1 )
print( r2 )
print( r3 )
```



Module primal.py

```
def is_factor( x, y ) :

    rem = x % y
    if ( rem == 0 ) :
        result = True
    else:
        result = False

    return result


def are_relative_primes( x, y ) :

    ...
```

Program Trace – *cpu control is passed back to program*

Program factoid.py

```
import primal

a1, b1 = 6, 2
a2, b2 = 7, 3
a3, b3 = 4, 5

r1 = primal.is_factor( a1, b1 )
r2 = primal.is_factor( a2, b2 )
r3 = primal.is_factor( a3, b3 )

print( r1 )
print( r2 )
print( r3 )
```

main	
a1	6
b1	2
a2	7
b2	3
a3	4
b3	5
r1	True

Module primal.py

```
def is_factor( x, y ) :

    rem = x % y
    if ( rem == 0 ) :
        result = True
    else:
        result = False

    return result


def are_relative_primes( x, y ) :

    ...
```

Program Trace – *program steps through its code*

Program factoid.py

```
import primal

a1, b1 = 6, 2
a2, b2 = 7, 3
a3, b3 = 4, 5

r1 = primal.is_factor( a1, b1 )
r2 = primal.is_factor( a2, b2 )
r3 = primal.is_factor( a3, b3 )

print( r1 )
print( r2 )
print( r3 )
```

main	
a1	6
b1	2
a2	7
b2	3
a3	4
b3	5
r1	True
r2	???

Module primal.py

```
def is_factor( x, y ) :

    rem = x % y
    if ( rem == 0 ) :
        result = True
    else:
        result = False

    return result


def are_relative_primes( x, y ) :

    ...
```

Program Trace – *function invocation causes transfers of control*

Program factoid.py

```
import primal

a1, b1 = 6, 2
a2, b2 = 7, 3
a3, b3 = 4, 5

r1 = primal.is_factor( a1, b1 )
r2 = primal.is_factor( a2, b2 )
r3 = primal.is_factor( a3, b3 )

print( r1 )
print( r2 )
print( r3 )
```

main	
a1	6
b1	2
a2	7
b2	3
a3	4
b3	5
r1	True
r2	???

Module primal.py

```
def is_factor( x, y ) :

    rem = x % y
    if ( rem == 0 ) :
        result = True
    else:
        result = False

    return result


def are_relative_primes( x, y ) :

    ...
```

Program Trace – *copies of arguments are passed (assigned) to parameters*

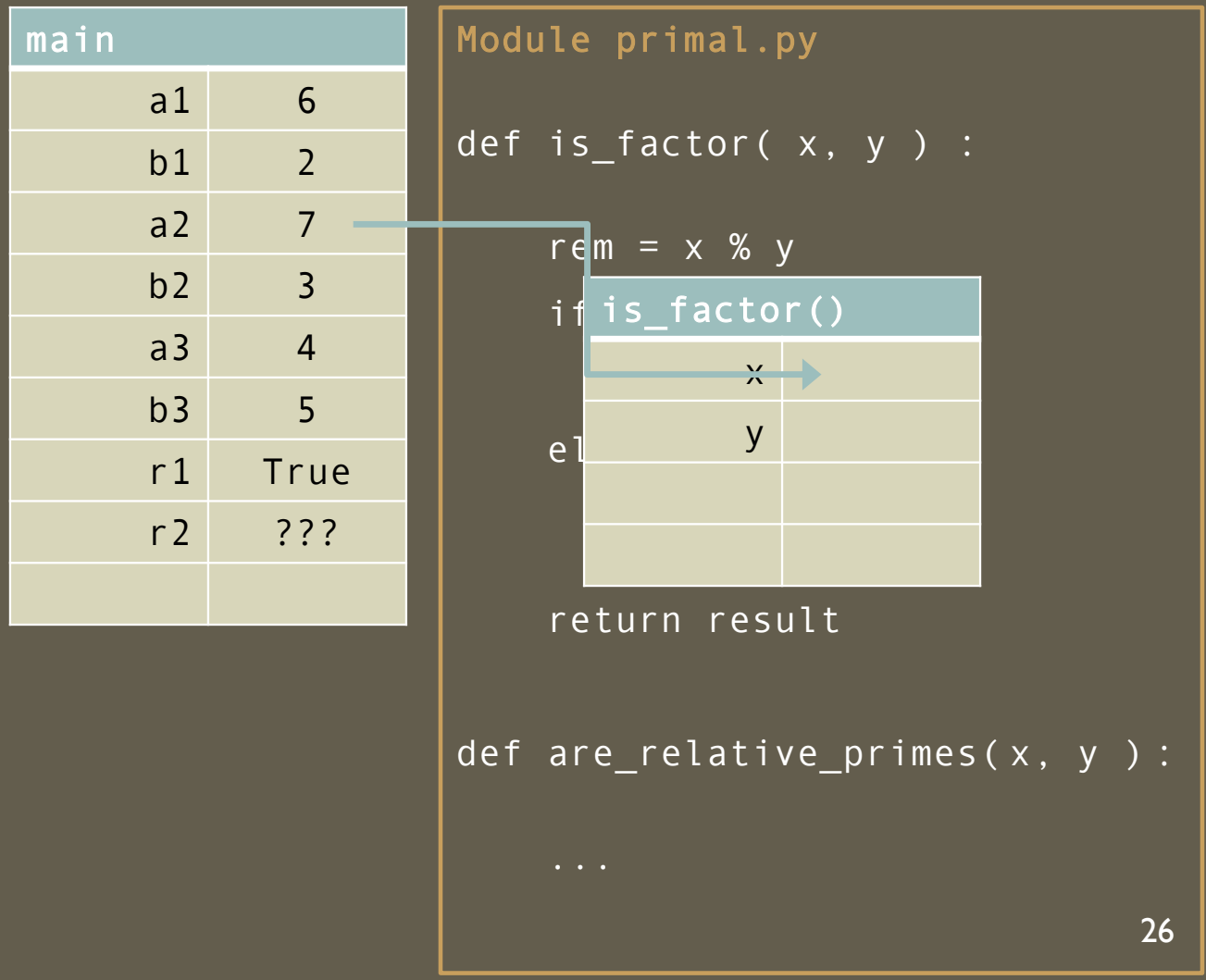
Program factoid.py

```
import primal

a1, b1 = 6, 2
a2, b2 = 7, 3
a3, b3 = 4, 5

r1 = primal.is_factor( a1, b1 )
r2 = primal.is_factor( a2, b2 )
r3 = primal.is_factor( a3, b3 )

print( r1 )
print( r2 )
print( r3 )
```



Program Trace – *copies of arguments are passed (assigned) to parameters*

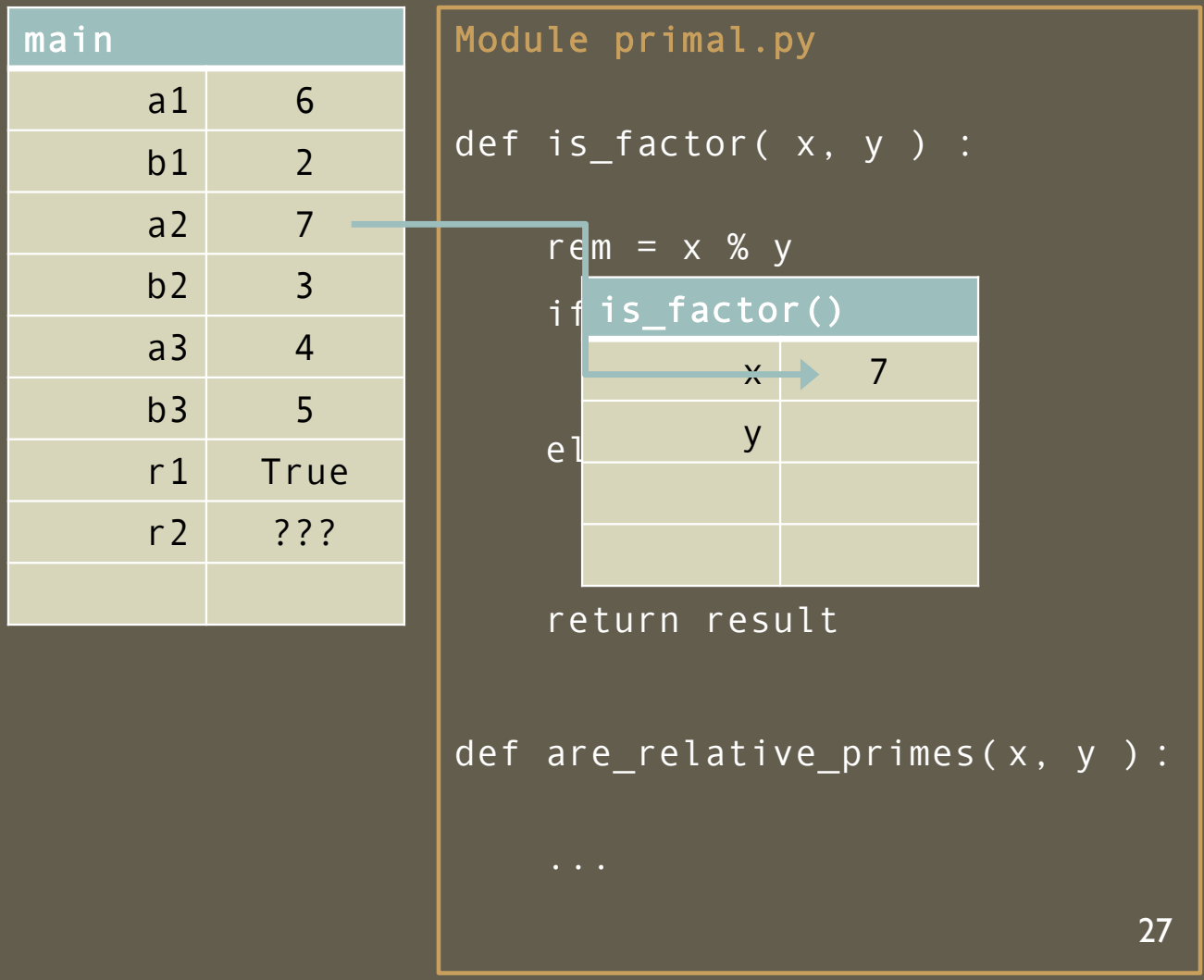
Program factoid.py

```
import primal

a1, b1 = 6, 2
a2, b2 = 7, 3
a3, b3 = 4, 5

r1 = primal.is_factor( a1, b1 )
r2 = primal.is_factor( a2, b2 )
r3 = primal.is_factor( a3, b3 )

print( r1 )
print( r2 )
print( r3 )
```



Program Trace – *copies of arguments are passed (assigned) to parameters*

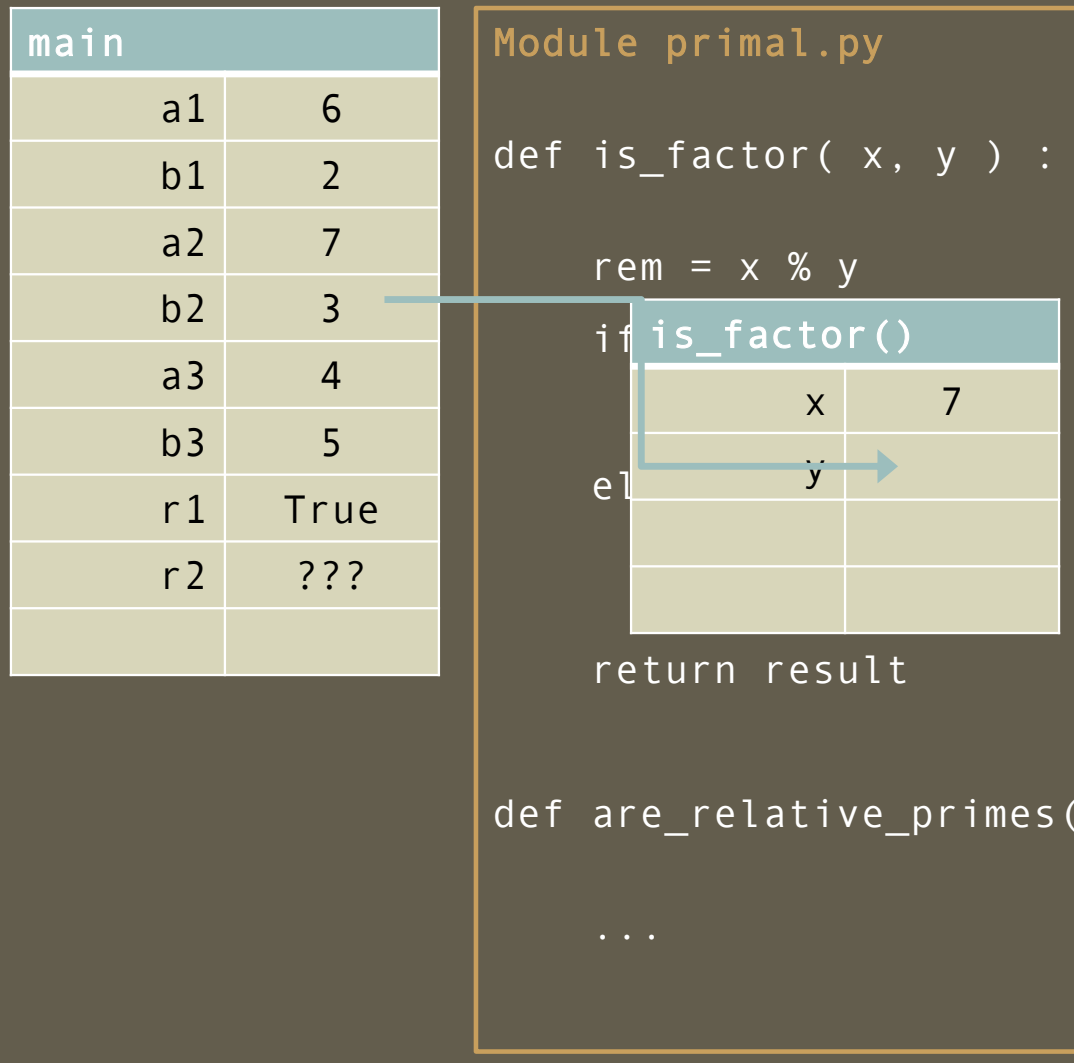
Program factoid.py

```
import primal

a1, b1 = 6, 2
a2, b2 = 7, 3
a3, b3 = 4, 5

r1 = primal.is_factor( a1, b1 )
r2 = primal.is_factor( a2, b2 )
r3 = primal.is_factor( a3, b3 )

print( r1 )
print( r2 )
print( r3 )
```



Module primal.py

```
def is_factor( x, y ) :

    rem = x % y
    if
    el
    return result

def are_relative_primes( x, y ) :

    ...
```

Program Trace – *copies of arguments are passed (assigned) to parameters*

Program factoid.py

```
import primal

a1, b1 = 6, 2
a2, b2 = 7, 3
a3, b3 = 4, 5

r1 = primal.is_factor( a1, b1 )
r2 = primal.is_factor( a2, b2 )
r3 = primal.is_factor( a3, b3 )

print( r1 )
print( r2 )
print( r3 )
```

main

a1	6
b1	2
a2	7
b2	3
a3	4
b3	5
r1	True
r2	???

Module primal.py

```
def is_factor( x, y ) :

    rem = x % y
    if
    el
    return result

def are_relative_primes( x, y ) :

    ...
```

is_factor()

x	7
y	3

Program Trace – *cpu control is passed to function*

Program factoid.py

```
import primal

a1, b1 = 6, 2
a2, b2 = 7, 3
a3, b3 = 4, 5

r1 = primal.is_factor( a1, b1 )
r2 = primal.is_factor( a2, b2 )
r3 = primal.is_factor( a3, b3 )

print( r1 )
print( r2 )
print( r3 )
```

main

a1	6
b1	2
a2	7
b2	3
a3	4
b3	5
r1	True
r2	???

Module primal.py

```
def is_factor( x, y ) :

    rem = x % y
    if
    el

    return result


def are_relative_primes( x, y ) :

    ...
```

is_factor()

x	7
y	3

Program Trace – *cpu control is passed to function*

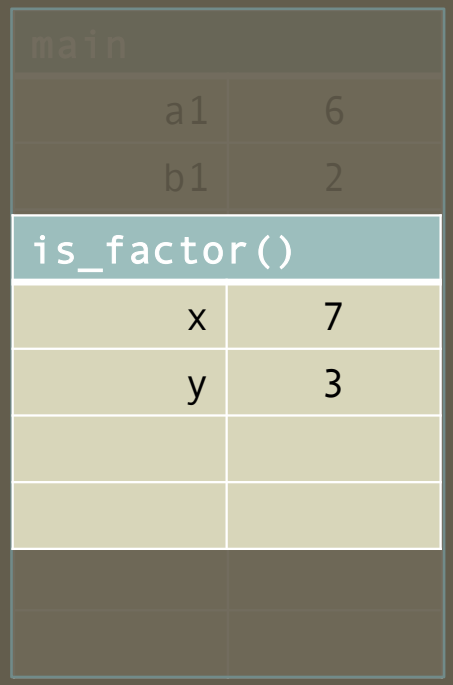
Program factoid.py

```
import primal

a1, b1 = 6, 2
a2, b2 = 7, 3
a3, b3 = 4, 5

r1 = primal.is_factor( a1, b1 )
r2 = primal.is_factor( a2, b2 )
r3 = primal.is_factor( a3, b3 )

print( r1 )
print( r2 )
print( r3 )
```



Module primal.py

```
def is_factor( x, y ) :

    rem = x % y
    if ( rem == 0 ) :
        result = True
    else:
        result = False

    return result


def are_relative_primes( x, y ) :

    ...
```

Program Trace – *function executes*

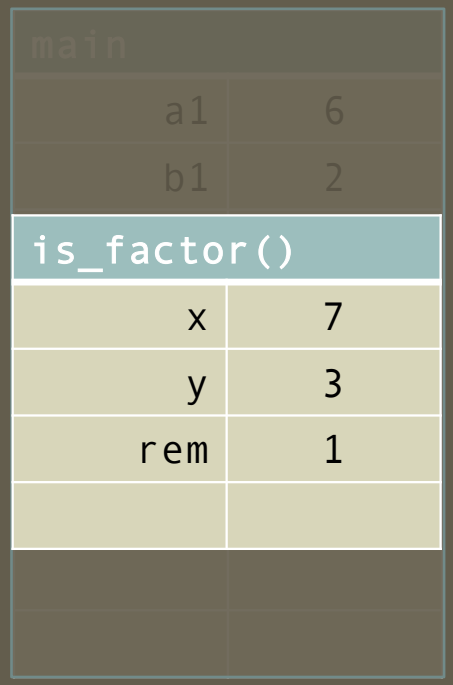
Program factoid.py

```
import primal

a1, b1 = 6, 2
a2, b2 = 7, 3
a3, b3 = 4, 5

r1 = primal.is_factor( a1, b1 )
r2 = primal.is_factor( a2, b2 )
r3 = primal.is_factor( a3, b3 )

print( r1 )
print( r2 )
print( r3 )
```



Module primal.py

```
def is_factor( x, y ) :

    rem = x % y
    if ( rem == 0 ) :
        result = True
    else:
        result = False

    return result


def are_relative_primes( x, y ) :

    ...
```

Program Trace – *function executes*

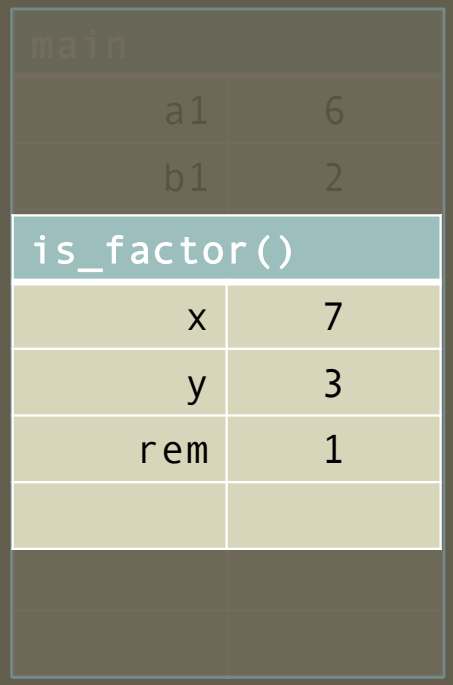
Program factoid.py

```
import primal

a1, b1 = 6, 2
a2, b2 = 7, 3
a3, b3 = 4, 5

r1 = primal.is_factor( a1, b1 )
r2 = primal.is_factor( a2, b2 )
r3 = primal.is_factor( a3, b3 )

print( r1 )
print( r2 )
print( r3 )
```



Module primal.py

```
def is_factor( x, y ) :

    rem = x % y
    if ( rem == 0 ) :
        result = True
    else:
        result = False

    return result


def are_relative_primes( x, y ) :

    ...
```

Program Trace – *function executes*

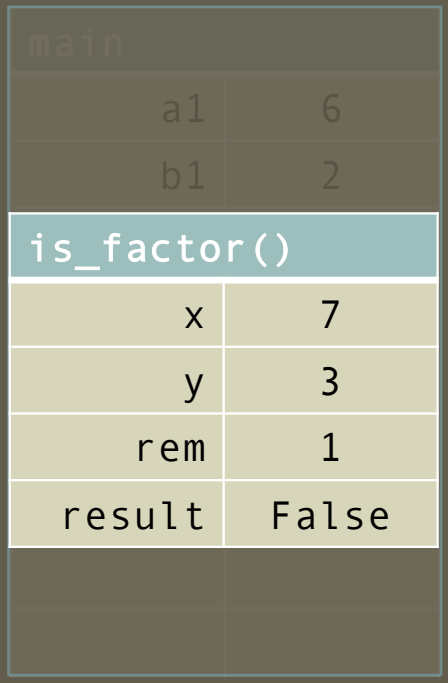
Program factoid.py

```
import primal

a1, b1 = 6, 2
a2, b2 = 7, 3
a3, b3 = 4, 5

r1 = primal.is_factor( a1, b1 )
r2 = primal.is_factor( a2, b2 )
r3 = primal.is_factor( a3, b3 )

print( r1 )
print( r2 )
print( r3 )
```



Module primal.py

```
def is_factor( x, y ) :

    rem = x % y
    if ( rem == 0 ) :
        result = True
    else:
        result = False

    return result


def are_relative_primes( x, y ) :

    ...
```

Program Trace – *function executes*

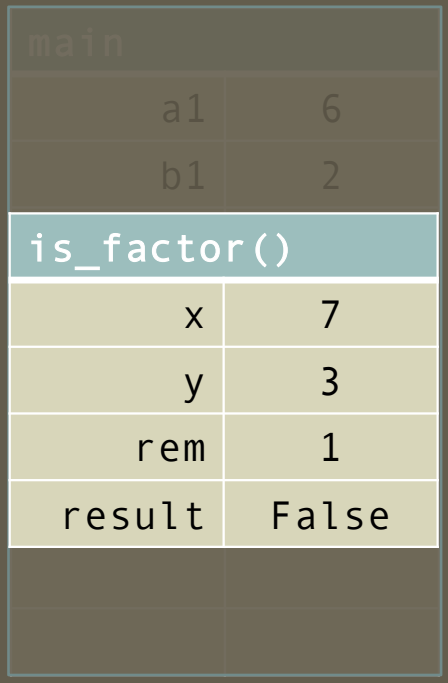
Program factoid.py

```
import primal

a1, b1 = 6, 2
a2, b2 = 7, 3
a3, b3 = 4, 5

r1 = primal.is_factor( a1, b1 )
r2 = primal.is_factor( a2, b2 )
r3 = primal.is_factor( a3, b3 )

print( r1 )
print( r2 )
print( r3 )
```



Module primal.py

```
def is_factor( x, y ) :

    rem = x % y
    if ( rem == 0 ) :
        result = True
    else:
        result = False

    return result


def are_relative_primes( x, y ) :

    ...
```

Program Trace – *return expression is the value of the invocation*

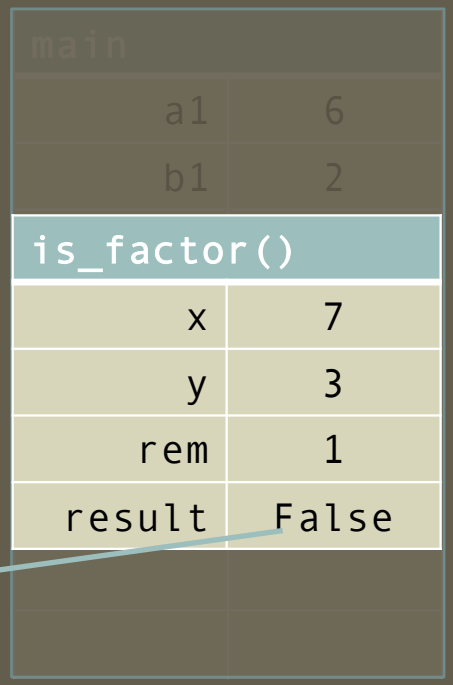
Program factoid.py

```
import primal

a1, b1 = 6, 2
a2, b2 = 7, 3
a3, b3 = 4, 5

r1 = primal.is_factor( a1, b1 )
r2 = primal.is_factor( a2, b2 )
r3 = primal.is_factor( a3, b3 )

print( r1 )
print( r2 )
print( r3 )
```



Module primal.py

```
def is_factor( x, y ) :

    rem = x % y
    if ( rem == 0 ) :
        result = True
    else:
        result = False

    return result


def are_relative_primes( x, y ) :

    ...
```

Program Trace – *cpu control is passed back to program*

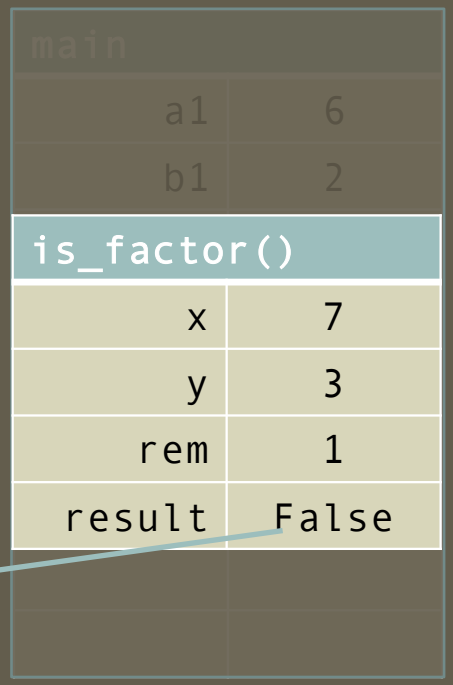
Program factoid.py

```
import primal

a1, b1 = 6, 2
a2, b2 = 7, 3
a3, b3 = 4, 5

r1 = primal.is_factor( a1, b1 )
r2 = False
r3 = primal.is_factor( a3, b3 )

print( r1 )
print( r2 )
print( r3 )
```



Module primal.py

```
def is_factor( x, y ) :

    rem = x % y
    if ( rem == 0 ) :
        result = True
    else:
        result = False

    return result


def are_relative_primes( x, y ) :

    ...
```

Program Trace – *cpu control is passed back to program*

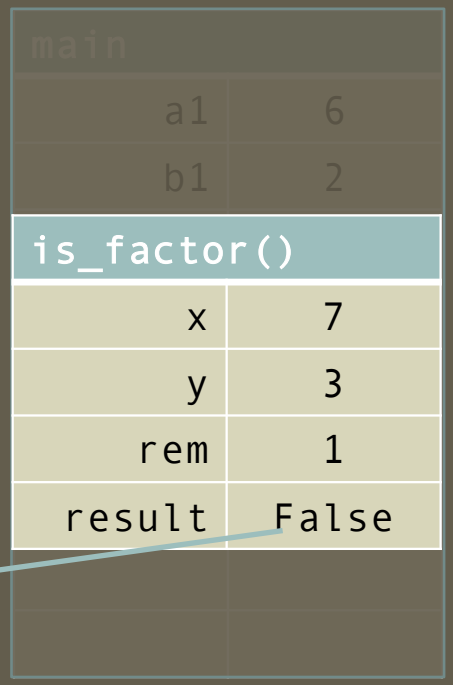
Program factoid.py

```
import primal

a1, b1 = 6, 2
a2, b2 = 7, 3
a3, b3 = 4, 5

r1 = primal.is_factor( a1, b1 )
r2 = primal.is_factor( a2, b2 )
r3 = primal.is_factor( a3, b3 )

print( r1 )
print( r2 )
print( r3 )
```



Module primal.py

```
def is_factor( x, y ) :

    rem = x % y
    if ( rem == 0 ) :
        result = True
    else:
        result = False

    return result


def are_relative_primes( x, y ) :

    ...
```

Program Trace – *cpu control is passed back to program*

Program factoid.py

```
import primal

a1, b1 = 6, 2
a2, b2 = 7, 3
a3, b3 = 4, 5

r1 = primal.is_factor( a1, b1 )
r2 = primal.is_factor( a2, b2 )
r3 = primal.is_factor( a3, b3 )

print( r1 )
print( r2 )
print( r3 )
```

main	
a1	6
b1	2
a2	7
b2	3
a3	4
b3	5
r1	True
r2	False

Module primal.py

```
def is_factor( x, y ) :

    rem = x % y
    if ( rem == 0 ) :
        result = True
    else:
        result = False

    return result


def are_relative_primes( x, y ) :

    ...
```

Program Trace – *program steps through its code*

Program factoid.py

```
import primal

a1, b1 = 6, 2
a2, b2 = 7, 3
a3, b3 = 4, 5

r1 = primal.is_factor( a1, b1 )
r2 = primal.is_factor( a2, b2 )
r3 = primal.is_factor( a3, b3 )

print( r1 )
print( r2 )
print( r3 )
```

main	
a1	6
b1	2
a2	7
b2	3
a3	4
b3	5
r1	True
r2	False
r3	???

Module primal.py

```
def is_factor( x, y ) :

    rem = x % y
    if ( rem == 0 ) :
        result = True
    else:
        result = False

    return result


def are_relative_primes( x, y ) :

    ...
```

Program Trace – *function invocation causes transfers of control*

Program factoid.py

```
import primal

a1, b1 = 6, 2
a2, b2 = 7, 3
a3, b3 = 4, 5

r1 = primal.is_factor( a1, b1 )
r2 = primal.is_factor( a2, b2 )
r3 = primal.is_factor( a3, b3 )

print( r1 )
print( r2 )
print( r3 )
```

main	
a1	6
b1	2
a2	7
b2	3
a3	4
b3	5
r1	True
r2	False
r3	???

Module primal.py

```
def is_factor( x, y ) :

    rem = x % y
    if ( rem == 0 ) :
        result = True
    else:
        result = False

    return result


def are_relative_primes( x, y ) :

    ...
```

Program Trace – *copies of arguments are passed (assigned) to parameters*

Program factoid.py

```
import primal

a1, b1 = 6, 2
a2, b2 = 7, 3
a3, b3 = 4, 5

r1 = primal.is_factor( a1, b1 )
r2 = primal.is_factor( a2, b2 )
r3 = primal.is_factor( a3, b3 )

print( r1 )
print( r2 )
print( r3 )
```

main	
a1	6
b1	2
a2	7
b2	3
a3	4
b3	5
r1	True
r2	False
r3	???

Module primal.py

```
def is_factor( x, y ) :

    rem = x % y
    if
    el
    return result

def are_relative_primes( x, y ) :

    ...
```

is_factor()	
x	
y	

Program Trace – *copies of arguments are passed (assigned) to parameters*

Program factoid.py

```
import primal

a1, b1 = 6, 2
a2, b2 = 7, 3
a3, b3 = 4, 5

r1 = primal.is_factor( a1, b1 )
r2 = primal.is_factor( a2, b2 )
r3 = primal.is_factor( a3, b3 )

print( r1 )
print( r2 )
print( r3 )
```

main	
a1	6
b1	2
a2	7
b2	3
a3	4
b3	5
r1	True
r2	False
r3	???

Module primal.py

```
def is_factor( x, y ) :

    rem = x % y
    if
    el

    return result


def are_relative_primes( x, y ) :

    ...
```

is_factor()	
x	4
y	

Program Trace – *copies of arguments are passed (assigned) to parameters*

Program factoid.py

```
import primal

a1, b1 = 6, 2
a2, b2 = 7, 3
a3, b3 = 4, 5

r1 = primal.is_factor( a1, b1 )
r2 = primal.is_factor( a2, b2 )
r3 = primal.is_factor( a3, b3 )

print( r1 )
print( r2 )
print( r3 )
```

main	
a1	6
b1	2
a2	7
b2	3
a3	4
b3	5
r1	True
r2	False
r3	False

Module primal.py

```
def is_factor( x, y ) :

    rem = x % y
    if
    el

    return result


def are_relative_primes( x, y ) :

    ...
```

is_factor()	
x	4
y	

Program Trace – *copies of arguments are passed (assigned) to parameters*

Program factoid.py

```
import primal

a1, b1 = 6, 2
a2, b2 = 7, 3
a3, b3 = 4, 5

r1 = primal.is_factor( a1, b1 )
r2 = primal.is_factor( a2, b2 )
r3 = primal.is_factor( a3, b3 )

print( r1 )
print( r2 )
print( r3 )
```

main	
a1	6
b1	2
a2	7
b2	3
a3	4
b3	5
r1	True
r2	False
r3	???

Module primal.py

```
def is_factor( x, y ) :

    rem = x % y
    if
    el
    return result


def are_relative_primes( x, y ) :

    ...
```

is_factor()	
x	4
y	5

Program Trace – *cpu control is passed to function*

Program factoid.py

```
import primal

a1, b1 = 6, 2
a2, b2 = 7, 3
a3, b3 = 4, 5

r1 = primal.is_factor( a1, b1 )
r2 = primal.is_factor( a2, b2 )
r3 = primal.is_factor( a3, b3 )

print( r1 )
print( r2 )
print( r3 )
```

main

a1	6
b1	2
a2	7
b2	3
a3	4
b3	5
r1	True
r2	True
r3	???

Module primal.py

```
def is_factor( x, y ) :

    rem = x % y
    if
    el

    return result


def are_relative_primes( x, y ) :

    ...
```

is_factor()

x	4
y	5

Program Trace – *cpu control is passed to function*

Program factoid.py

```
import primal

a1, b1 = 6, 2
a2, b2 = 7, 3
a3, b3 = 4, 5

r1 = primal.is_factor( a1, b1 )
r2 = primal.is_factor( a2, b2 )
r3 = primal.is_factor( a3, b3 )

print( r1 )
print( r2 )
print( r3 )
```

main

a1	6
b1	2
a2	7
b2	3
a3	4
b3	5
r1	True
r2	???

Module primal.py

```
def is_factor( x, y ) :

    rem = x % y
    if
    el

    return result


def are_relative_primes( x, y ) :

    ...
```

is_factor()

x	4
y	5

Program Trace – *function executes*

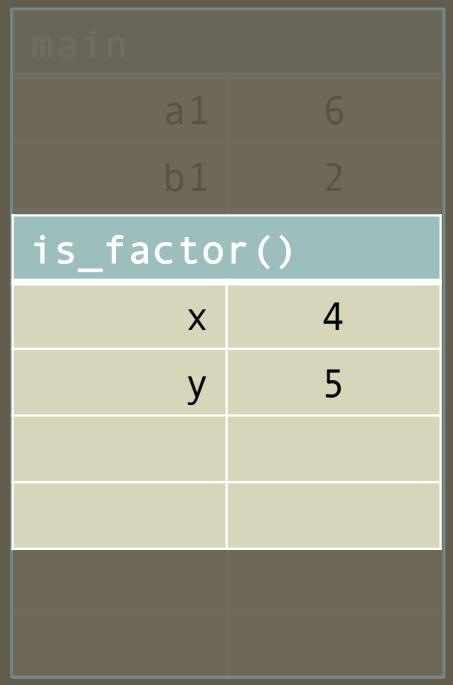
Program factoid.py

```
import primal

a1, b1 = 6, 2
a2, b2 = 7, 3
a3, b3 = 4, 5

r1 = primal.is_factor( a1, b1 )
r2 = primal.is_factor( a2, b2 )
r3 = primal.is_factor( a3, b3 )

print( r1 )
print( r2 )
print( r3 )
```



Module primal.py

```
def is_factor( x, y ) :

    rem = x % y
    if ( rem == 0 ) :
        result = True
    else:
        result = False

    return result


def are_relative_primes( x, y ) :

    ...
```

Program Trace – *function executes*

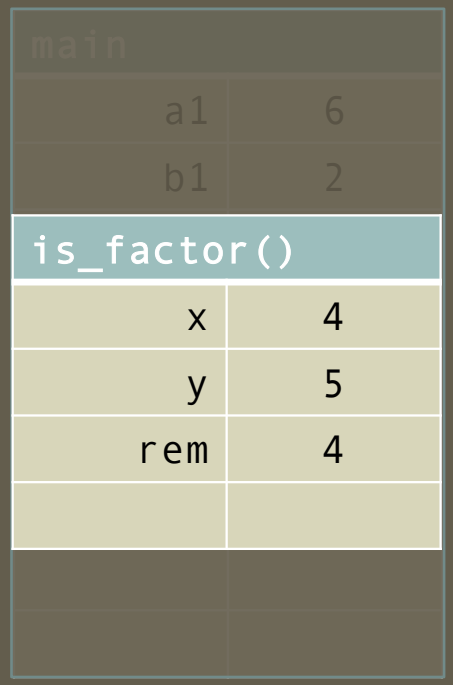
Program factoid.py

```
import primal

a1, b1 = 6, 2
a2, b2 = 7, 3
a3, b3 = 4, 5

r1 = primal.is_factor( a1, b1 )
r2 = primal.is_factor( a2, b2 )
r3 = primal.is_factor( a3, b3 )

print( r1 )
print( r2 )
print( r3 )
```



Module primal.py

```
def is_factor( x, y ) :

    rem = x % y
    if ( rem == 0 ) :
        result = True
    else:
        result = False

    return result


def are_relative_primes( x, y ) :

    ...
```

Program Trace – *function executes*

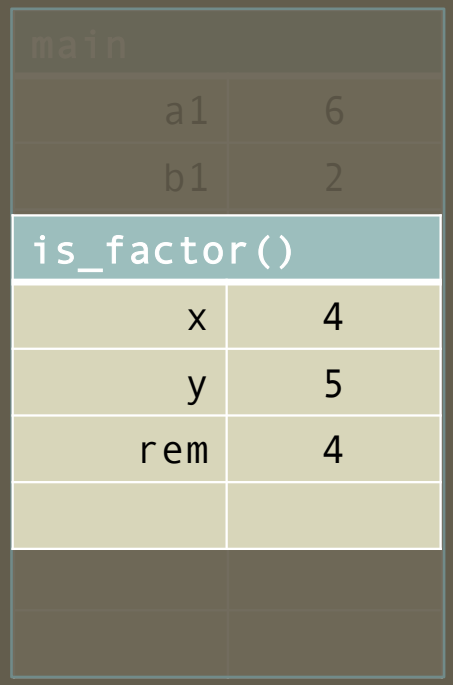
Program factoid.py

```
import primal

a1, b1 = 6, 2
a2, b2 = 7, 3
a3, b3 = 4, 5

r1 = primal.is_factor( a1, b1 )
r2 = primal.is_factor( a2, b2 )
r3 = primal.is_factor( a3, b3 )

print( r1 )
print( r2 )
print( r3 )
```



Module primal.py

```
def is_factor( x, y ) :

    rem = x % y
    if ( rem == 0 ) :
        result = True
    else:
        result = False

    return result


def are_relative_primes( x, y ) :

    ...
```

Program Trace – *function executes*

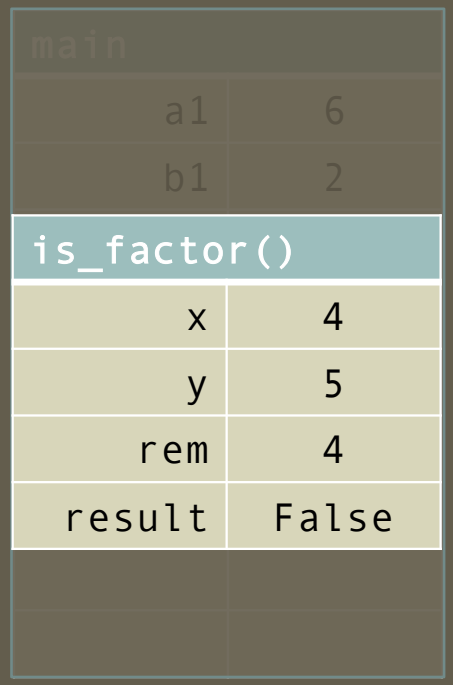
Program factoid.py

```
import primal

a1, b1 = 6, 2
a2, b2 = 7, 3
a3, b3 = 4, 5

r1 = primal.is_factor( a1, b1 )
r2 = primal.is_factor( a2, b2 )
r3 = primal.is_factor( a3, b3 )

print( r1 )
print( r2 )
print( r3 )
```



Module primal.py

```
def is_factor( x, y ) :

    rem = x % y
    if ( rem == 0 ) :
        result = True
    else:
        result = False

    return result


def are_relative_primes( x, y ) :

    ...
```

Program Trace – *function executes*

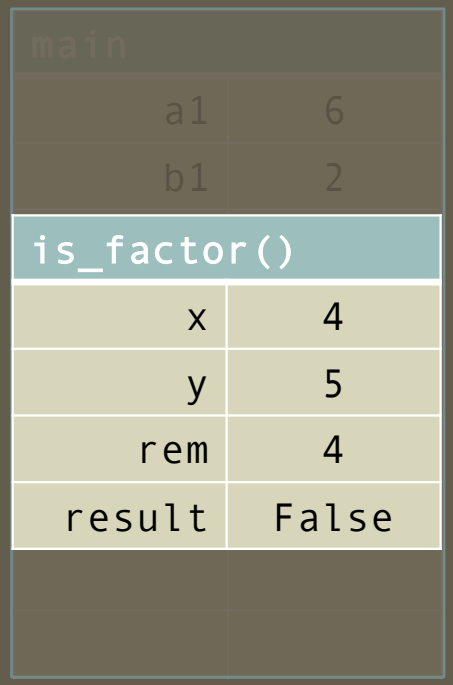
Program factoid.py

```
import primal

a1, b1 = 6, 2
a2, b2 = 7, 3
a3, b3 = 4, 5

r1 = primal.is_factor( a1, b1 )
r2 = primal.is_factor( a2, b2 )
r3 = primal.is_factor( a3, b3 )

print( r1 )
print( r2 )
print( r3 )
```



Module primal.py

```
def is_factor( x, y ) :

    rem = x % y
    if ( rem == 0 ) :
        result = True
    else:
        result = False

    return result


def are_relative_primes( x, y ) :

    ...
```

Program Trace – *return expression is the value of the invocation*

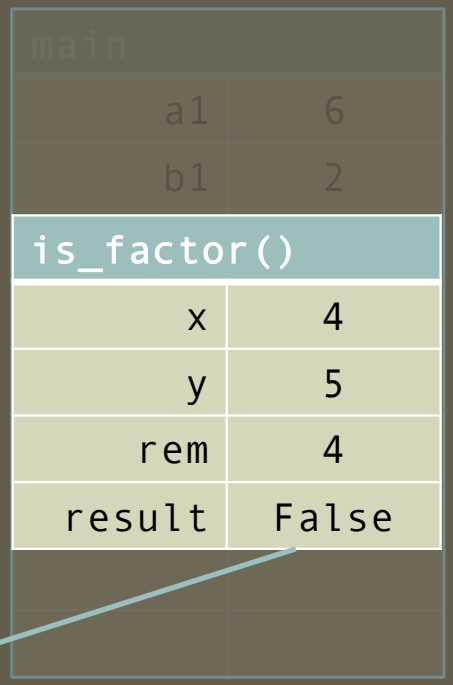
Program factoid.py

```
import primal

a1, b1 = 6, 2
a2, b2 = 7, 3
a3, b3 = 4, 5

r1 = primal.is_factor( a1, b1 )
r2 = primal.is_factor( a2, b2 )
r3 = False

print( r1 )
print( r2 )
print( r3 )
```



Module primal.py

```
def is_factor( x, y ) :

    rem = x % y
    if ( rem == 0 ) :
        result = True
    else:
        result = False

    return result


def are_relative_primes( x, y ) :

    ...
```

Program Trace – *cpu control is passed back to program*

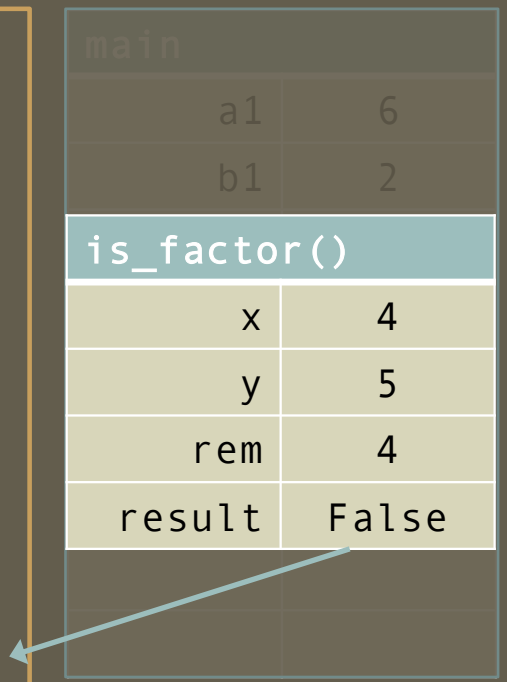
Program factoid.py

```
import primal

a1, b1 = 6, 2
a2, b2 = 7, 3
a3, b3 = 4, 5

r1 = primal.is_factor( a1, b1 )
r2 = primal.is_factor( a2, b2 )
r3 = False

print( r1 )
print( r2 )
print( r3 )
```



Module primal.py

```
def is_factor( x, y ) :

    rem = x % y
    if ( rem == 0 ) :
        result = True
    else:
        result = False

    return result


def are_relative_primes( x, y ) :

    ...
```

Program Trace – *cpu control is passed back to program*

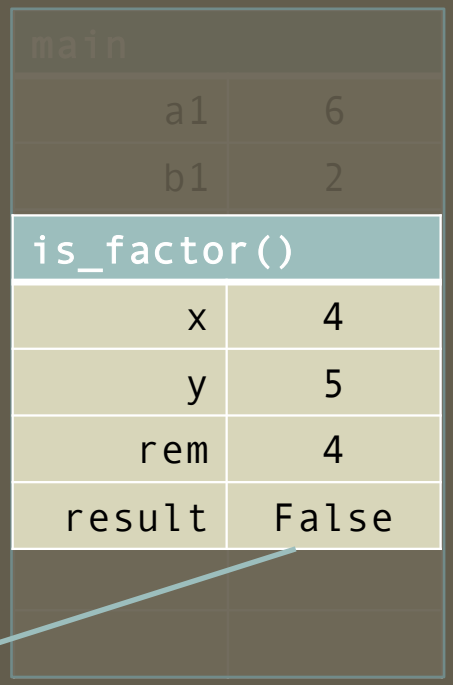
Program factoid.py

```
import primal

a1, b1 = 6, 2
a2, b2 = 7, 3
a3, b3 = 4, 5

r1 = primal.is_factor( a1, b1 )
r2 = primal.is_factor( a2, b2 )
r3 = primal.is_factor( a3, b3 )

print( r1 )
print( r2 )
print( r3 )
```



Module primal.py

```
def is_factor( x, y ) :

    rem = x % y
    if ( rem == 0 ) :
        result = True
    else:
        result = False

    return result


def are_relative_primes( x, y ) :

    ...
```

Program Trace – *cpu control is passed back to program*

Program factoid.py

```
import primal

a1, b1 = 6, 2
a2, b2 = 7, 3
a3, b3 = 4, 5

r1 = primal.is_factor( a1, b1 )
r2 = primal.is_factor( a2, b2 )
r3 = primal.is_factor( a3, b3 )

print( r1 )
print( r2 )
print( r3 )
```

main	
a1	6
b1	2
a2	7
b2	3
a3	4
b3	5
r1	True
r2	False
r3	False

Module primal.py

```
def is_factor( x, y ) :

    rem = x % y
    if ( rem == 0 ) :
        result = True
    else:
        result = False

    return result


def are_relative_primes( x, y ) :

    ...
```

Program Trace – *program steps through its code*

Program factoid.py

```
import primal

a1, b1 = 6, 2
a2, b2 = 7, 3
a3, b3 = 4, 5

r1 = primal.is_factor( a1, b1 )
r2 = primal.is_factor( a2, b2 )
r3 = primal.is_factor( a3, b3 )

print( r1 )
print( r2 )
print( r3 )
```

main	
a1	6
b1	2
a2	7
b2	3
a3	4
b3	5
r1	True
r2	False
r3	False

Module primal.py

```
def is_factor( x, y ) :

    rem = x % y
    if ( rem == 0 ) :
        result = True
    else:
        result = False

    return result


def are_relative_primes( x, y ) :

    ...
```

Program Trace – *cpu control is passed to function*

Program factoid.py

```
import primal

a1, b1 = 6, 2
a2, b2 = 7, 3
a3, b3 = 4, 5

r1 = primal.is_factor( a1, b1 )
r2 = primal.is_factor( a2, b2 )
r3 = primal.is_factor( a3, b3 )

print( r1 )
print( r2 )
print( r3 )
```

main	
a1	6
b1	2
a2	7
b2	3
a3	4
b3	5
r1	True
r2	False
r3	False

Module primal.py

```
def is_factor( x, y ) :

    rem = x % y
    if
    el
    return result


def are_relative_primes( x, y ) :

    ...
```

print()	
???	???
???	True
???	???
???	???

Program Trace – *cpu control is passed to function*

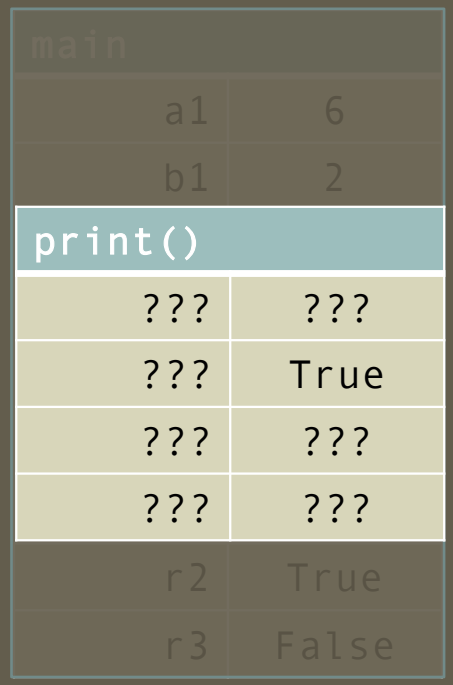
Program factoid.py

```
import primal

a1, b1 = 6, 2
a2, b2 = 7, 3
a3, b3 = 4, 5

r1 = primal.is_factor( a1, b1 )
r2 = primal.is_factor( a2, b2 )
r3 = primal.is_factor( a3, b3 )

print( r1 )
print( r2 )
print( r3 )
```




```
True
```

Module primal.py

```
def is_factor( x, y ) :

    rem = x % y
    if ( rem == 0 ) :
        result = True
    else:
        result = False

    return result


def are_relative_primes( x, y ) :

    ...
```

Program Trace – *cpu control is passed to function*

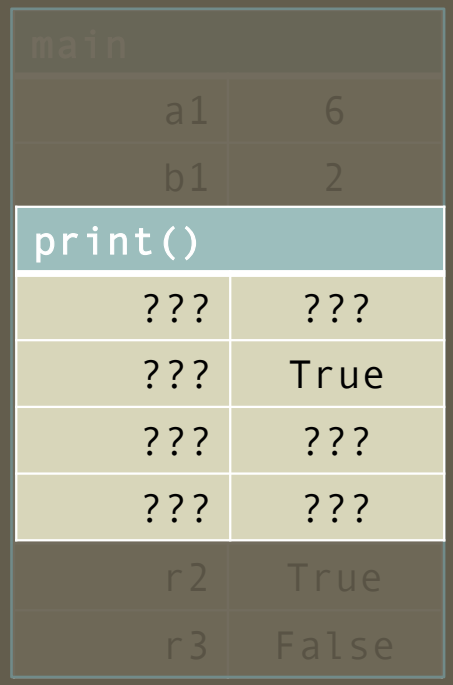
Program factoid.py

```
import primal

a1, b1 = 6, 2
a2, b2 = 7, 3
a3, b3 = 4, 5

r1 = primal.is_factor( a1, b1 )
r2 = primal.is_factor( a2, b2 )
r3 = primal.is_factor( a3, b3 )

print( r1 )
print( r2 )
print( r3 )
```




```
True
```

Module primal.py

```
def is_factor( x, y ) :

    rem = x % y
    if ( rem == 0 ) :
        result = True
    else:
        result = False

    return result


def are_relative_primes( x, y ) :

    ...
```

Program Trace – *program steps through its code*

Program factoid.py

```
import primal

a1, b1 = 6, 2
a2, b2 = 7, 3
a3, b3 = 4, 5

r1 = primal.is_factor( a1, b1 )
r2 = primal.is_factor( a2, b2 )
r3 = primal.is_factor( a3, b3 )


print( r1 )
print( r2 )
print( r3 )
```

main	
a1	6
b1	2
a2	7
b2	3
a3	4
b3	5
r1	True
r2	False
r3	False

```
True
```

Module primal.py

```
def is_factor( x, y ) :

    rem = x % y
    if ( rem == 0 ) :
        result = True
    else:
        result = False

    return result


def are_relative_primes( x, y ) :

    ...
```

Program Trace – *cpu control is passed to function*

Program factoid.py

```
import primal

a1, b1 = 6, 2
a2, b2 = 7, 3
a3, b3 = 4, 5

r1 = primal.is_factor( a1, b1 )
r2 = primal.is_factor( a2, b2 )
r3 = primal.is_factor( a3, b3 )

print( r1 )
print( r2 )
print( r3 )
```

main	
a1	6
b1	2
a2	7
b2	3
a3	4
b3	5
r1	True
r2	False
r3	False

True

Module primal.py

```
def is_factor( x, y ) :

    rem = x % y
    if
    el
    return result


def are_relative_primes( x, y ) :

    ...
```

print()	
???	???
???	???
???	False
???	???

Program Trace – *cpu control is passed to function*

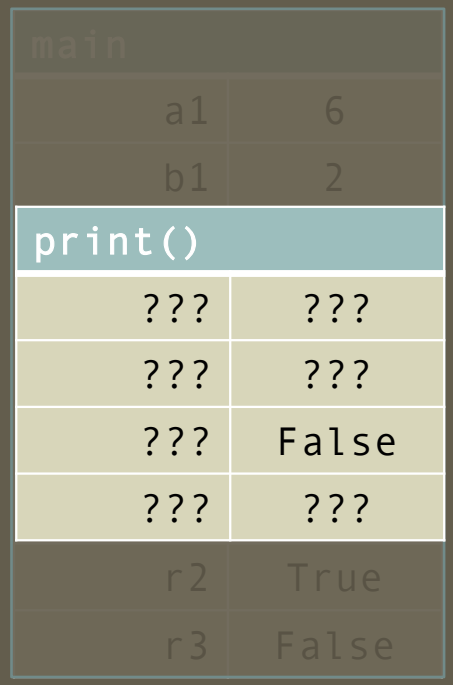
Program factoid.py

```
import primal

a1, b1 = 6, 2
a2, b2 = 7, 3
a3, b3 = 4, 5

r1 = primal.is_factor( a1, b1 )
r2 = primal.is_factor( a2, b2 )
r3 = primal.is_factor( a3, b3 )

print( r1 )
print( r2 )
print( r3 )
```




```
True
False
```

Module primal.py

```
def is_factor( x, y ) :

    rem = x % y
    if ( rem == 0 ) :
        result = True
    else:
        result = False

    return result


def are_relative_primes( x, y ) :

    ...
```

Program Trace – *cpu control is passed to function*

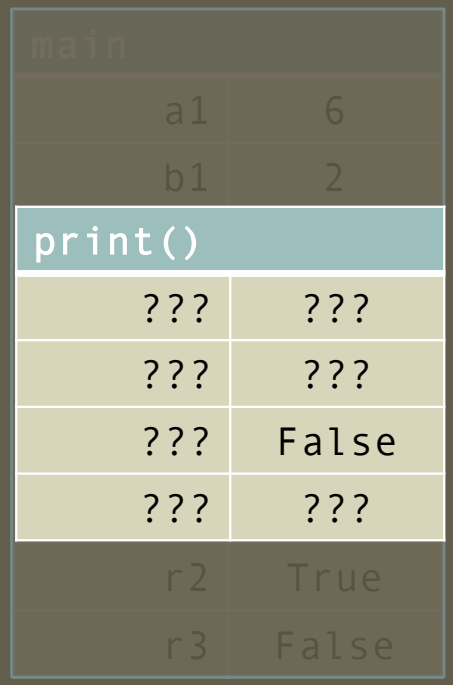
Program factoid.py

```
import primal

a1, b1 = 6, 2
a2, b2 = 7, 3
a3, b3 = 4, 5

r1 = primal.is_factor( a1, b1 )
r2 = primal.is_factor( a2, b2 )
r3 = primal.is_factor( a3, b3 )

print( r1 )
print( r2 )
print( r3 )
```




```
True
False
```

Module primal.py

```
def is_factor( x, y ) :

    rem = x % y
    if ( rem == 0 ) :
        result = True
    else:
        result = False

    return result


def are_relative_primes( x, y ) :

    ...
```

Program Trace – *program steps through its code*

Program factoid.py

```
import primal

a1, b1 = 6, 2
a2, b2 = 7, 3
a3, b3 = 4, 5

r1 = primal.is_factor( a1, b1 )
r2 = primal.is_factor( a2, b2 )
r3 = primal.is_factor( a3, b3 )


print( r1 )
print( r2 )
print( r3 )
```

main	
a1	6
b1	2
a2	7
b2	3
a3	4
b3	5
r1	True
r2	False
r3	False

```
True
False
```

Module primal.py

```
def is_factor( x, y ) :

    rem = x % y
    if ( rem == 0 ) :
        result = True
    else:
        result = False

    return result


def are_relative_primes( x, y ) :

    ...
```

Program Trace – *cpu control is passed to function*

Program factoid.py

```
import primal

a1, b1 = 6, 2
a2, b2 = 7, 3
a3, b3 = 4, 5

r1 = primal.is_factor( a1, b1 )
r2 = primal.is_factor( a2, b2 )
r3 = primal.is_factor( a3, b3 )

print( r1 )
print( r2 )
print( r3 )
```

main	
a1	6
b1	2
a2	7
b2	3
a3	4
b3	5
r1	True
r2	False
r3	False

```
True
False
```

Module primal.py

```
def is_factor( x, y ) :

    rem = x % y
    if
    el
    return result


def are_relative_primes( x, y ) :

    ...
```

print()	
???	False
???	???
???	???
???	???

Program Trace – *cpu control is passed to function*

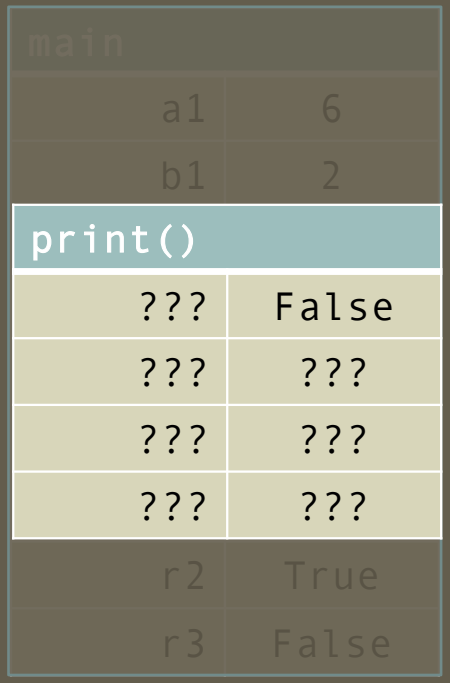
Program factoid.py

```
import primal

a1, b1 = 6, 2
a2, b2 = 7, 3
a3, b3 = 4, 5

r1 = primal.is_factor( a1, b1 )
r2 = primal.is_factor( a2, b2 )
r3 = primal.is_factor( a3, b3 )

print( r1 )
print( r2 )
print( r3 )
```




```
True
False
False
```

Module primal.py

```
def is_factor( x, y ) :

    rem = x % y
    if ( rem == 0 ) :
        result = True
    else:
        result = False

    return result


def are_relative_primes( x, y ) :

    ...
```

Program Trace – *cpu control is passed to function*

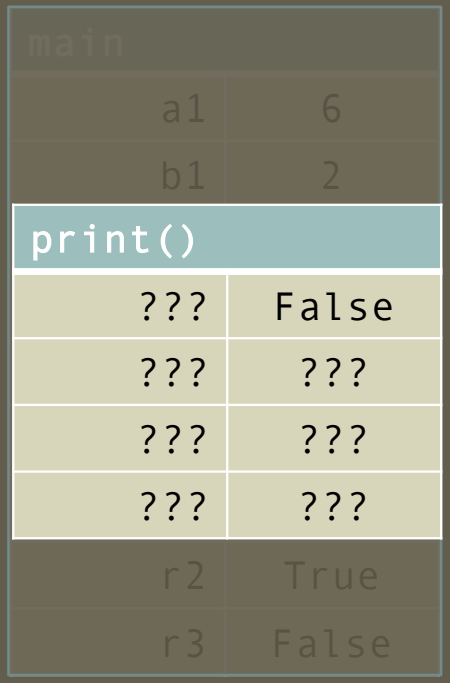
Program factoid.py

```
import primal

a1, b1 = 6, 2
a2, b2 = 7, 3
a3, b3 = 4, 5

r1 = primal.is_factor( a1, b1 )
r2 = primal.is_factor( a2, b2 )
r3 = primal.is_factor( a3, b3 )

print( r1 )
print( r2 )
print( r3 )
```




```
True
False
False
```

Module primal.py

```
def is_factor( x, y ) :

    rem = x % y
    if ( rem == 0 ) :
        result = True
    else:
        result = False

    return result


def are_relative_primes( x, y ) :

    ...
```

Program Trace – *program steps through its code*

Program factoid.py

```
import primal

a1, b1 = 6, 2
a2, b2 = 7, 3
a3, b3 = 4, 5

r1 = primal.is_factor( a1, b1 )
r2 = primal.is_factor( a2, b2 )
r3 = primal.is_factor( a3, b3 )


print( r1 )
print( r2 )
print( r3 )
```

main	
a1	6
b1	2
a2	7
b2	3
a3	4
b3	5
r1	True
r2	False
r3	False

```
True
False
False
```

Module primal.py

```
def is_factor( x, y ) :

    rem = x % y
    if ( rem == 0 ) :
        result = True
    else:
        result = False

    return result


def are_relative_primes( x, y ) :

    ...
```

Program Trace – *program steps through its code*

Program factoid.py

```
import primal

a1, b1 = 6, 2
a2, b2 = 7, 3
a3, b3 = 4, 5

r1 = primal.is_factor( a1, b1 )
r2 = primal.is_factor( a2, b2 )
r3 = primal.is_factor( a3, b3 )


print( r1 )
print( r2 )
print( r3 )
```

```
True
False
False
```

Module primal.py

```
def is_factor( x, y ) :

    rem = x % y
    if ( rem == 0 ) :
        result = True
    else:
        result = False

    return result


def are_relative_primes( x, y ) :

    ...
```

Program Trace – *program steps through its code*

```
True
False
False
```