

[320] OOP and Recursion

Department of Computer Sciences
University of Wisconsin-Madison

1. the parent class of Dog is Pet. Does Pet have a parent type? If so, what is it?
2. how many arguments does line C pass?
3. how many arguments does line B pass?
4. on another paper, draw what the frames and object(s) will look like after line A.
(check with PythonTutor)

```
class Pet:
    def __init__(self, name):
        self.name = name # A

class Dog(Pet):
    def __init__(self, name, age):
        self.age = age
        Pet.__init__(self, name) # B

pup = Dog("Sam", 1) # C
```

1. the parent class of Dog is Pet. Does Pet have a parent type? If so, what is it?

object

2. how many arguments does line C pass?

3. how many arguments does line B pass?

4. on another paper, draw what the frames and object(s) will look like after line A.
(check with PythonTutor)

```
class Pet:
    def __init__(self, name):
        self.name = name # A

class Dog(Pet):
    def __init__(self, name, age):
        self.age = age
        Pet.__init__(self, name) # B

pup = Dog("Sam", 1) # C
```

1. the parent class of Dog is Pet. Does Pet have a parent type? If so, what is it?

object

2. how many arguments does line C pass?

3

3. how many arguments does line B pass?

4. on another paper, draw what the frames and object(s) will look like after line A.
(check with PythonTutor)

```
class Pet:
    def __init__(self, name):
        self.name = name # A

class Dog(Pet):
    def __init__(self, name, age):
        self.age = age
        Pet.__init__(self, name) # B

pup = Dog("Sam", 1) # C
```

1. the parent class of Dog is Pet. Does Pet have a parent type? If so, what is it?

object

2. how many arguments does line C pass?

3

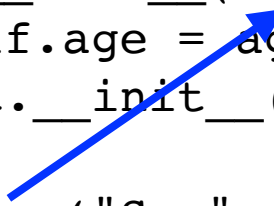
3. how many arguments does line B pass?

4. on another paper, draw what the frames and object(s) will look like after line A.
(check with PythonTutor)

```
class Pet:
    def __init__(self, name):
        self.name = name # A

class Dog(Pet):
    def __init__(self, name, age):
        self.age = age
        Pet.__init__(self, name) # B

pup = Dog("Sam", 1) # C
```



1. the parent class of Dog is Pet. Does Pet have a parent type? If so, what is it?

object

2. how many arguments does line C pass?

3

3. how many arguments does line B pass?

4. on another paper, draw what the frames and object(s) will look like after line A.
(check with PythonTutor)

```
class Pet:
    def __init__(self, name):
        self.name = name # A

class Dog(Pet):
    def __init__(self, name, age):
        self.age = age
        Pet.__init__(self, name) # B

pup = Dog("Sam", 1) # C
```

1. the parent class of Dog is Pet. Does Pet have a parent type? If so, what is it?

object

2. how many arguments does line C pass?

3

3. how many arguments does line B pass?

4. on another paper, draw what the frames and object(s) will look like after line A.
(check with PythonTutor)

```
class Pet:
    def __init__(self, name):
        self.name = name # A

class Dog(Pet):
    def __init__(self, name, age):
        self.age = age
        Pet.__init__(self, name) # B

pup = Dog("Sam", 1) # C
```

1. the parent class of Dog is Pet. Does Pet have a parent type? If so, what is it?

object

2. how many arguments does line C pass?

3

3. how many arguments does line B pass?

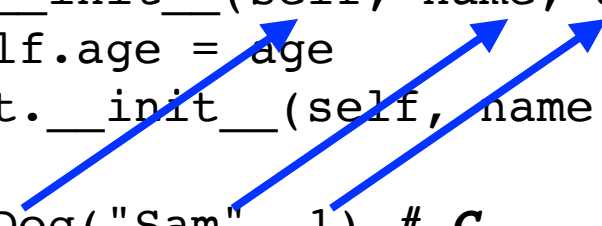
2

4. on another paper, draw what the frames and object(s) will look like after line A.
(check with PythonTutor)

```
class Pet:
    def __init__(self, name):
        self.name = name # A

class Dog(Pet):
    def __init__(self, name, age):
        self.age = age
        Pet.__init__(self, name) # B

pup = Dog("Sam", 1) # C
```



1. the parent class of Dog is Pet. Does Pet have a parent type? If so, what is it?

object

2. how many arguments does line C pass?

3

3. how many arguments does line B pass?

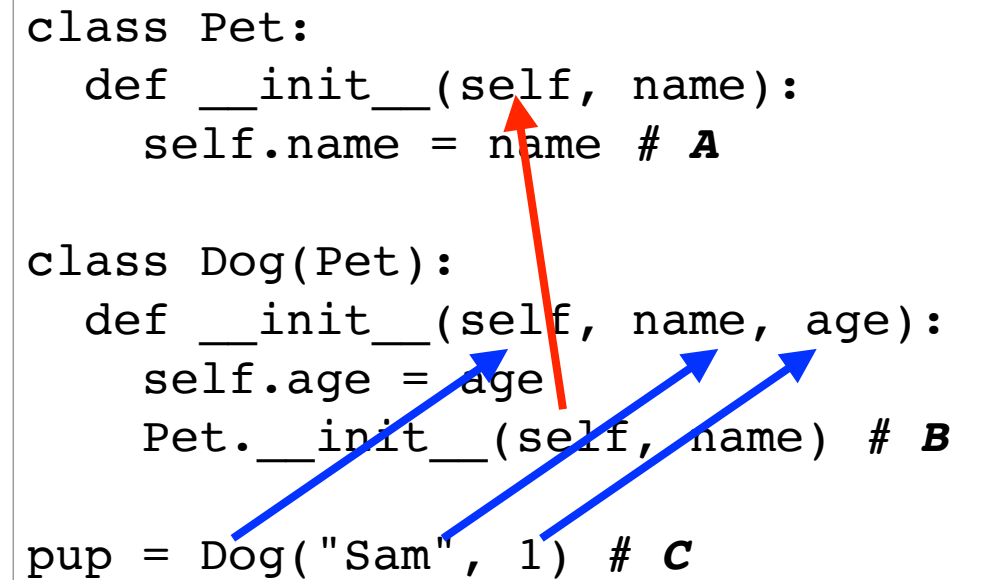
2

4. on another paper, draw what the frames and object(s) will look like after line A.
(check with PythonTutor)

```
class Pet:
    def __init__(self, name):
        self.name = name # A

class Dog(Pet):
    def __init__(self, name, age):
        self.age = age
        Pet.__init__(self, name) # B

pup = Dog("Sam", 1) # C
```



1. the parent class of Dog is Pet. Does Pet have a parent type? If so, what is it?

object

2. how many arguments does line C pass?

3

3. how many arguments does line B pass?

2

4. on another paper, draw what the frames and object(s) will look like after line A.
(check with PythonTutor)

```
class Pet:
    def __init__(self, name):
        self.name = name # A

class Dog(Pet):
    def __init__(self, name, age):
        self.age = age
        Pet.__init__(self, name) # B

pup = Dog("Sam", 1) # C
```

1. the parent class of Dog is Pet. Does Pet have a parent type? If so, what is it?

object

2. how many arguments does line C pass?

3

3. how many arguments does line B pass?

2

4. on another paper, draw what the frames and object(s) will look like after line A.
(check with PythonTutor)

Demo with Python Tutor

```
class Pet:
    def __init__(self, name):
        self.name = name # A

class Dog(Pet):
    def __init__(self, name, age):
        self.age = age
        Pet.__init__(self, name) # B

pup = Dog("Sam", 1) # C
```

The diagram illustrates argument passing in the provided Python code. Red arrows indicate the flow of arguments for the `Pet.__init__` method: one from the `name` parameter in line A to the `self` parameter in line B, and another from the `name` parameter in line B to the `name` parameter in line A. Blue arrows indicate the flow of arguments for the `Dog.__init__` method: one from the `name` parameter in line C to the `self` parameter in line B, and another from the `age` parameter in line C to the `age` parameter in line B. A third blue arrow points from the `Pet.__init__` call in line B to the `Pet` class name in line B.

Frames

Objects

```
class Pet:
    def __init__(self, name):
        self.name = name # A

class Dog(Pet):
    def __init__(self, name, age):
        self.age = age
        Pet.__init__(self, name) # B

pup = Dog("Sam", 1) # C
```

1

Arrows from Pet and Dog are not drawn to keep the figure clean.

Frames



Objects

```
class Pet:
    def __init__(self, name):
        self.name = name # A

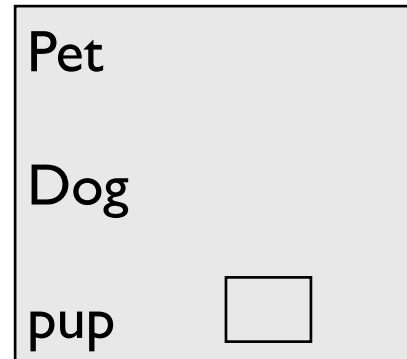
class Dog(Pet):
    def __init__(self, name, age):
        self.age = age
        Pet.__init__(self, name) # B

pup = Dog("Sam", 1) # C
```

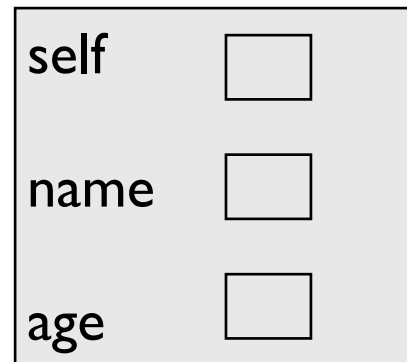
1

Arrows from Pet and Dog are not drawn to keep the figure clean.

Frames



Dog.__init__



Objects

age	
name	

```
class Pet:
    def __init__(self, name):
        self.name = name # A

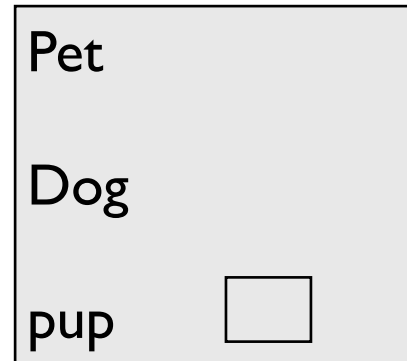
class Dog(Pet):
    def __init__(self, name, age):
        self.age = age
        Pet.__init__(self, name) # B

pup = Dog("Sam", 1) # C
```

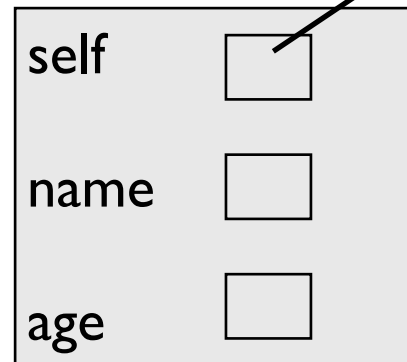
1

Arrows from Pet and Dog are not drawn to keep the figure clean.

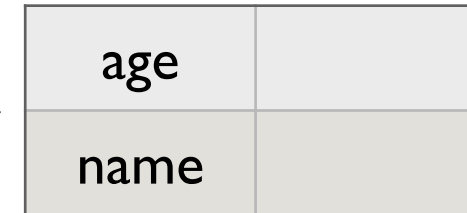
Frames



Dog.__init__



Objects



```
class Pet:
    def __init__(self, name):
        self.name = name # A

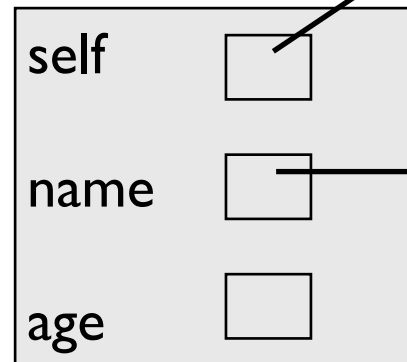
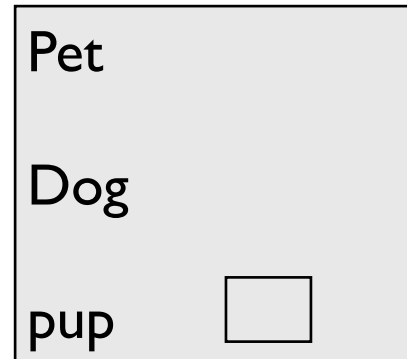
class Dog(Pet):
    def __init__(self, name, age):
        self.age = age
        Pet.__init__(self, name) # B

pup = Dog("Sam", 1) # C
```

1

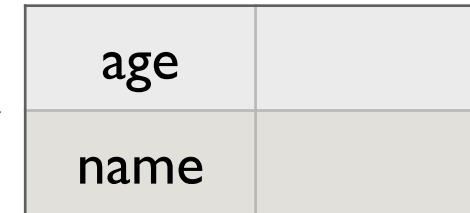
Arrows from Pet and Dog are not drawn to keep the figure clean.

Frames



Dog.__init__

Objects



"Sam"

```
class Pet:
    def __init__(self, name):
        self.name = name # A

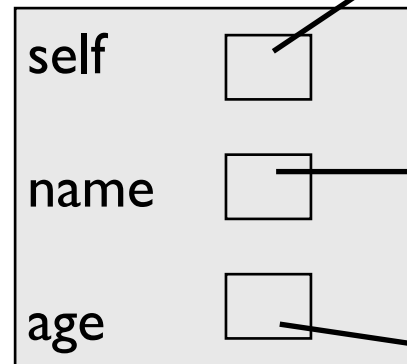
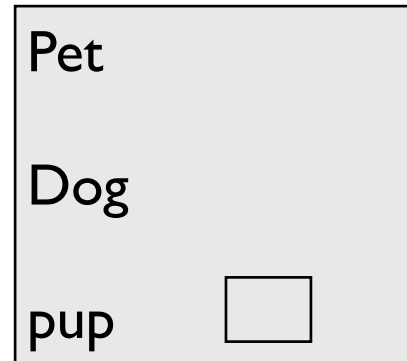
class Dog(Pet):
    def __init__(self, name, age):
        self.age = age
        Pet.__init__(self, name) # B

pup = Dog("Sam", 1) # C
```

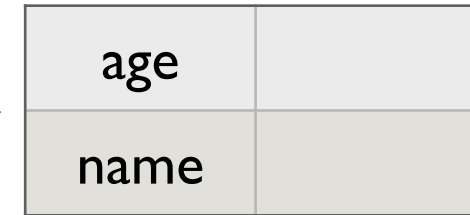

1

Arrows from Pet and Dog are not drawn to keep the figure clean.

Frames



Objects



Dog.__init__

"Sam"

|

```
class Pet:
    def __init__(self, name):
        self.name = name # A

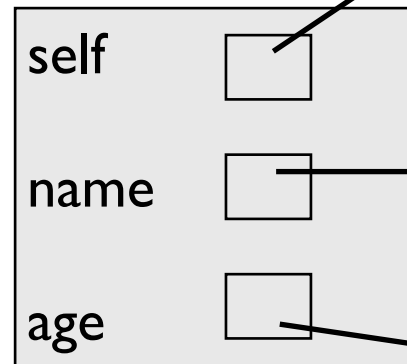
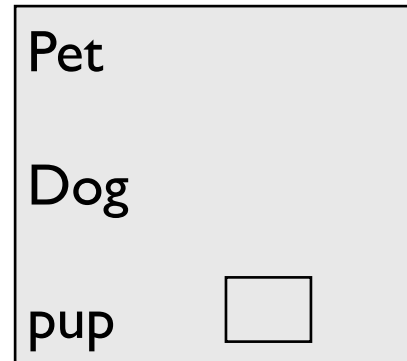
class Dog(Pet):
    def __init__(self, name, age):
        self.age = age
        Pet.__init__(self, name) # B

pup = Dog("Sam", 1) # C
```

1

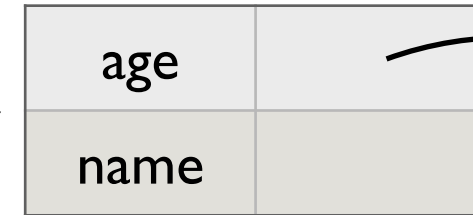
Arrows from Pet and Dog are not drawn to keep the figure clean.

Frames



Dog.__init__

Objects



"Sam"

|

```
class Pet:
    def __init__(self, name):
        self.name = name # A

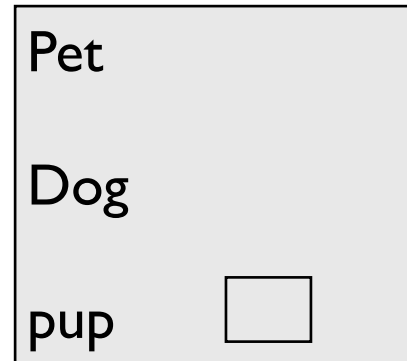
class Dog(Pet):
    def __init__(self, name, age):
        self.age = age
        Pet.__init__(self, name) # B

pup = Dog("Sam", 1) # C
```

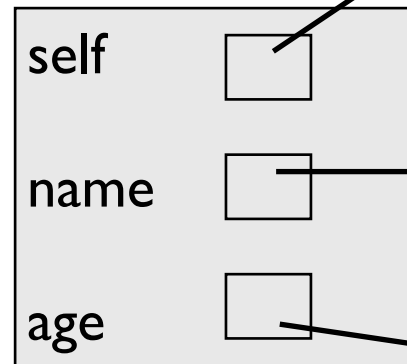
1

Arrows from Pet and Dog are not drawn to keep the figure clean.

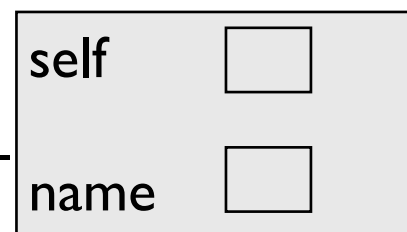
Frames



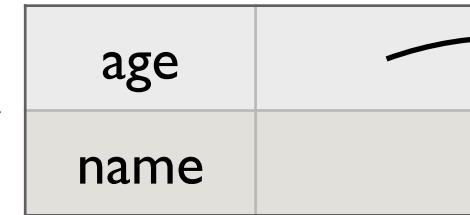
Dog.__init__



Pet.__init__



Objects



"Sam"

|

```
class Pet:
    def __init__(self, name):
        self.name = name # A

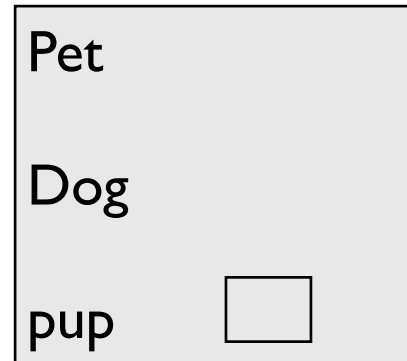
class Dog(Pet):
    def __init__(self, name, age):
        self.age = age
        Pet.__init__(self, name) # B

pup = Dog("Sam", 1) # C
```

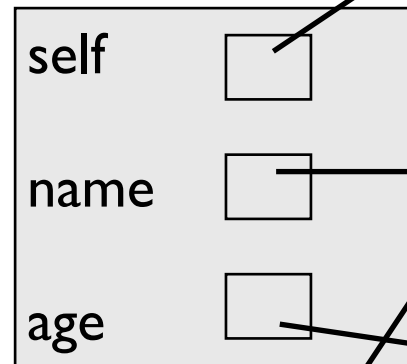
1

Arrows from Pet and Dog are not drawn to keep the figure clean.

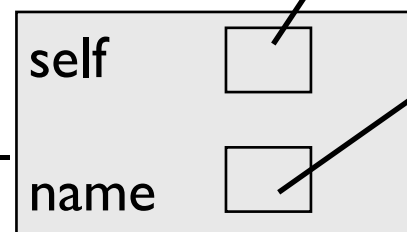
Frames



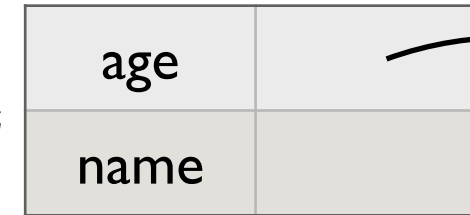
Dog.__init__



Pet.__init__



Objects



"Sam"

|

```
class Pet:
    def __init__(self, name):
        self.name = name # A

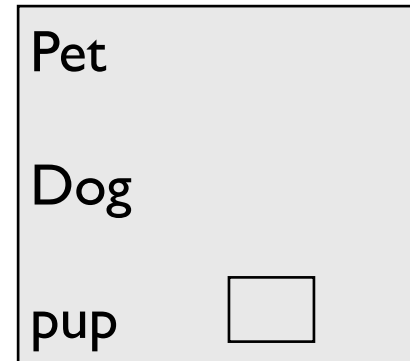
class Dog(Pet):
    def __init__(self, name, age):
        self.age = age
        Pet.__init__(self, name) # B

pup = Dog("Sam", 1) # C
```

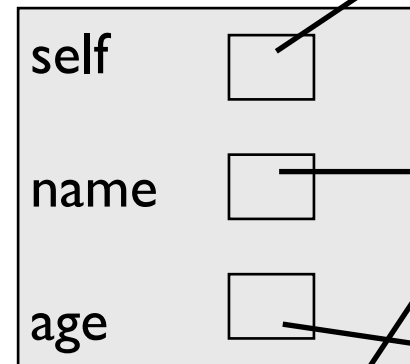
1

Arrows from Pet and Dog are not drawn to keep the figure clean.

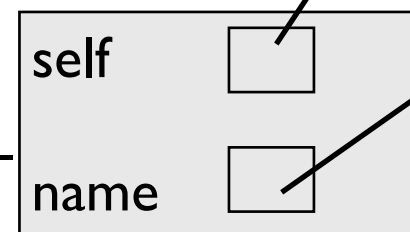
Frames



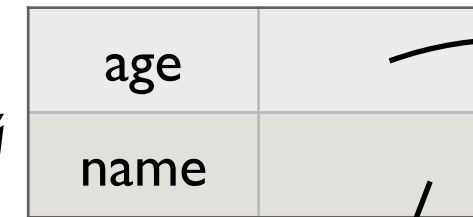
Dog.__init__



Pet.__init__



Objects



"Sam"

|

```
class Pet:
    def __init__(self, name):
        self.name = name # A

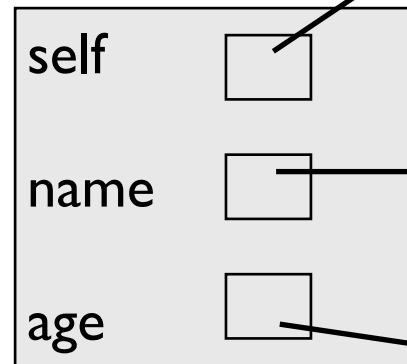
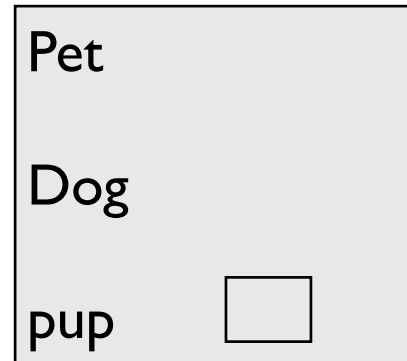
class Dog(Pet):
    def __init__(self, name, age):
        self.age = age
        Pet.__init__(self, name) # B

pup = Dog("Sam", 1) # C
```

1

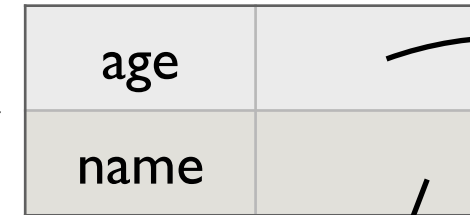
Arrows from Pet and Dog are not drawn to keep the figure clean.

Frames



Dog.__init__

Objects



"Sam"

|

```
class Pet:
    def __init__(self, name):
        self.name = name # A

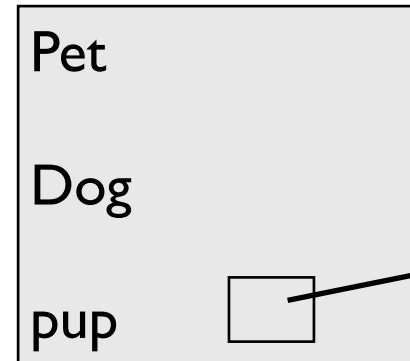
class Dog(Pet):
    def __init__(self, name, age):
        self.age = age
        Pet.__init__(self, name) # B

pup = Dog("Sam", 1) # C
```

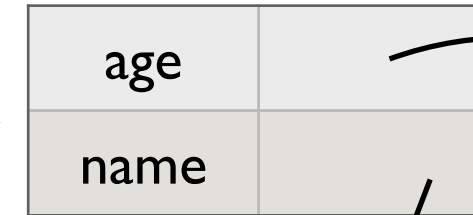
1

Arrows from Pet and Dog are not drawn to keep the figure clean.

Frames



Objects



"Sam"

|

```
class Pet:
    def __init__(self, name):
        self.name = name # A

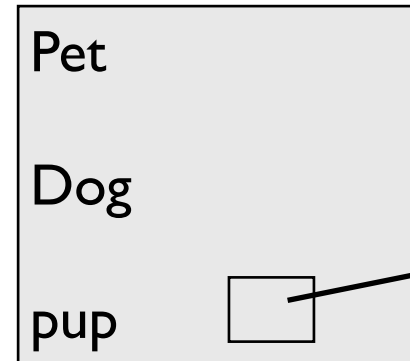
class Dog(Pet):
    def __init__(self, name, age):
        self.age = age
        Pet.__init__(self, name) # B

pup = Dog("Sam", 1) # C
```

1

Arrows from Pet and Dog are not drawn to keep the figure clean.

Frames



Objects

age	"Sam"
name	I

```
class Pet:
    def __init__(self, name):
        self.name = name # A

class Dog(Pet):
    def __init__(self, name, age):
        self.age = age
        Pet.__init__(self, name) # B

pup = Dog("Sam", 1) # C
```


2

```
def fact(n):  
    if n == 0:  
        return 1  
    return n * fact(n-1)  
  
# what is fact(5)
```

```
def fib(n):  
    if n < 2:  
        return n  
    return fib(n-1) + fib(n-2)  
  
# what is fib(6)?
```

2 a

```
def fact(n):  
    if n == 0:  
        return 1  
    return n * fact(n-1)  
  
# what is fact(5)
```

2 a

```
def fact(n):  
    if n == 0:  
        return 1  
    return n * fact(n-1)  
  
# what is fact(5)
```

5!

2 a

```
def fact(n):  
    if n == 0:  
        return 1  
    return n * fact(n-1)  
  
# what is fact(5)
```

$$5! = 5 * 4 * 3 * 2 * 1$$

2 a

```
def fact(n):  
    if n == 0:  
        return 1  
    return n * fact(n-1)  
  
# what is fact(5)
```

$$\begin{aligned} 5! &= 5 * 4 * 3 * 2 * 1 \\ &= 5 * 4! \end{aligned}$$

2 a

```
def fact(n):  
    if n == 0:  
        return 1  
    return n * fact(n-1)  
  
# what is fact(5)
```

$$\begin{aligned}5! &= 5 * 4 * 3 * 2 * 1 \\ &= 5 * 4! \\ 4! &= 4 * 3!\end{aligned}$$

2 a

```
def fact(n):  
    if n == 0:  
        return 1  
    return n * fact(n-1)  
  
# what is fact(5)
```

$$\begin{aligned}5! &= 5 * 4 * 3 * 2 * 1 \\ &= 5 * 4! \\ 4! &= 4 * 3! \\ &\dots\end{aligned}$$

2 a

```
def fact(n):  
    if n == 0:  
        return 1  
    return n * fact(n-1)  
  
# what is fact(5)
```

$$5! = 5 * 4 * 3 * 2 * 1$$

$$= 5 * 4!$$

$$4! = 4 * 3!$$

...

In general, $N! = N * (N-1)!$

2 a

```
def fact(n):  
    if n == 0:  
        return 1  
    return n * fact(n-1)  
  
# what is fact(5)
```

$$\begin{aligned}5! &= 5 * 4 * 3 * 2 * 1 \\ &= 5 * 4! \\ 4! &= 4 * 3! \\ &\dots\end{aligned}$$

In general, $N! = N * (N-1)!$
And $\text{fact}(N) = N!$

2 a

```
def fact(n):  
    if n == 0:  
        return 1  
    return n * fact(n-1)  
  
# what is fact(5)
```

$$\begin{aligned}5! &= 5 * 4 * 3 * 2 * 1 \\ &= 5 * 4! \\ 4! &= 4 * 3! \\ &\dots\end{aligned}$$

In general, $N! = N * (N-1)!$
And $\text{fact}(N) = N!$

`fact(5)`

2 a

```
def fact(n):  
    if n == 0:  
        return 1  
    return n * fact(n-1)  
  
# what is fact(5)
```

$$\begin{aligned}5! &= 5 * 4 * 3 * 2 * 1 \\ &= 5 * 4! \\ 4! &= 4 * 3! \\ &\dots\end{aligned}$$

In general, $N! = N * (N-1)!$
And $\text{fact}(N) = N!$

$$\text{fact}(5) = 5 * \text{fact}(4)$$

2 a

```
def fact(n):  
    if n == 0:  
        return 1  
    return n * fact(n-1)  
  
# what is fact(5)
```

$$\begin{aligned}5! &= 5 * 4 * 3 * 2 * 1 \\ &= 5 * 4! \\ 4! &= 4 * 3! \\ &\dots\end{aligned}$$

In general, $N! = N * (N-1)!$
And $\text{fact}(N) = N!$

$$\begin{aligned}\text{fact}(5) &= 5 * \text{fact}(4) \\ \text{fact}(4) &= 4 * \text{fact}(3)\end{aligned}$$

2 a

```
def fact(n):  
    if n == 0:  
        return 1  
    return n * fact(n-1)  
  
# what is fact(5)
```

$$\begin{aligned}5! &= 5 * 4 * 3 * 2 * 1 \\ &= 5 * 4! \\ 4! &= 4 * 3! \\ &\dots\end{aligned}$$

In general, $N! = N * (N-1)!$
And $\text{fact}(N) = N!$

$$\begin{aligned}\text{fact}(5) &= 5 * \text{fact}(4) \\ \text{fact}(4) &= 4 * \text{fact}(3) \\ \text{fact}(3) &= 3 * \text{fact}(2)\end{aligned}$$

2 a

```
def fact(n):  
    if n == 0:  
        return 1  
    return n * fact(n-1)  
  
# what is fact(5)
```

$$\begin{aligned}5! &= 5 * 4 * 3 * 2 * 1 \\ &= 5 * 4! \\ 4! &= 4 * 3! \\ &\dots\end{aligned}$$

In general, $N! = N * (N-1)!$
And $\text{fact}(N) = N!$

$$\begin{aligned}\text{fact}(5) &= 5 * \text{fact}(4) \\ \text{fact}(4) &= 4 * \text{fact}(3) \\ \text{fact}(3) &= 3 * \text{fact}(2) \\ \text{fact}(2) &= 2 * \text{fact}(1)\end{aligned}$$

2 a

```
def fact(n):  
    if n == 0:  
        return 1  
    return n * fact(n-1)  
  
# what is fact(5)
```

$$\begin{aligned}5! &= 5 * 4 * 3 * 2 * 1 \\ &= 5 * 4! \\ 4! &= 4 * 3! \\ &\dots\end{aligned}$$

In general, $N! = N * (N-1)!$
And $\text{fact}(N) = N!$

$$\begin{aligned}\text{fact}(5) &= 5 * \text{fact}(4) \\ \text{fact}(4) &= 4 * \text{fact}(3) \\ \text{fact}(3) &= 3 * \text{fact}(2) \\ \text{fact}(2) &= 2 * \text{fact}(1) \\ \text{fact}(1) &= 1 * \text{fact}(0)\end{aligned}$$

2 a

```
def fact(n):  
    if n == 0:  
        return 1  
    return n * fact(n-1)  
  
# what is fact(5)
```

$$\begin{aligned}5! &= 5 * 4 * 3 * 2 * 1 \\ &= 5 * 4! \\ 4! &= 4 * 3! \\ &\dots\end{aligned}$$

In general, $N! = N * (N-1)!$
And $\text{fact}(N) = N!$

$$\begin{aligned}\text{fact}(5) &= 5 * \text{fact}(4) \\ \text{fact}(4) &= 4 * \text{fact}(3) \\ \text{fact}(3) &= 3 * \text{fact}(2) \\ \text{fact}(2) &= 2 * \text{fact}(1) \\ \text{fact}(1) &= 1 * \text{fact}(0) \\ \text{fact}(0) &= 1\end{aligned}$$

2 a

```
def fact(n):  
    if n == 0:  
        return 1  
    return n * fact(n-1)  
  
# what is fact(5)
```

$$\begin{aligned}5! &= 5 * 4 * 3 * 2 * 1 \\ &= 5 * 4! \\ 4! &= 4 * 3! \\ &\dots\end{aligned}$$

In general, $N! = N * (N-1)!$
And $\text{fact}(N) = N!$

$$\begin{aligned}\text{fact}(5) &= 5 * \text{fact}(4) \\ \text{fact}(4) &= 4 * \text{fact}(3) \\ \text{fact}(3) &= 3 * \text{fact}(2) \\ \text{fact}(2) &= 2 * \text{fact}(1) \\ \text{fact}(1) &= 1 * \text{fact}(0) \\ \text{fact}(0) &= 1 \text{ (base case)}\end{aligned}$$

2 a

```
def fact(n):  
    if n == 0:  
        return 1  
    return n * fact(n-1)  
  
# what is fact(5)
```

$$\begin{aligned}5! &= 5 * 4 * 3 * 2 * 1 \\ &= 5 * 4! \\ 4! &= 4 * 3! \\ &\dots\end{aligned}$$

In general, $N! = N * (N-1)!$
And $\text{fact}(N) = N!$

$$\begin{aligned}\text{fact}(5) &= 5 * \text{fact}(4) \\ \text{fact}(4) &= 4 * \text{fact}(3) \\ \text{fact}(3) &= 3 * \text{fact}(2) \\ \text{fact}(2) &= 2 * \text{fact}(1) \\ \text{fact}(1) &= 1 * \text{fact}(0) = 1 * 1 \\ \text{fact}(0) &= 1 \text{ (base case)}\end{aligned}$$

2 a

```
def fact(n):  
    if n == 0:  
        return 1  
    return n * fact(n-1)  
  
# what is fact(5)
```

$$\begin{aligned}5! &= 5 * 4 * 3 * 2 * 1 \\ &= 5 * 4! \\ 4! &= 4 * 3! \\ &\dots\end{aligned}$$

In general, $N! = N * (N-1)!$
And $\text{fact}(N) = N!$

$$\begin{aligned}\text{fact}(5) &= 5 * \text{fact}(4) \\ \text{fact}(4) &= 4 * \text{fact}(3) \\ \text{fact}(3) &= 3 * \text{fact}(2) \\ \text{fact}(2) &= 2 * \text{fact}(1) \\ \text{fact}(1) &= 1 * \text{fact}(0) = 1 * 1 = 1 \\ \text{fact}(0) &= 1 \text{ (base case)}\end{aligned}$$

2 a

```
def fact(n):  
    if n == 0:  
        return 1  
    return n * fact(n-1)  
  
# what is fact(5)
```

$$\begin{aligned}5! &= 5 * 4 * 3 * 2 * 1 \\ &= 5 * 4! \\ 4! &= 4 * 3! \\ &\dots\end{aligned}$$

In general, $N! = N * (N-1)!$
And $\text{fact}(N) = N!$

$$\begin{aligned}\text{fact}(5) &= 5 * \text{fact}(4) \\ \text{fact}(4) &= 4 * \text{fact}(3) \\ \text{fact}(3) &= 3 * \text{fact}(2) \\ \text{fact}(2) &= 2 * \text{fact}(1) = 2 * 1 \\ \text{fact}(1) &= 1 * \text{fact}(0) = 1 * 1 = 1 \\ \text{fact}(0) &= 1 \text{ (base case)}\end{aligned}$$

2 a

```
def fact(n):  
    if n == 0:  
        return 1  
    return n * fact(n-1)  
  
# what is fact(5)
```

$$\begin{aligned}5! &= 5 * 4 * 3 * 2 * 1 \\ &= 5 * 4! \\ 4! &= 4 * 3! \\ &\dots\end{aligned}$$

In general, $N! = N * (N-1)!$
And $\text{fact}(N) = N!$

$$\begin{aligned}\text{fact}(5) &= 5 * \text{fact}(4) \\ \text{fact}(4) &= 4 * \text{fact}(3) \\ \text{fact}(3) &= 3 * \text{fact}(2) \\ \text{fact}(2) &= 2 * \text{fact}(1) = 2 * 1 = 2 \\ \text{fact}(1) &= 1 * \text{fact}(0) = 1 * 1 = 1 \\ \text{fact}(0) &= 1 \text{ (base case)}\end{aligned}$$

2 a

```
def fact(n):  
    if n == 0:  
        return 1  
    return n * fact(n-1)  
  
# what is fact(5)
```

$$\begin{aligned}5! &= 5 * 4 * 3 * 2 * 1 \\ &= 5 * 4! \\ 4! &= 4 * 3! \\ &\dots\end{aligned}$$

In general, $N! = N * (N-1)!$
And $\text{fact}(N) = N!$

$$\begin{aligned}\text{fact}(5) &= 5 * \text{fact}(4) \\ \text{fact}(4) &= 4 * \text{fact}(3) \\ \text{fact}(3) &= 3 * \text{fact}(2) = 3 * 2 \\ \text{fact}(2) &= 2 * \text{fact}(1) = 2 * 1 = 2 \\ \text{fact}(1) &= 1 * \text{fact}(0) = 1 * 1 = 1 \\ \text{fact}(0) &= 1 \text{ (base case)}\end{aligned}$$

2 a

```
def fact(n):  
    if n == 0:  
        return 1  
    return n * fact(n-1)  
  
# what is fact(5)
```

$$\begin{aligned}5! &= 5 * 4 * 3 * 2 * 1 \\ &= 5 * 4! \\ 4! &= 4 * 3! \\ &\dots\end{aligned}$$

In general, $N! = N * (N-1)!$
And $\text{fact}(N) = N!$

$$\begin{aligned}\text{fact}(5) &= 5 * \text{fact}(4) \\ \text{fact}(4) &= 4 * \text{fact}(3) \\ \text{fact}(3) &= 3 * \text{fact}(2) = 3 * 2 = 6 \\ \text{fact}(2) &= 2 * \text{fact}(1) = 2 * 1 = 2 \\ \text{fact}(1) &= 1 * \text{fact}(0) = 1 * 1 = 1 \\ \text{fact}(0) &= 1 \text{ (base case)}\end{aligned}$$

2 a

```
def fact(n):  
    if n == 0:  
        return 1  
    return n * fact(n-1)  
  
# what is fact(5)
```

$$\begin{aligned}5! &= 5 * 4 * 3 * 2 * 1 \\ &= 5 * 4! \\ 4! &= 4 * 3! \\ &\dots\end{aligned}$$

In general, $N! = N * (N-1)!$
And $\text{fact}(N) = N!$

$$\begin{aligned}\text{fact}(5) &= 5 * \text{fact}(4) \\ \text{fact}(4) &= 4 * \text{fact}(3) = 4 * 6 \\ \text{fact}(3) &= 3 * \text{fact}(2) = 3 * 2 = 6 \\ \text{fact}(2) &= 2 * \text{fact}(1) = 2 * 1 = 2 \\ \text{fact}(1) &= 1 * \text{fact}(0) = 1 * 1 = 1 \\ \text{fact}(0) &= 1 \text{ (base case)}\end{aligned}$$

2 a

```
def fact(n):  
    if n == 0:  
        return 1  
    return n * fact(n-1)  
  
# what is fact(5)
```

$$\begin{aligned}5! &= 5 * 4 * 3 * 2 * 1 \\ &= 5 * 4! \\ 4! &= 4 * 3! \\ &\dots\end{aligned}$$

In general, $N! = N * (N-1)!$
And $\text{fact}(N) = N!$

$$\begin{aligned}\text{fact}(5) &= 5 * \text{fact}(4) \\ \text{fact}(4) &= 4 * \text{fact}(3) = 4 * 6 = 24 \\ \text{fact}(3) &= 3 * \text{fact}(2) = 3 * 2 = 6 \\ \text{fact}(2) &= 2 * \text{fact}(1) = 2 * 1 = 2 \\ \text{fact}(1) &= 1 * \text{fact}(0) = 1 * 1 = 1 \\ \text{fact}(0) &= 1 \text{ (base case)}\end{aligned}$$

2 a

```
def fact(n):  
    if n == 0:  
        return 1  
    return n * fact(n-1)  
  
# what is fact(5)
```

$$\begin{aligned}5! &= 5 * 4 * 3 * 2 * 1 \\ &= 5 * 4! \\ 4! &= 4 * 3! \\ &\dots\end{aligned}$$

In general, $N! = N * (N-1)!$
And $\text{fact}(N) = N!$

$$\begin{aligned}\text{fact}(5) &= 5 * \text{fact}(4) = 5 * 24 \\ \text{fact}(4) &= 4 * \text{fact}(3) = 4 * 6 = 24 \\ \text{fact}(3) &= 3 * \text{fact}(2) = 3 * 2 = 6 \\ \text{fact}(2) &= 2 * \text{fact}(1) = 2 * 1 = 2 \\ \text{fact}(1) &= 1 * \text{fact}(0) = 1 * 1 = 1 \\ \text{fact}(0) &= 1 \text{ (base case)}\end{aligned}$$

2 a

```
def fact(n):  
    if n == 0:  
        return 1  
    return n * fact(n-1)  
  
# what is fact(5)
```

$$\begin{aligned}5! &= 5 * 4 * 3 * 2 * 1 \\ &= 5 * 4! \\ 4! &= 4 * 3! \\ &\dots\end{aligned}$$

In general, $N! = N * (N-1)!$
And $\text{fact}(N) = N!$

$$\begin{aligned}\text{fact}(5) &= 5 * \text{fact}(4) = 5 * 24 = 120 \\ \text{fact}(4) &= 4 * \text{fact}(3) = 4 * 6 = 24 \\ \text{fact}(3) &= 3 * \text{fact}(2) = 3 * 2 = 6 \\ \text{fact}(2) &= 2 * \text{fact}(1) = 2 * 1 = 2 \\ \text{fact}(1) &= 1 * \text{fact}(0) = 1 * 1 = 1 \\ \text{fact}(0) &= 1 \text{ (base case)}\end{aligned}$$

2 a

```
def fact(n):  
    if n == 0:  
        return 1  
    return n * fact(n-1)  
  
# what is fact(5)
```

$$\begin{aligned}5! &= 5 * 4 * 3 * 2 * 1 \\ &= 5 * 4! \\ 4! &= 4 * 3! \\ &\dots\end{aligned}$$

In general, $N! = N * (N-1)!$
And $\text{fact}(N) = N!$

$$\begin{aligned}\text{fact}(5) &= 5 * \text{fact}(4) = 5 * 24 = 120 \\ \text{fact}(4) &= 4 * \text{fact}(3) = 4 * 6 = 24 \\ \text{fact}(3) &= 3 * \text{fact}(2) = 3 * 2 = 6 \\ \text{fact}(2) &= 2 * \text{fact}(1) = 2 * 1 = 2 \\ \text{fact}(1) &= 1 * \text{fact}(0) = 1 * 1 = 1 \\ \text{fact}(0) &= 1 \text{ (base case)}\end{aligned}$$

Answer: $\text{fact}(5) = 120$

2 b

```
def fib(n):  
    if n < 2:  
        return n  
    return fib(n-1) + fib(n-2)  
  
# what is fib(6)?
```

2 b

```
def fib(n):  
    if n < 2:  
        return n  
    return fib(n-1) + fib(n-2)  
  
# what is fib(6)?
```

fib(6)

2 b

```
def fib(n):  
    if n < 2:  
        return n  
    return fib(n-1) + fib(n-2)  
  
# what is fib(6)?
```

$\text{fib}(6) = \text{fib}(5) + \text{fib}(4)$

2 b

```
def fib(n):  
    if n < 2:  
        return n  
    return fib(n-1) + fib(n-2)  
  
# what is fib(6)?
```

$\text{fib}(6) = \text{fib}(5) + \text{fib}(4)$

$\text{fib}(5) = \text{fib}(4) + \text{fib}(3)$

2 b

```
def fib(n):  
    if n < 2:  
        return n  
    return fib(n-1) + fib(n-2)  
  
# what is fib(6)?
```

$\text{fib}(6) = \text{fib}(5) + \text{fib}(4)$

$\text{fib}(5) = \text{fib}(4) + \text{fib}(3)$

$\text{fib}(4) = \text{fib}(3) + \text{fib}(2)$

2 b

```
def fib(n):  
    if n < 2:  
        return n  
    return fib(n-1) + fib(n-2)  
  
# what is fib(6)?
```

$\text{fib}(6) = \text{fib}(5) + \text{fib}(4)$

$\text{fib}(5) = \text{fib}(4) + \text{fib}(3)$

$\text{fib}(4) = \text{fib}(3) + \text{fib}(2)$

$\text{fib}(3) = \text{fib}(2) + \text{fib}(1)$

2 b

```
def fib(n):  
    if n < 2:  
        return n  
    return fib(n-1) + fib(n-2)  
  
# what is fib(6)?
```

$\text{fib}(6) = \text{fib}(5) + \text{fib}(4)$

$\text{fib}(5) = \text{fib}(4) + \text{fib}(3)$

$\text{fib}(4) = \text{fib}(3) + \text{fib}(2)$

$\text{fib}(3) = \text{fib}(2) + \text{fib}(1)$

$\text{fib}(2) = \text{fib}(1) + \text{fib}(0)$

2 b

```
def fib(n):  
    if n < 2:  
        return n  
    return fib(n-1) + fib(n-2)  
  
# what is fib(6)?
```

$\text{fib}(6) = \text{fib}(5) + \text{fib}(4)$

$\text{fib}(5) = \text{fib}(4) + \text{fib}(3)$

$\text{fib}(4) = \text{fib}(3) + \text{fib}(2)$

$\text{fib}(3) = \text{fib}(2) + \text{fib}(1)$

$\text{fib}(2) = \text{fib}(1) + \text{fib}(0)$

$\text{fib}(1) = 1$

2 b

```
def fib(n):  
    if n < 2:  
        return n  
    return fib(n-1) + fib(n-2)  
  
# what is fib(6)?
```

$\text{fib}(6) = \text{fib}(5) + \text{fib}(4)$

$\text{fib}(5) = \text{fib}(4) + \text{fib}(3)$

$\text{fib}(4) = \text{fib}(3) + \text{fib}(2)$

$\text{fib}(3) = \text{fib}(2) + \text{fib}(1)$

$\text{fib}(2) = \text{fib}(1) + \text{fib}(0)$

$\text{fib}(1) = 1$

$\text{fib}(0) = 0$

2 b

```
def fib(n):  
    if n < 2:  
        return n  
    return fib(n-1) + fib(n-2)  
  
# what is fib(6)?
```

$$\text{fib}(6) = \text{fib}(5) + \text{fib}(4)$$

$$\text{fib}(5) = \text{fib}(4) + \text{fib}(3)$$

$$\text{fib}(4) = \text{fib}(3) + \text{fib}(2)$$

$$\text{fib}(3) = \text{fib}(2) + \text{fib}(1)$$

$$\text{fib}(2) = \text{fib}(1) + \text{fib}(0) = 1 + 0$$

$$\text{fib}(1) = 1$$

$$\text{fib}(0) = 0$$

2 b

```
def fib(n):  
    if n < 2:  
        return n  
    return fib(n-1) + fib(n-2)  
  
# what is fib(6)?
```

$$\text{fib}(6) = \text{fib}(5) + \text{fib}(4)$$

$$\text{fib}(5) = \text{fib}(4) + \text{fib}(3)$$

$$\text{fib}(4) = \text{fib}(3) + \text{fib}(2)$$

$$\text{fib}(3) = \text{fib}(2) + \text{fib}(1)$$

$$\text{fib}(2) = \text{fib}(1) + \text{fib}(0) = 1 + 0 = 1$$

$$\text{fib}(1) = 1$$

$$\text{fib}(0) = 0$$

2 b

```
def fib(n):  
    if n < 2:  
        return n  
    return fib(n-1) + fib(n-2)  
  
# what is fib(6)?
```

$$\text{fib}(6) = \text{fib}(5) + \text{fib}(4)$$

$$\text{fib}(5) = \text{fib}(4) + \text{fib}(3)$$

$$\text{fib}(4) = \text{fib}(3) + \text{fib}(2)$$

$$\text{fib}(3) = \text{fib}(2) + \text{fib}(1) = 1 + 1$$

$$\text{fib}(2) = \text{fib}(1) + \text{fib}(0) = 1 + 0 = 1$$

$$\text{fib}(1) = 1$$

$$\text{fib}(0) = 0$$

2 b

```
def fib(n):  
    if n < 2:  
        return n  
    return fib(n-1) + fib(n-2)  
  
# what is fib(6)?
```

$$\text{fib}(6) = \text{fib}(5) + \text{fib}(4)$$

$$\text{fib}(5) = \text{fib}(4) + \text{fib}(3)$$

$$\text{fib}(4) = \text{fib}(3) + \text{fib}(2)$$

$$\text{fib}(3) = \text{fib}(2) + \text{fib}(1) = 1 + 1 = 2$$

$$\text{fib}(2) = \text{fib}(1) + \text{fib}(0) = 1 + 0 = 1$$

$$\text{fib}(1) = 1$$

$$\text{fib}(0) = 0$$

2 b

```
def fib(n):  
    if n < 2:  
        return n  
    return fib(n-1) + fib(n-2)  
  
# what is fib(6)?
```

$$\text{fib}(6) = \text{fib}(5) + \text{fib}(4)$$

$$\text{fib}(5) = \text{fib}(4) + \text{fib}(3)$$

$$\text{fib}(4) = \text{fib}(3) + \text{fib}(2) = 2 + 1$$

$$\text{fib}(3) = \text{fib}(2) + \text{fib}(1) = 1 + 1 = 2$$

$$\text{fib}(2) = \text{fib}(1) + \text{fib}(0) = 1 + 0 = 1$$

$$\text{fib}(1) = 1$$

$$\text{fib}(0) = 0$$

2 b

```
def fib(n):  
    if n < 2:  
        return n  
    return fib(n-1) + fib(n-2)  
  
# what is fib(6)?
```

$$\text{fib}(6) = \text{fib}(5) + \text{fib}(4)$$

$$\text{fib}(5) = \text{fib}(4) + \text{fib}(3)$$

$$\text{fib}(4) = \text{fib}(3) + \text{fib}(2) = 2 + 1 = 3$$

$$\text{fib}(3) = \text{fib}(2) + \text{fib}(1) = 1 + 1 = 2$$

$$\text{fib}(2) = \text{fib}(1) + \text{fib}(0) = 1 + 0 = 1$$

$$\text{fib}(1) = 1$$

$$\text{fib}(0) = 0$$

2 b

```
def fib(n):  
    if n < 2:  
        return n  
    return fib(n-1) + fib(n-2)  
  
# what is fib(6)?
```

$$\text{fib}(6) = \text{fib}(5) + \text{fib}(4)$$

$$\text{fib}(5) = \text{fib}(4) + \text{fib}(3) = 3 + 2$$

$$\text{fib}(4) = \text{fib}(3) + \text{fib}(2) = 2 + 1 = 3$$

$$\text{fib}(3) = \text{fib}(2) + \text{fib}(1) = 1 + 1 = 2$$

$$\text{fib}(2) = \text{fib}(1) + \text{fib}(0) = 1 + 0 = 1$$

$$\text{fib}(1) = 1$$

$$\text{fib}(0) = 0$$

2 b

```
def fib(n):  
    if n < 2:  
        return n  
    return fib(n-1) + fib(n-2)  
  
# what is fib(6)?
```

$$\text{fib}(6) = \text{fib}(5) + \text{fib}(4)$$

$$\text{fib}(5) = \text{fib}(4) + \text{fib}(3) = 3 + 2 = 5$$

$$\text{fib}(4) = \text{fib}(3) + \text{fib}(2) = 2 + 1 = 3$$

$$\text{fib}(3) = \text{fib}(2) + \text{fib}(1) = 1 + 1 = 2$$

$$\text{fib}(2) = \text{fib}(1) + \text{fib}(0) = 1 + 0 = 1$$

$$\text{fib}(1) = 1$$

$$\text{fib}(0) = 0$$

2 b

```
def fib(n):  
    if n < 2:  
        return n  
    return fib(n-1) + fib(n-2)  
  
# what is fib(6)?
```

$$\text{fib}(6) = \text{fib}(5) + \text{fib}(4) = 5 + 3$$

$$\text{fib}(5) = \text{fib}(4) + \text{fib}(3) = 3 + 2 = 5$$

$$\text{fib}(4) = \text{fib}(3) + \text{fib}(2) = 2 + 1 = 3$$

$$\text{fib}(3) = \text{fib}(2) + \text{fib}(1) = 1 + 1 = 2$$

$$\text{fib}(2) = \text{fib}(1) + \text{fib}(0) = 1 + 0 = 1$$

$$\text{fib}(1) = 1$$

$$\text{fib}(0) = 0$$

2 b

```
def fib(n):  
    if n < 2:  
        return n  
    return fib(n-1) + fib(n-2)  
  
# what is fib(6)?
```

$$\text{fib}(6) = \text{fib}(5) + \text{fib}(4) = 5 + 3 = 8$$

$$\text{fib}(5) = \text{fib}(4) + \text{fib}(3) = 3 + 2 = 5$$

$$\text{fib}(4) = \text{fib}(3) + \text{fib}(2) = 2 + 1 = 3$$

$$\text{fib}(3) = \text{fib}(2) + \text{fib}(1) = 1 + 1 = 2$$

$$\text{fib}(2) = \text{fib}(1) + \text{fib}(0) = 1 + 0 = 1$$

$$\text{fib}(1) = 1$$

$$\text{fib}(0) = 0$$

2 b

```
def fib(n):  
    if n < 2:  
        return n  
    return fib(n-1) + fib(n-2)  
  
# what is fib(6)?
```

$$\text{fib}(6) = \text{fib}(5) + \text{fib}(4) = 5 + 3 = 8$$

$$\text{fib}(5) = \text{fib}(4) + \text{fib}(3) = 3 + 2 = 5$$

$$\text{fib}(4) = \text{fib}(3) + \text{fib}(2) = 2 + 1 = 3$$

$$\text{fib}(3) = \text{fib}(2) + \text{fib}(1) = 1 + 1 = 2$$

$$\text{fib}(2) = \text{fib}(1) + \text{fib}(0) = 1 + 0 = 1$$

$$\text{fib}(1) = 1$$

$$\text{fib}(0) = 0$$

Answer: $\text{fib}(6) = 8$

3

```
def f(n):  
    print(n)  
    if n < 9:  
        f(n + 1)
```

what does f(7)
print?

```
def g(n):  
    if n < 9:  
        g(n + 1)  
    print(n)
```

what does g(7)
print?

3 a

```
def f(n):  
    print(n)  
    if n < 9:  
        f(n + 1)
```

```
# what does f(7)  
print?
```

3 a

```
def f(n):  
    print(n)  
    if n < 9:  
        f(n + 1)
```

```
# what does f(7)  
print?
```

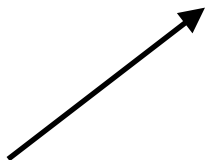
f(7)

3 a

```
def f(n):  
    print(n)  
    if n < 9:  
        f(n + 1)
```

```
# what does f(7)  
print?
```

f(7)

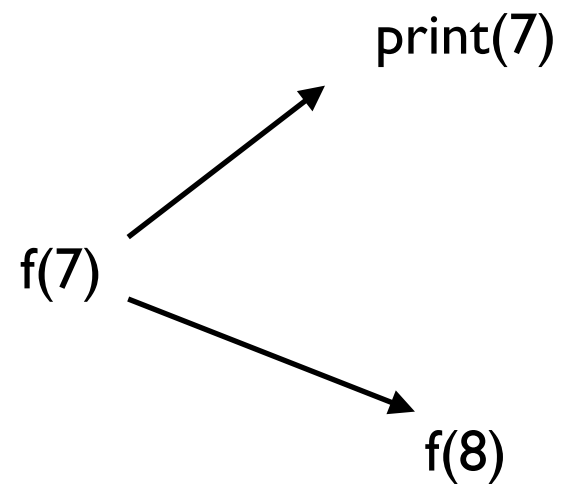


print(7)

3 a

```
def f(n):  
    print(n)  
    if n < 9:  
        f(n + 1)
```

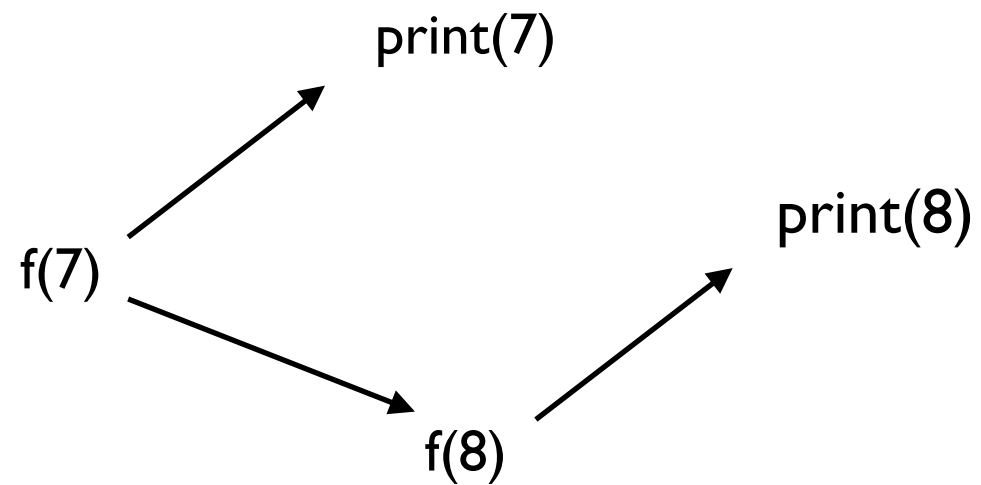
```
# what does f(7)  
print?
```



3 a

```
def f(n):  
    print(n)  
    if n < 9:  
        f(n + 1)
```

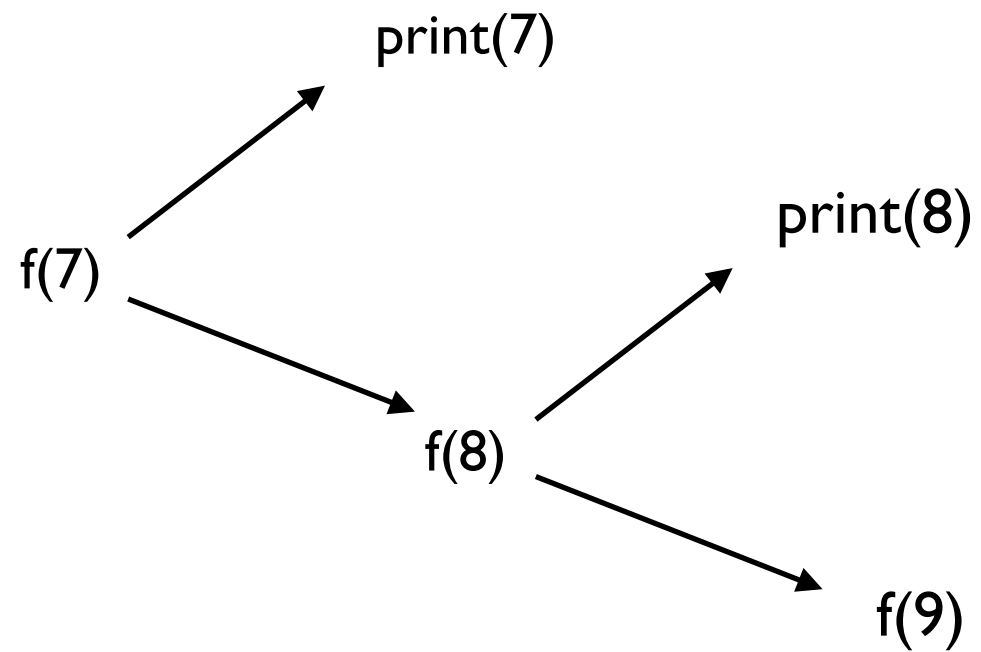
```
# what does f(7)  
print?
```



3 a

```
def f(n):  
    print(n)  
    if n < 9:  
        f(n + 1)
```

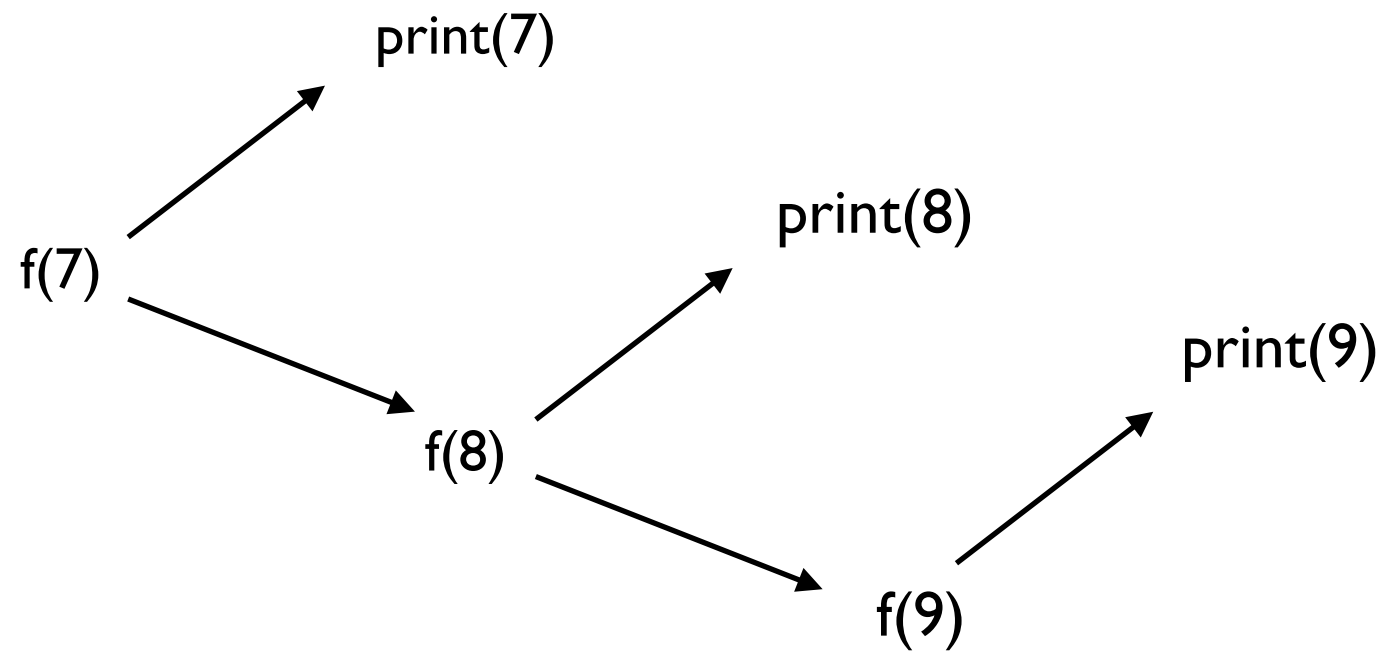
```
# what does f(7)  
print?
```



3 a

```
def f(n):  
    print(n)  
    if n < 9:  
        f(n + 1)
```

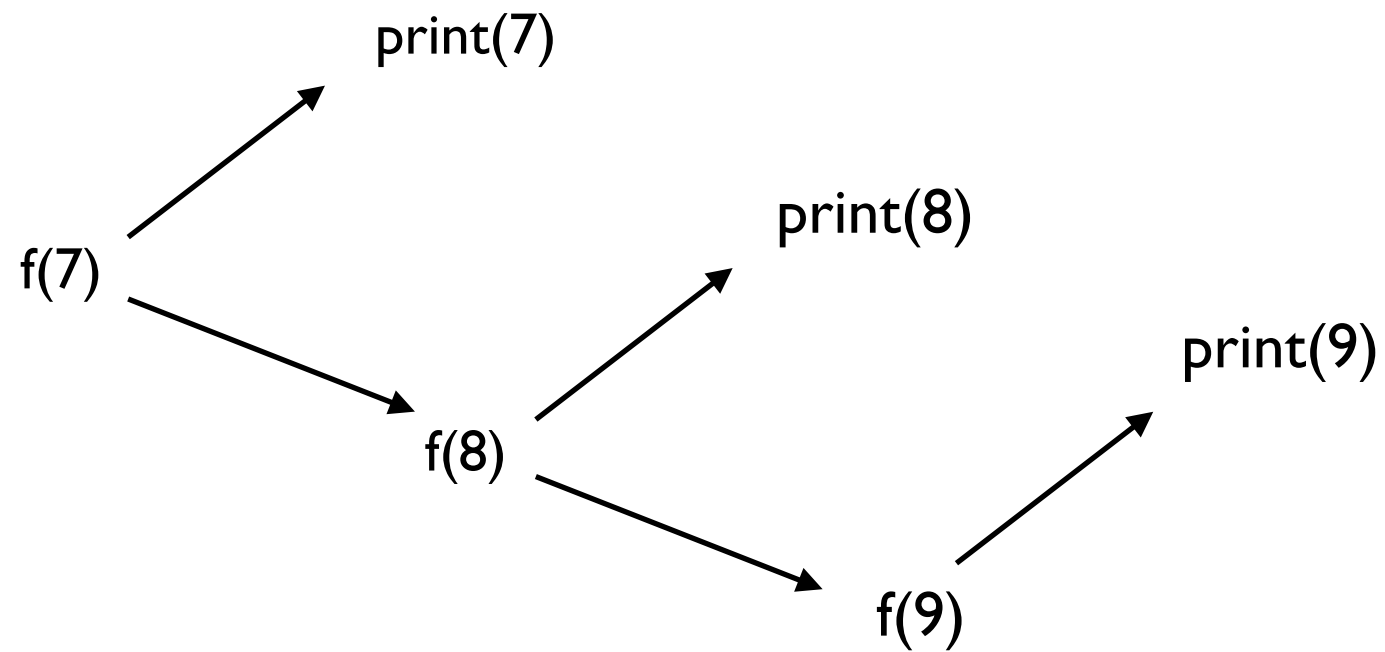
```
# what does f(7)  
print?
```



3 a

```
def f(n):  
    print(n)  
    if n < 9:  
        f(n + 1)
```

```
# what does f(7)  
print?
```



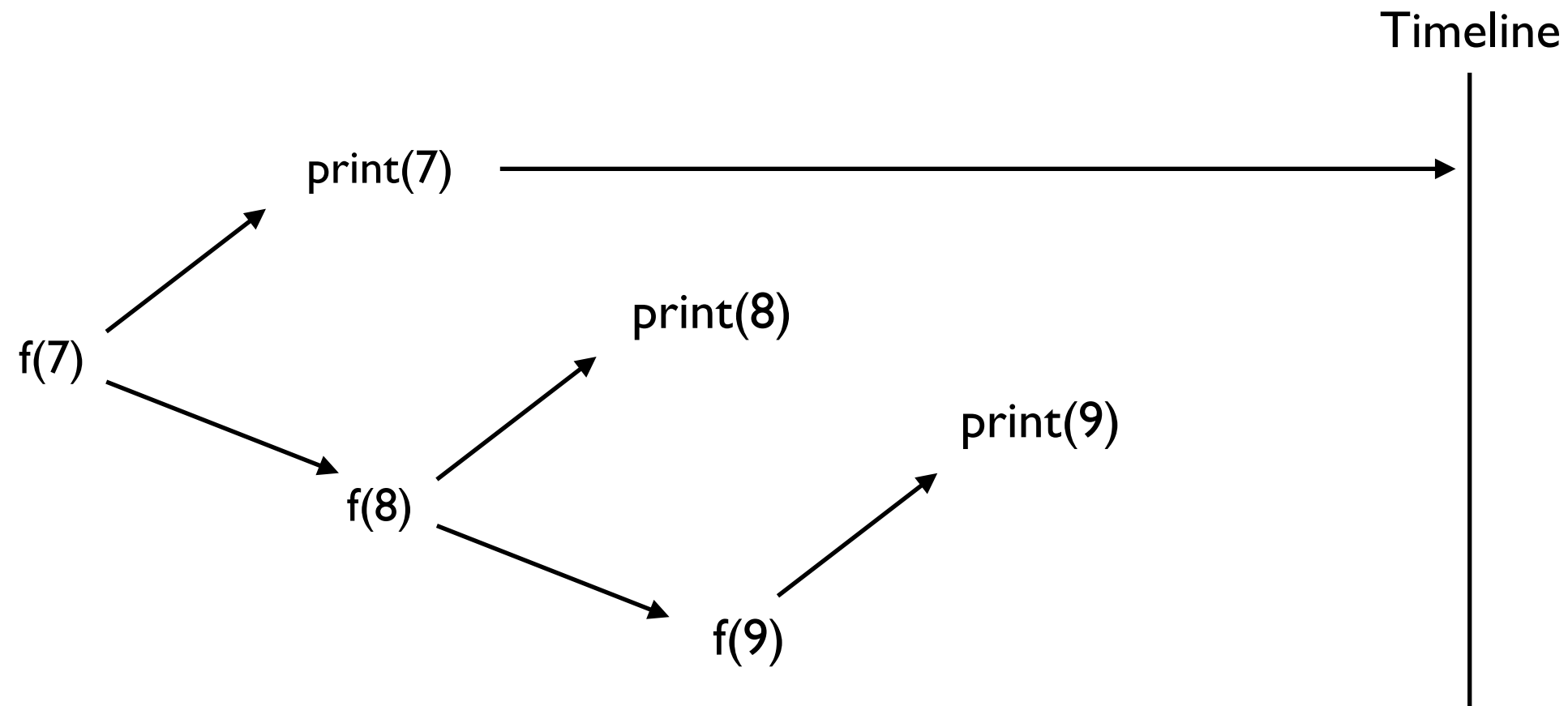
Timeline



3 a

```
def f(n):  
    print(n)  
    if n < 9:  
        f(n + 1)
```

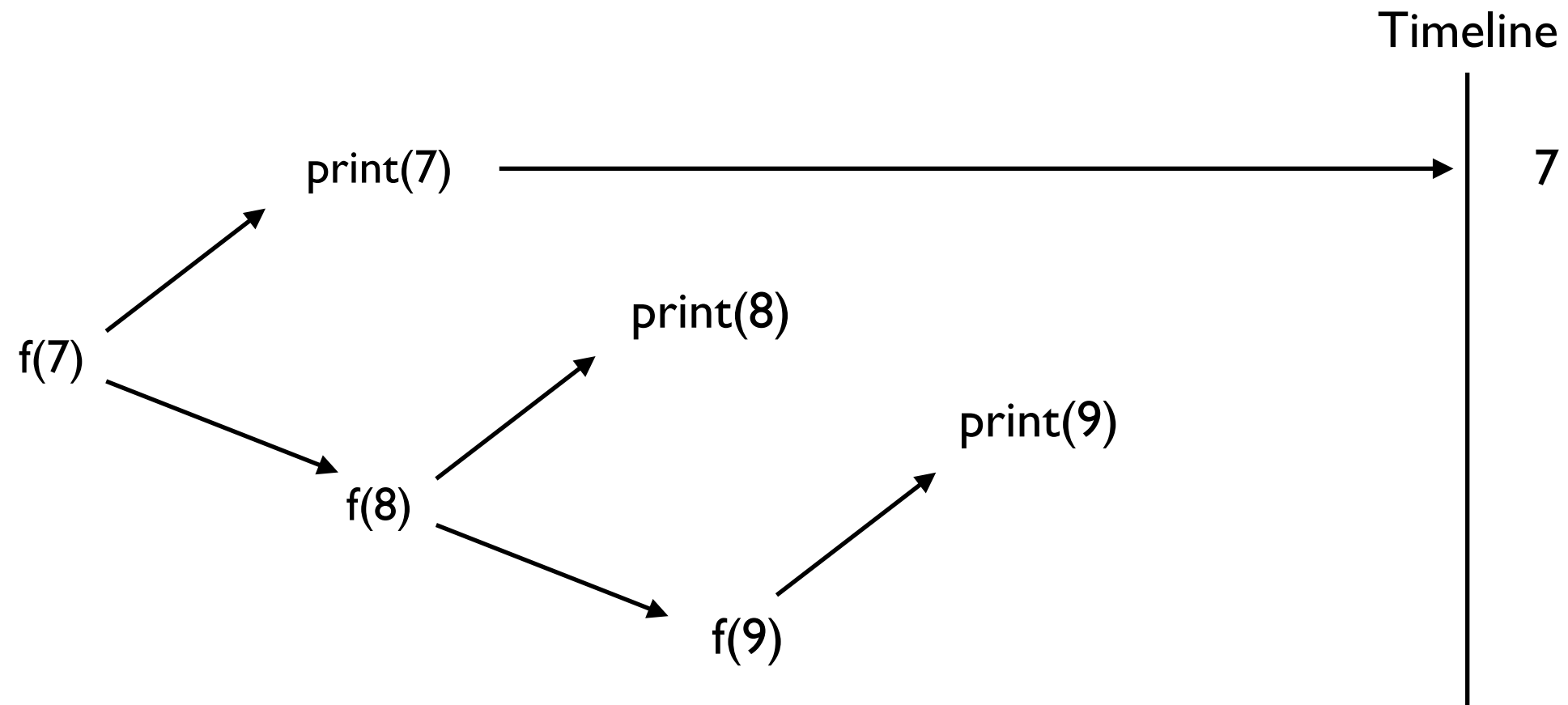
```
# what does f(7)  
print?
```



3 a

```
def f(n):  
    print(n)  
    if n < 9:  
        f(n + 1)
```

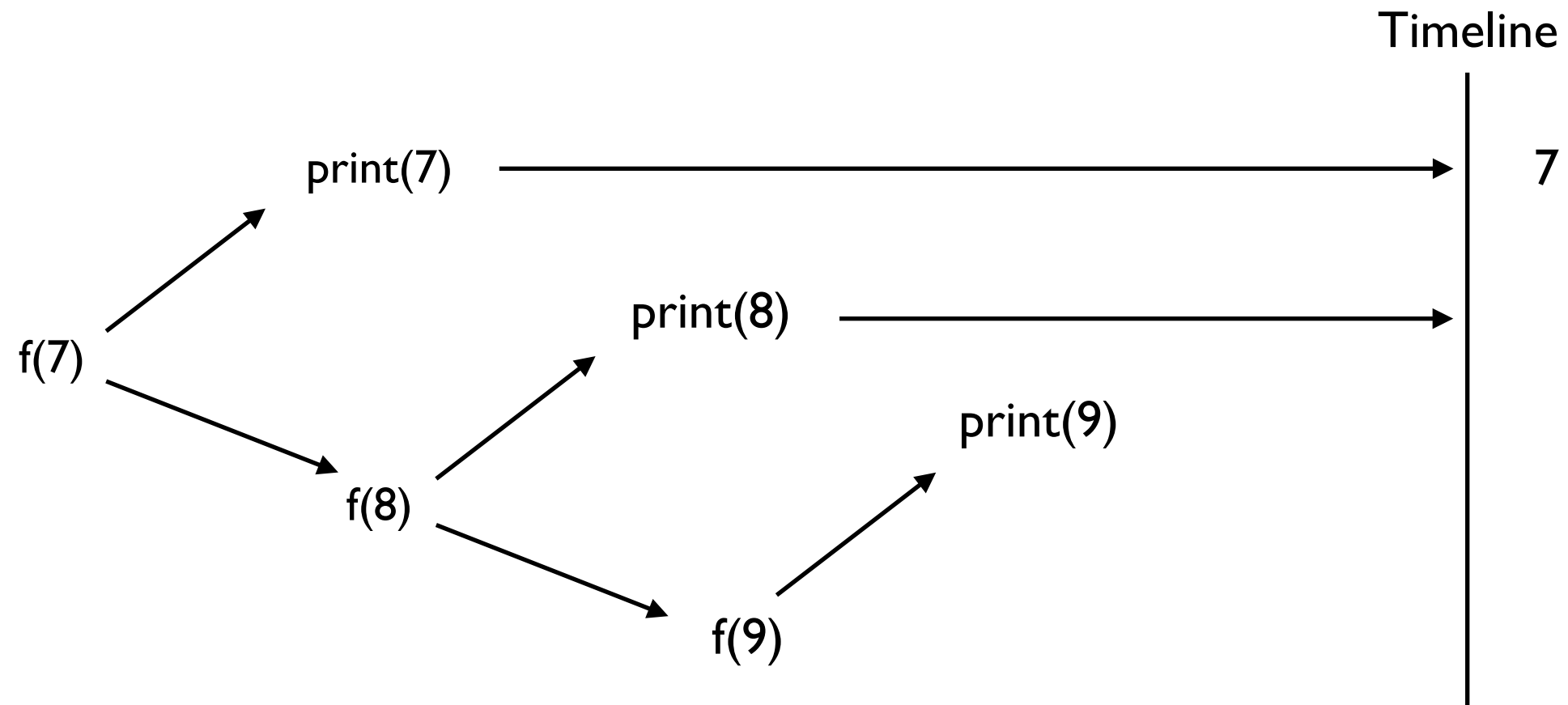
```
# what does f(7)  
print?
```



3 a

```
def f(n):  
    print(n)  
    if n < 9:  
        f(n + 1)
```

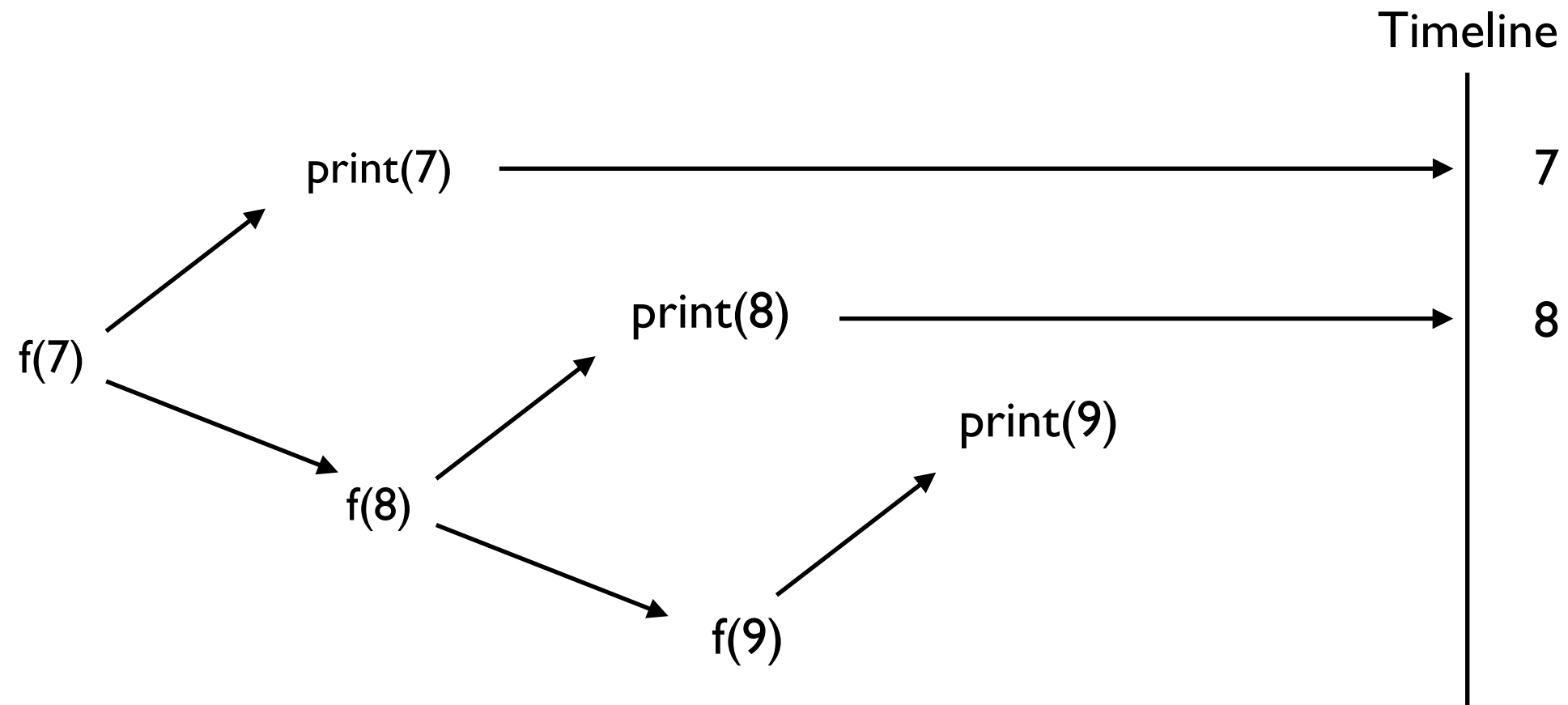
```
# what does f(7)  
print?
```



3 a

```
def f(n):  
    print(n)  
    if n < 9:  
        f(n + 1)
```

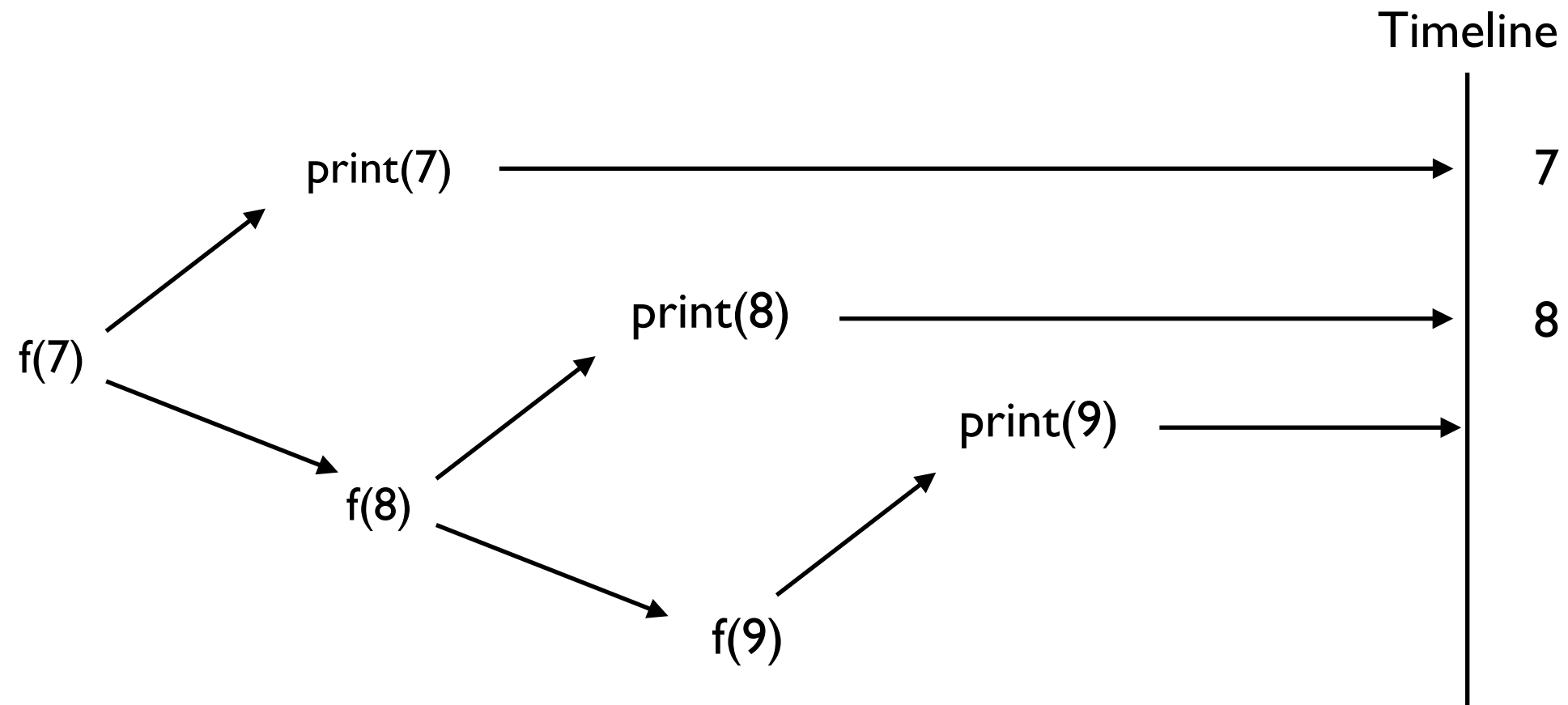
```
# what does f(7)  
print?
```



3 a

```
def f(n):  
    print(n)  
    if n < 9:  
        f(n + 1)
```

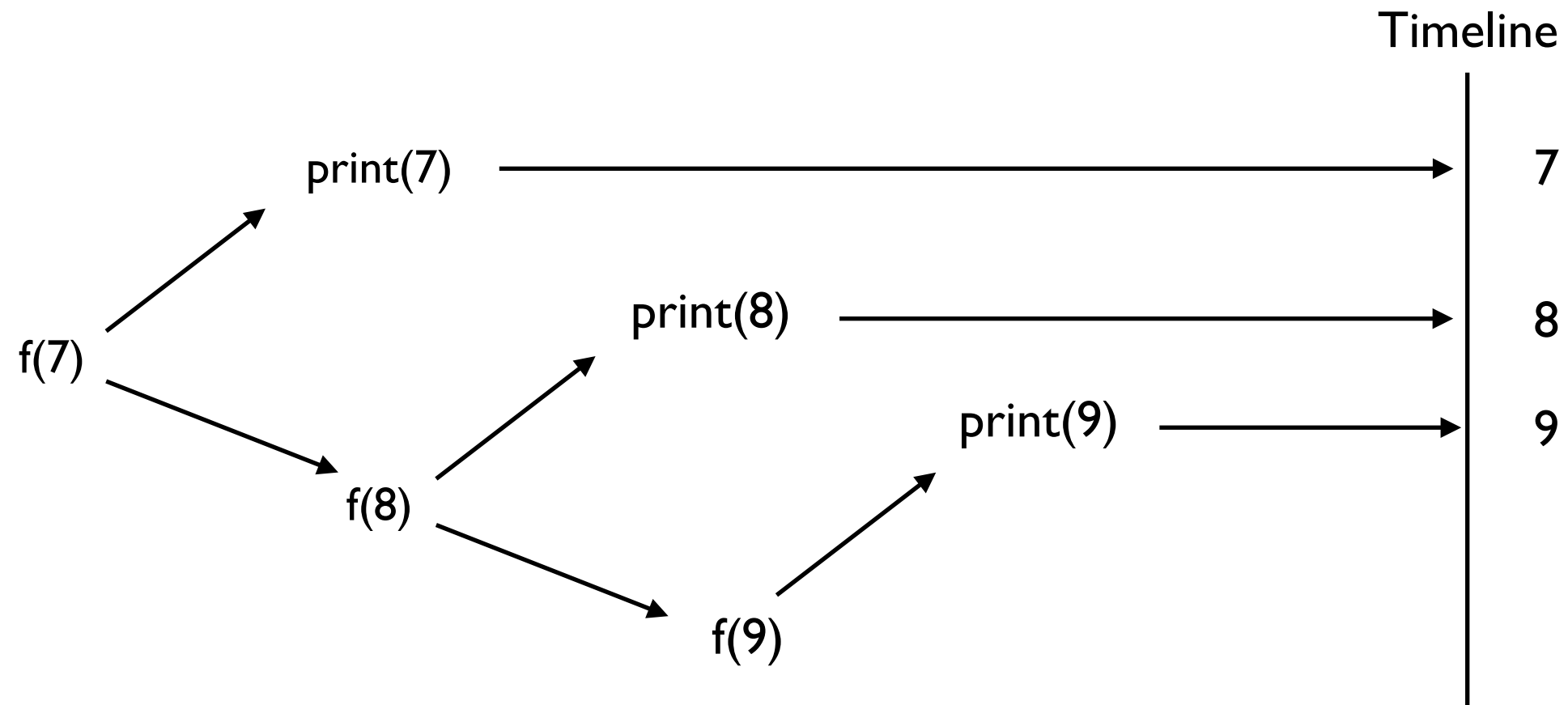
```
# what does f(7)  
print?
```



3 a

```
def f(n):  
    print(n)  
    if n < 9:  
        f(n + 1)
```

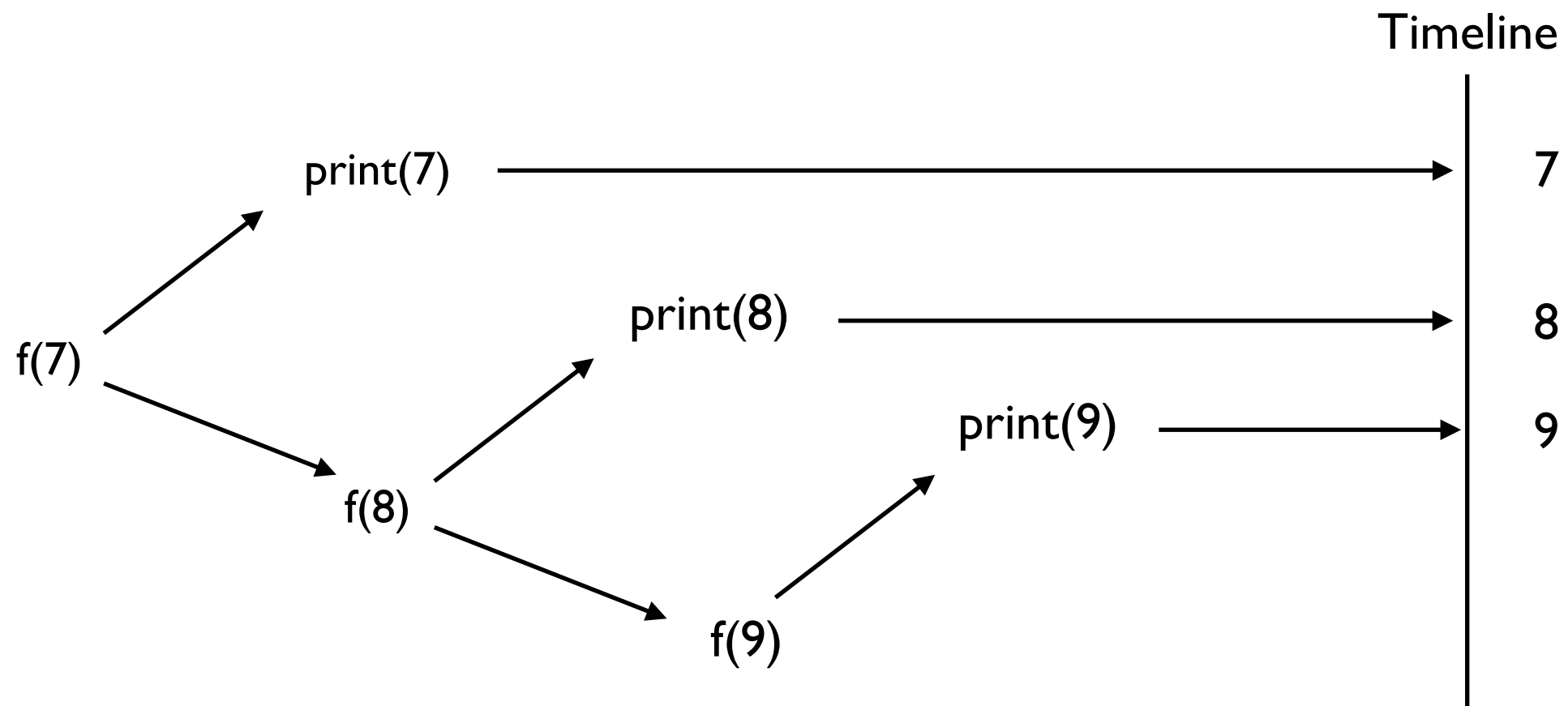
```
# what does f(7)  
print?
```



3 a

```
def f(n):  
    print(n)  
    if n < 9:  
        f(n + 1)
```

```
# what does f(7)  
print?
```



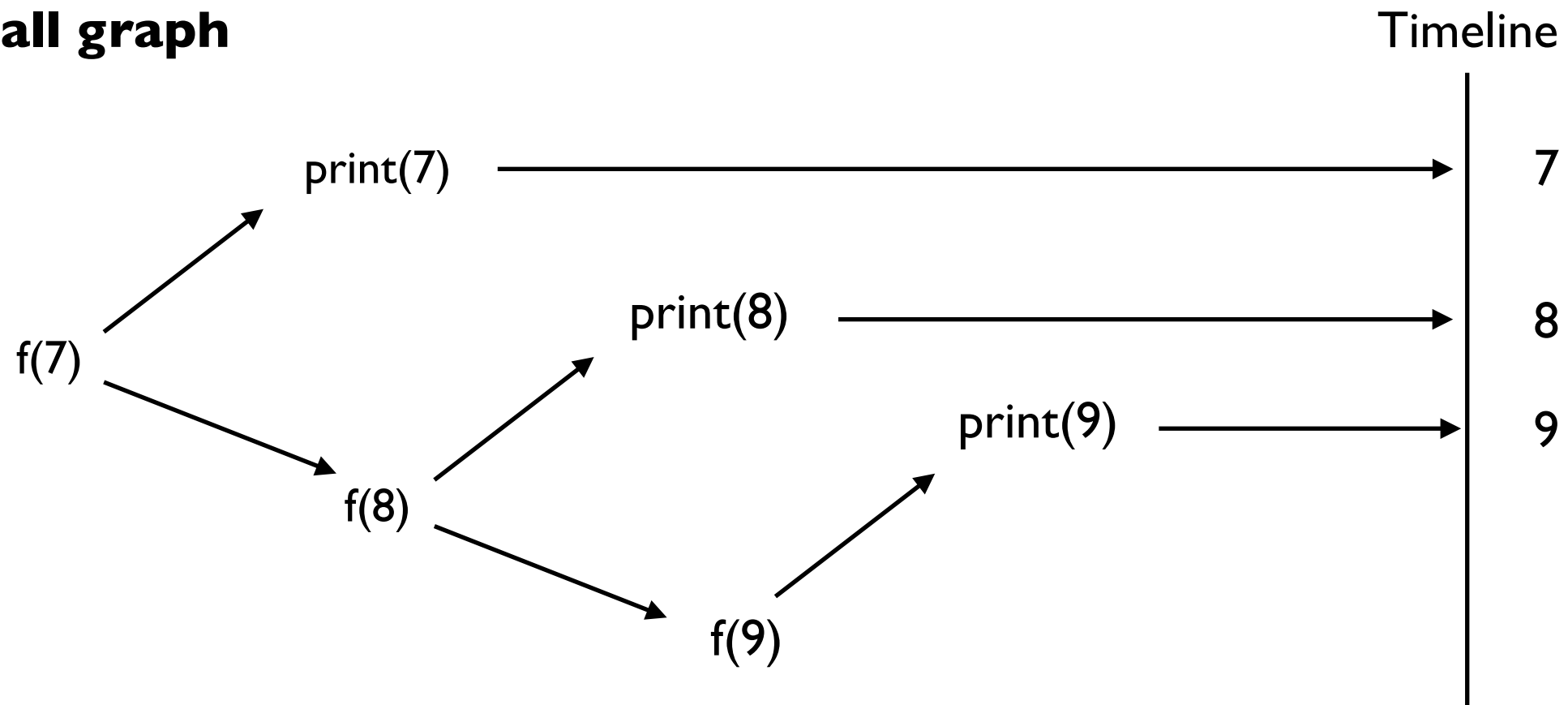
Answer: 7, 8, 9

3 a

```
def f(n):  
    print(n)  
    if n < 9:  
        f(n + 1)
```

```
# what does f(7)  
print?
```

Call graph



Answer: 7, 8, 9

3 b

```
def g(n):  
    if n < 9:  
        g(n + 1)  
    print(n)
```

```
# what does g(7)  
print?
```

3 b

```
def g(n):  
    if n < 9:  
        g(n + 1)  
    print(n)
```

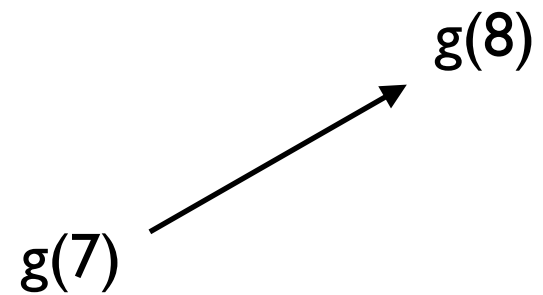
```
# what does g(7)  
print?
```

g(7)

3 b

```
def g(n):  
    if n < 9:  
        g(n + 1)  
    print(n)
```

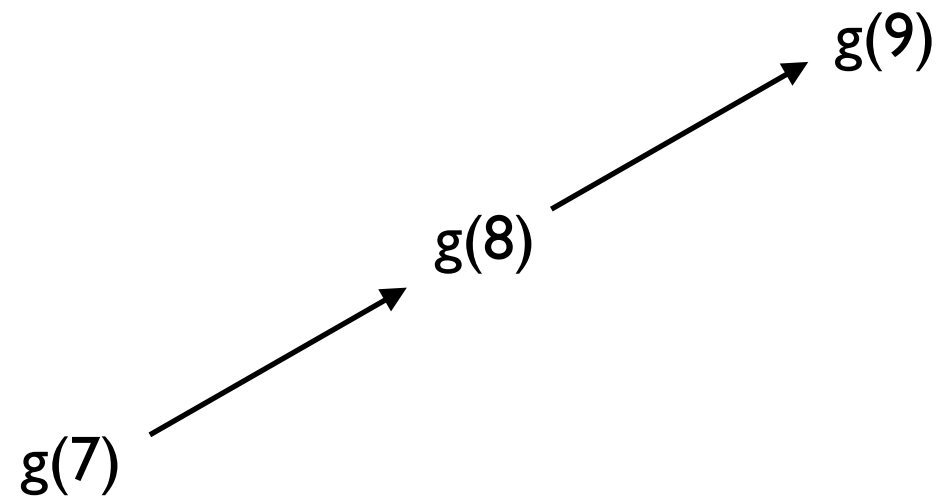
```
# what does g(7)  
print?
```



3 b

```
def g(n):  
    if n < 9:  
        g(n + 1)  
    print(n)
```

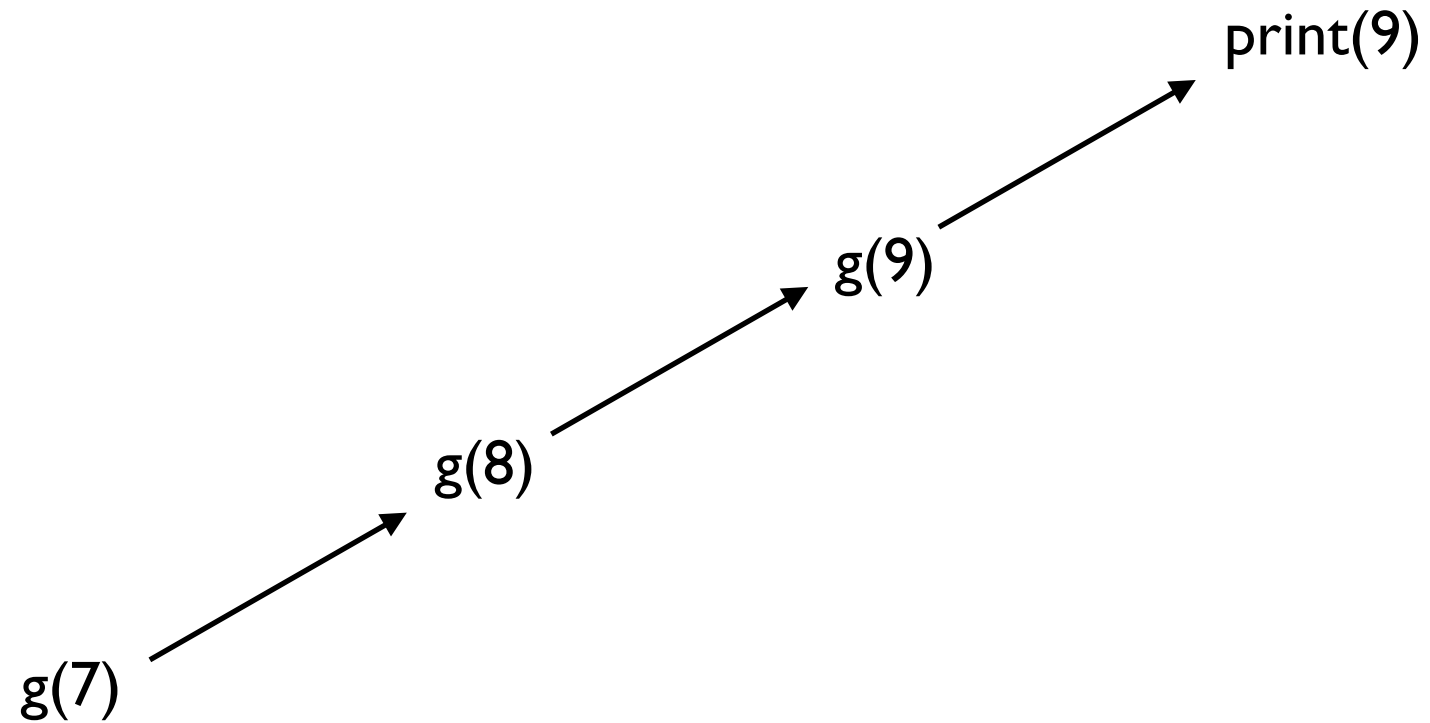
```
# what does g(7)  
print?
```



3 b

```
def g(n):  
    if n < 9:  
        g(n + 1)  
    print(n)
```

```
# what does g(7)  
print?
```

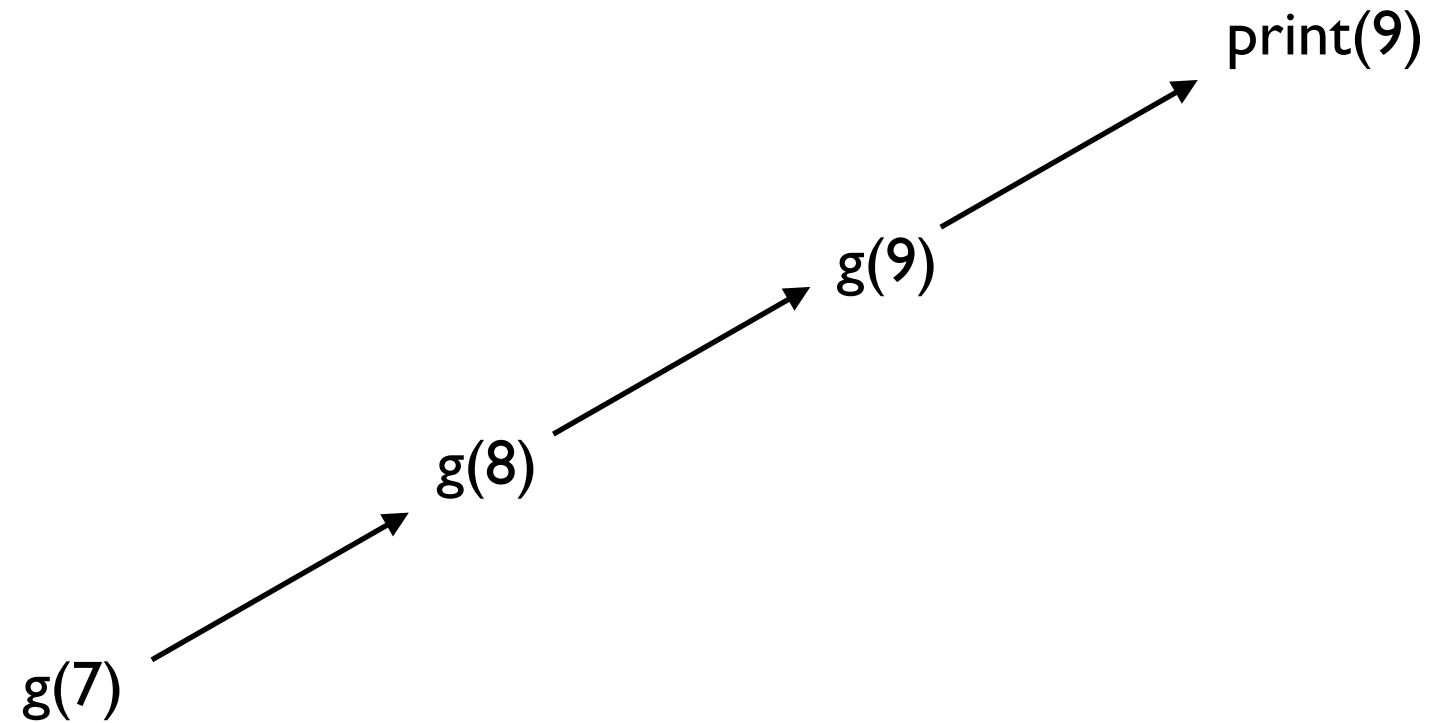


3 b

```
def g(n):  
    if n < 9:  
        g(n + 1)  
    print(n)
```

```
# what does g(7)  
print?
```

Timeline

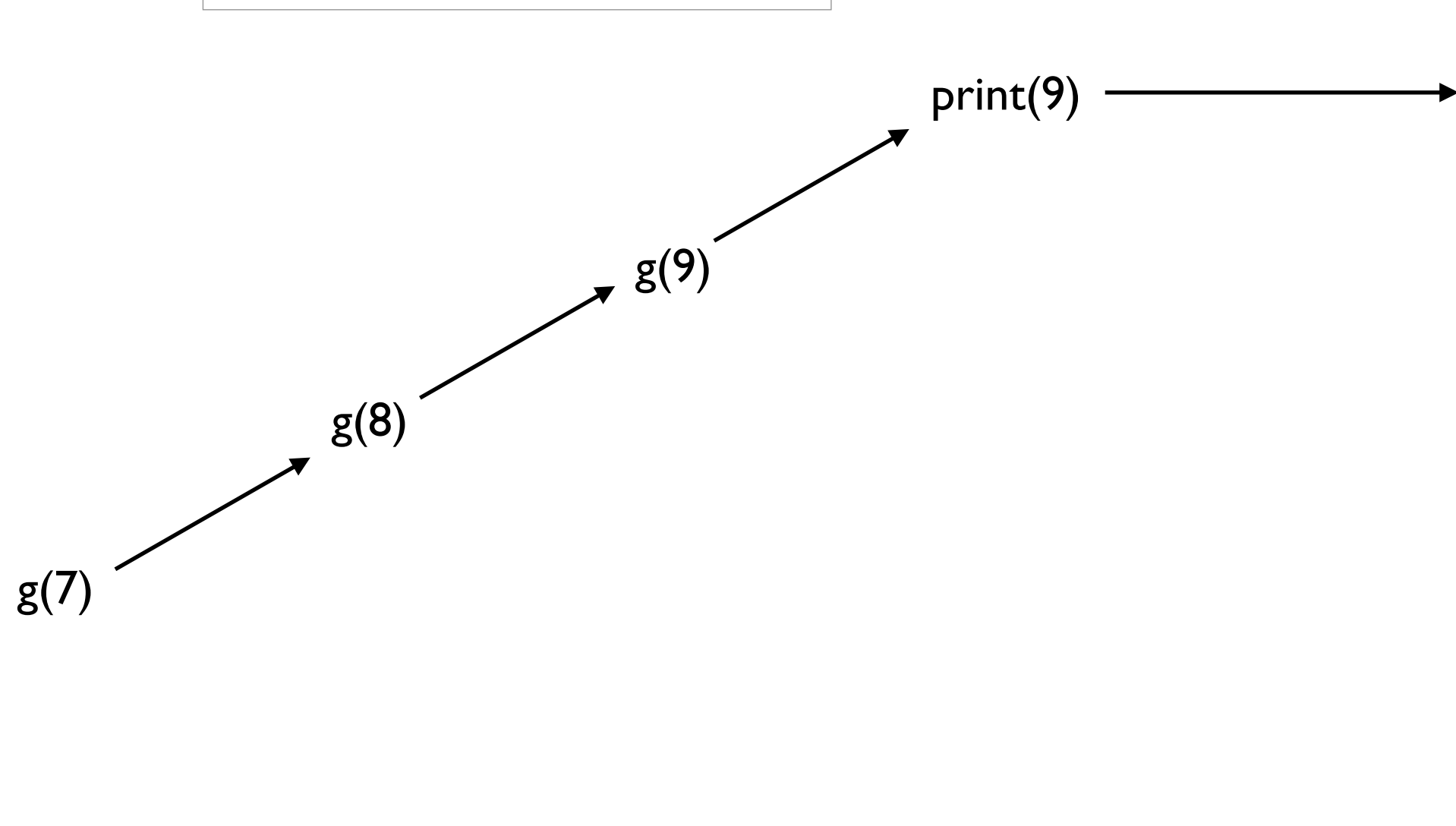


3 b

```
def g(n):  
    if n < 9:  
        g(n + 1)  
    print(n)
```

```
# what does g(7)  
print?
```

Timeline

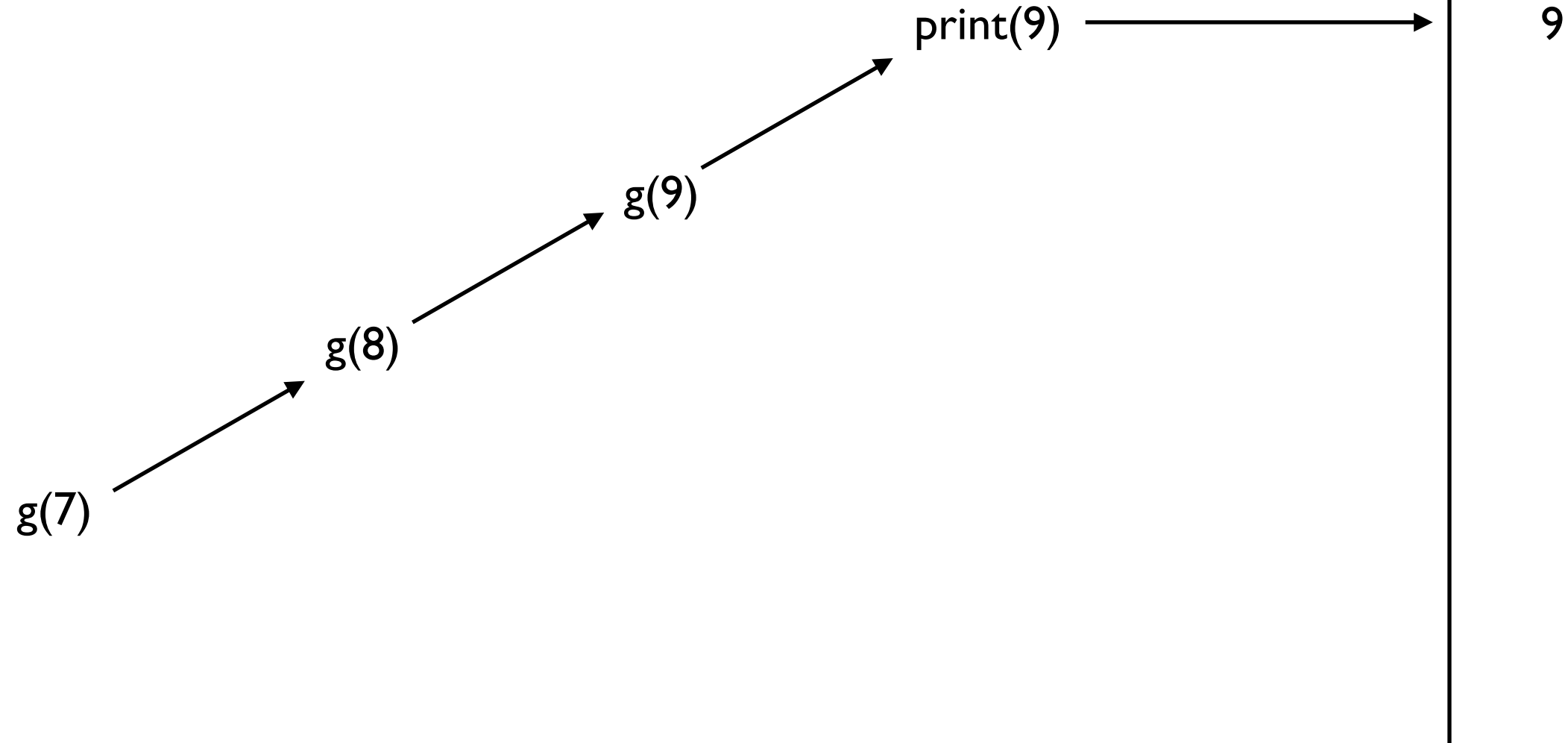


3 b

```
def g(n):  
    if n < 9:  
        g(n + 1)  
    print(n)
```

```
# what does g(7)  
print?
```

Timeline

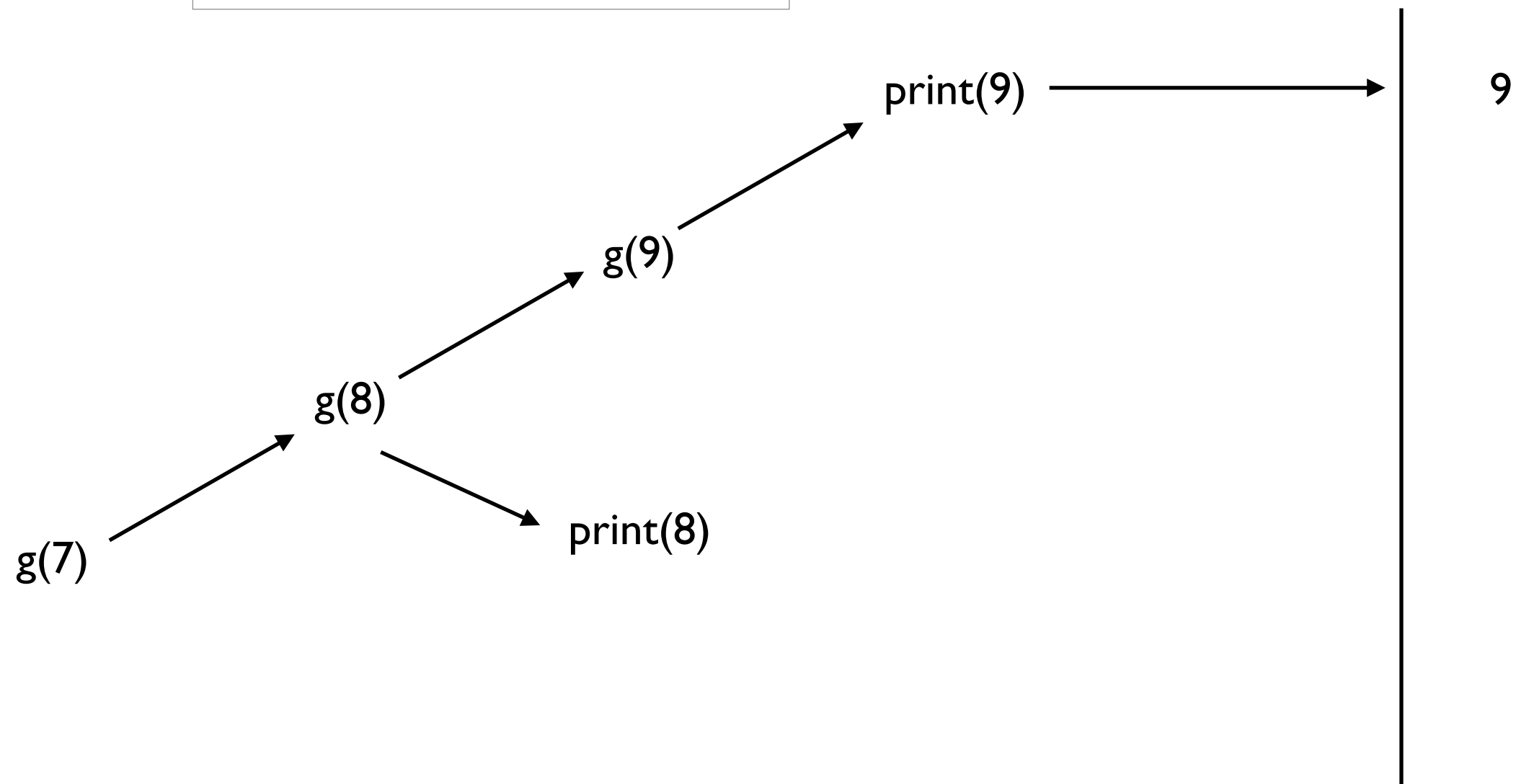


3 b

```
def g(n):  
    if n < 9:  
        g(n + 1)  
    print(n)
```

what does g(7)
print?

Timeline

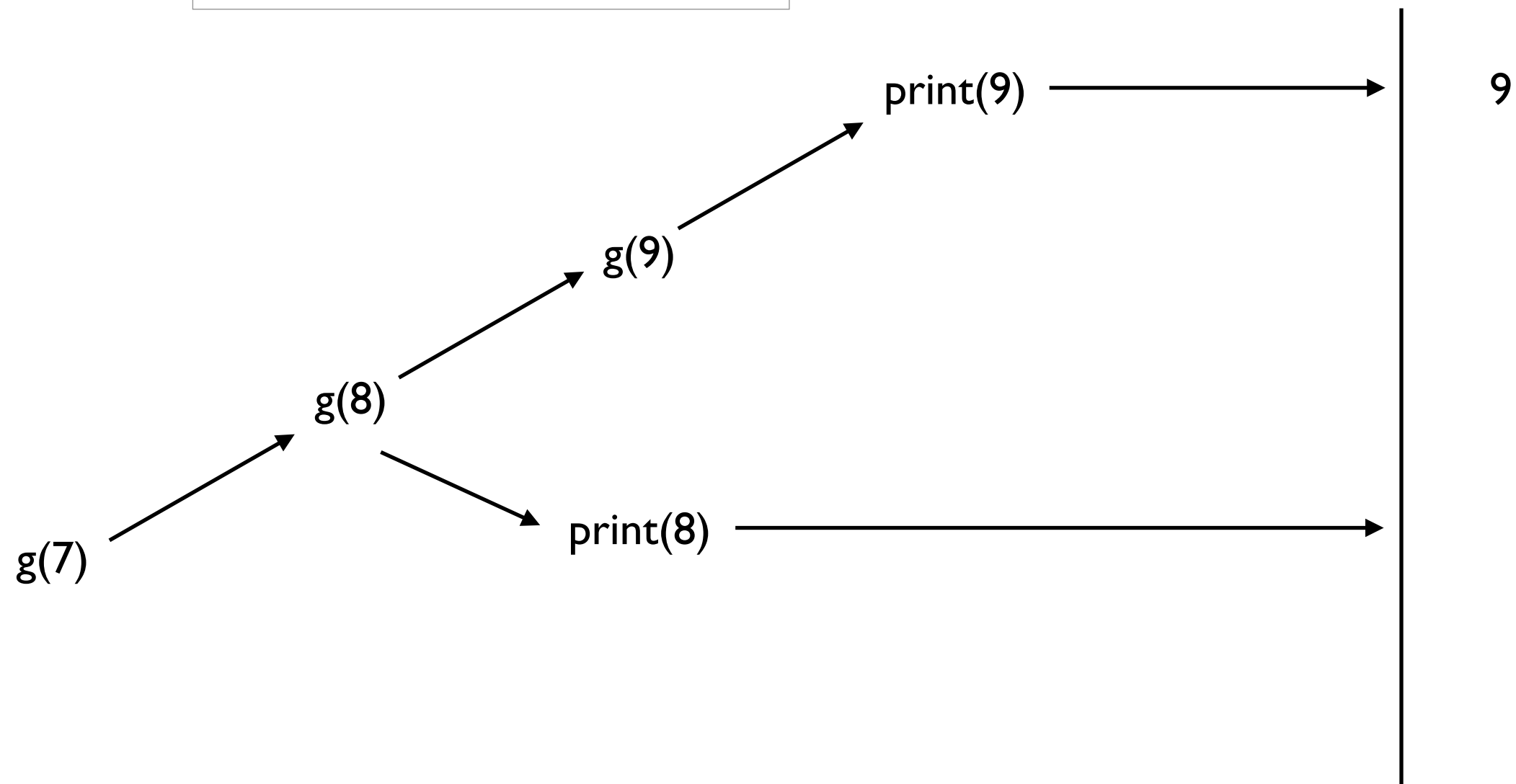


3 b

```
def g(n):  
    if n < 9:  
        g(n + 1)  
    print(n)
```

what does g(7)
print?

Timeline

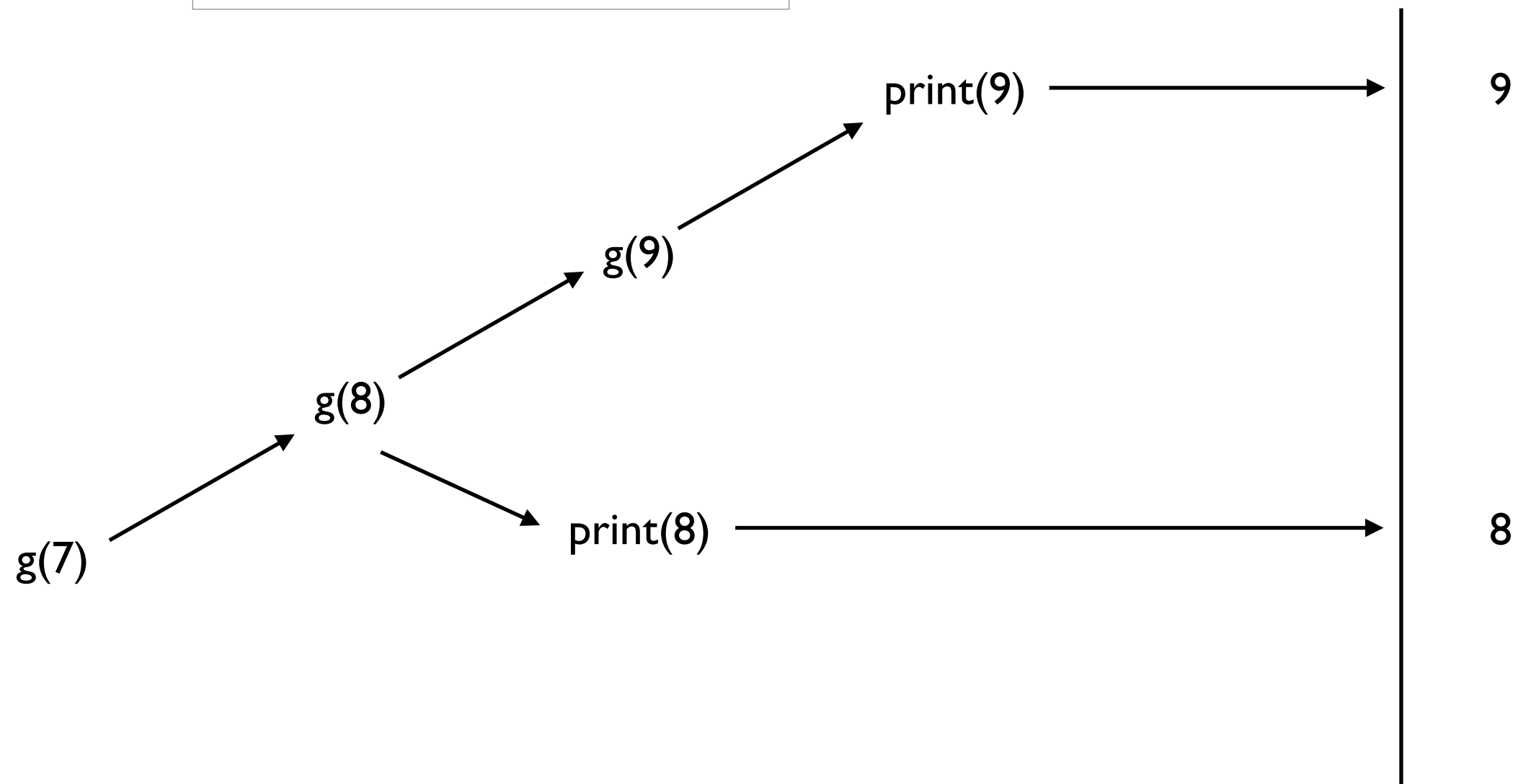


3 b

```
def g(n):  
    if n < 9:  
        g(n + 1)  
    print(n)
```

what does g(7)
print?

Timeline

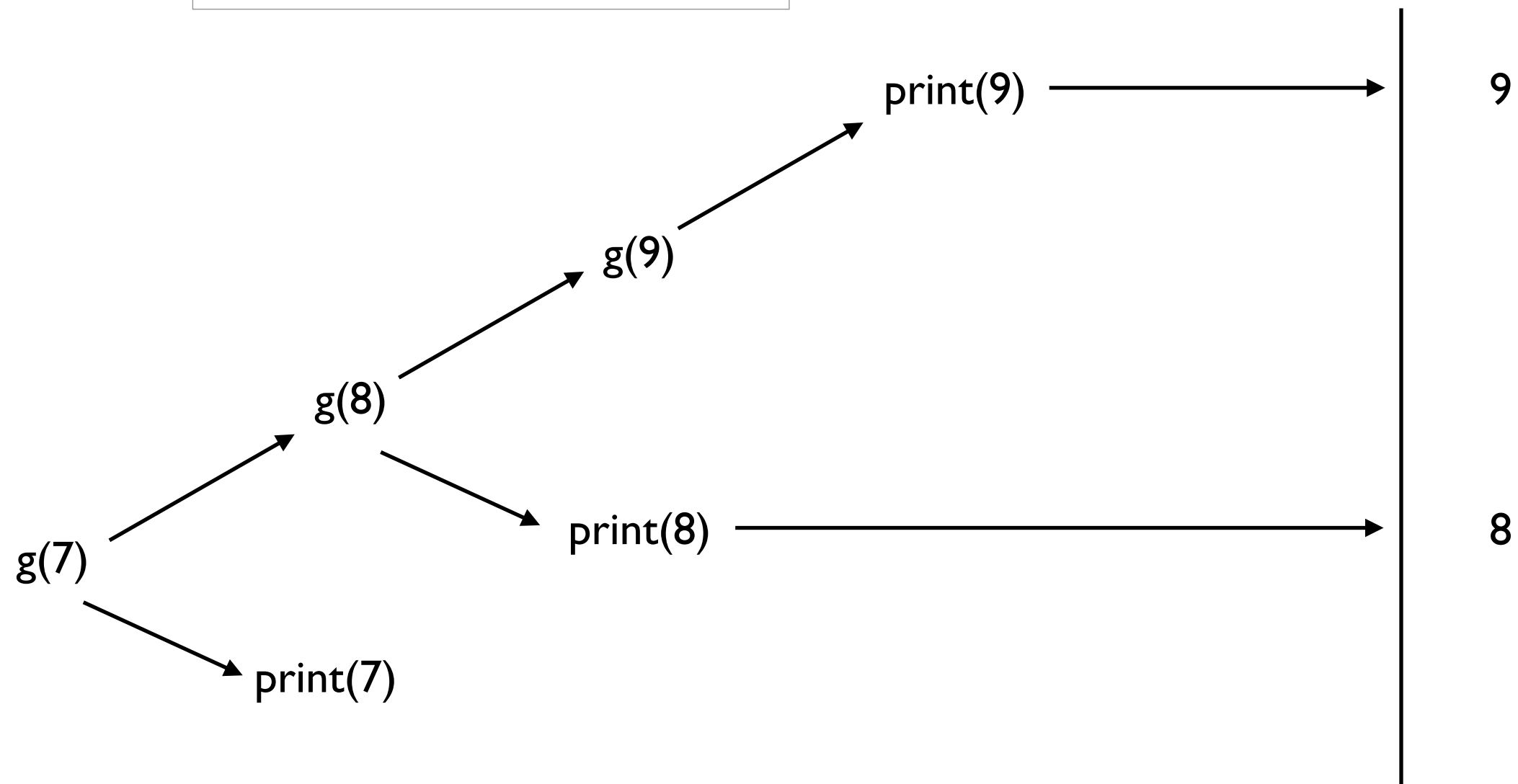


3 b

```
def g(n):  
    if n < 9:  
        g(n + 1)  
    print(n)
```

what does g(7)
print?

Timeline

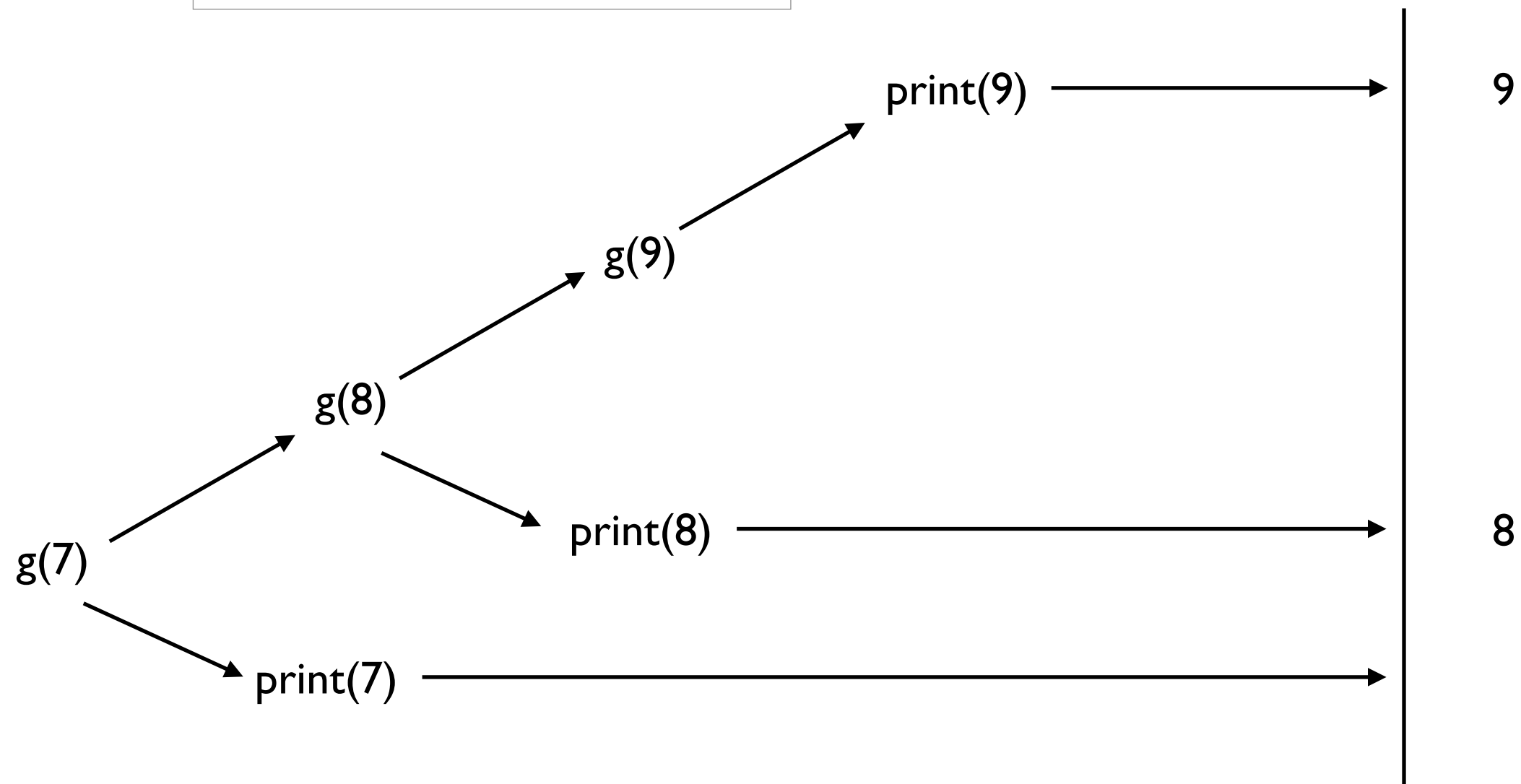


3 b

```
def g(n):  
    if n < 9:  
        g(n + 1)  
    print(n)
```

what does g(7)
print?

Timeline

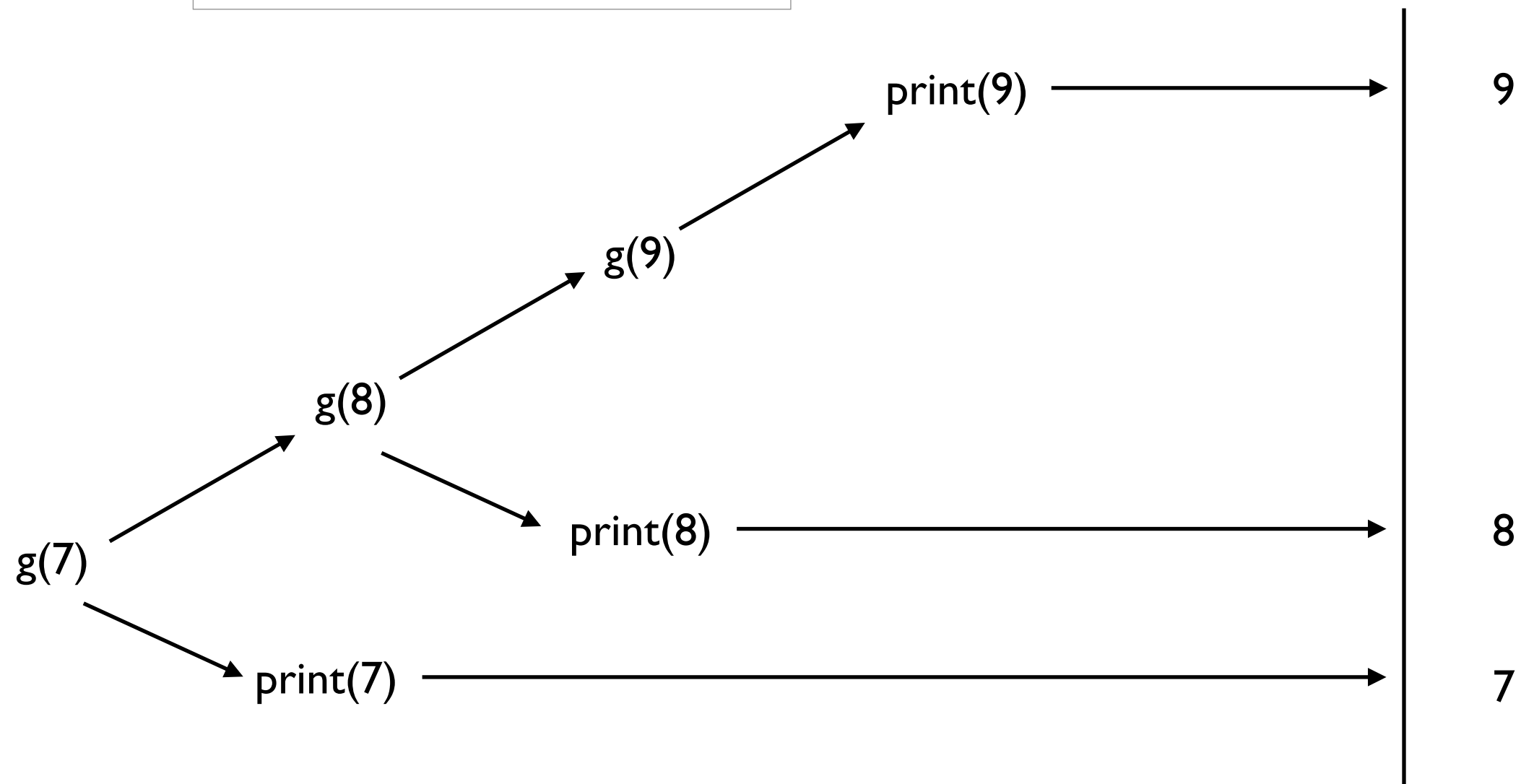


3 b

```
def g(n):  
    if n < 9:  
        g(n + 1)  
    print(n)
```

what does g(7)
print?

Timeline

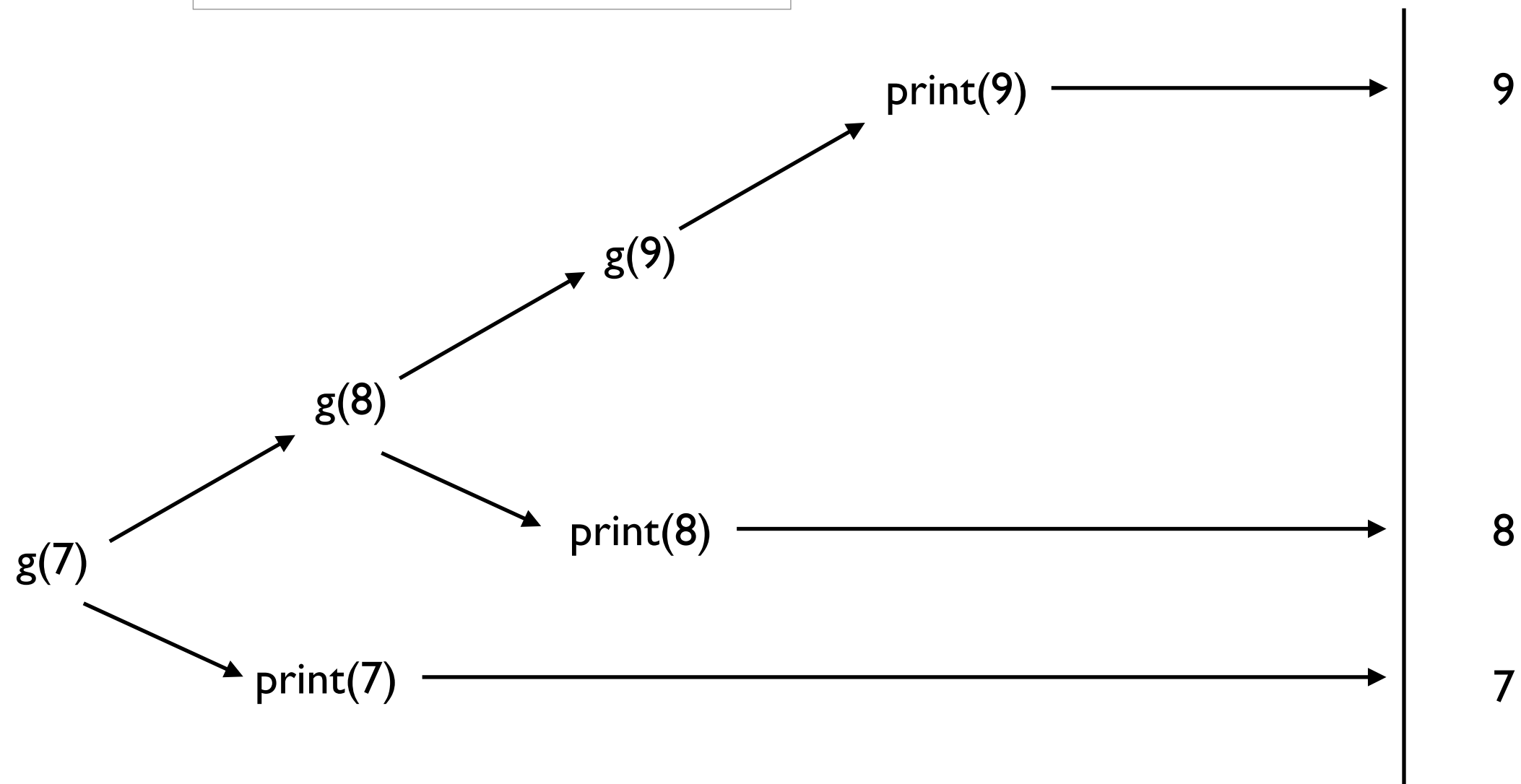


3 b

```
def g(n):  
    if n < 9:  
        g(n + 1)  
    print(n)
```

what does g(7)
print?

Timeline



Answer: 9, 8, 7

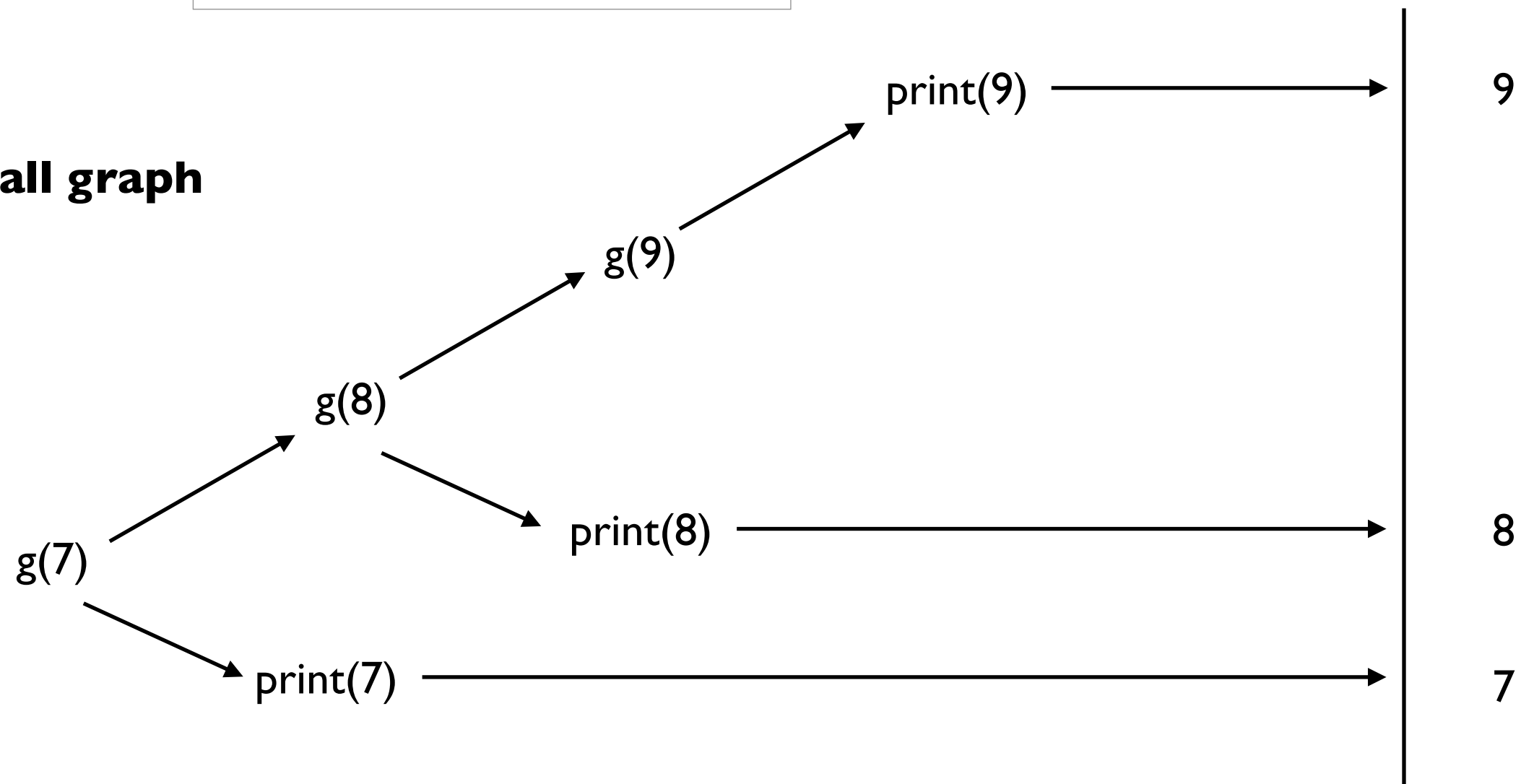
3 b

```
def g(n):  
    if n < 9:  
        g(n + 1)  
    print(n)
```

```
# what does g(7)  
print?
```

Timeline

Call graph



Answer: 9, 8, 7

```
def M(n):  
    print(n)  
    if n > 1:  
        M(n-1)  
    print(n)
```

what does M(3)
print?

```
B = []  
def h(A):  
    if len(A) > 0:  
        h(A[1:])
```

```
B.append(A[0])  
h([2, 5, 6, 3])  
# what is in B?
```

4 a

```
def M(n):  
    print(n)  
    if n > 1:  
        M(n-1)  
        print(n)
```

what does M(3)
print?

4 a

```
def M(n):  
    print(n)  
    if n > 1:  
        M(n-1)  
        print(n)
```

```
# what does M(3)  
print?
```

M(3)

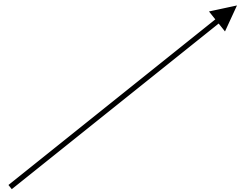
4 a

```
def M(n):  
    print(n)  
    if n > 1:  
        M(n-1)  
        print(n)
```

what does M(3)
print?

M(3)

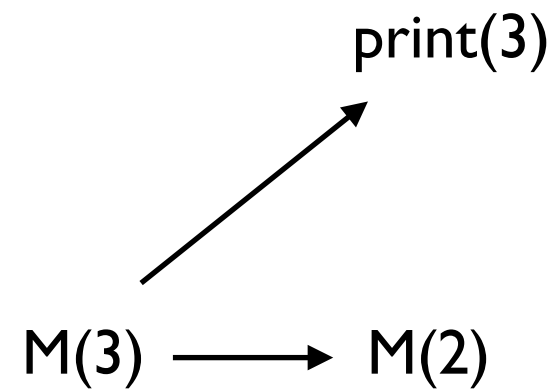
print(3)



4 a

```
def M(n):  
    print(n)  
    if n > 1:  
        M(n-1)  
    print(n)
```

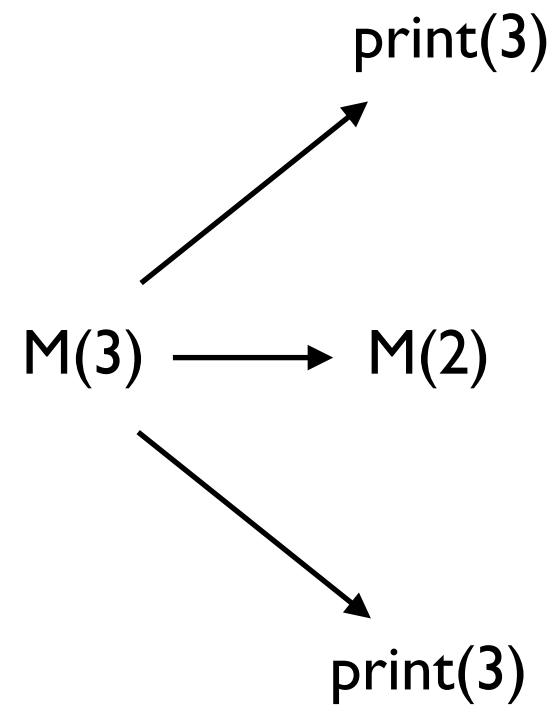
what does M(3)
print?



4 a

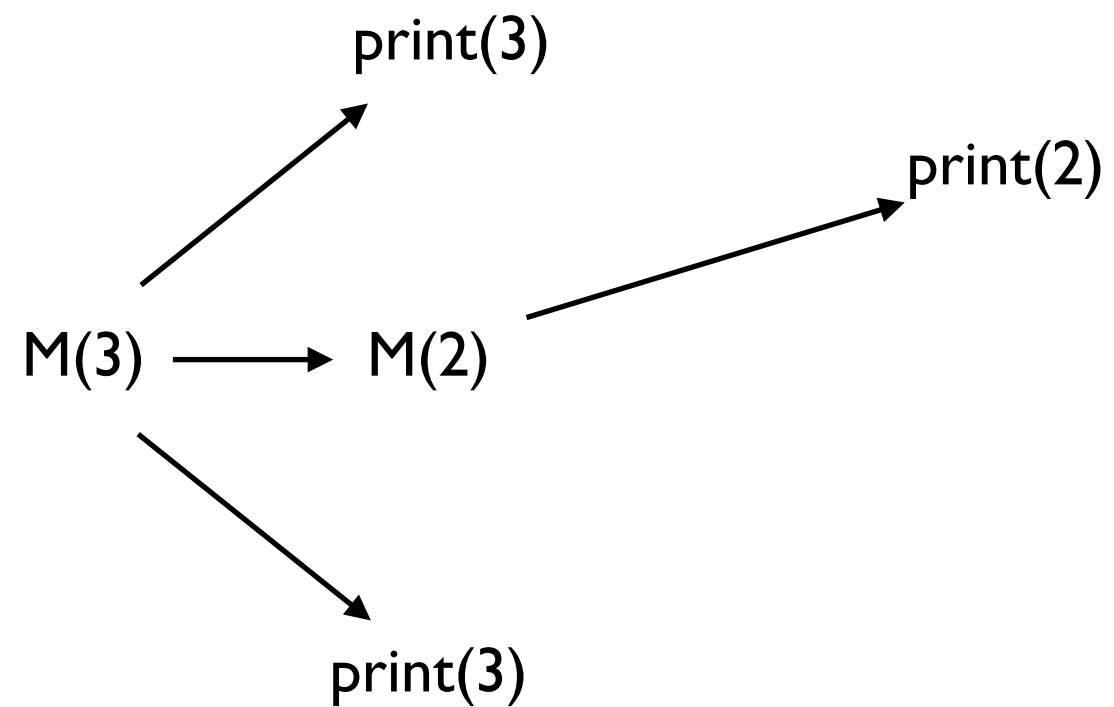
```
def M(n):  
    print(n)  
    if n > 1:  
        M(n-1)  
    print(n)
```

what does M(3)
print?



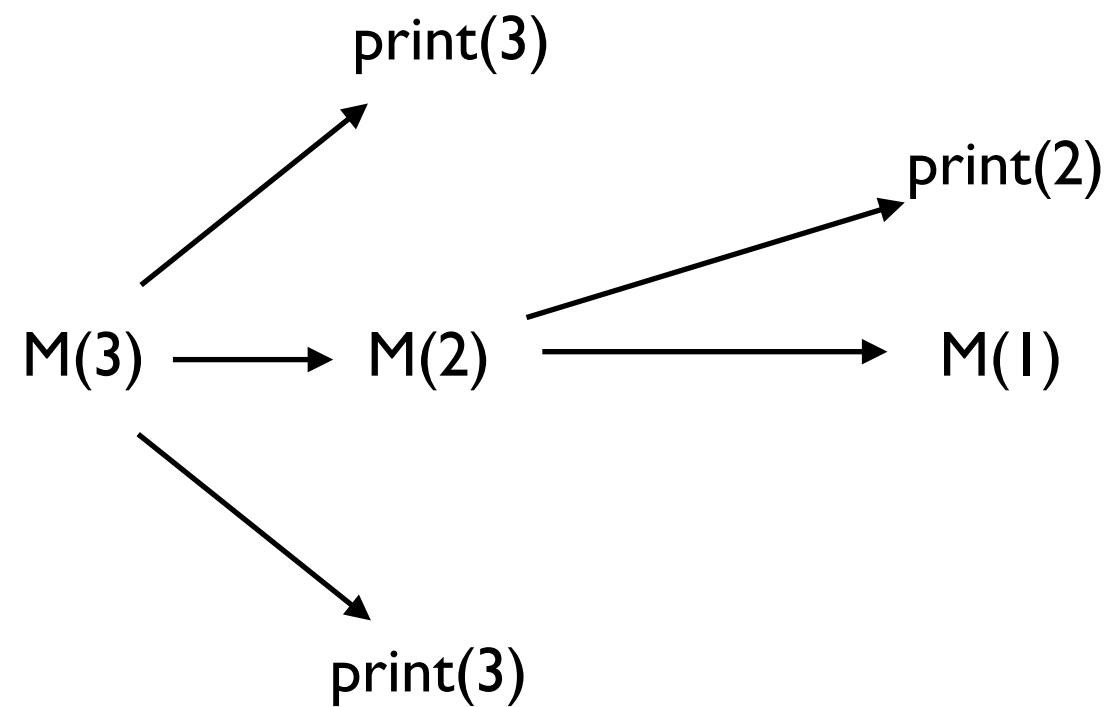
4 a

```
def M(n):  
    print(n)  
    if n > 1:  
        M(n-1)  
    print(n)  
  
# what does M(3)  
print?
```



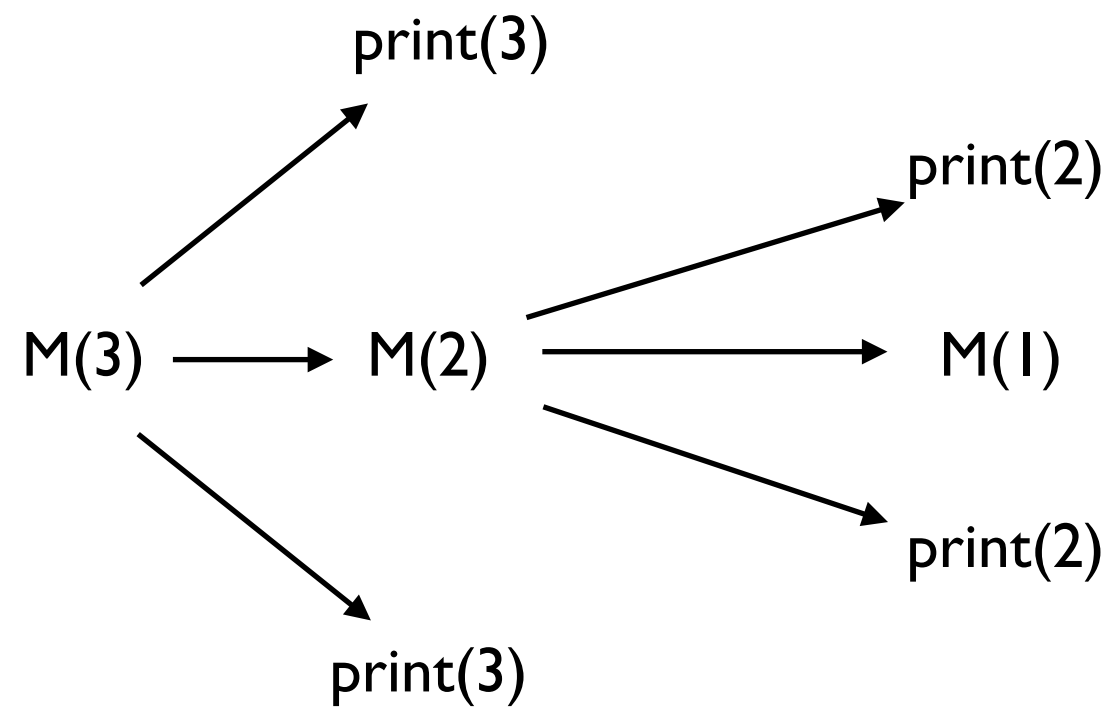
4 a

```
def M(n):  
    print(n)  
    if n > 1:  
        M(n-1)  
    print(n)  
  
# what does M(3)  
print?
```



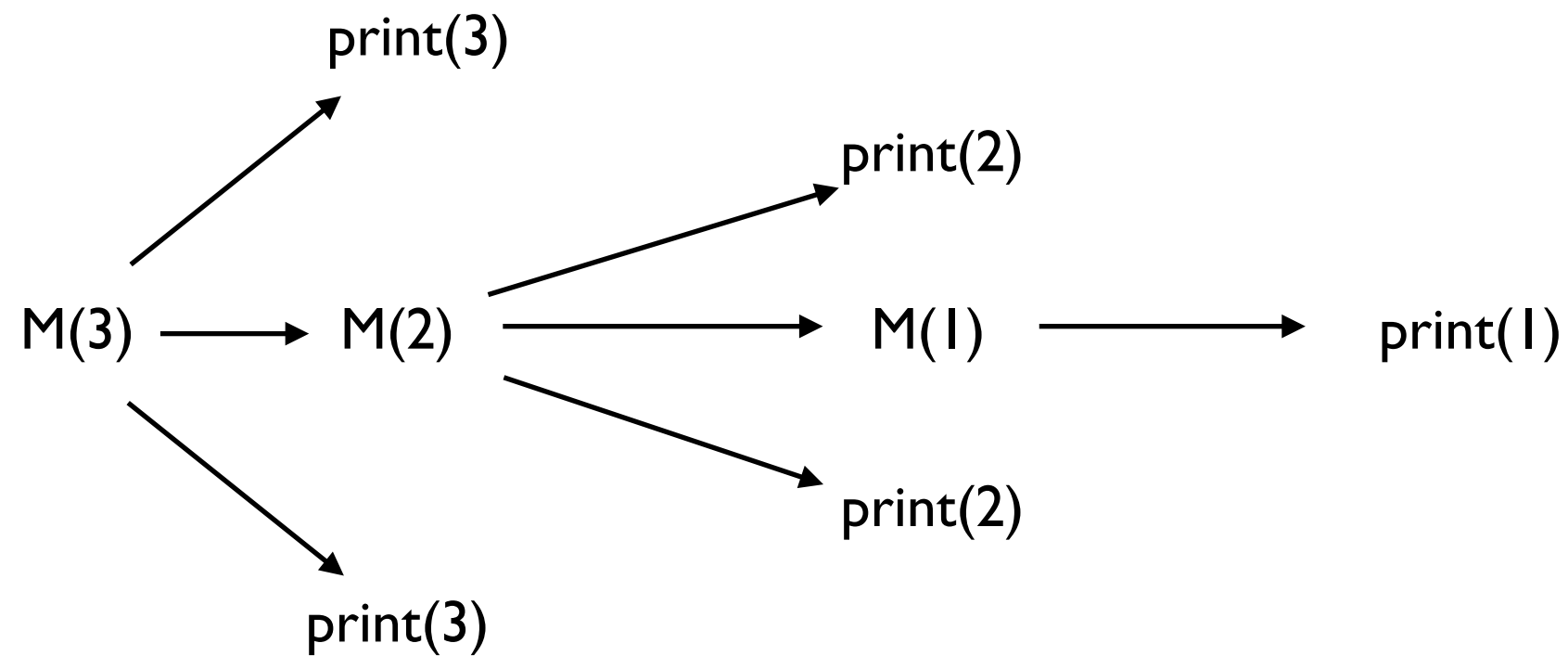
4 a

```
def M(n):  
    print(n)  
    if n > 1:  
        M(n-1)  
    print(n)  
  
# what does M(3)  
print?
```



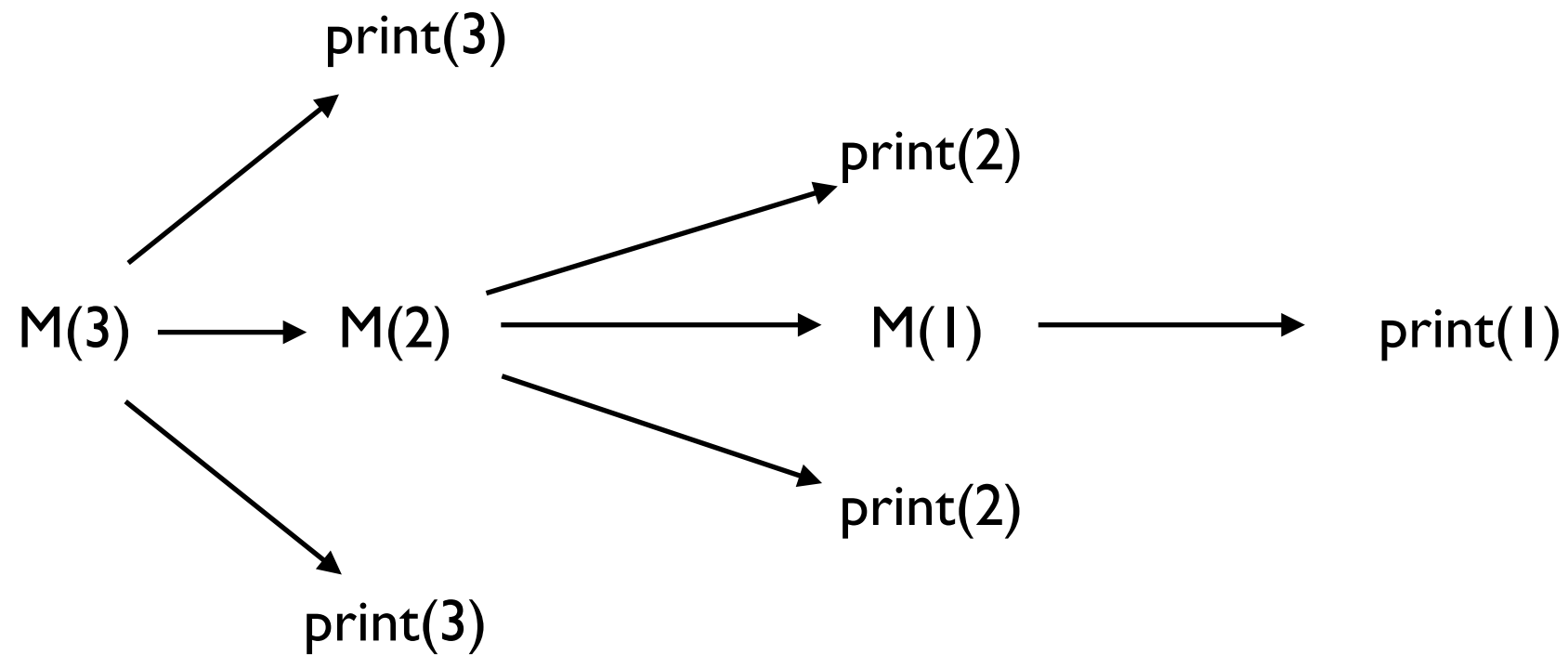
4 a

```
def M(n):  
    print(n)  
    if n > 1:  
        M(n-1)  
    print(n)  
  
# what does M(3)  
print?
```



4 a

```
def M(n):  
    print(n)  
    if n > 1:  
        M(n-1)  
    print(n)  
  
# what does M(3)  
print?
```

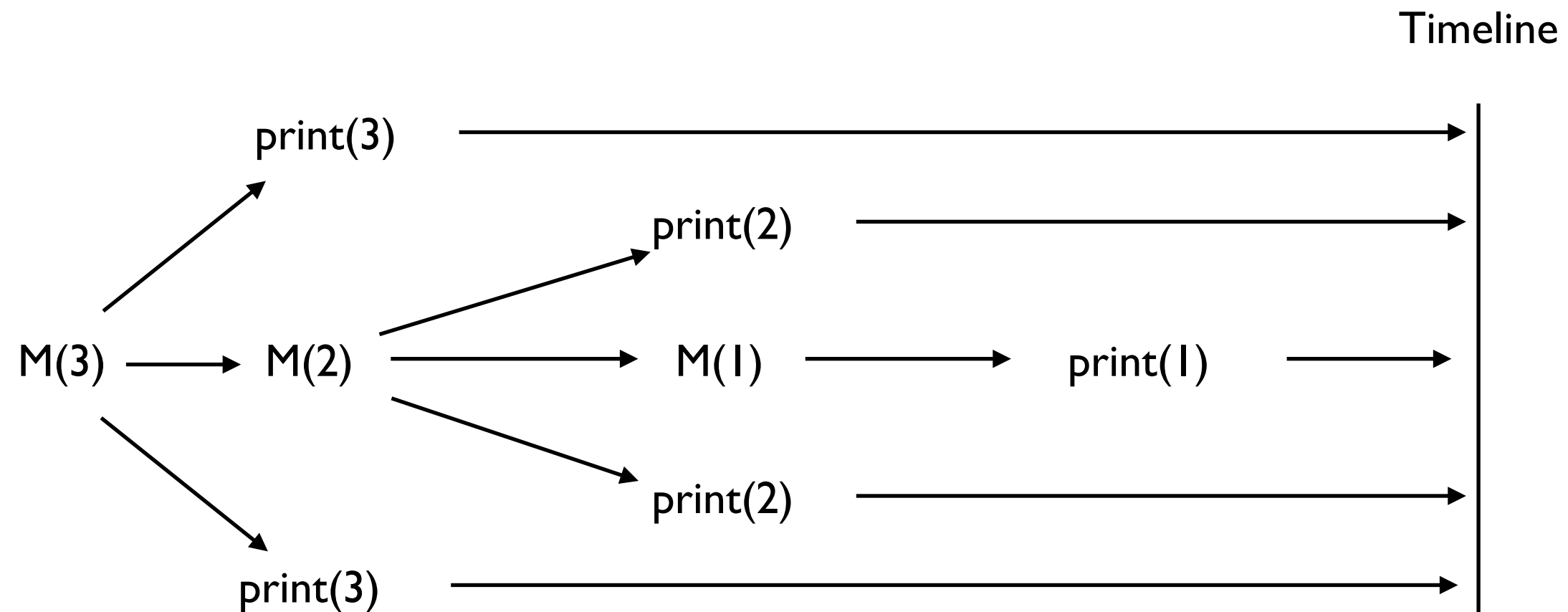


Timeline



4 a

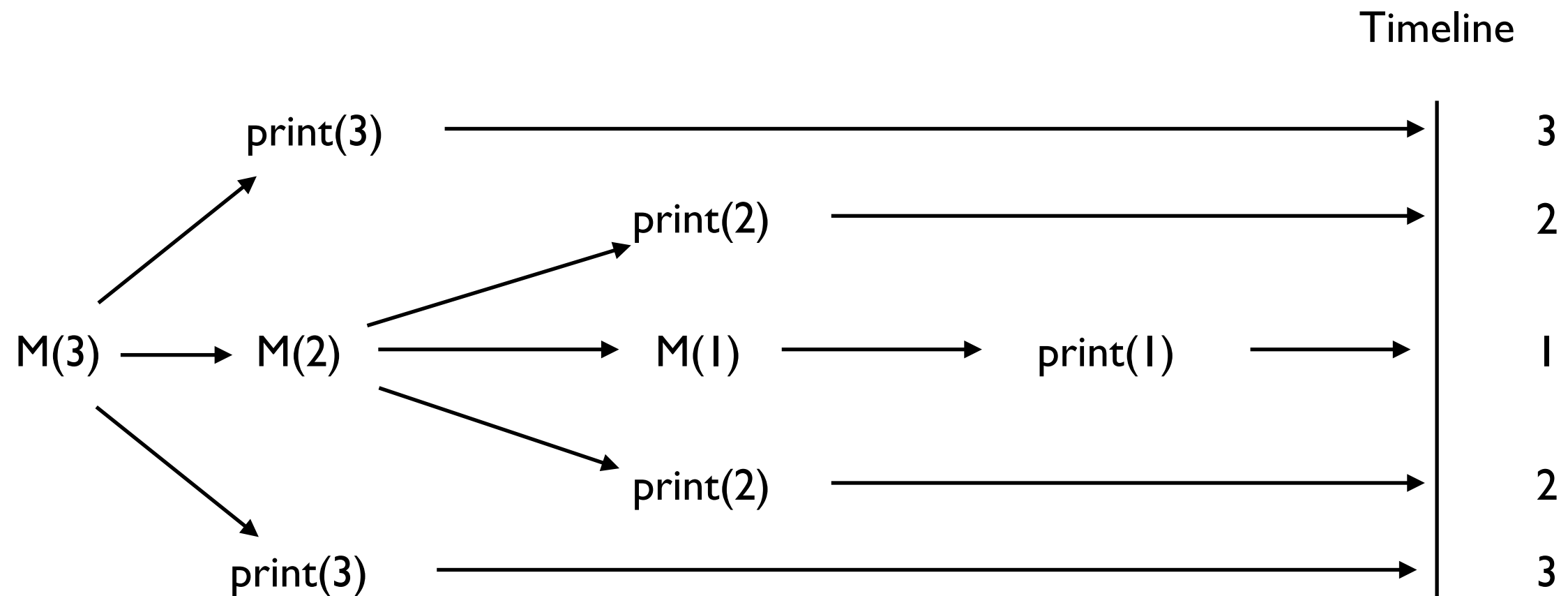
```
def M(n):  
    print(n)  
    if n > 1:  
        M(n-1)  
    print(n)  
  
# what does M(3)  
print?
```



4 a

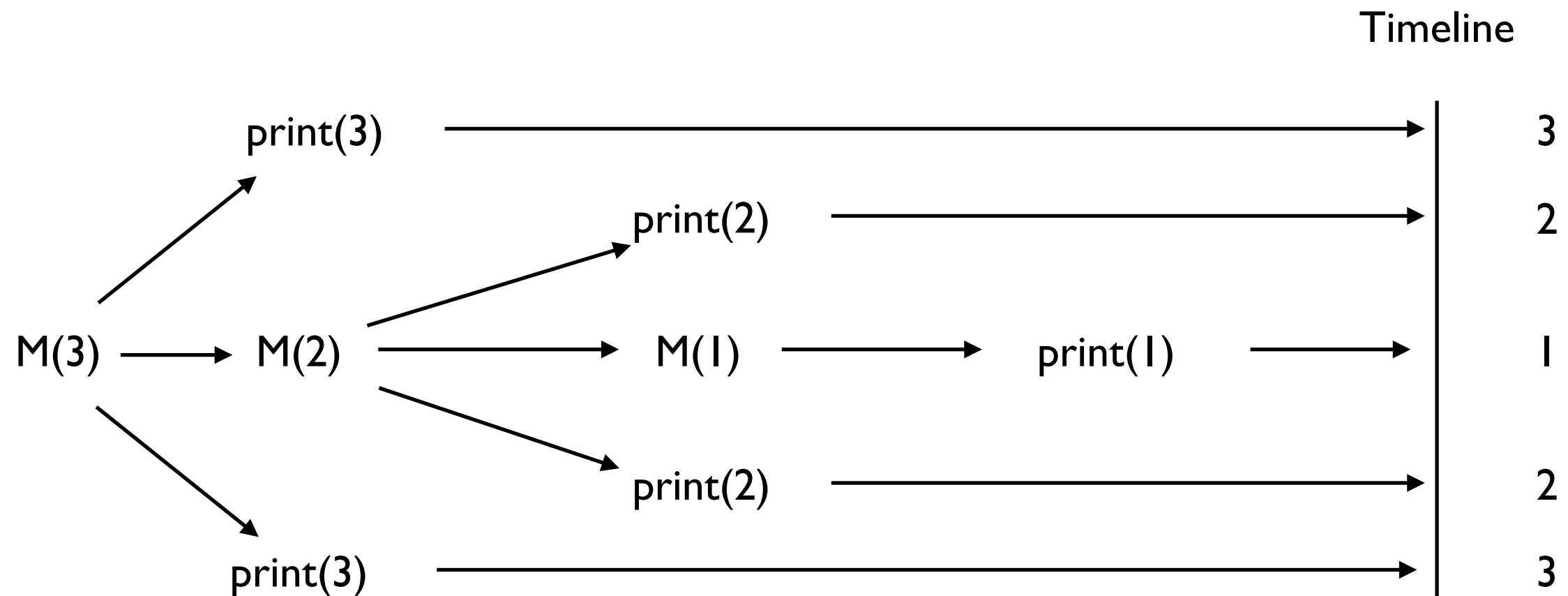
```
def M(n):  
    print(n)  
    if n > 1:  
        M(n-1)  
    print(n)
```

what does M(3)
print?



4 a

```
def M(n):  
    print(n)  
    if n > 1:  
        M(n-1)  
    print(n)  
  
# what does M(3)  
print?
```

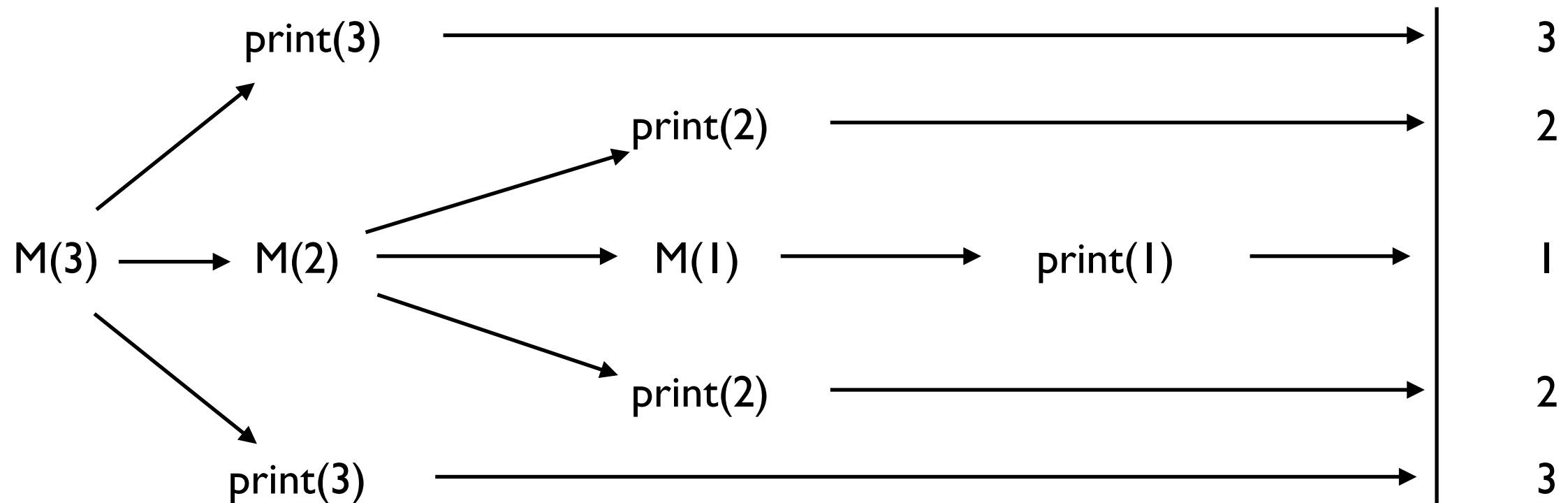


Answer: 3, 2, 1, 2, 3

4 a

```
def M(n):  
    print(n)  
    if n > 1:  
        M(n-1)  
    print(n)  
  
# what does M(3)  
print?
```

Call graph



Answer: 3, 2, 1, 2, 3

4 b

```
B = []
def h(A):
    if len(A) > 0:
        h(A[1:])
        B.append(A[0])
h([2, 5, 6, 3])
# what is in B?
```

4 b

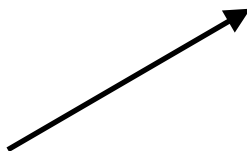
```
B = []
def h(A):
    if len(A) > 0:
        h(A[1:])
        B.append(A[0])
h([2, 5, 6, 3])
# what is in B?
```

h([2, 5, 6, 3])

4 b

```
B = []  
def h(A):  
    if len(A) > 0:  
        h(A[1:])  
        B.append(A[0])  
h([2, 5, 6, 3])  
# what is in B?
```

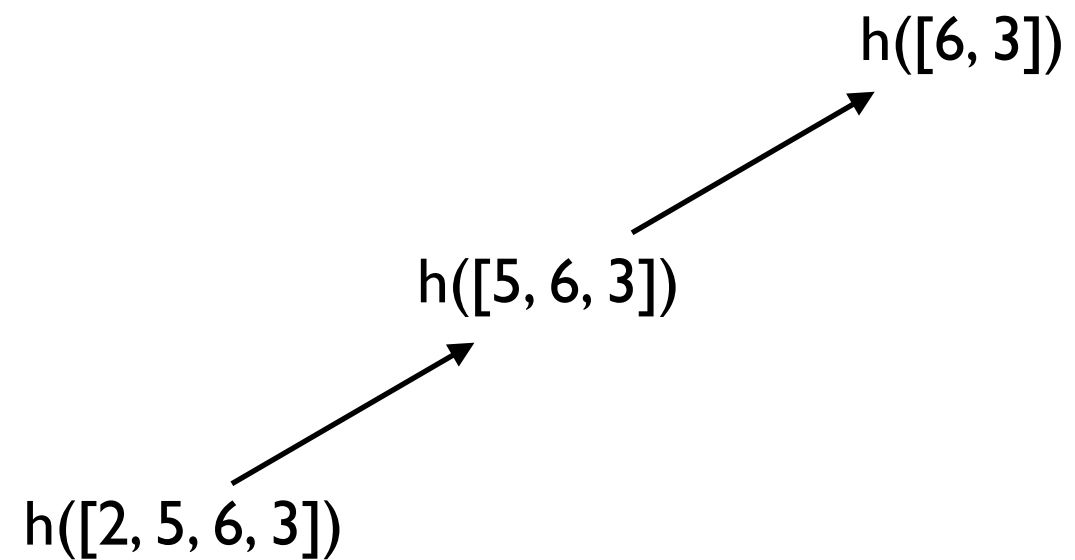
h([2, 5, 6, 3])



h([5, 6, 3])

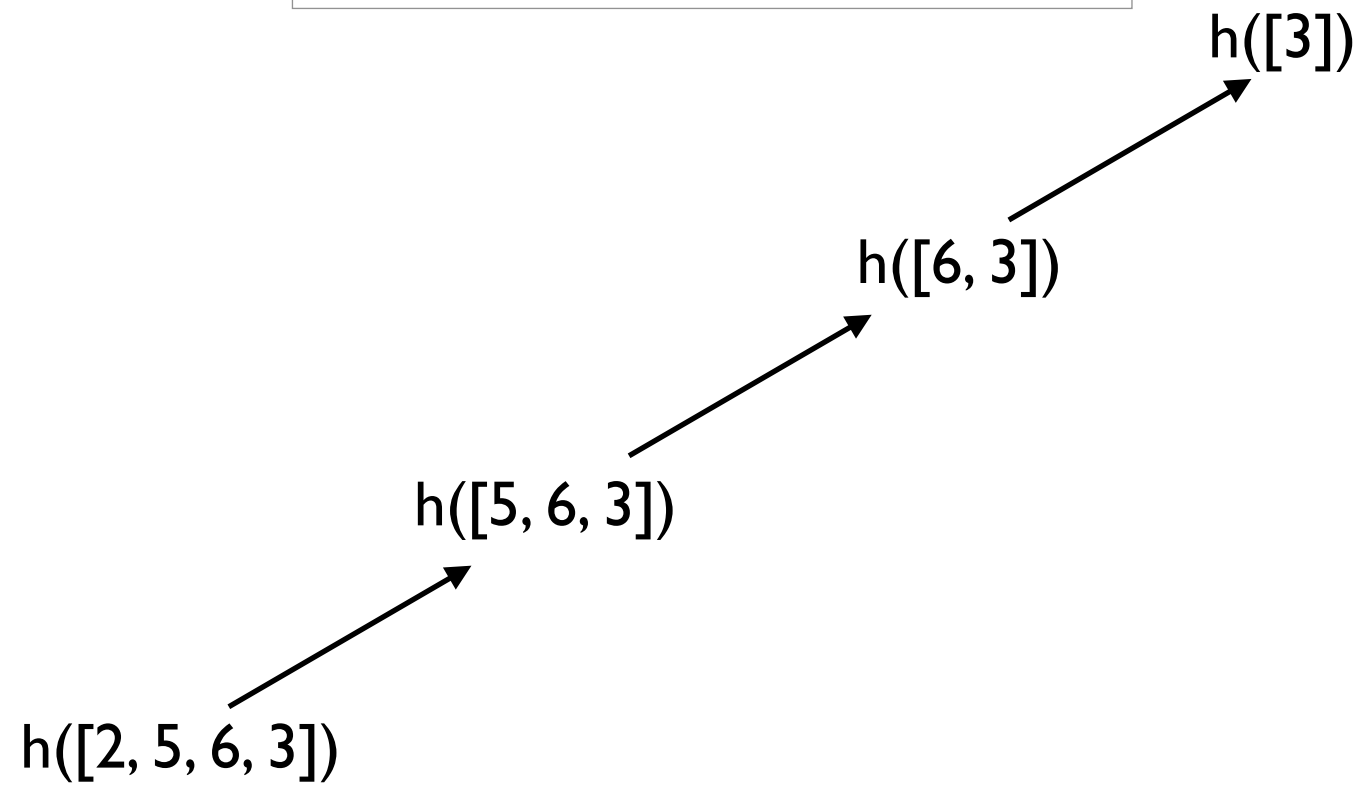
4 b

```
B = []  
def h(A):  
    if len(A) > 0:  
        h(A[1:])  
        B.append(A[0])  
h([2, 5, 6, 3])  
# what is in B?
```



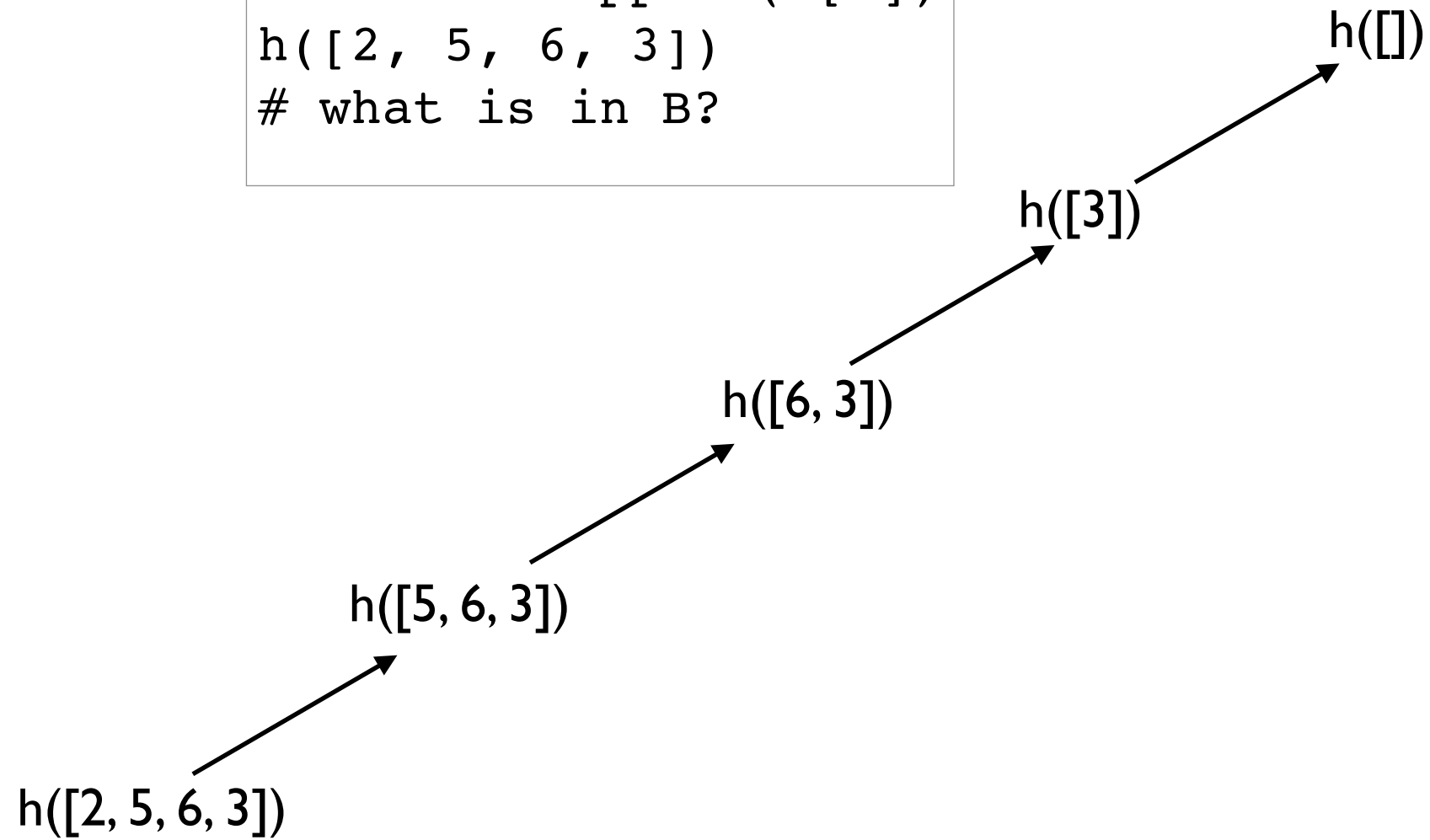
4 b

```
B = []  
def h(A):  
    if len(A) > 0:  
        h(A[1:])  
        B.append(A[0])  
h([2, 5, 6, 3])  
# what is in B?
```



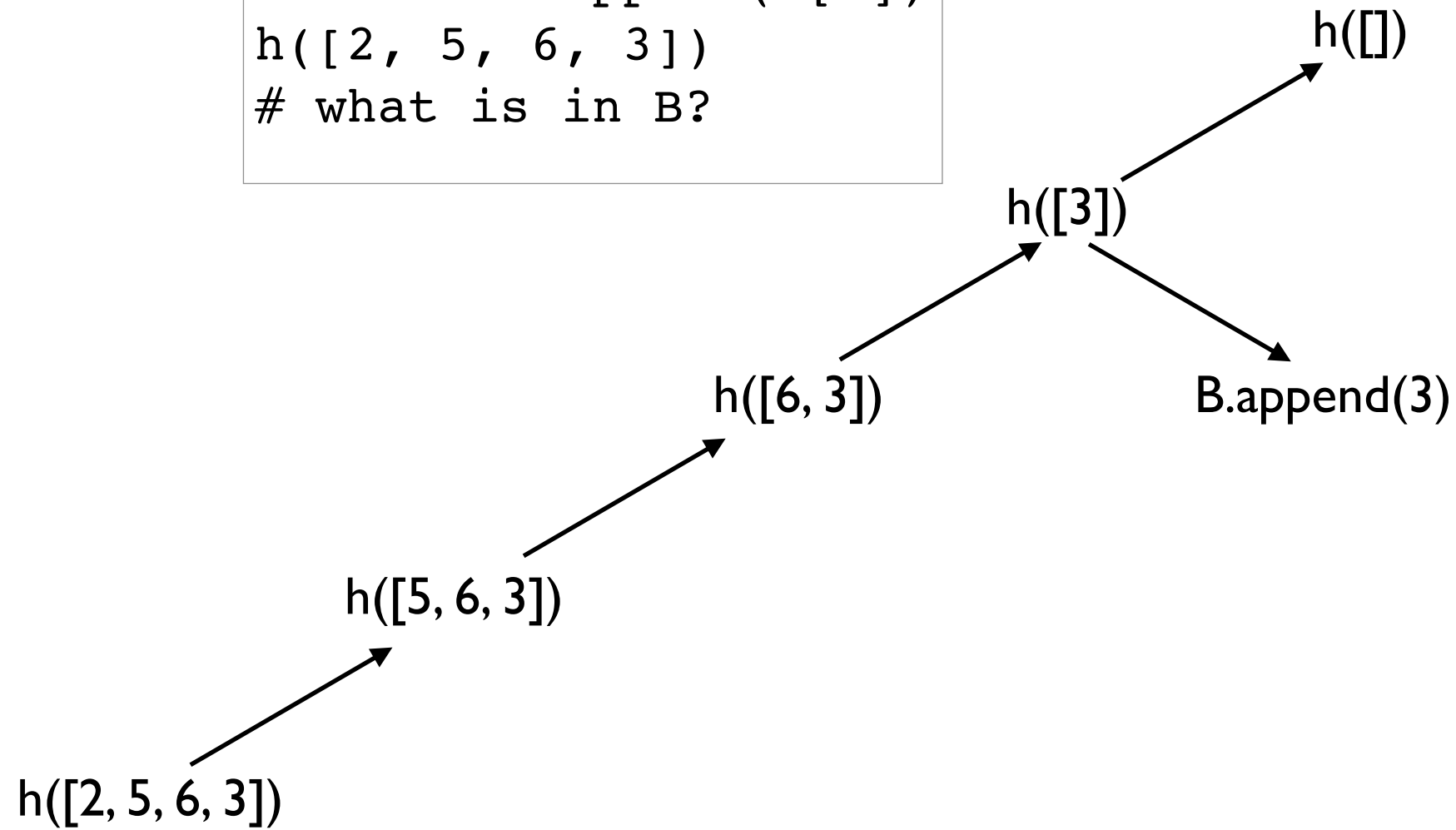
4 b

```
B = []  
def h(A):  
    if len(A) > 0:  
        h(A[1:])  
        B.append(A[0])  
h([2, 5, 6, 3])  
# what is in B?
```



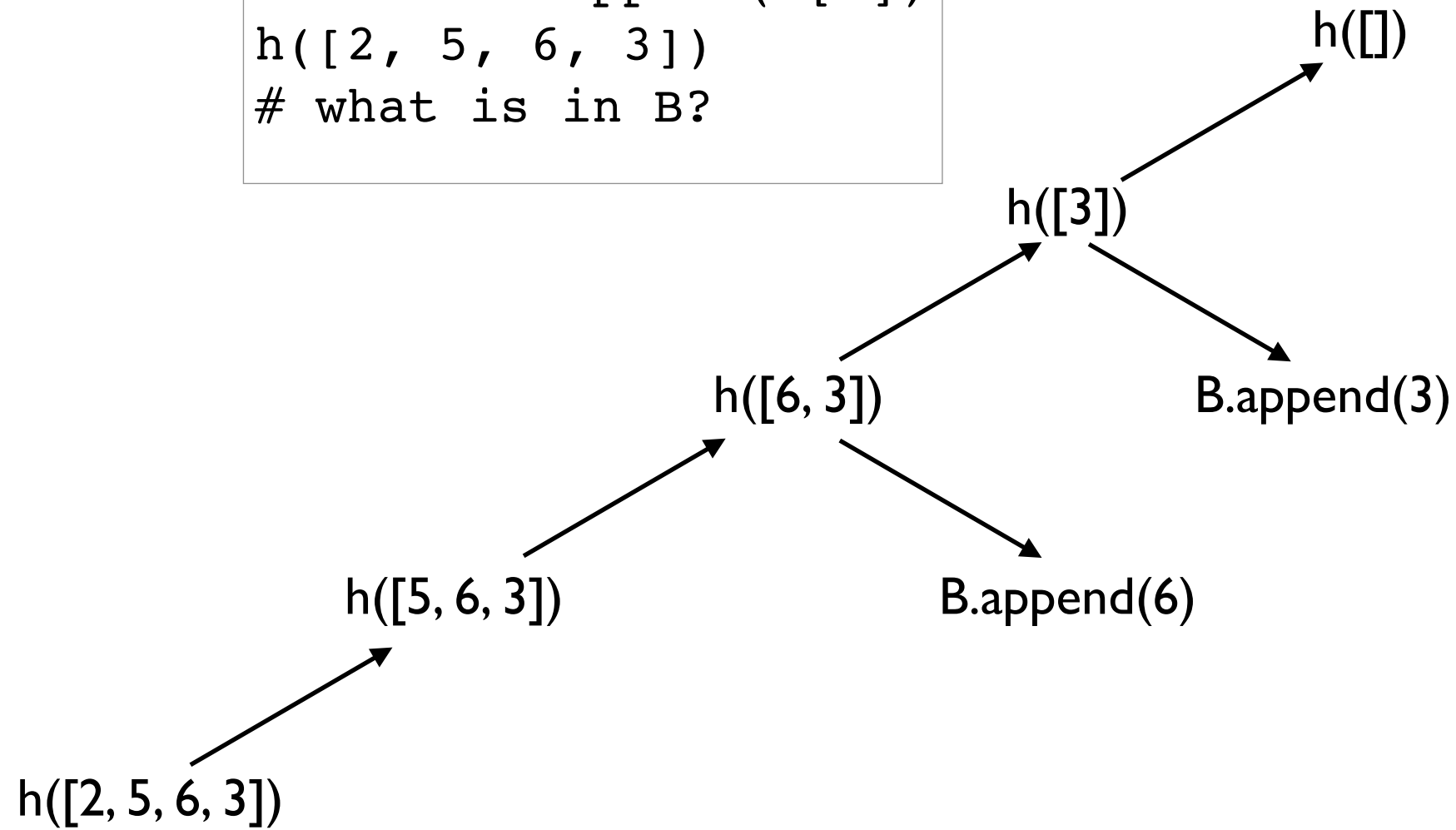
4 b

```
B = []  
def h(A):  
    if len(A) > 0:  
        h(A[1:])  
        B.append(A[0])  
h([2, 5, 6, 3])  
# what is in B?
```



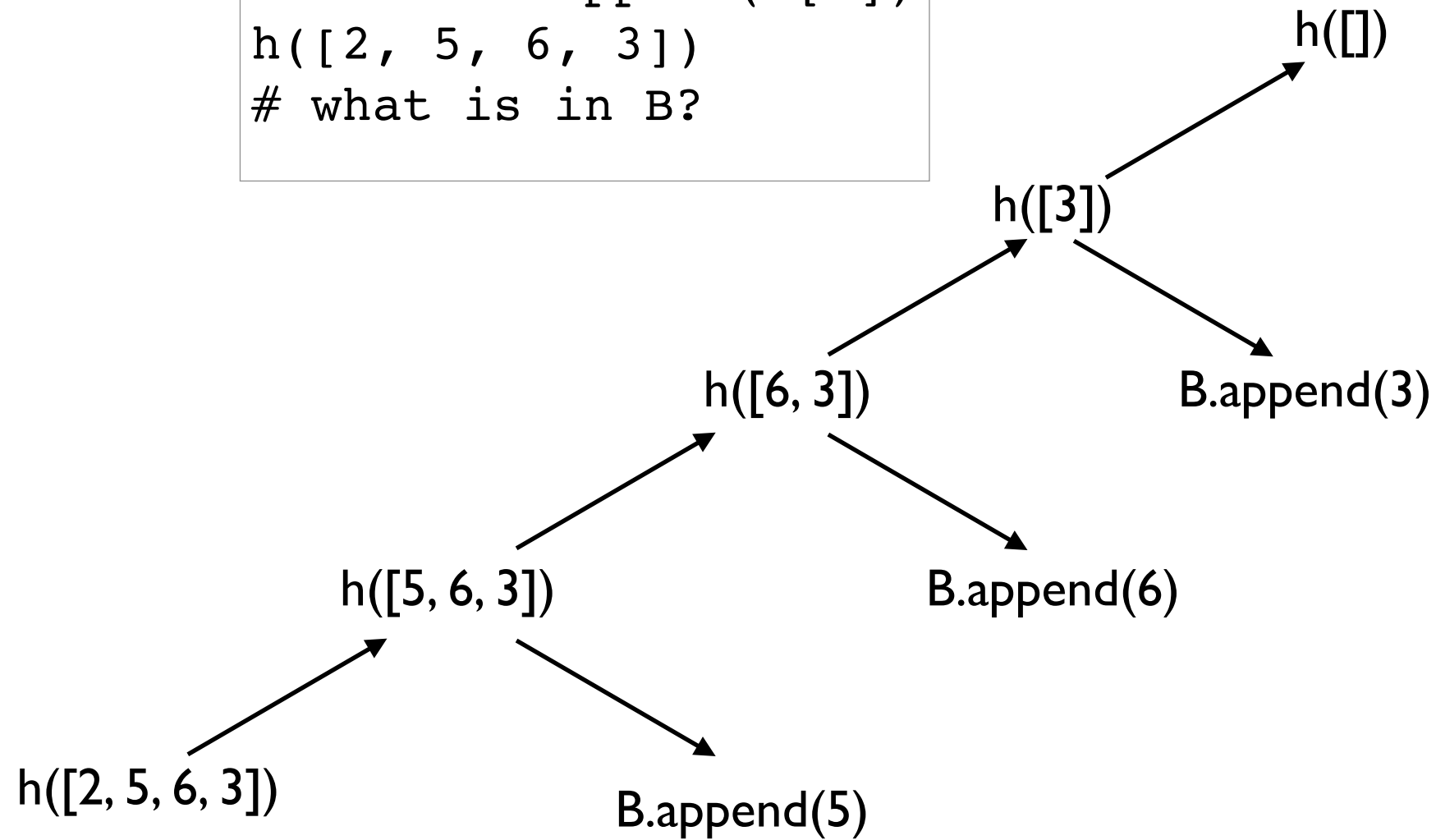
4 b

```
B = []  
def h(A):  
    if len(A) > 0:  
        h(A[1:])  
        B.append(A[0])  
h([2, 5, 6, 3])  
# what is in B?
```



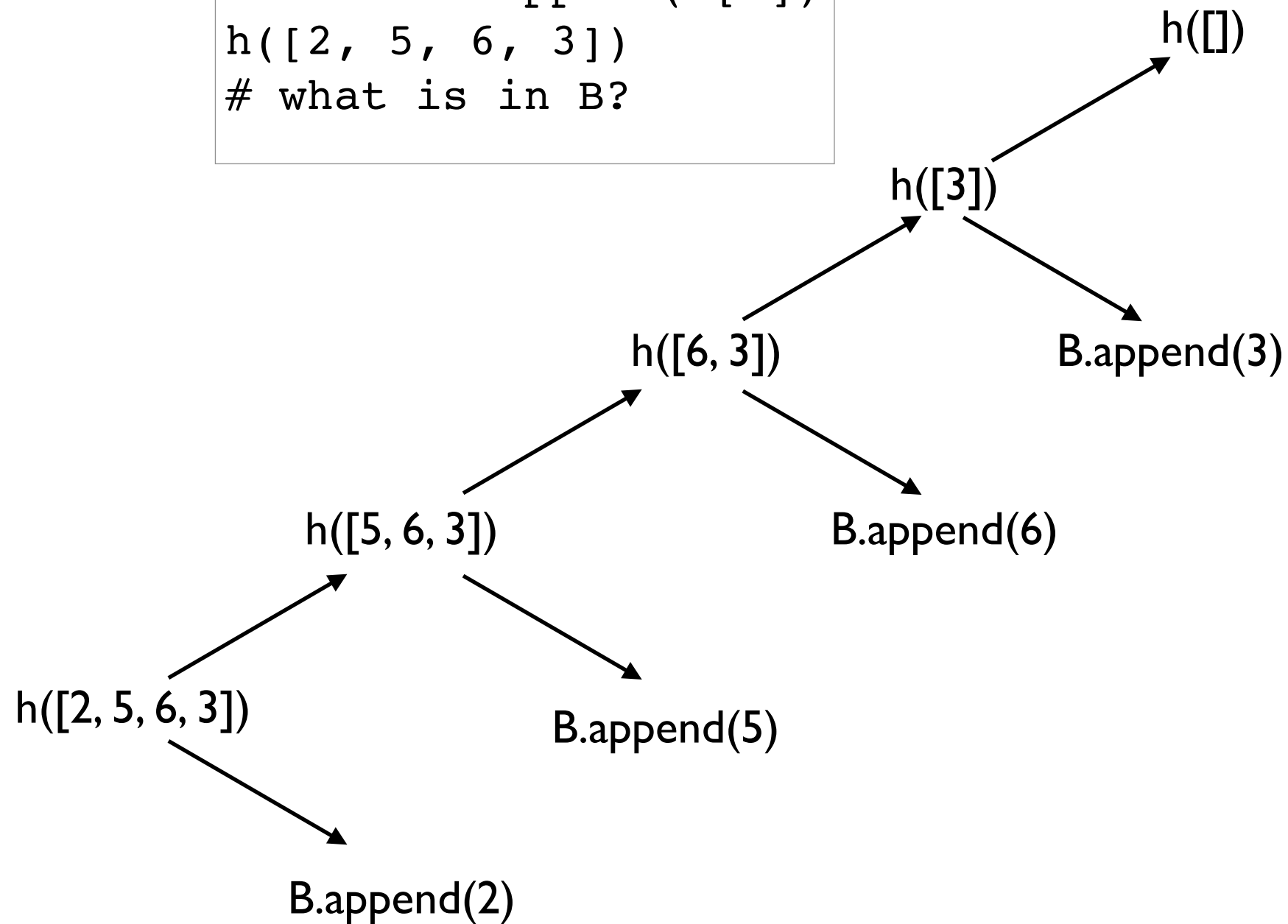
4 b

```
B = []  
def h(A):  
    if len(A) > 0:  
        h(A[1:])  
        B.append(A[0])  
h([2, 5, 6, 3])  
# what is in B?
```



4 b

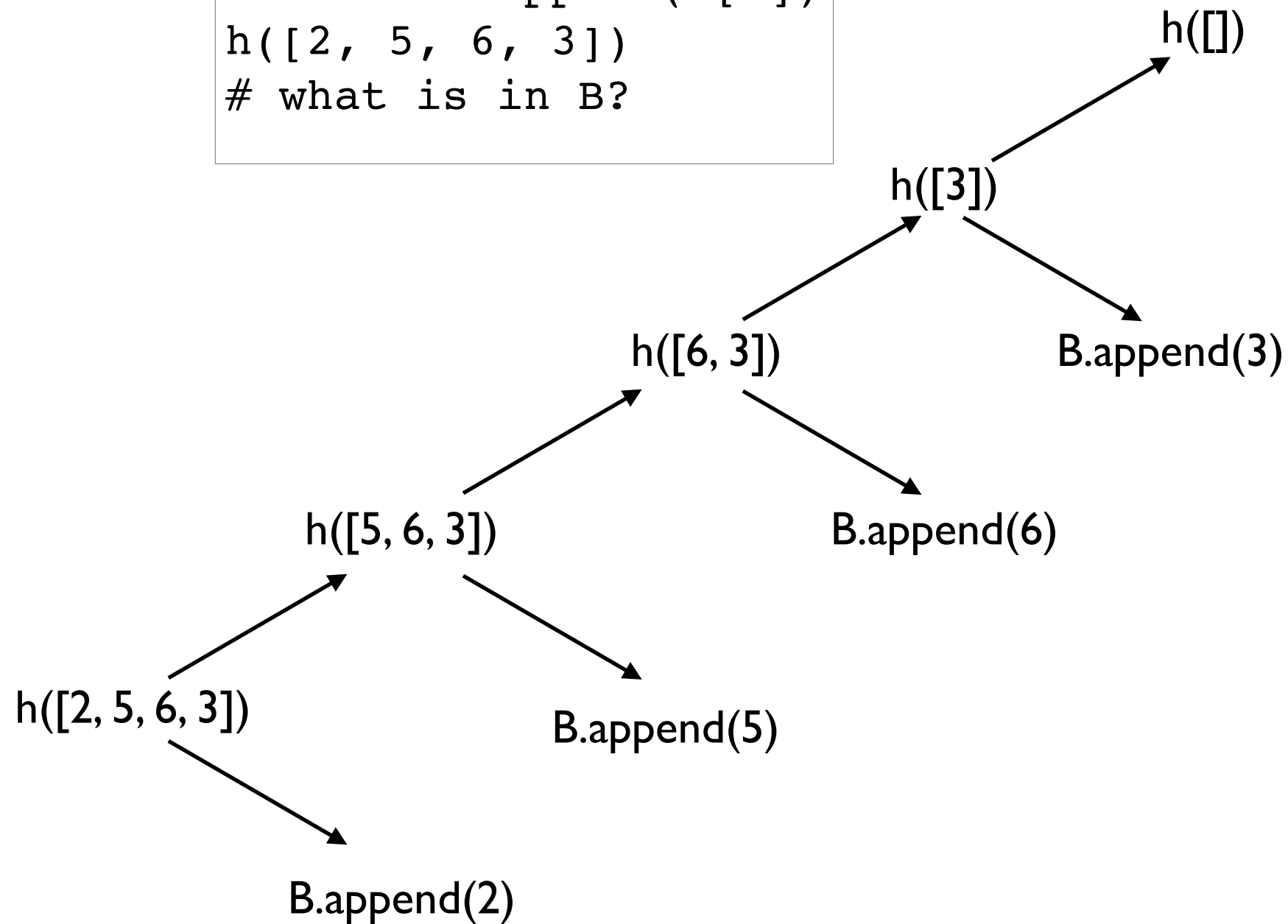
```
B = []  
def h(A):  
    if len(A) > 0:  
        h(A[1:])  
        B.append(A[0])  
h([2, 5, 6, 3])  
# what is in B?
```



4 b

```
B = []  
def h(A):  
    if len(A) > 0:  
        h(A[1:])  
        B.append(A[0])  
h([2, 5, 6, 3])  
# what is in B?
```

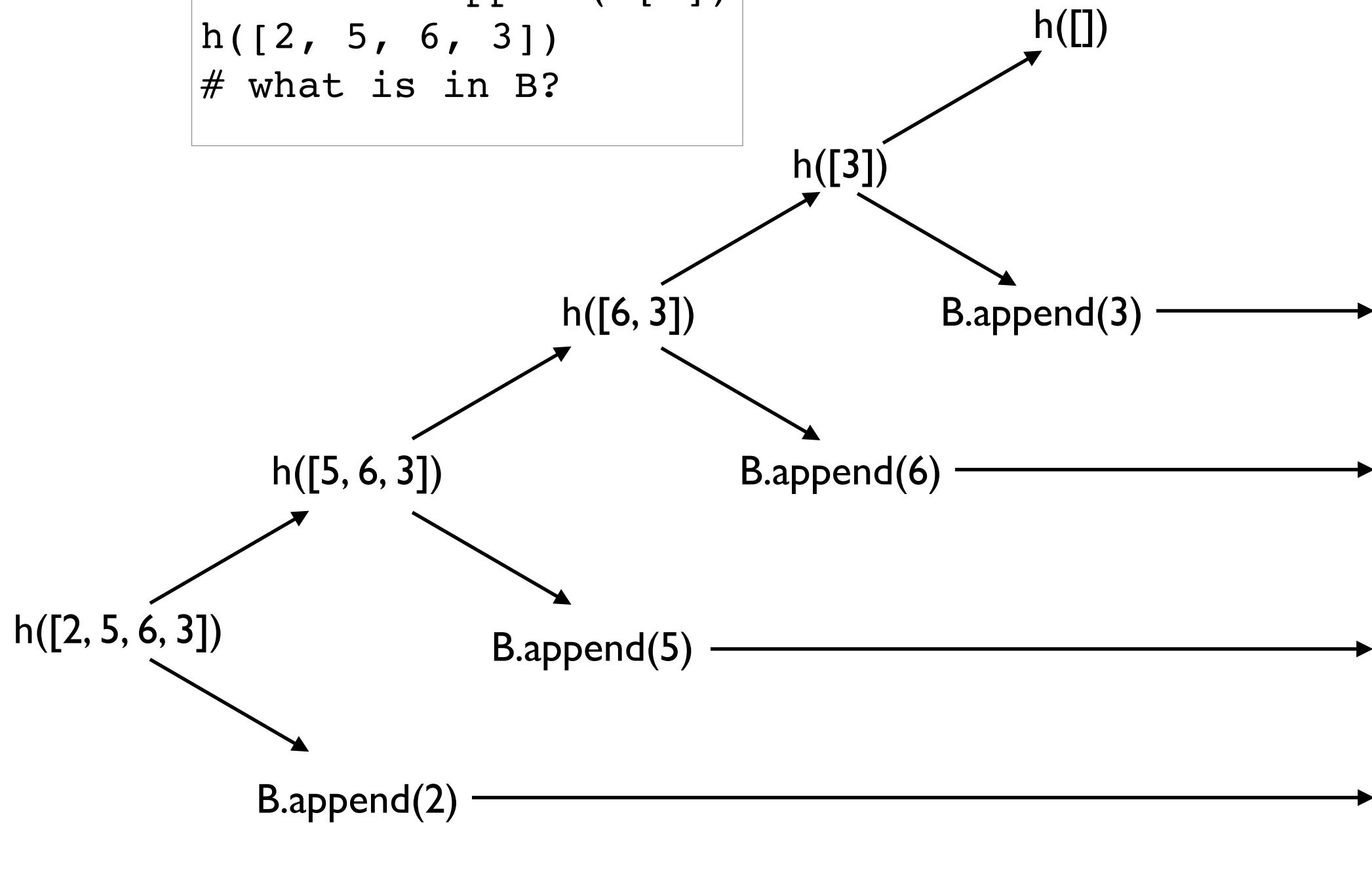
Timeline



4 b

```
B = []
def h(A):
    if len(A) > 0:
        h(A[1:])
        B.append(A[0])
h([2, 5, 6, 3])
# what is in B?
```

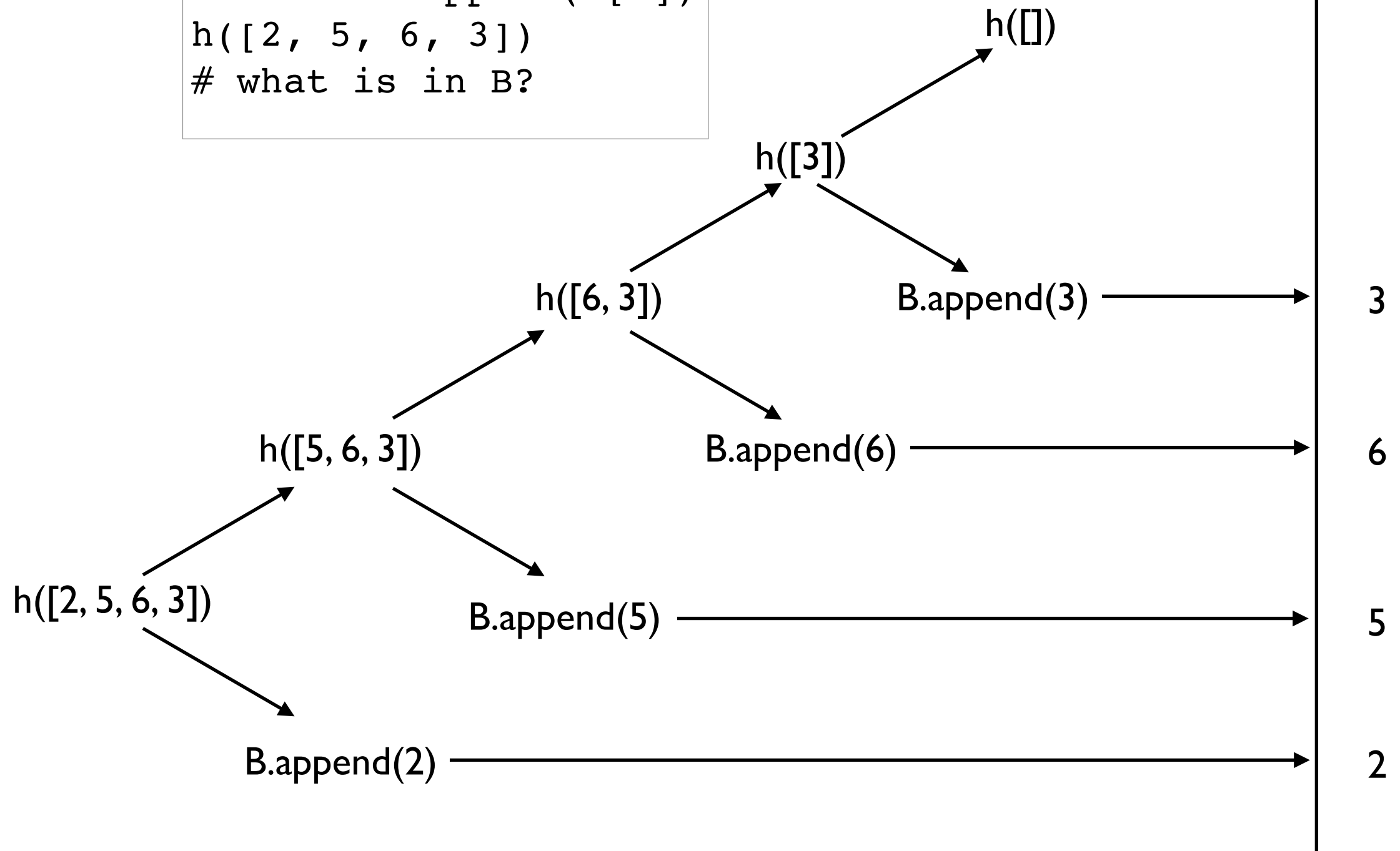
Timeline



4 b

```
B = []  
def h(A):  
    if len(A) > 0:  
        h(A[1:])  
        B.append(A[0])  
h([2, 5, 6, 3])  
# what is in B?
```

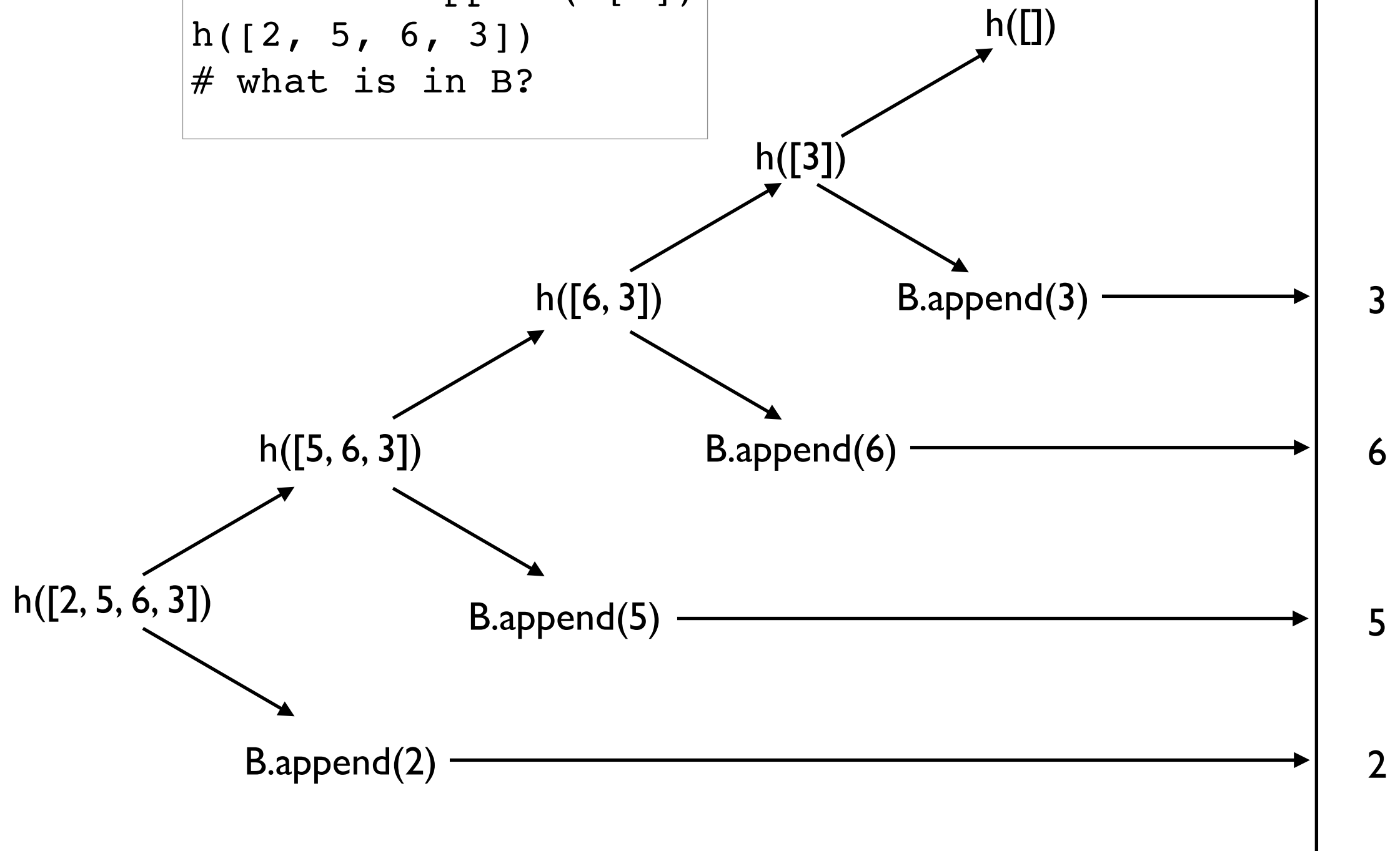
Timeline



4 b

```
B = []
def h(A):
    if len(A) > 0:
        h(A[1:])
        B.append(A[0])
h([2, 5, 6, 3])
# what is in B?
```

Timeline



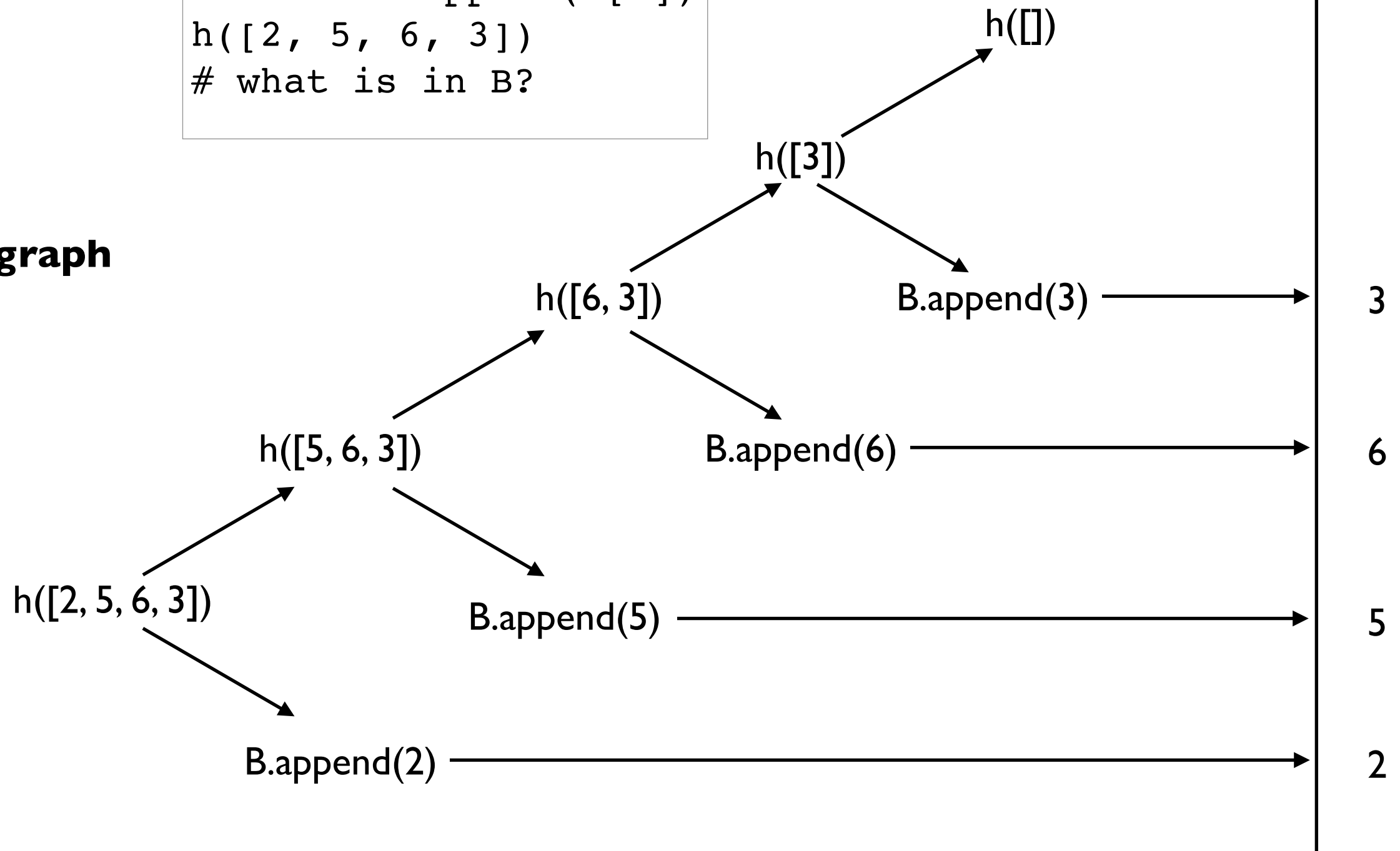
Answer: [3, 6, 5, 2]

4 b

```
B = []
def h(A):
    if len(A) > 0:
        h(A[1:])
        B.append(A[0])
h([2, 5, 6, 3])
# what is in B?
```

Call graph

Timeline



Answer: [3, 6, 5, 2]

```

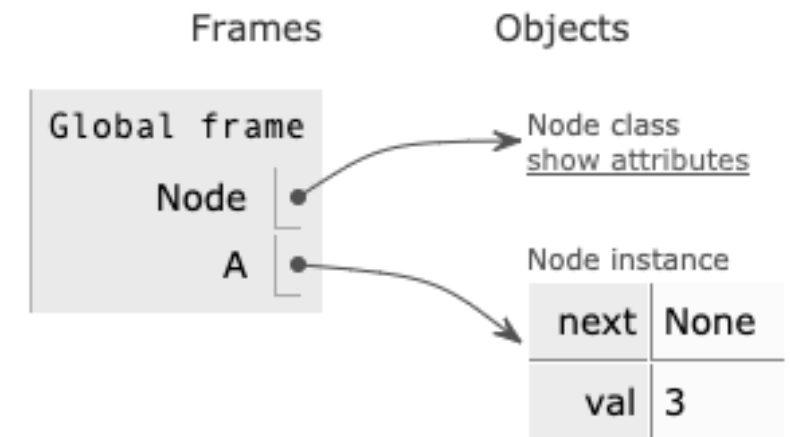
class Node:
    def __init__(self, val):
        self.val = val
        self.next = None

    def tot(self):
        if self.next == None:
            return self.val
        return self.val + self.next.tot()

    def __getitem__(self, idx):
        if idx == 0:
            return self.val
        return self.next[idx-1]

A = Node(3)
B = Node(5)
C = Node(7)
A.next = B
B.next = C

```



1. finish the PythonTutor picture on the right
2. what is **C.tot()**? **B.tot()**? **A.tot()**?
3. what is **A[0]**? **A[2]**?
4. what kind of error does **A[-1]** produce?
5. how would the PythonTutor change if we added **C.next = A**?
6. what would **C[3]** be, given above change?
7. what would **A.tot()** do, give above change?


```

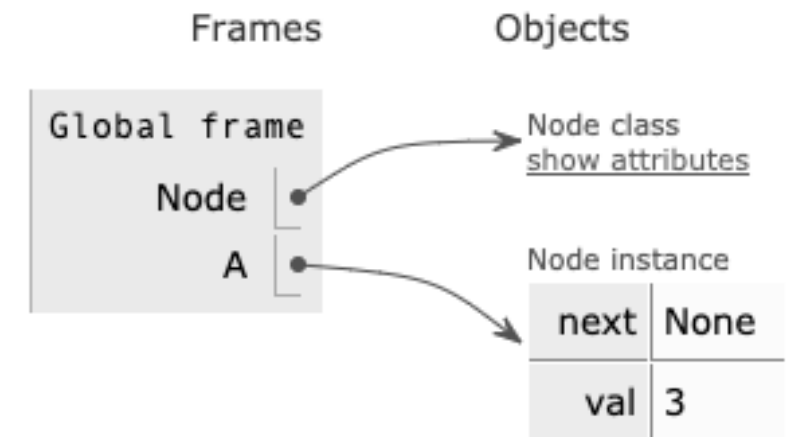
class Node:
    def __init__(self, val):
        self.val = val
        self.next = None

    def tot(self):
        if self.next == None:
            return self.val
        return self.val + self.next.tot()

    def __getitem__(self, idx):
        if idx == 0:
            return self.val
        return self.next[idx-1]

A = Node(3)
B = Node(5)
C = Node(7)
A.next = B
B.next = C

```



Demo with Python Tutor

1. finish the PythonTutor picture on the right
2. what is **C.tot()**? **B.tot()**? **A.tot()**?
3. what is **A[0]**? **A[2]**?
4. what kind of error does **A[-1]** produce?
5. how would the PythonTutor change if we added **C.next = A**?
6. what would **C[3]** be, given above change?
7. what would **A.tot()** do, give above change?