CS21: INTRODUCTION TO COMPUTER SCIENCE

Prof. Mathieson

Fall 2017

Swarthmore College

Outline Dec 6:

- Recursion (example: Fibonacci)
- Stack diagrams for recursion
- Start: graphics examples
- Go over Quiz 5

Notes

•Lab 11 is optional but STRONGLY recommended
•Lab 11 attendance is NOT optional
•Ninja session tonight and Friday
•Office hours on Friday

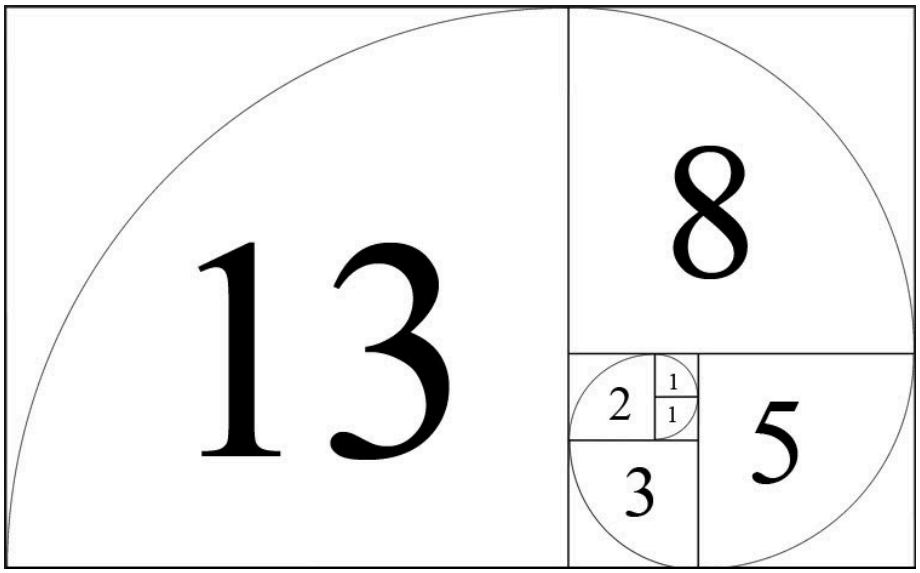
Fibonacci Example

Fibonacci numbers

Each Fibonacci number is the sum of the previous two Fibonacci numbers

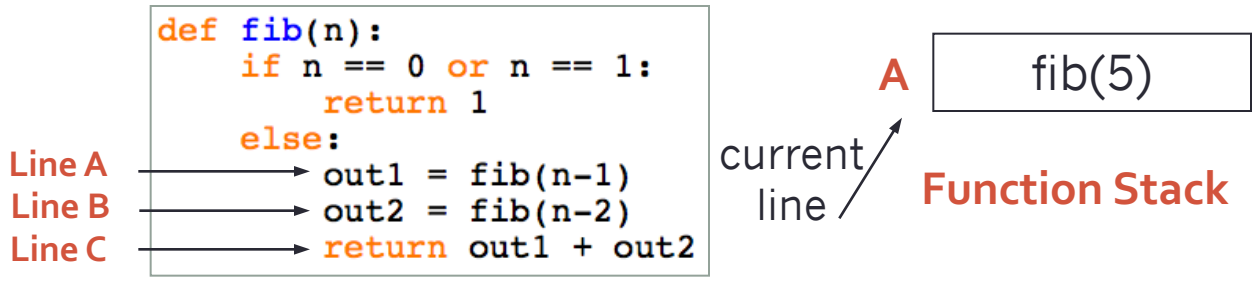
Recursion: $F_n = F_{n-1} + F_{n-2}$

Base cases: $F_0 = 1$ and $F_1 = 1$

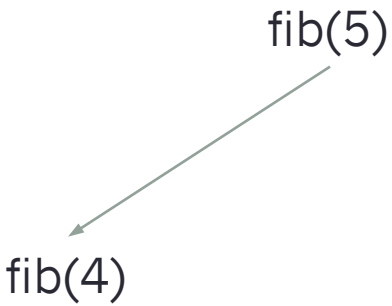


Fibonacci Function Stack

fib(5)



Fibonacci Function Stack




```
def fib(n):
    if n == 0 or n == 1:
        return 1
    else:
        out1 = fib(n-1)
        out2 = fib(n-2)
        return out1 + out2
```

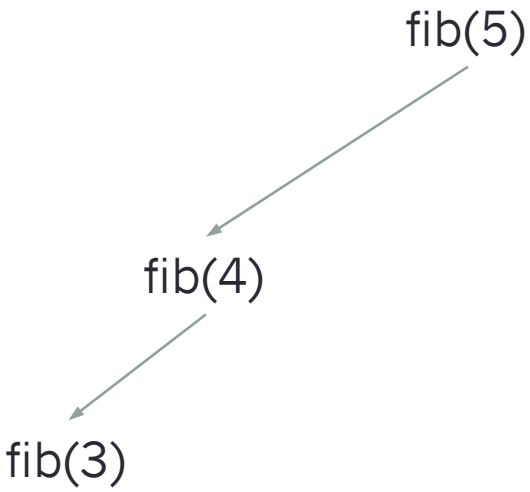
Line A
Line B
Line C

current line

A	fib(4)
A	fib(5)

Function Stack

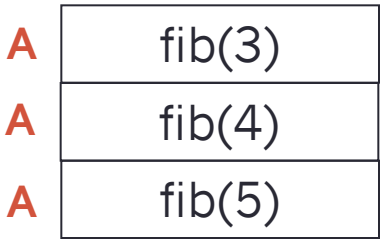
Fibonacci Function Stack




```
def fib(n):
    if n == 0 or n == 1:
        return 1
    else:
        out1 = fib(n-1)
        out2 = fib(n-2)
        return out1 + out2
```

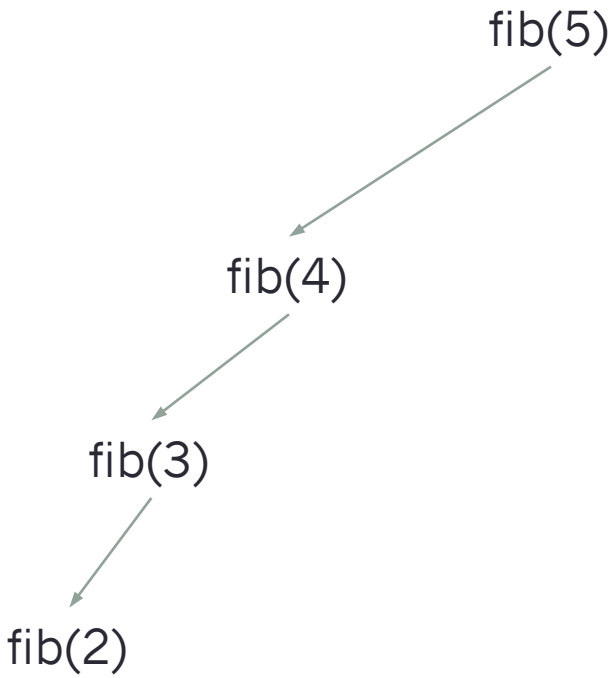
Line A
Line B
Line C

current line



Function Stack

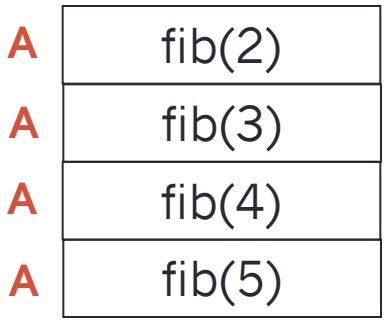
Fibonacci Function Stack




```
def fib(n):
    if n == 0 or n == 1:
        return 1
    else:
        out1 = fib(n-1)
        out2 = fib(n-2)
        return out1 + out2
```

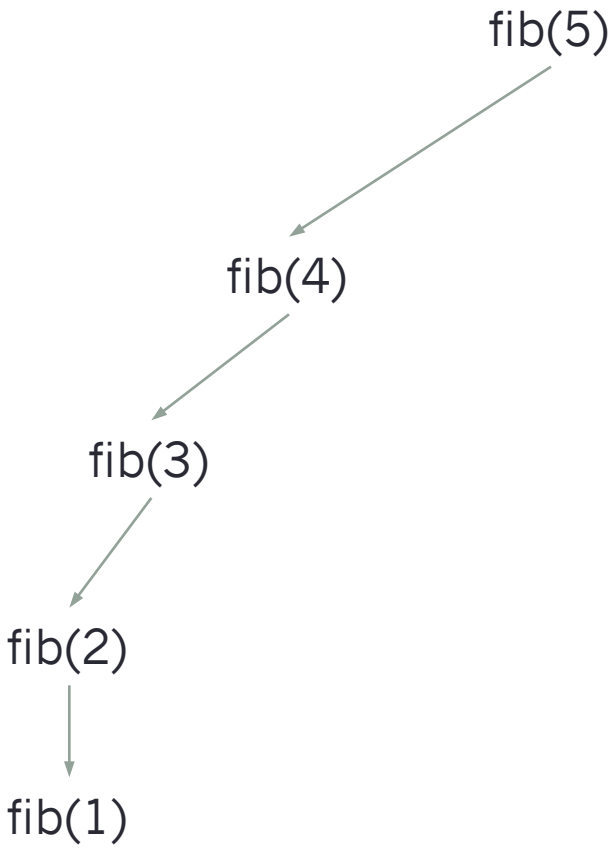
Line A
Line B
Line C

current line



Function Stack

Fibonacci Function Stack



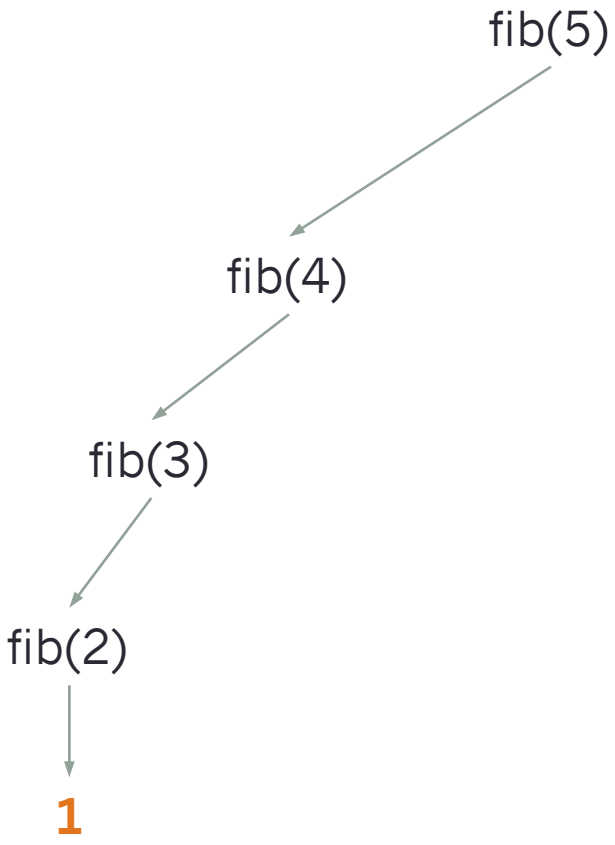

```
def fib(n):
    if n == 0 or n == 1:
        return 1
    else:
        out1 = fib(n-1)
        out2 = fib(n-2)
        return out1 + out2
```

Line A
Line B
Line C

current line

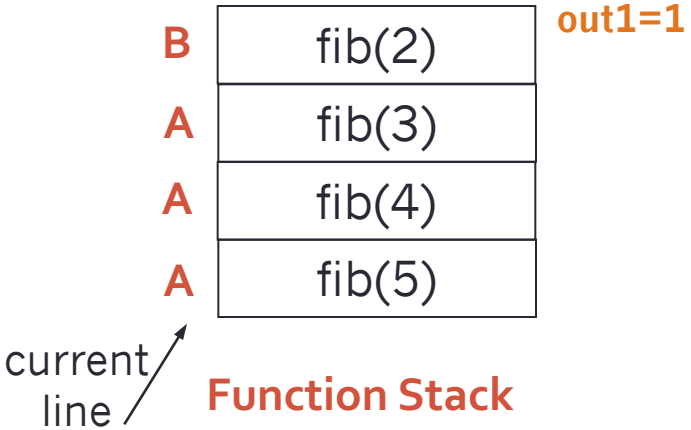
Function Stack

Fibonacci Function Stack

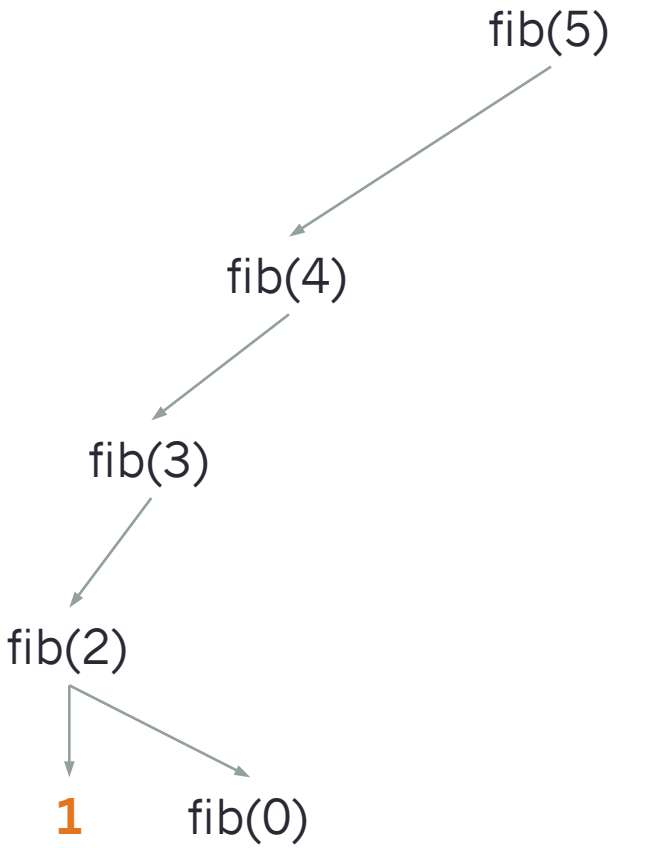



```
def fib(n):
    if n == 0 or n == 1:
        return 1
    else:
        out1 = fib(n-1)
        out2 = fib(n-2)
        return out1 + out2
```

Line A
Line B
Line C



Fibonacci Function Stack



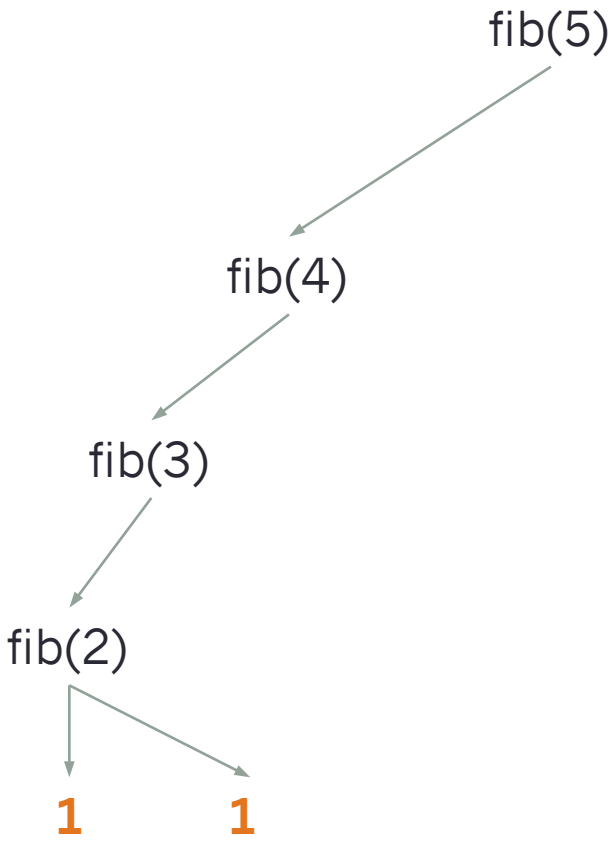

```
def fib(n):
    if n == 0 or n == 1:
        return 1
    else:
        out1 = fib(n-1)
        out2 = fib(n-2)
        return out1 + out2
```

Line A
Line B
Line C

current line

Function Stack

Fibonacci Function Stack



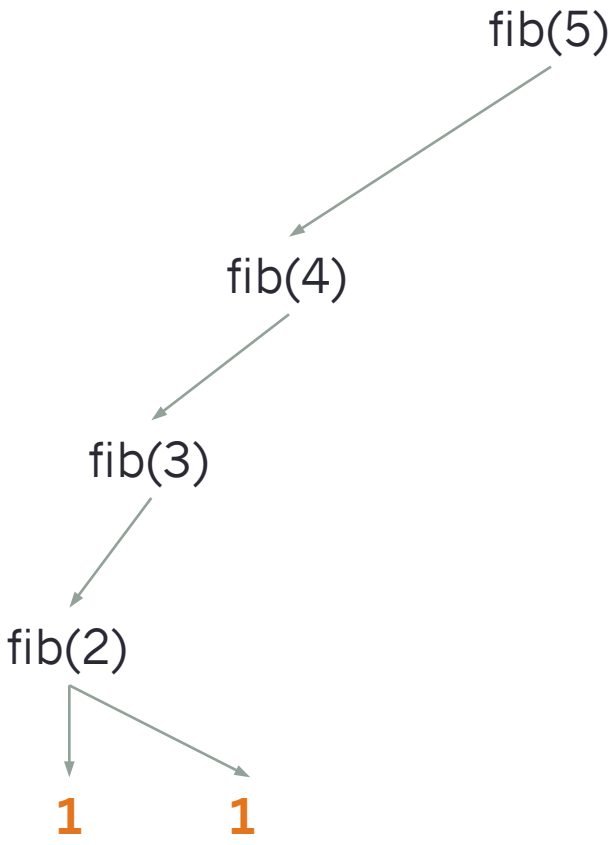

```
def fib(n):
    if n == 0 or n == 1:
        return 1
    else:
        out1 = fib(n-1)
        out2 = fib(n-2)
        return out1 + out2
```

Line A
Line B
Line C

current line

Function Stack

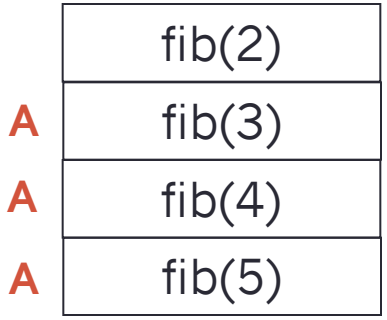
Fibonacci Function Stack




```
def fib(n):
    if n == 0 or n == 1:
        return 1
    else:
        out1 = fib(n-1)
        out2 = fib(n-2)
        return out1 + out2
```

Line A
Line B
Line C

current line

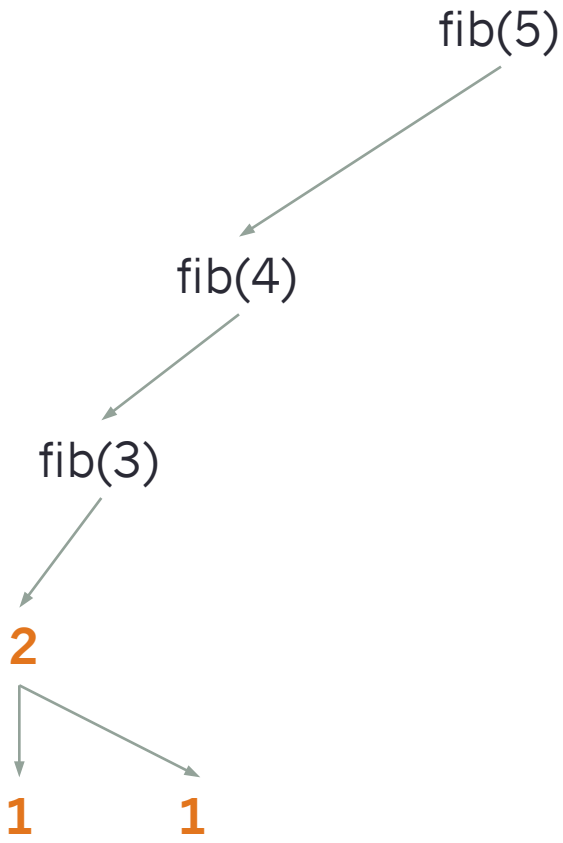


A
A
A

return 2

Function Stack

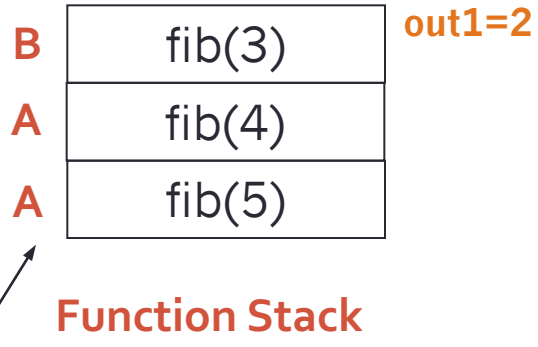
Fibonacci Function Stack



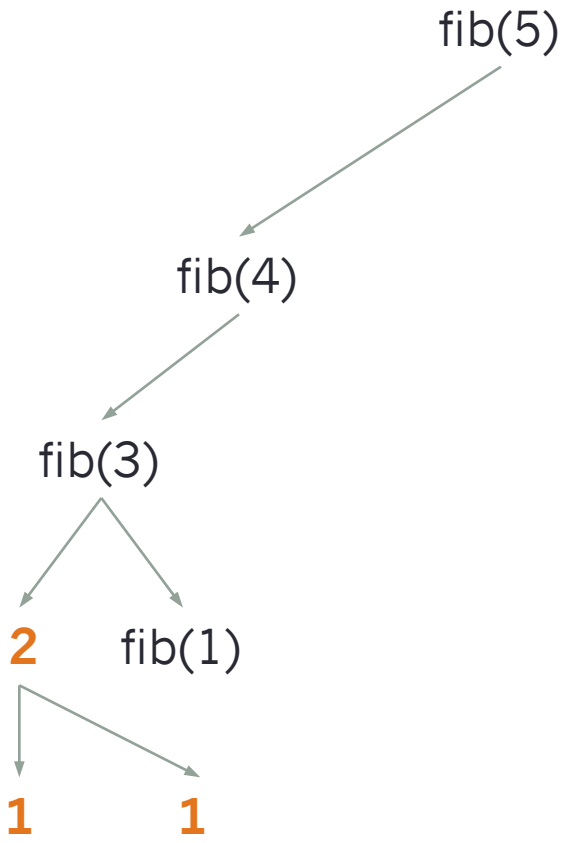

```
def fib(n):
    if n == 0 or n == 1:
        return 1
    else:
        out1 = fib(n-1)
        out2 = fib(n-2)
        return out1 + out2
```

Line A
Line B
Line C

current line



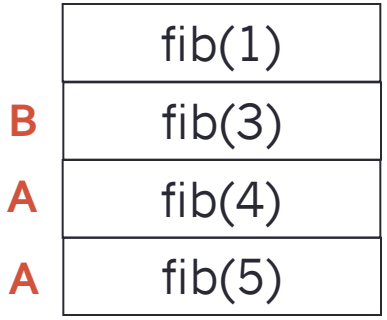
Fibonacci Function Stack




```
def fib(n):
    if n == 0 or n == 1:
        return 1
    else:
        out1 = fib(n-1)
        out2 = fib(n-2)
        return out1 + out2
```

Line A
Line B
Line C

current line

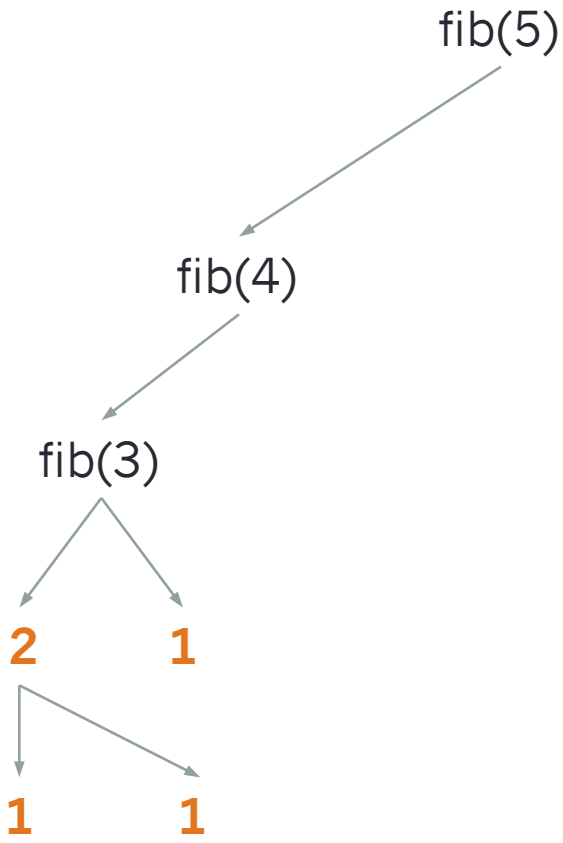


B
A
A

return 1
out1=2

Function Stack

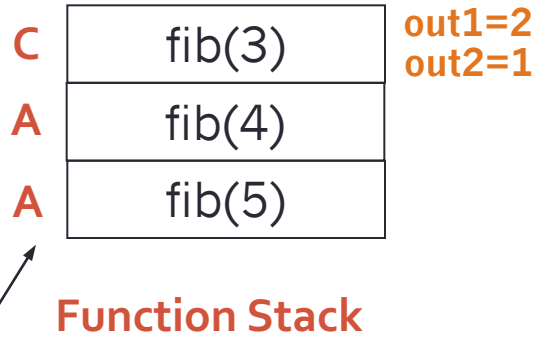
Fibonacci Function Stack



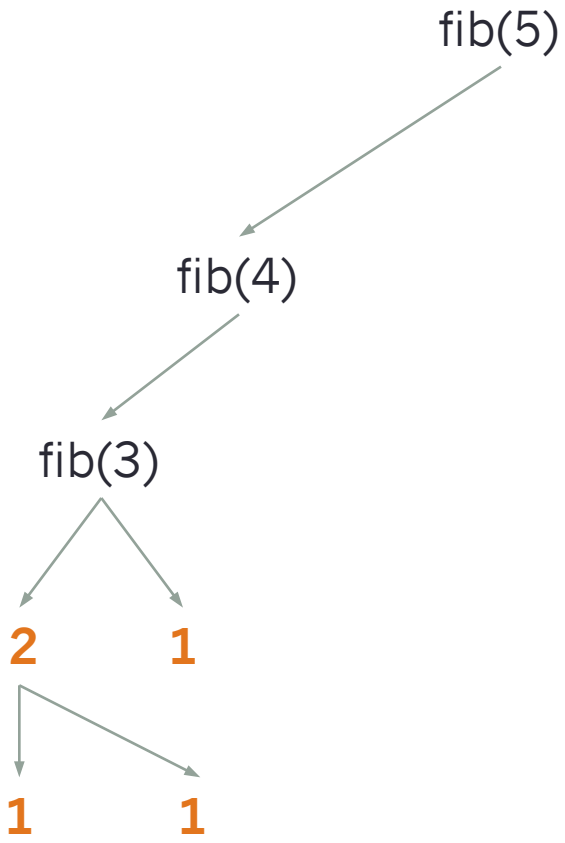

```
def fib(n):
    if n == 0 or n == 1:
        return 1
    else:
        out1 = fib(n-1)
        out2 = fib(n-2)
        return out1 + out2
```

Line A
Line B
Line C

current line



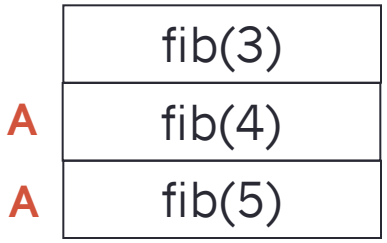
Fibonacci Function Stack




```
def fib(n):
    if n == 0 or n == 1:
        return 1
    else:
        out1 = fib(n-1)
        out2 = fib(n-2)
        return out1 + out2
```

Line A
Line B
Line C

current line

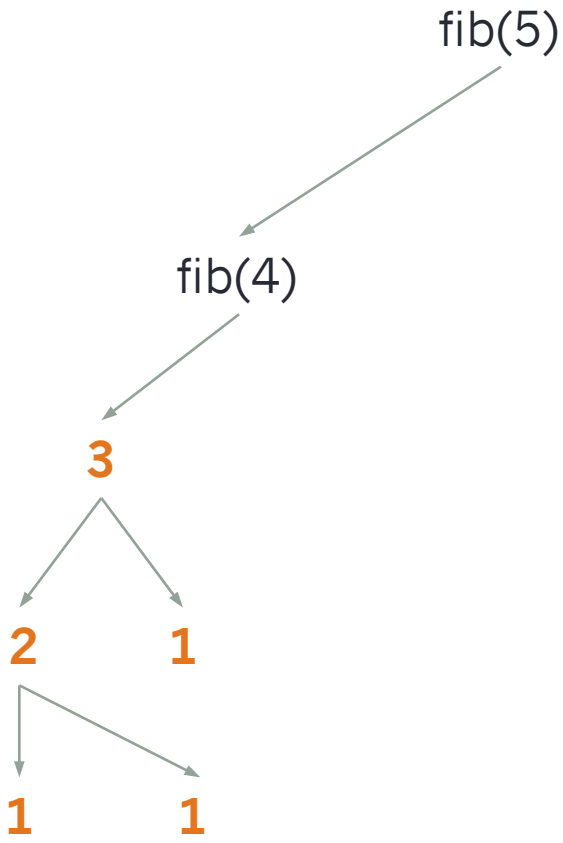


A
A

return 3

Function Stack

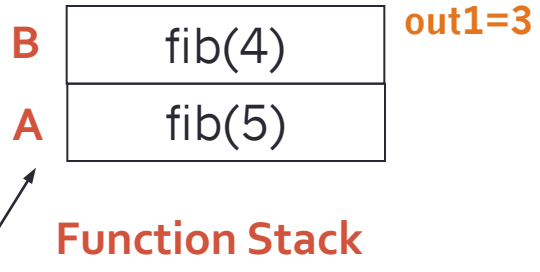
Fibonacci Function Stack



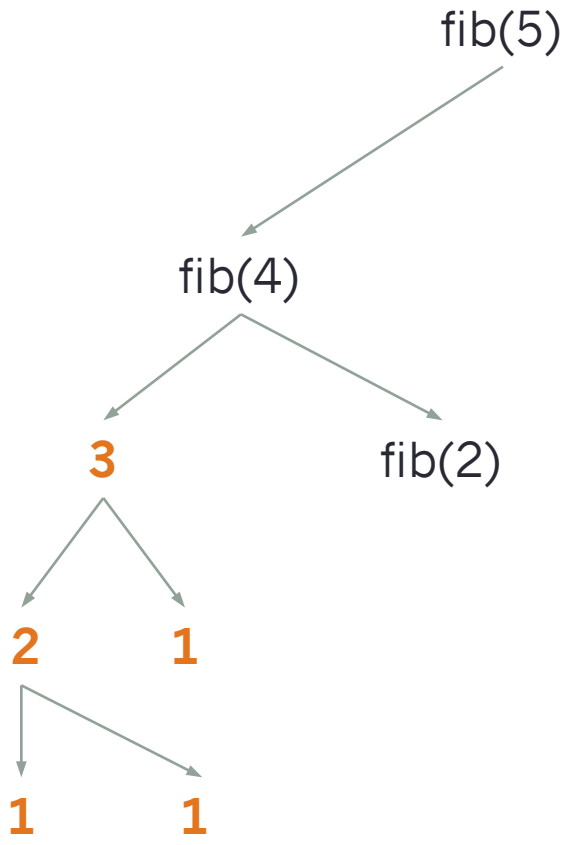

```
def fib(n):
    if n == 0 or n == 1:
        return 1
    else:
        out1 = fib(n-1)
        out2 = fib(n-2)
        return out1 + out2
```

Line A
Line B
Line C

current line



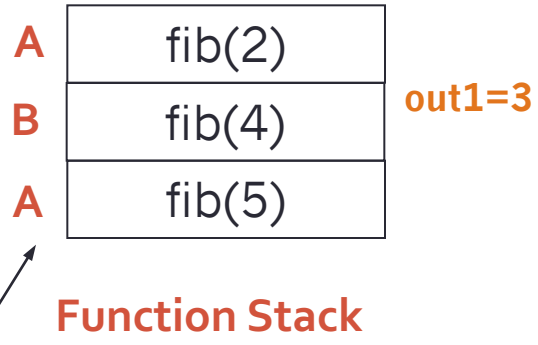
Fibonacci Function Stack



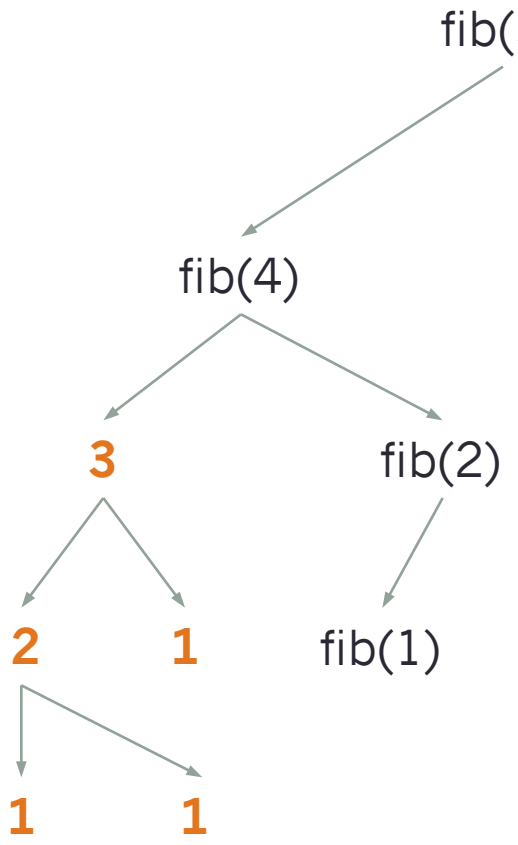

```
def fib(n):
    if n == 0 or n == 1:
        return 1
    else:
        out1 = fib(n-1)
        out2 = fib(n-2)
        return out1 + out2
```

Line A
Line B
Line C

current line



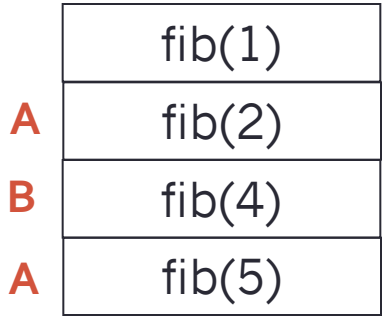
Fibonacci Function Stack




```
def fib(n):
    if n == 0 or n == 1:
        return 1
    else:
        out1 = fib(n-1)
        out2 = fib(n-2)
        return out1 + out2
```

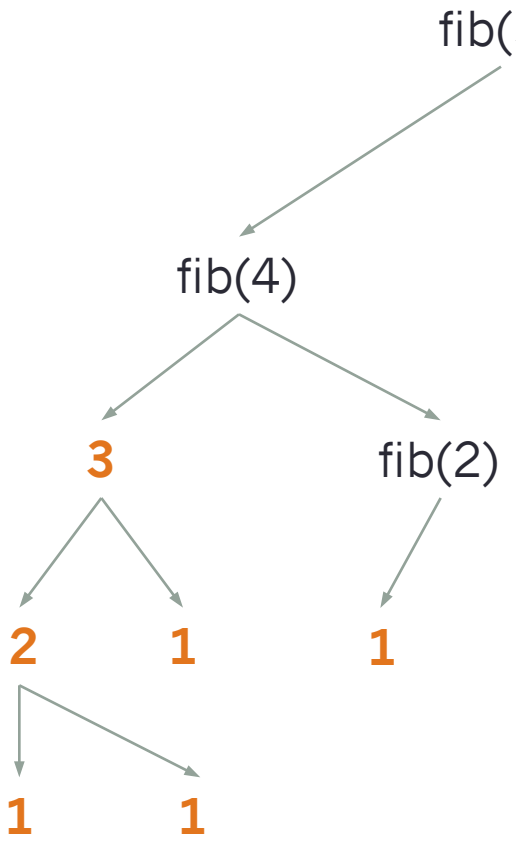
Line A
Line B
Line C

current line



Function Stack

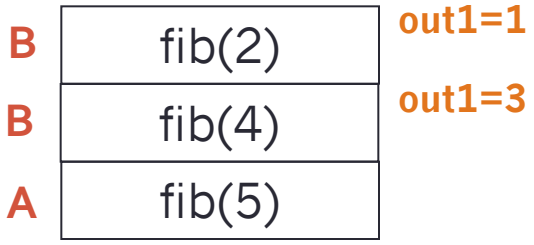
Fibonacci Function Stack




```
def fib(n):
    if n == 0 or n == 1:
        return 1
    else:
        out1 = fib(n-1)
        out2 = fib(n-2)
        return out1 + out2
```

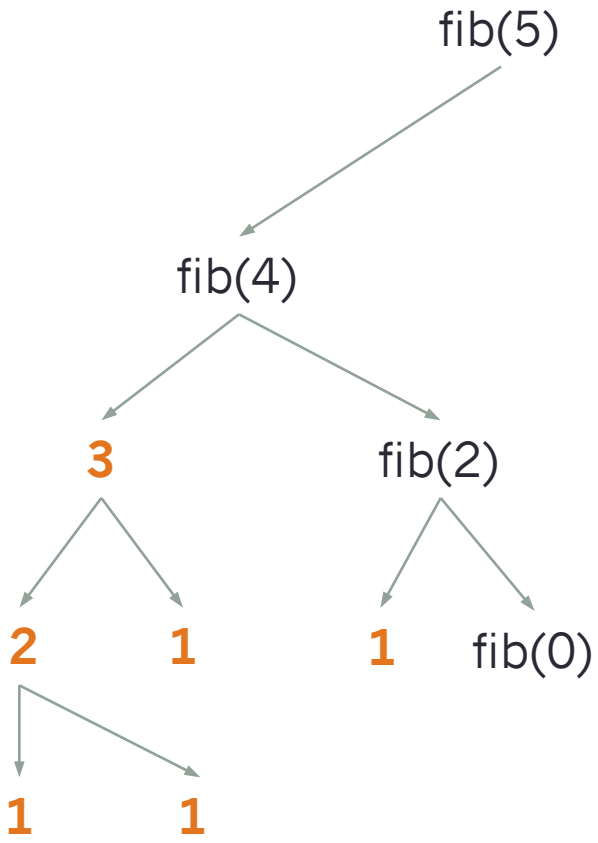
Line A
Line B
Line C

current line



Function Stack

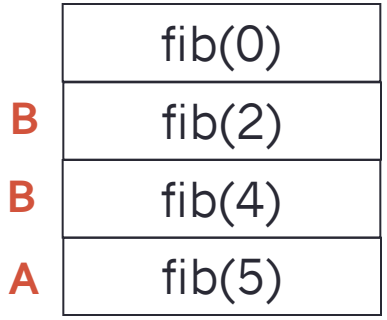
Fibonacci Function Stack




```
def fib(n):
    if n == 0 or n == 1:
        return 1
    else:
        out1 = fib(n-1)
        out2 = fib(n-2)
        return out1 + out2
```

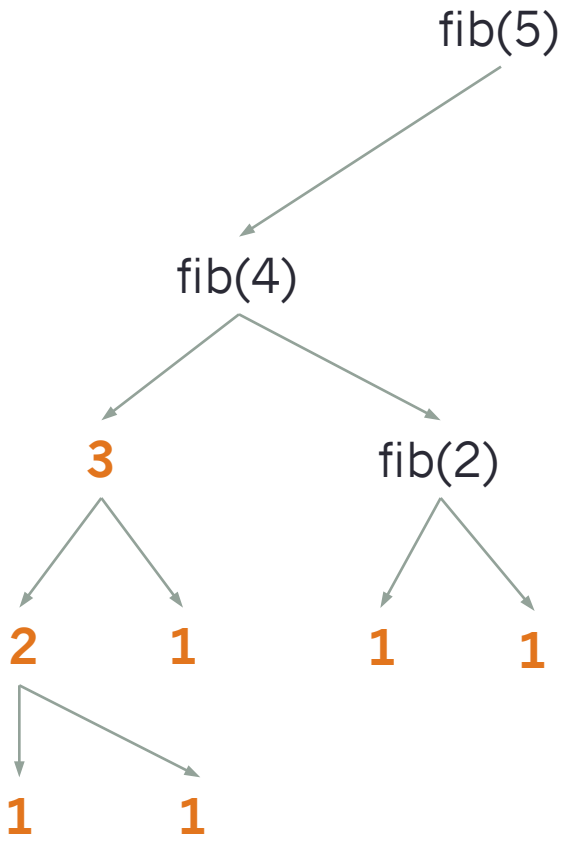
Line A
Line B
Line C

current line



Function Stack

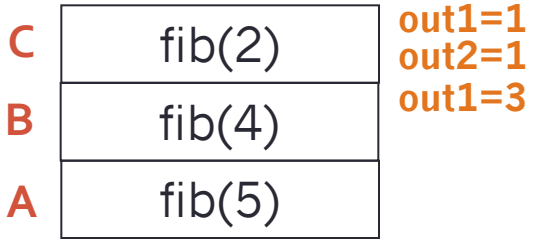
Fibonacci Function Stack




```
def fib(n):
    if n == 0 or n == 1:
        return 1
    else:
        out1 = fib(n-1)
        out2 = fib(n-2)
        return out1 + out2
```

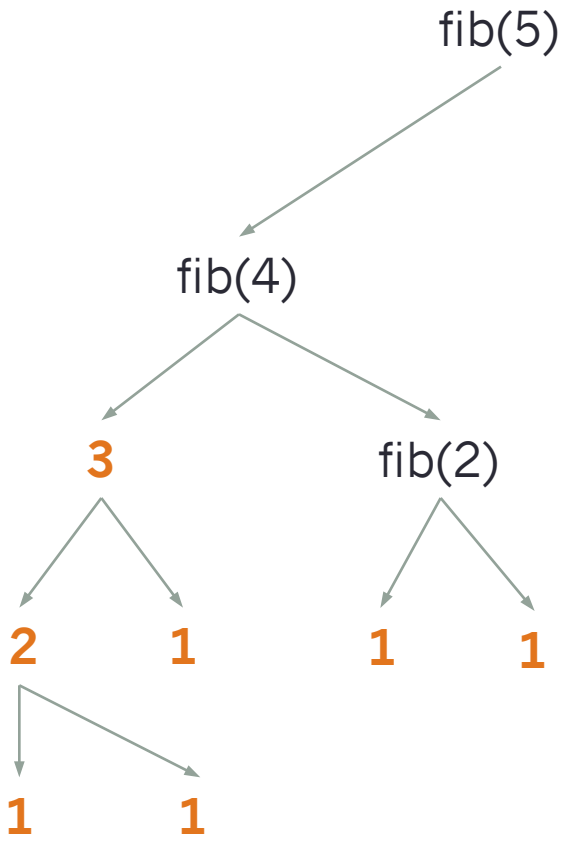
Line A
Line B
Line C

current line



Function Stack

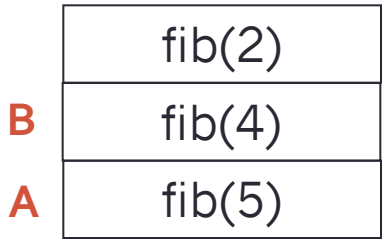
Fibonacci Function Stack




```
def fib(n):
    if n == 0 or n == 1:
        return 1
    else:
        out1 = fib(n-1)
        out2 = fib(n-2)
        return out1 + out2
```

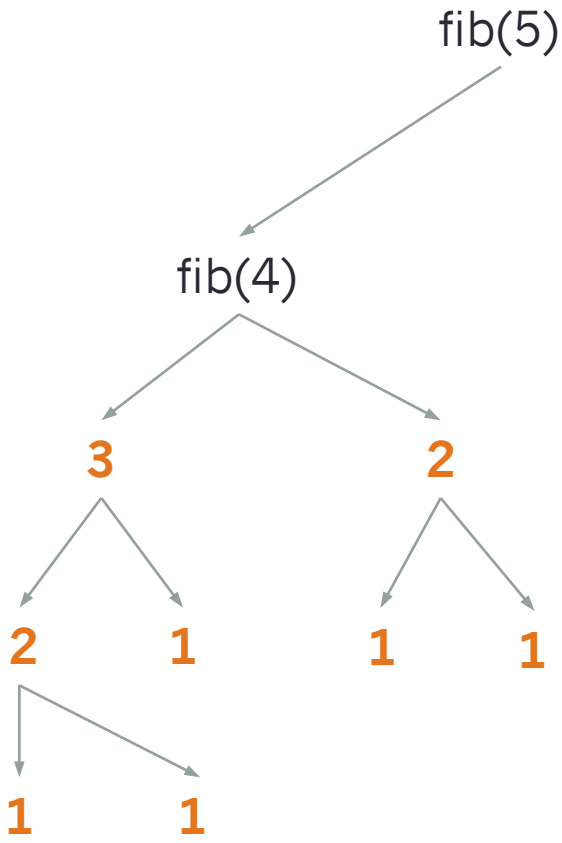
Line A
Line B
Line C

current line



Function Stack

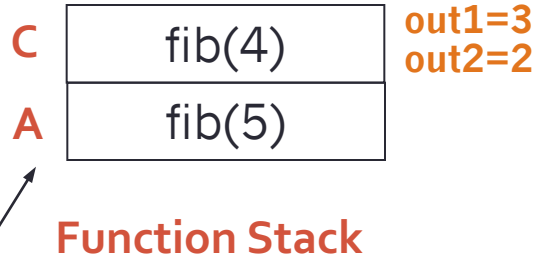
Fibonacci Function Stack



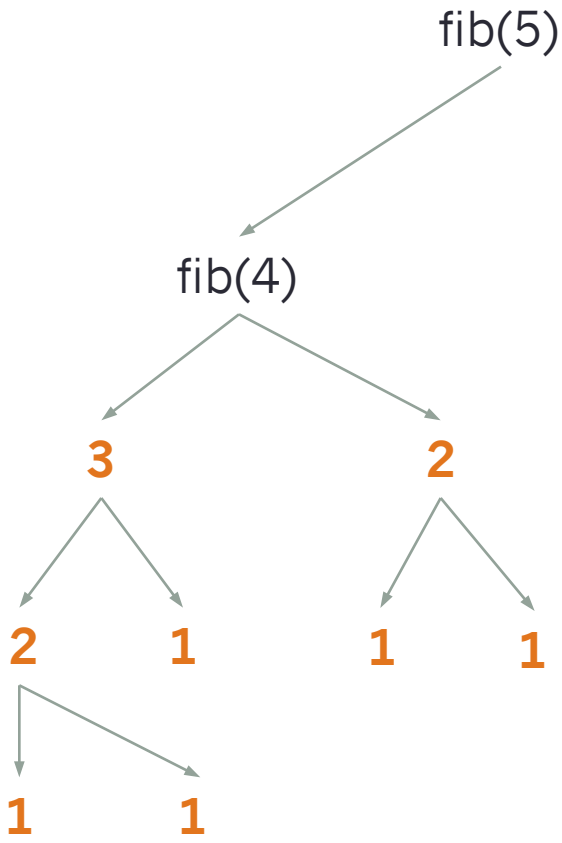

```
def fib(n):
    if n == 0 or n == 1:
        return 1
    else:
        out1 = fib(n-1)
        out2 = fib(n-2)
        return out1 + out2
```

Line A
Line B
Line C

current line



Fibonacci Function Stack




```
def fib(n):
    if n == 0 or n == 1:
        return 1
    else:
        out1 = fib(n-1)
        out2 = fib(n-2)
        return out1 + out2
```

Line A
Line B
Line C

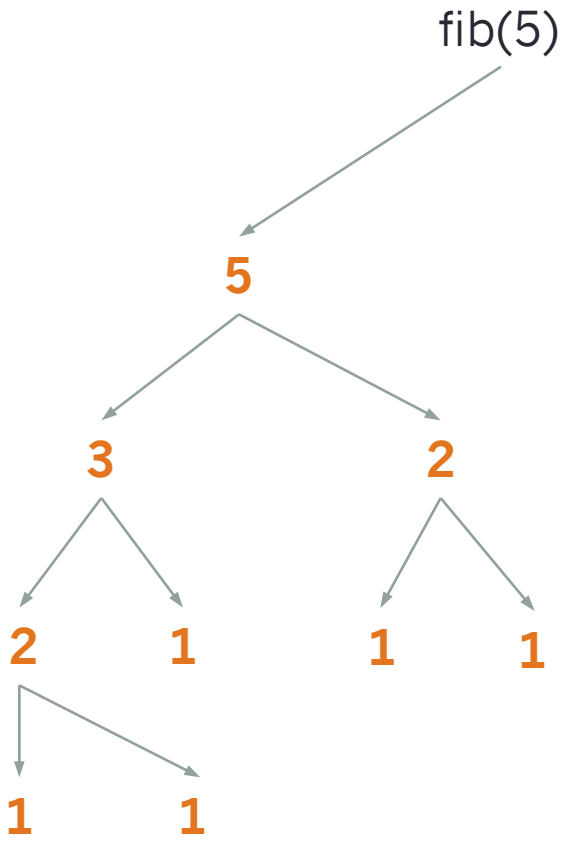
current line

A

return 5

Function Stack

Fibonacci Function Stack




```
def fib(n):
    if n == 0 or n == 1:
        return 1
    else:
        out1 = fib(n-1)
        out2 = fib(n-2)
        return out1 + out2
```

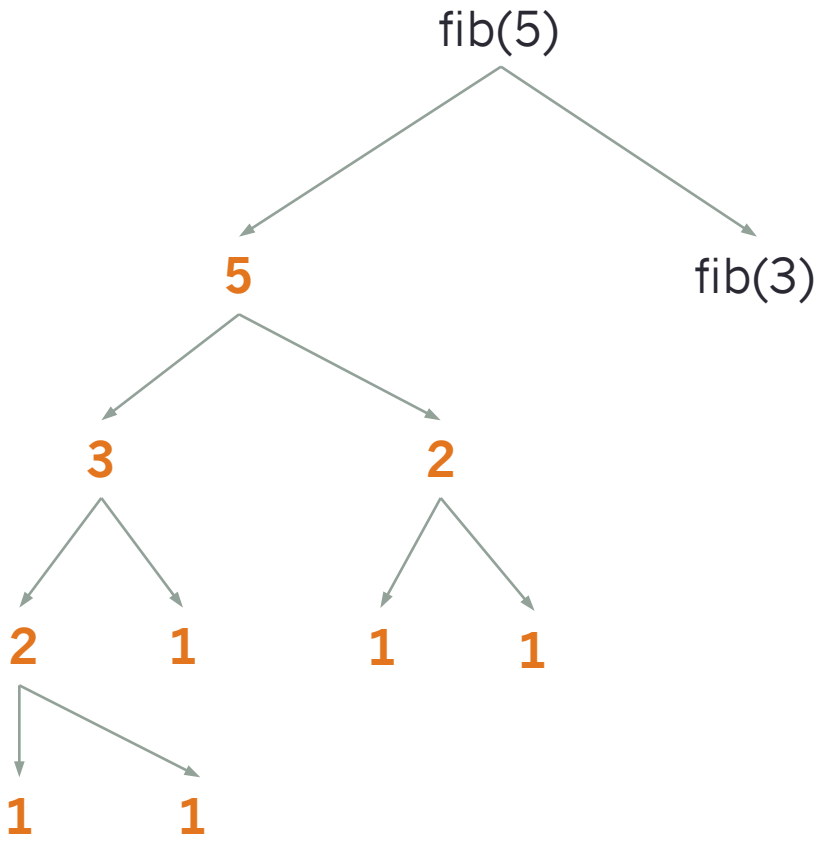
Line A
Line B
Line C

current line

B | fib(5) | out1=5

Function Stack

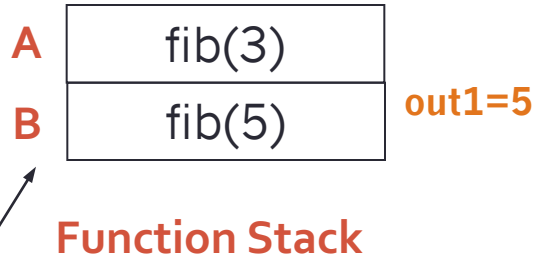
Fibonacci Function Stack



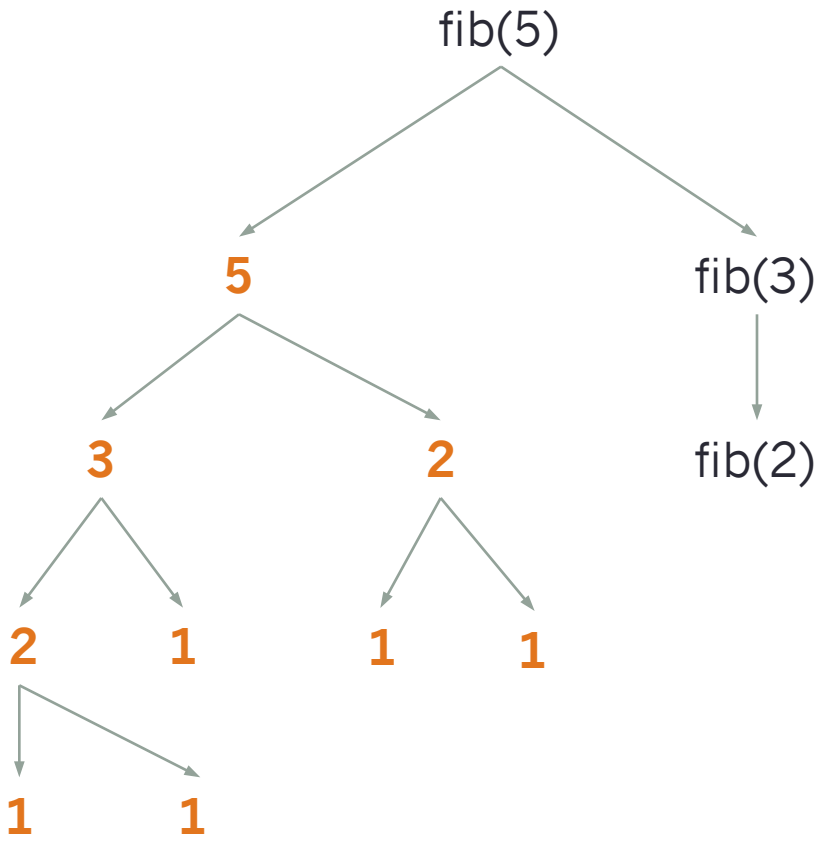

```
def fib(n):
    if n == 0 or n == 1:
        return 1
    else:
        out1 = fib(n-1)
        out2 = fib(n-2)
        return out1 + out2
```

Line A
Line B
Line C

current line



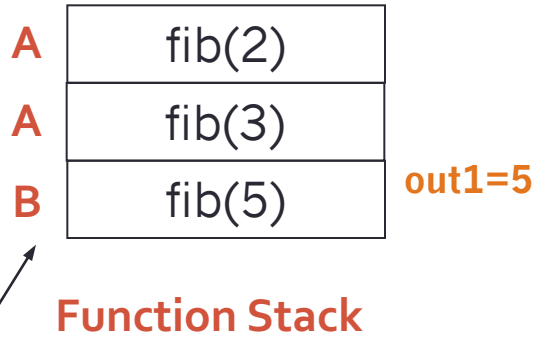
Fibonacci Function Stack



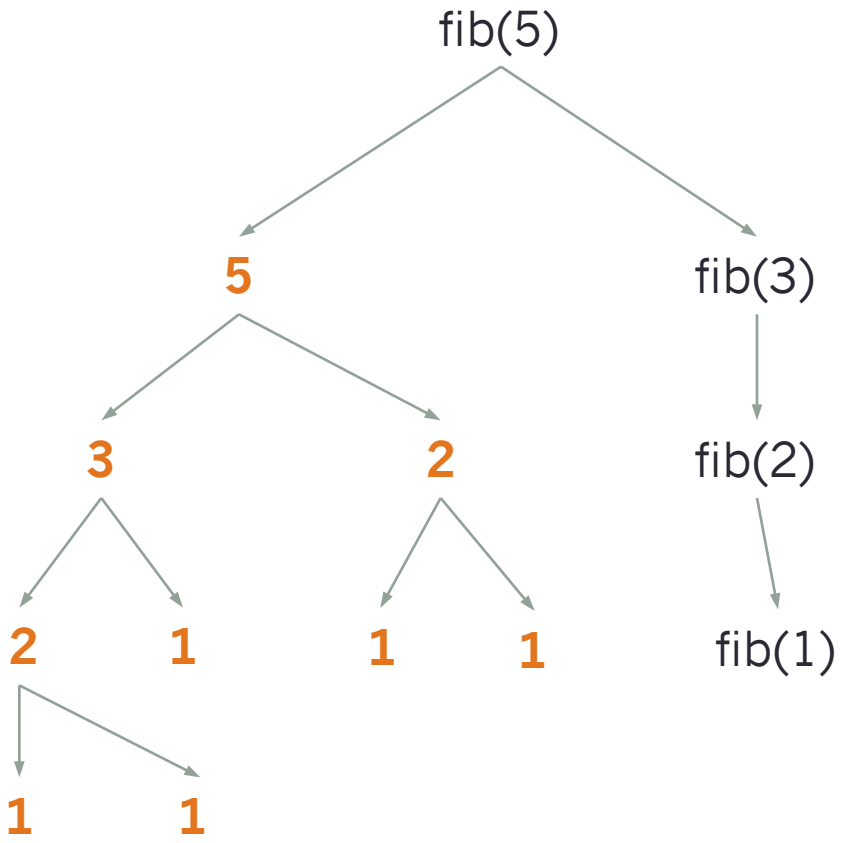

```
def fib(n):
    if n == 0 or n == 1:
        return 1
    else:
        out1 = fib(n-1)
        out2 = fib(n-2)
        return out1 + out2
```

Line A
Line B
Line C

current line



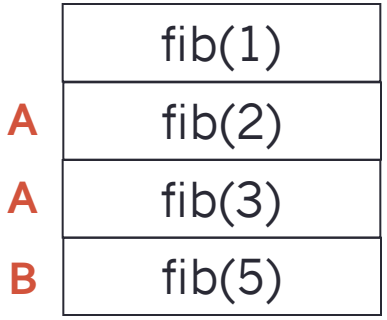
Fibonacci Function Stack




```
def fib(n):
    if n == 0 or n == 1:
        return 1
    else:
        out1 = fib(n-1)
        out2 = fib(n-2)
        return out1 + out2
```

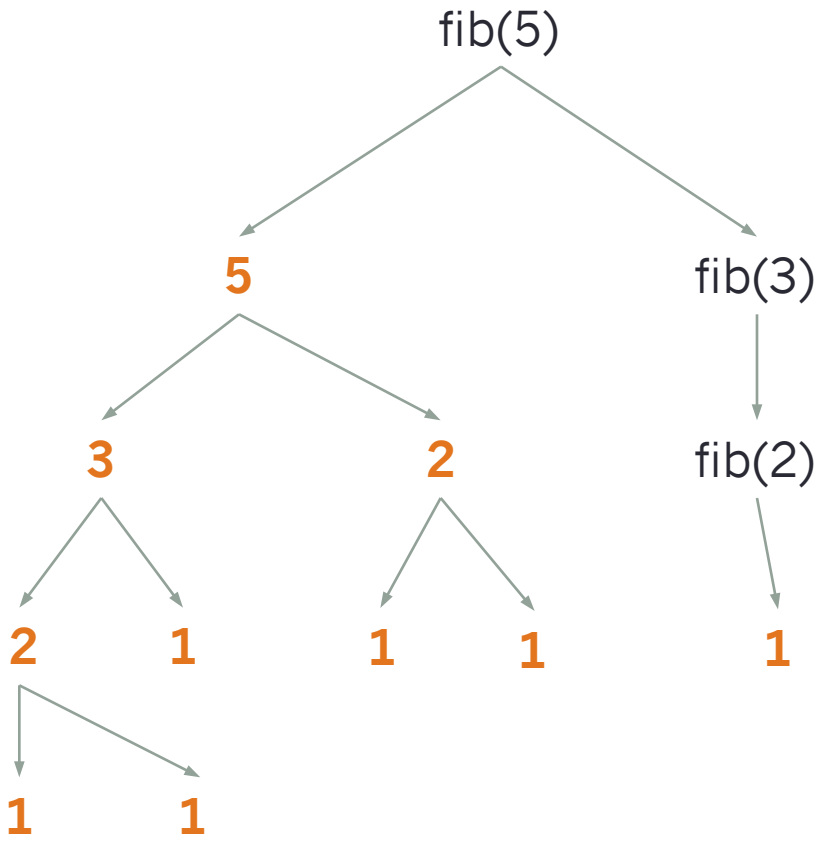
Line A
Line B
Line C

current line



Function Stack

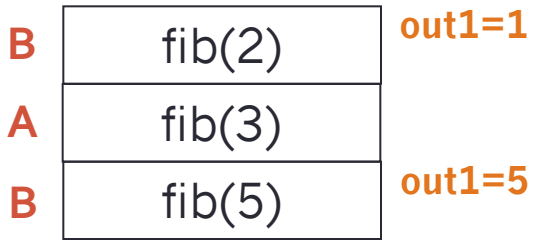
Fibonacci Function Stack




```
def fib(n):
    if n == 0 or n == 1:
        return 1
    else:
        out1 = fib(n-1)
        out2 = fib(n-2)
        return out1 + out2
```

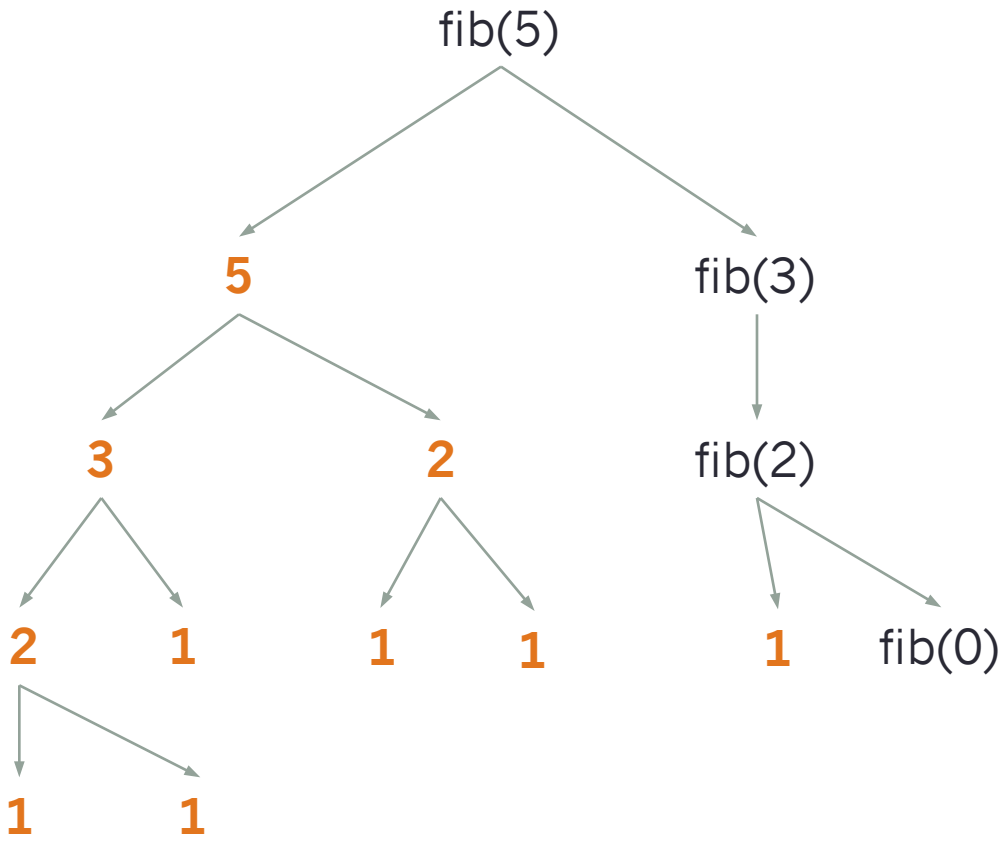
Line A
Line B
Line C

current line



Function Stack

Fibonacci Function Stack



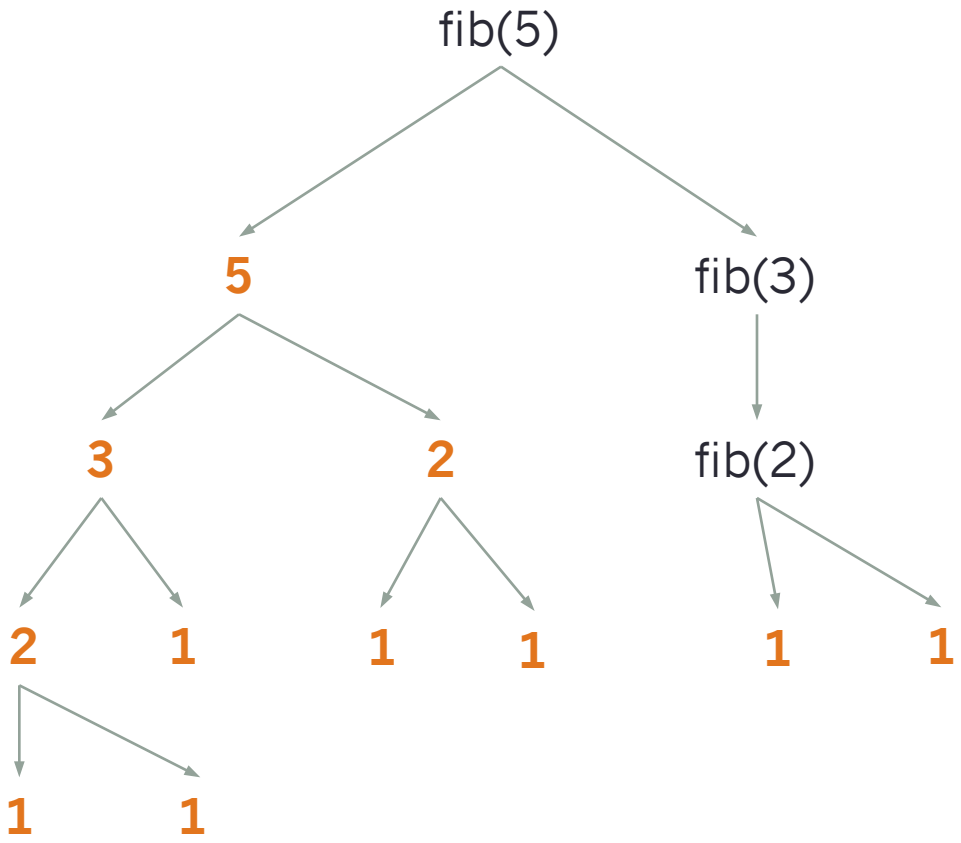

```
def fib(n):
    if n == 0 or n == 1:
        return 1
    else:
        out1 = fib(n-1)
        out2 = fib(n-2)
        return out1 + out2
```

Line A
Line B
Line C

current line

Function Stack

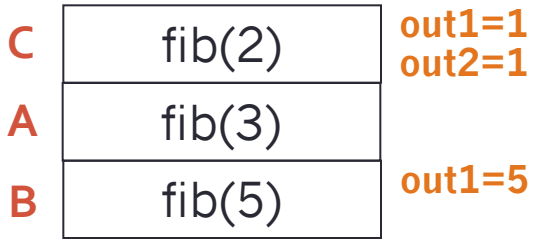
Fibonacci Function Stack




```
def fib(n):
    if n == 0 or n == 1:
        return 1
    else:
        out1 = fib(n-1)
        out2 = fib(n-2)
        return out1 + out2
```

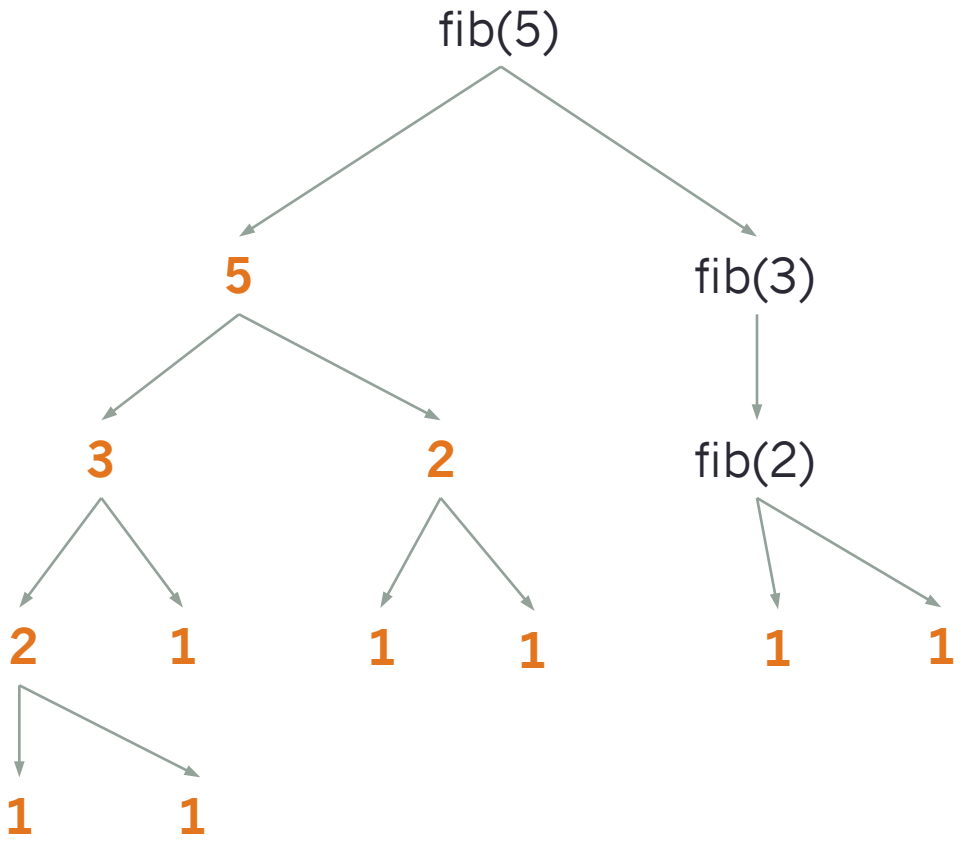
Line A
Line B
Line C

current line



Function Stack

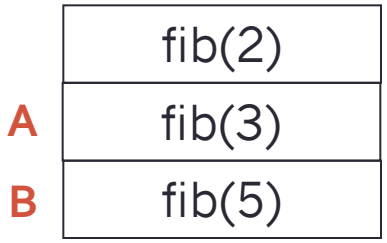
Fibonacci Function Stack




```
def fib(n):
    if n == 0 or n == 1:
        return 1
    else:
        out1 = fib(n-1)
        out2 = fib(n-2)
        return out1 + out2
```

Line A
Line B
Line C

current line



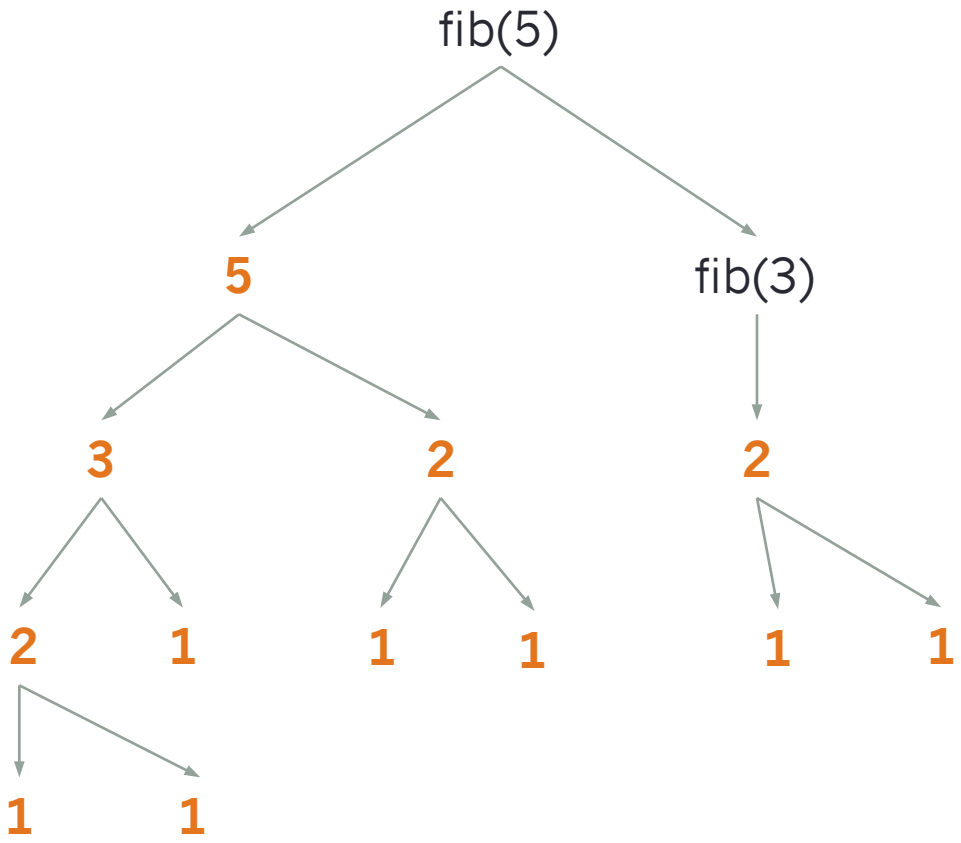
A
B

return 2

out1=5

Function Stack

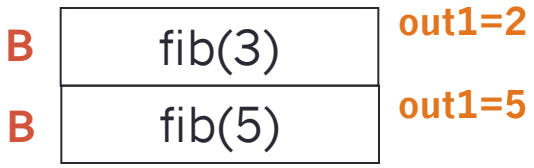
Fibonacci Function Stack




```
def fib(n):
    if n == 0 or n == 1:
        return 1
    else:
        out1 = fib(n-1)
        out2 = fib(n-2)
        return out1 + out2
```

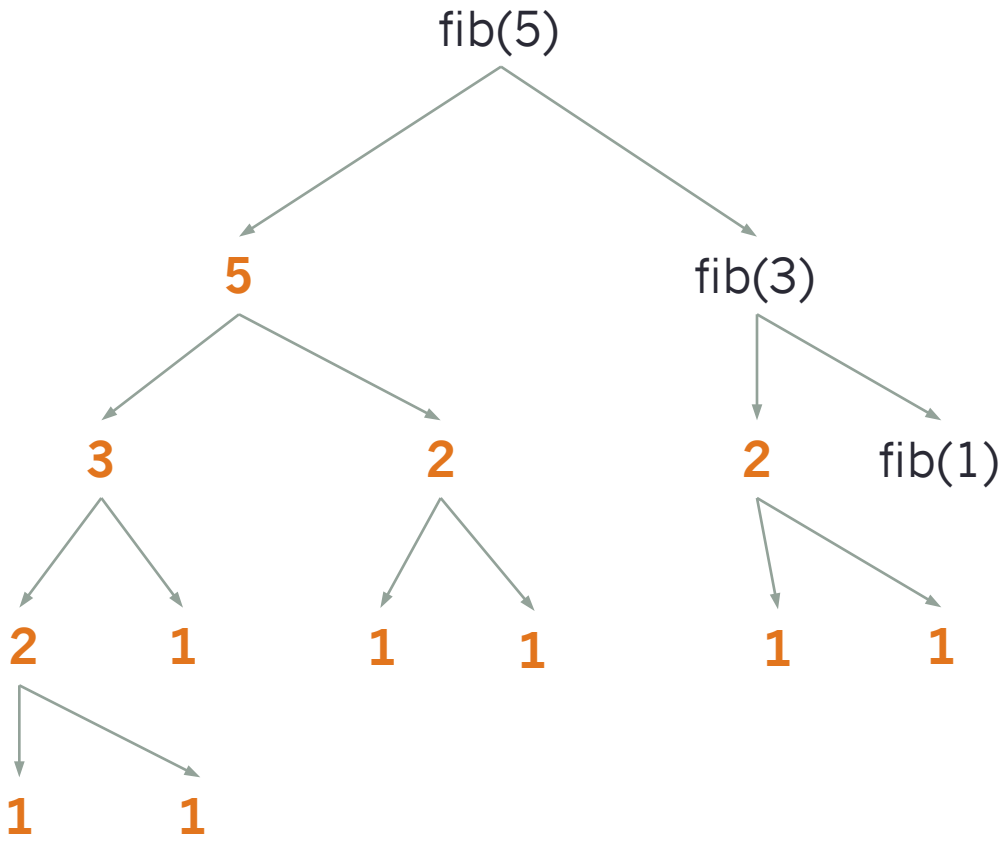
Line A
Line B
Line C

current line



Function Stack

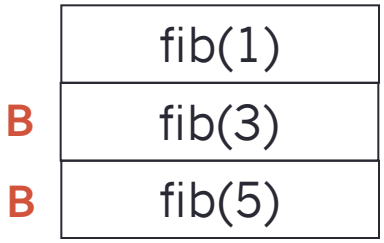
Fibonacci Function Stack




```
def fib(n):
    if n == 0 or n == 1:
        return 1
    else:
        out1 = fib(n-1)
        out2 = fib(n-2)
        return out1 + out2
```

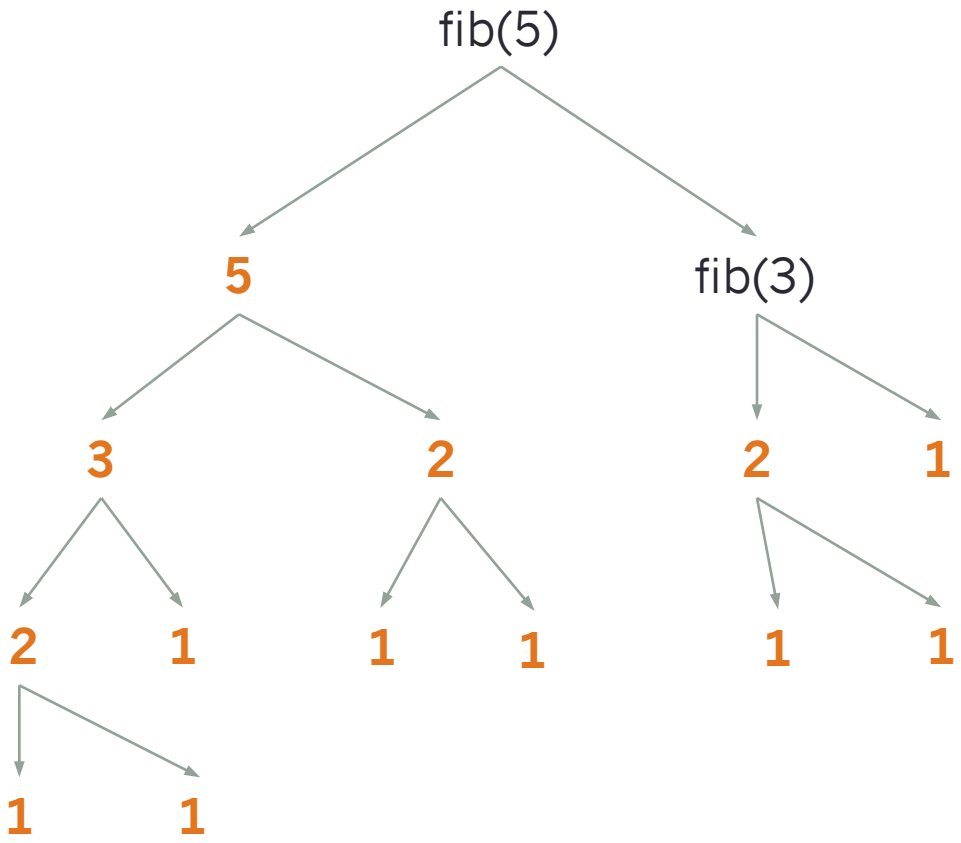
Line A
Line B
Line C

current line



Function Stack

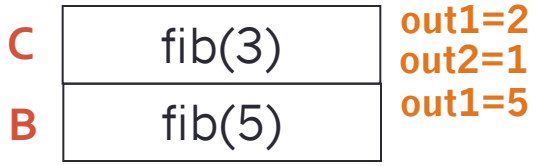
Fibonacci Function Stack




```
def fib(n):
    if n == 0 or n == 1:
        return 1
    else:
        out1 = fib(n-1)
        out2 = fib(n-2)
        return out1 + out2
```

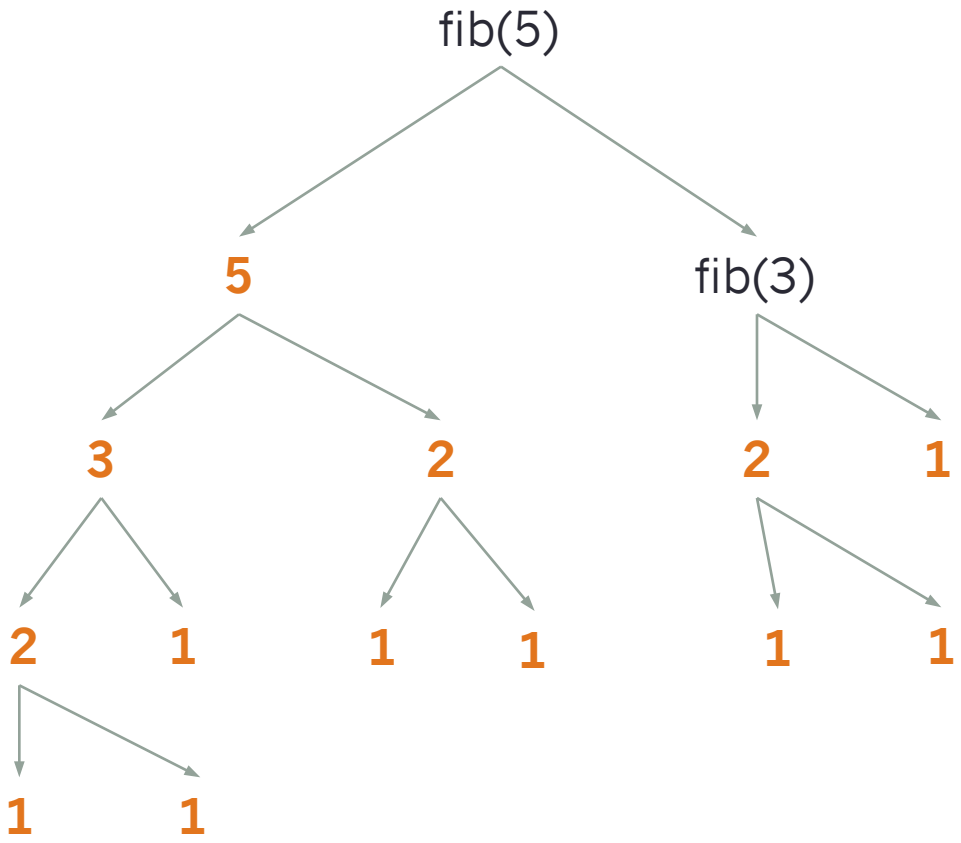
Line A
Line B
Line C

current line



Function Stack

Fibonacci Function Stack




```
def fib(n):
    if n == 0 or n == 1:
        return 1
    else:
        out1 = fib(n-1)
        out2 = fib(n-2)
        return out1 + out2
```

Line A
Line B
Line C

current line

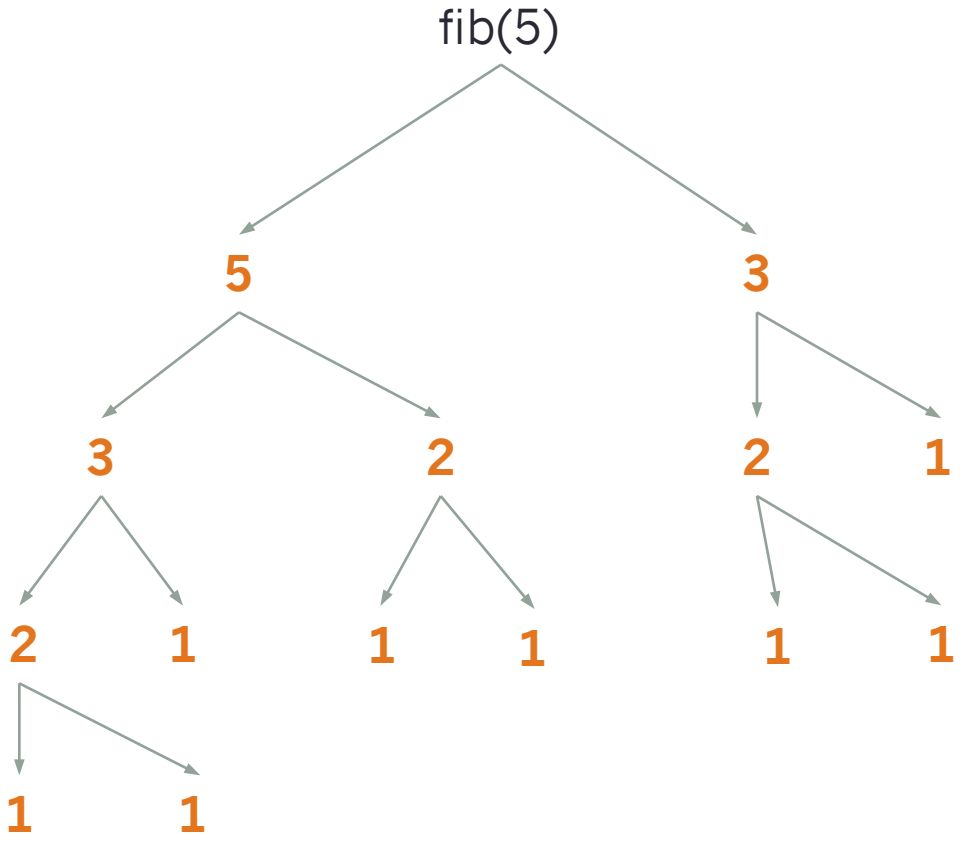
B

return 3

out1=5

Function Stack

Fibonacci Function Stack




```
def fib(n):
    if n == 0 or n == 1:
        return 1
    else:
        out1 = fib(n-1)
        out2 = fib(n-2)
        return out1 + out2
```

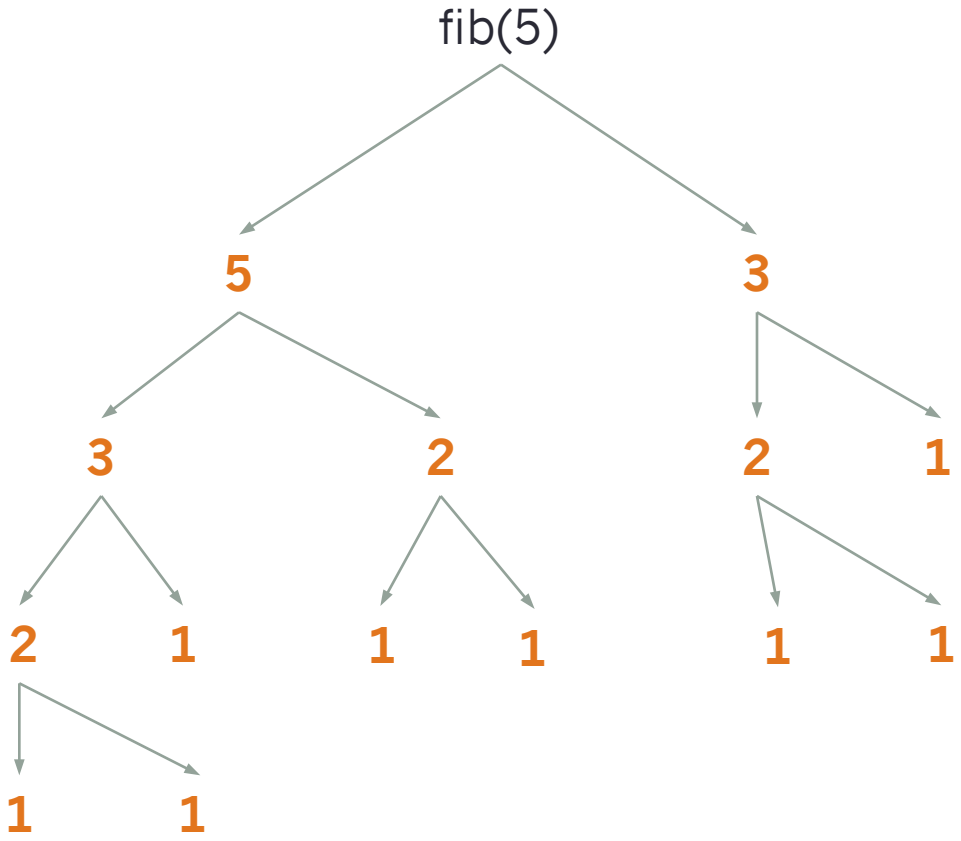
Line A
Line B
Line C

current line

C | fib(5) | out1=5 out2=3

Function Stack

Fibonacci Function Stack




```
def fib(n):
    if n == 0 or n == 1:
        return 1
    else:
        out1 = fib(n-1)
        out2 = fib(n-2)
        return out1 + out2
```

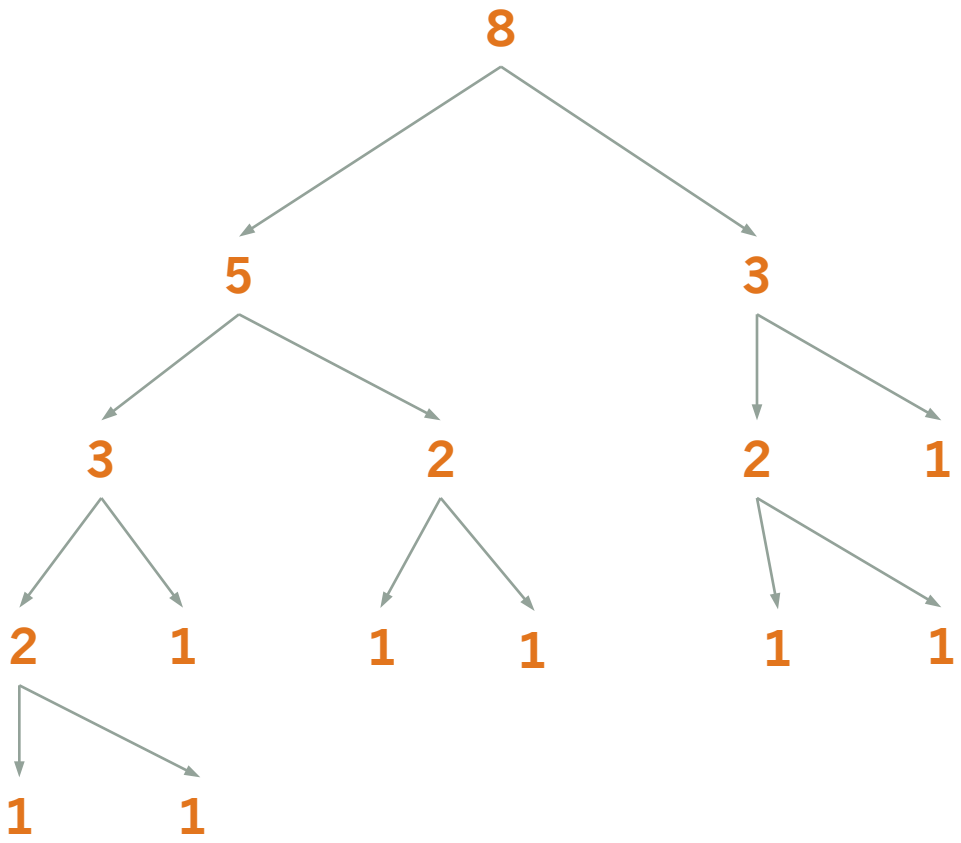
Line A
Line B
Line C

current line

fib(5) return 8

Function Stack

Fibonacci Function Stack



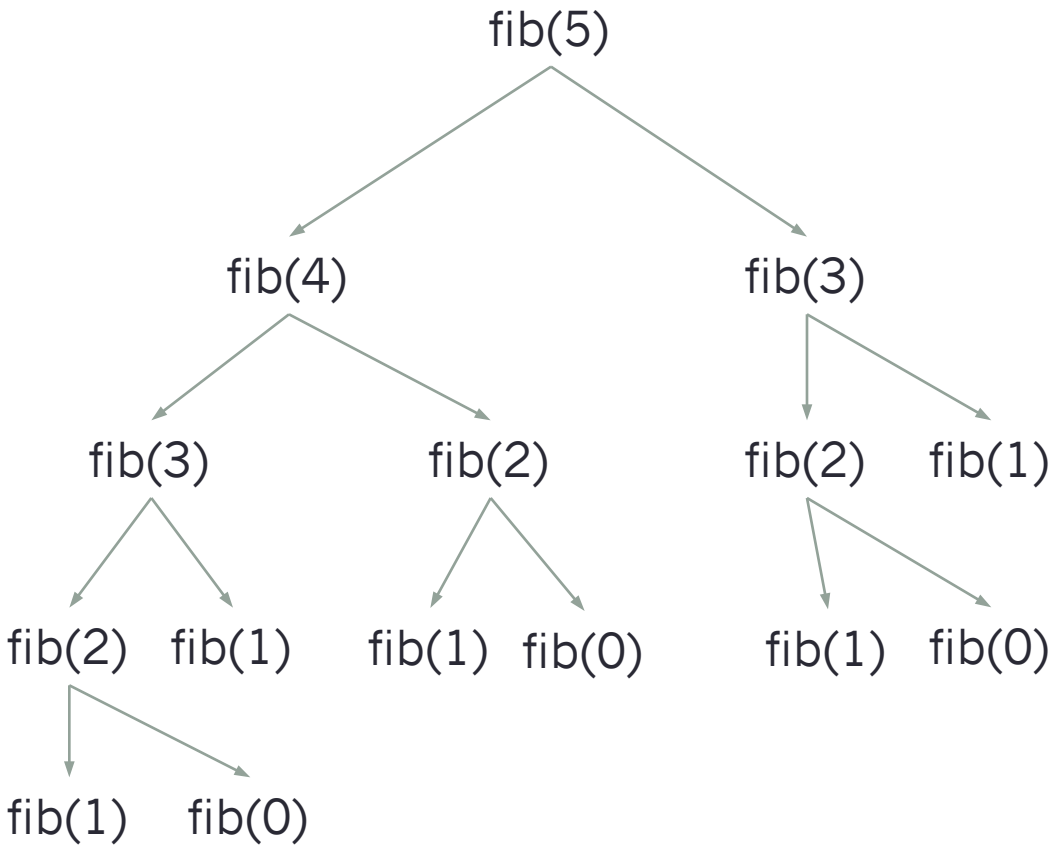

```
def fib(n):
    if n == 0 or n == 1:
        return 1
    else:
        out1 = fib(n-1)
        out2 = fib(n-2)
        return out1 + out2
```

Line A
Line B
Line C

empty!

Function Stack

Fibonacci Tree with Function Calls




```
def fib(n):
    if n == 0 or n == 1:
        return 1
    else:
        out1 = fib(n-1)
        out2 = fib(n-2)
        return out1 + out2
```

Line A
Line B
Line C

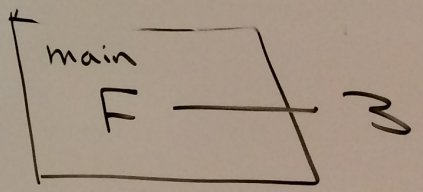
Fibonacci

n	0	1	2	3	4	5	6	7	...
F(n)	1	1	2	3	5	8	13	21	...

$$F(n) = F(n-2) + F(n-1)$$

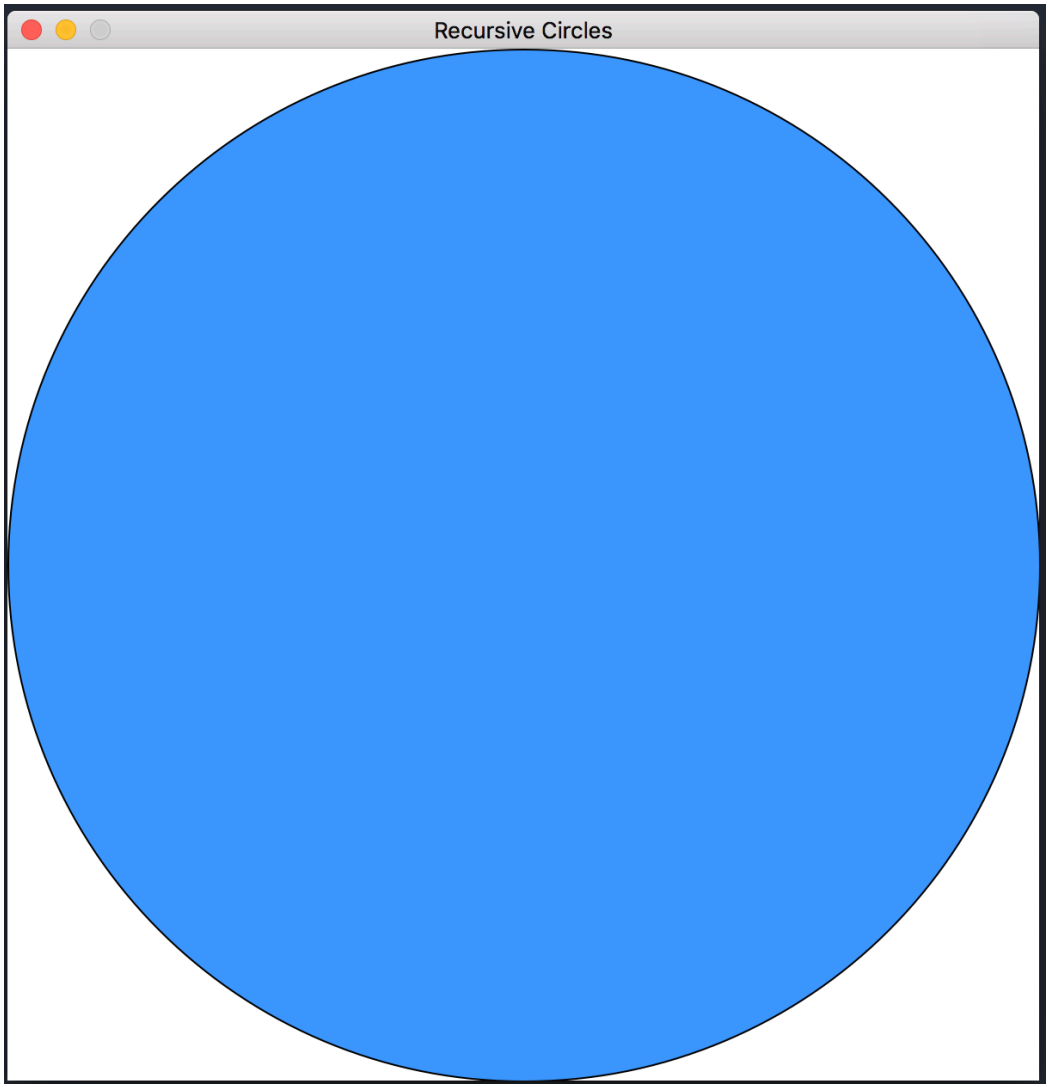
recursive call

fib(3)

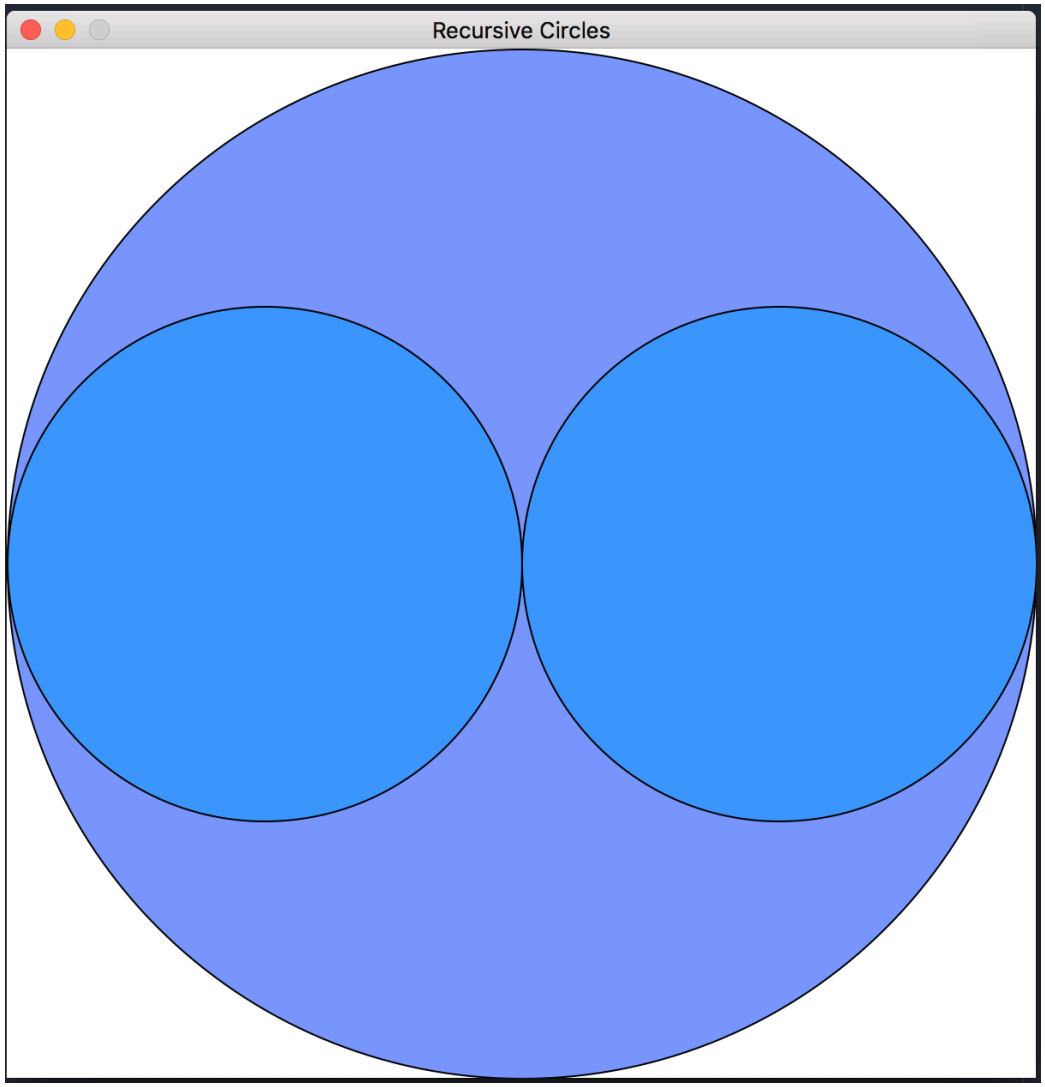


Recursive Circles Example

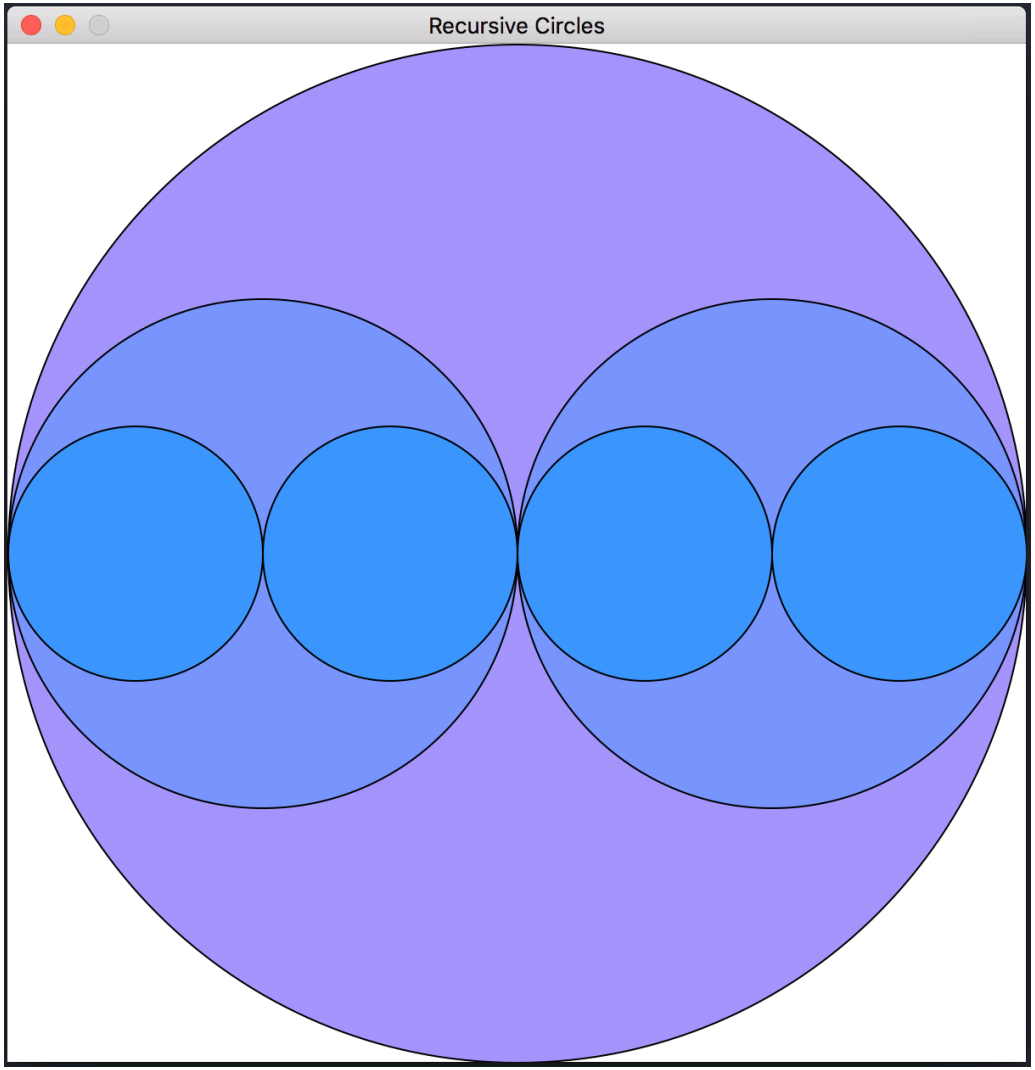
Recursive circles, n=1



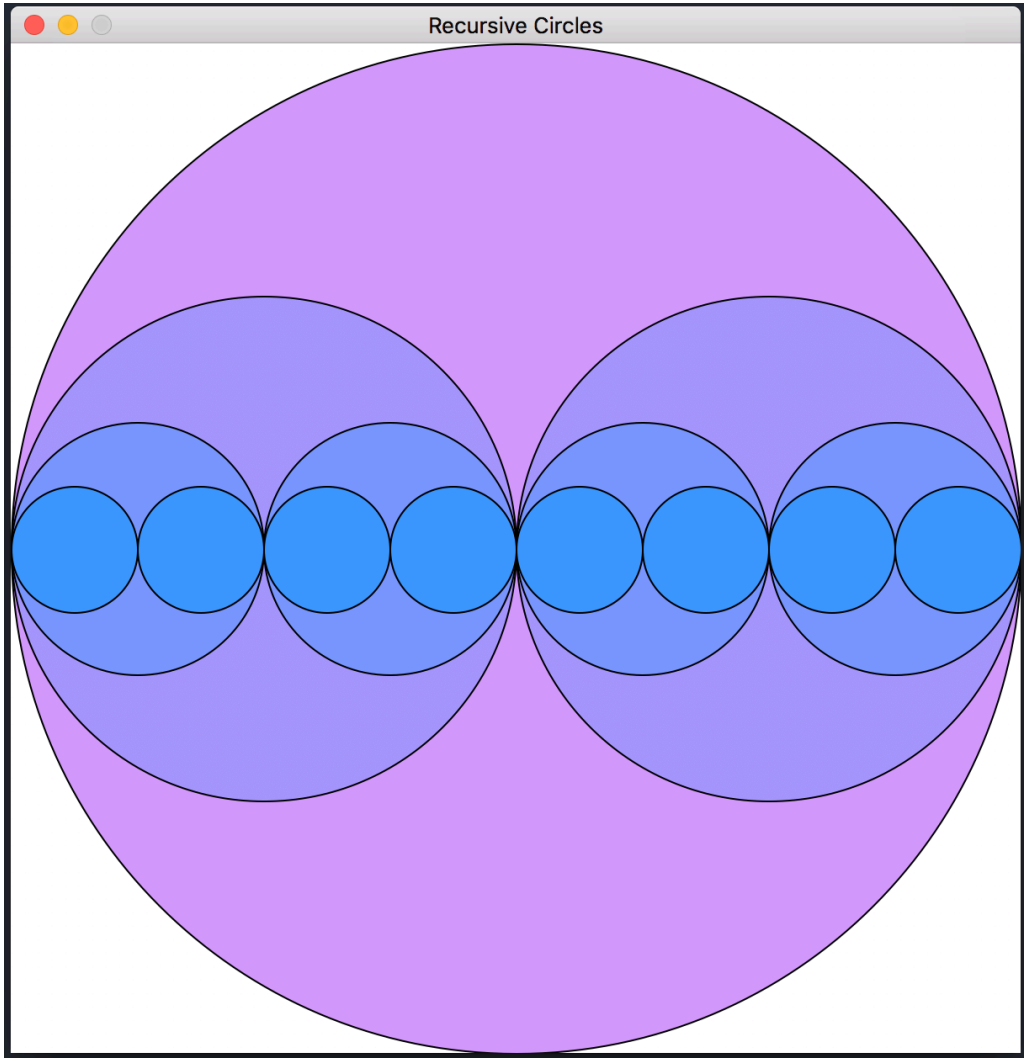
Recursive circles, n=2



Recursive circles, n=3



Recursive circles, n=4



Recursive circles, n=8