CSC 111:
Intro to Computer Science through Programming

Spring 2017
Prof. Sara Mathieson

Admin

- Homework 8 is due Tuesday April 11
- Homework 9 is due Tuesday April 18 (last homework)
- Final Project (roughly 1.5 homeworks) due Tuesday May 2
- Thursday office hours **10am-12pm (Ford 015)**
- Liberal Arts Module on Friday

Outline: 4/5

- Recap Homework 4
- Continue recursion (Fibonacci)
- Introduce Lab 8 and Homework 8
- Friday: liberal arts module (maps)

Notecards from Monday

- Question 1: understand well
  - Graphics
  - For-loops, if-statements
  - Nested structures
- Question 2: need to spend more time
  - While loops
  - Dictionaries
  - Different ways of reading files

Homework 4 Examples

(selected by Aditya)

Rock, paper, scissors by Yuhan

```
def game(score_comp, score_user):
    a = random.choice(["rock","paper","scissors"])
    b = input("enter rock, paper, or scissors: ")

    # see if it's a draw(user's choice = computer's)
    if a == b:
        print(a, "and", a, "is a tie")

    else:

        # if not a draw, find both user's and computer's choices in one sentence
        # of the list, then see which one starts the sentence.
        # The one that starts the sentence is the winner.

        for i in range(3):
            lst = ["rock smashes scissors","scissors cuts paper","paper covers rock"]
            string = lst[i].replace(a,"").replace(b,"")

            if string[0] == " " and string[-1] == " ":
                if a[0] == lst[i][0]:
                    print(lst[i]+", computer wins!")
                    score_comp = score_comp + 1
                else:
                    print(lst[i]+", user wins!")
                    score_user = score_user + 1

    return score_comp, score_user
```

Rock, paper, scissors by Yuhan

```
def main():
    score_comp = 0
    score_user = 0
    n = eval(input("Enter the number of rounds: "))


    # iteration of rounds
    for k in range(n):
        print("\nRound:",k+1)
        score_comp, score_user = game(score_comp, score_user)

        print("current score: user ",score_user,", computer ",score_comp, sep="")


    # Conclude final result
    print("\nThe final score is: user ",score_user,", computer ",score_comp, sep="")
    if score_user > score_comp:
        print("User wins")
    elif score_user < score_comp:
        print("Computer wins")
    else:
        print("Draw")
```

Rock, paper, scissors by Chloe

```
user_want_to_play = True          #A booleen variable to record if the user want to continue the game.

for i in range (rounds):
    if user_want_to_play == True:
        print ("round:",i)
        user_choice = input("Enter rock, paper, or scissors: ")

        #The booleen variable changes into False when the user doesn't want to play any more.
        if user_choice == "quit":
            user_want_to_play = False
            print ("Game Over!\n")

        else:
            computer_choice = random.choice(game_choices)

            if computer_choice == user_choice:
                print (user_choice , "and", computer_choice + " is a tie.")
                user_score = user_score
                computer_score = computer_score

            else:
                message(computer_choice,user_choice)        #Call the second helper function
                result = compare(computer_choice, user_choice)  #To change scores after one round
                if result.split()[-2] == "computer":
                    computer_score = computer_score + 1
                elif result.split()[-2] == "you":
                    user_score = user_score + 1

            print ("current score: user", user_score, ", computer", computer_score, "\n")
```

Rock, paper, scissors by Yingchuan

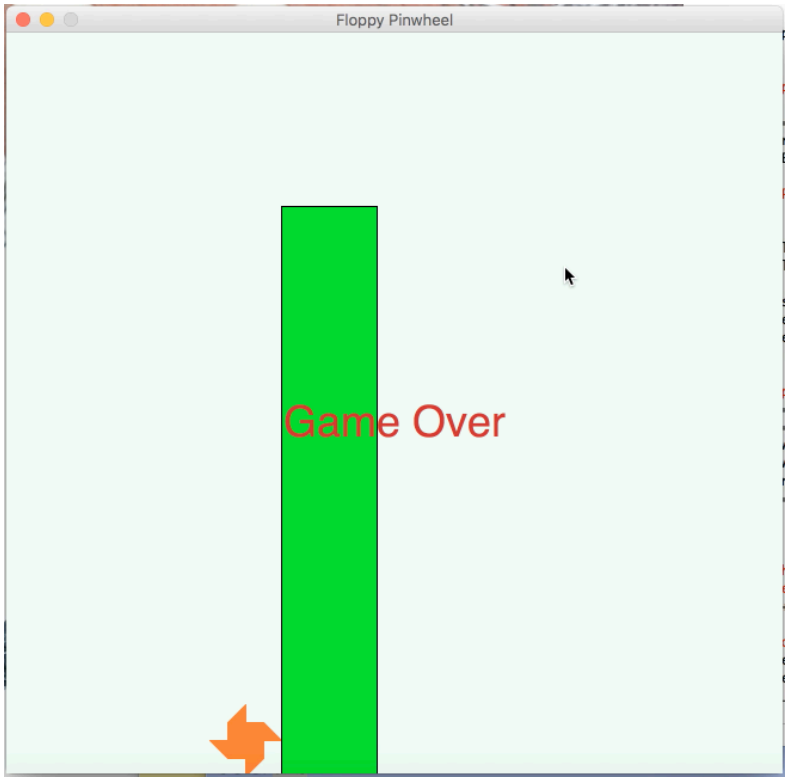
```
switch = True # Boolean variable

for i in range(round_number):
    if switch == True:
        user = input("enter rock, paper, or scissors: ")
        computer = random.choice(choice)

    if user == "quit": # stop the loop if user types 'quit'
        switch = False

    if switch == True:
        winner = game(user, computer)

        if winner == user:
            user_score = user_score + 1 # calculating the score in a cycle

        elif winner == computer:
            computer_score = computer_score +1

        print("current score: user",user_score, "computer",computer_score)
        print(" ")
```

Random partners by Chelsey

```
# create a function that performs random pairs function for 10 times
def random_pairs_10(lst):

    total_lst = []   # a list to store all the pairs we had before
    for i in range(1,11):
        print("For week",str(i)+":")
        print()

        # call the random pairs function and assign the value of the partner list into the variable "partner_lst"
        lab_lst = random_pairs(lst)

        for item in lab_lst:

            # if in the list we have the same pair as in the total list, we will do random pair function again
            # and create a new list until there is no pair that is same as those in the total list.
            if item in total_lst:
                lab_lst = random_pairs(lst)

        total_lst.append(lab_lst)  # update the total list to include week i's partner list
```

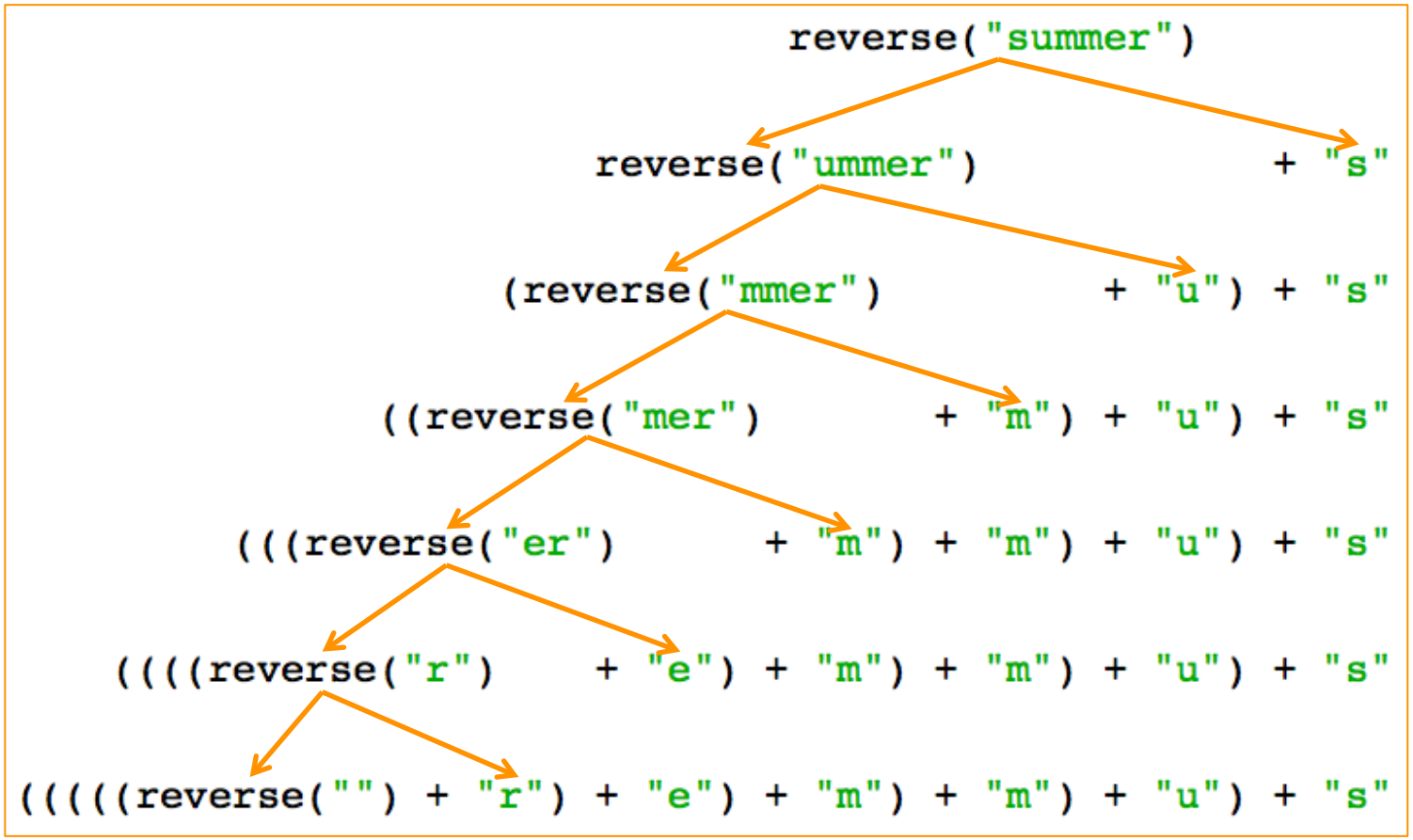
Homework 7 Extensions

Flappy Pinwheel by Mai and Butterfly Catcher by Isabelle



Continue Recursion

Reversing a string



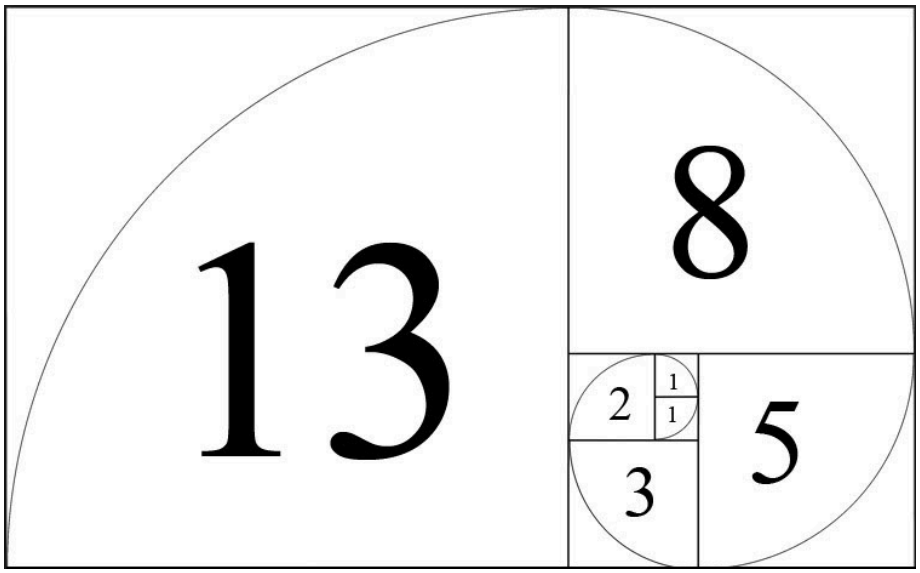
Fibonacci Example

Fibonacci numbers

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

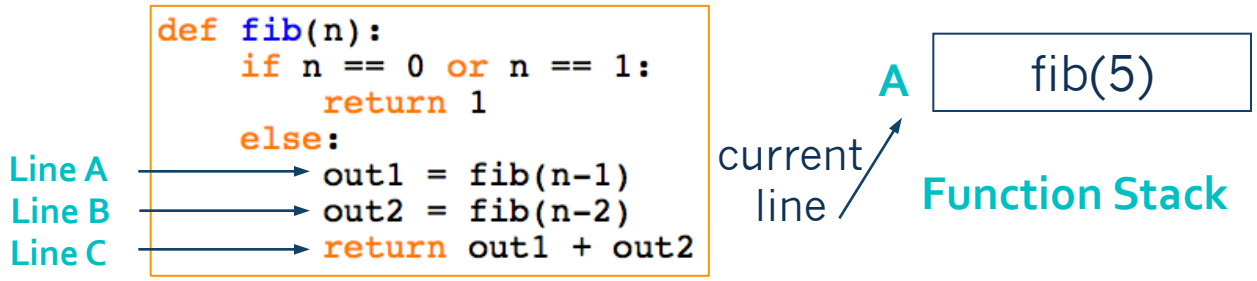
Recursion: $F_n = F_{n-1} + F_{n-2}$

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



Fibonacci Function Stack

fib(5)



Fibonacci Function Stack

fib(5)

fib(4)

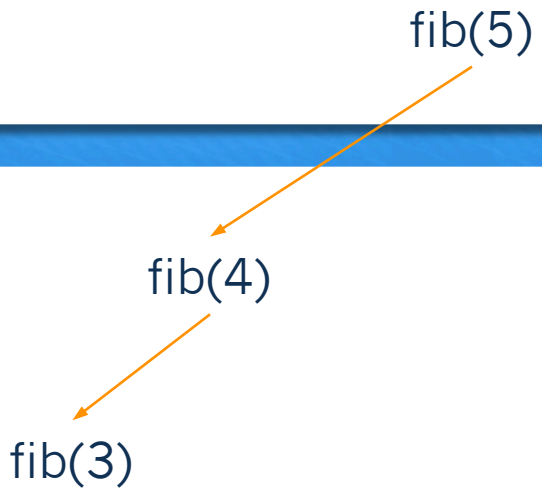
```
def fib(n):
    if n == 0 or n == 1:
        return 1
    else:
        out1 = fib(n-1)
        out2 = fib(n-2)
        return out1 + out2
```

Line A
Line B
Line C

current line

Function Stack

Fibonacci Function Stack



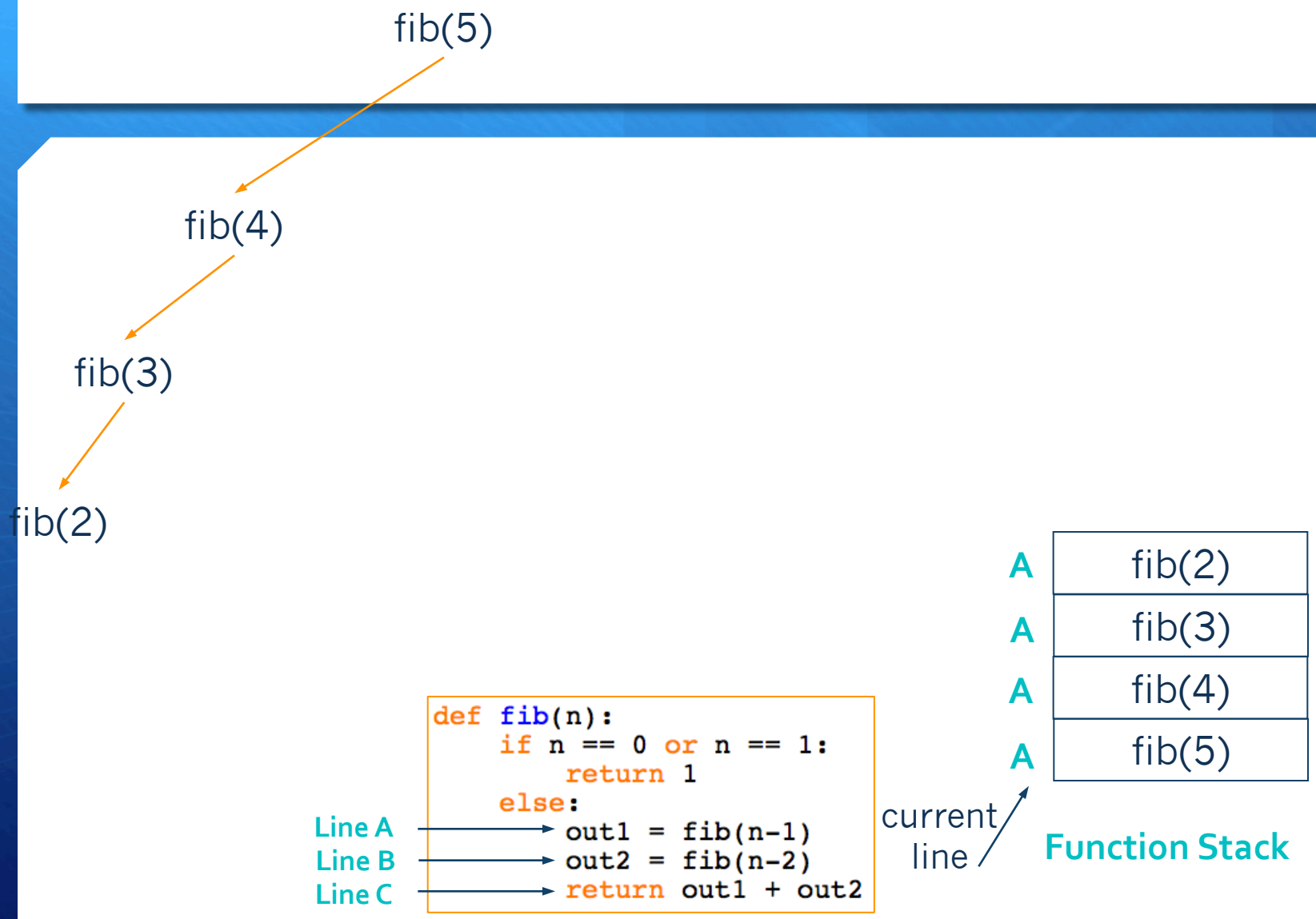

```
def fib(n):
    if n == 0 or n == 1:
        return 1
    else:
        out1 = fib(n-1)
        out2 = fib(n-2)
        return out1 + out2
```

Line A
Line B
Line C

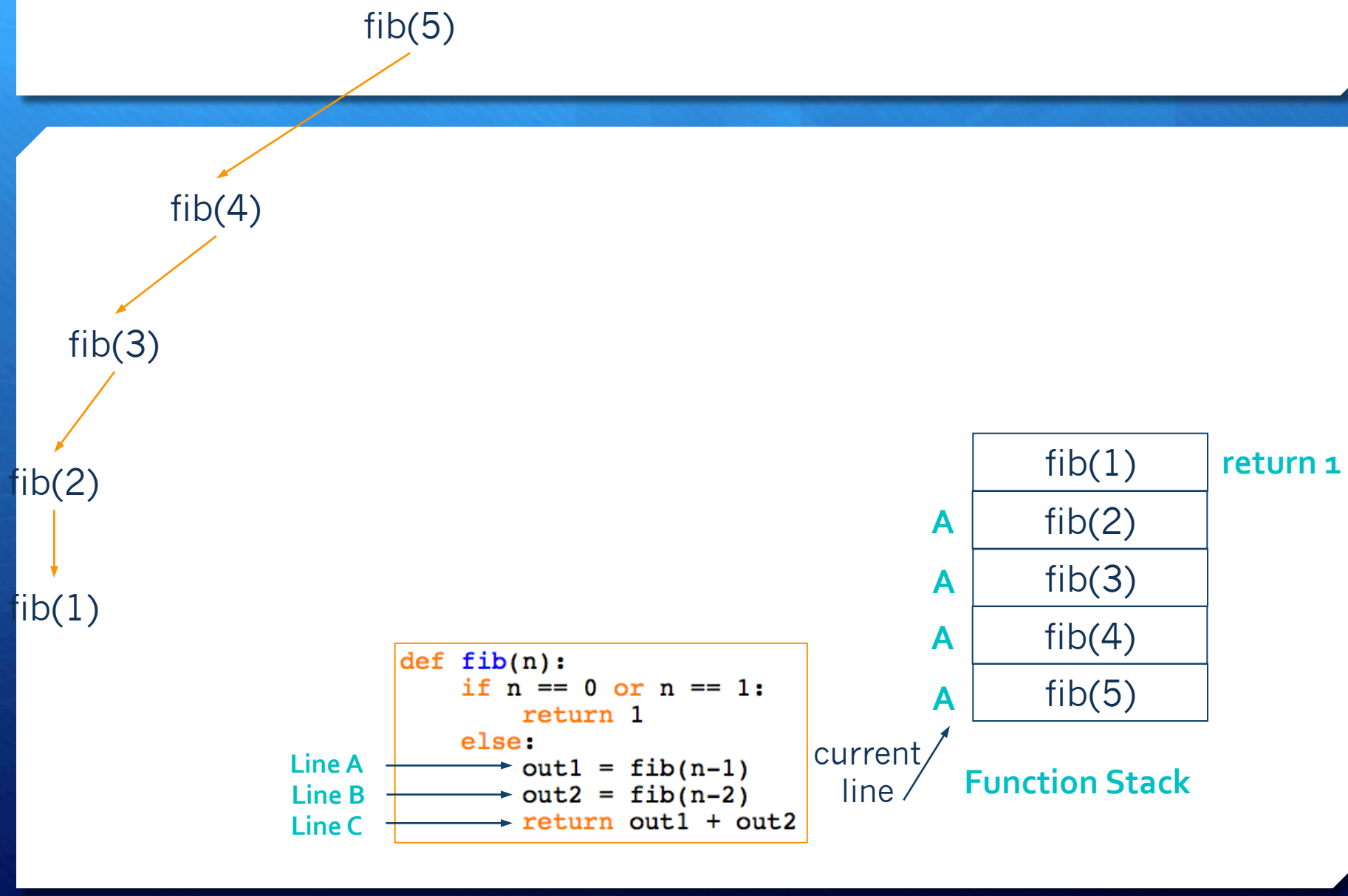
current line

Function Stack

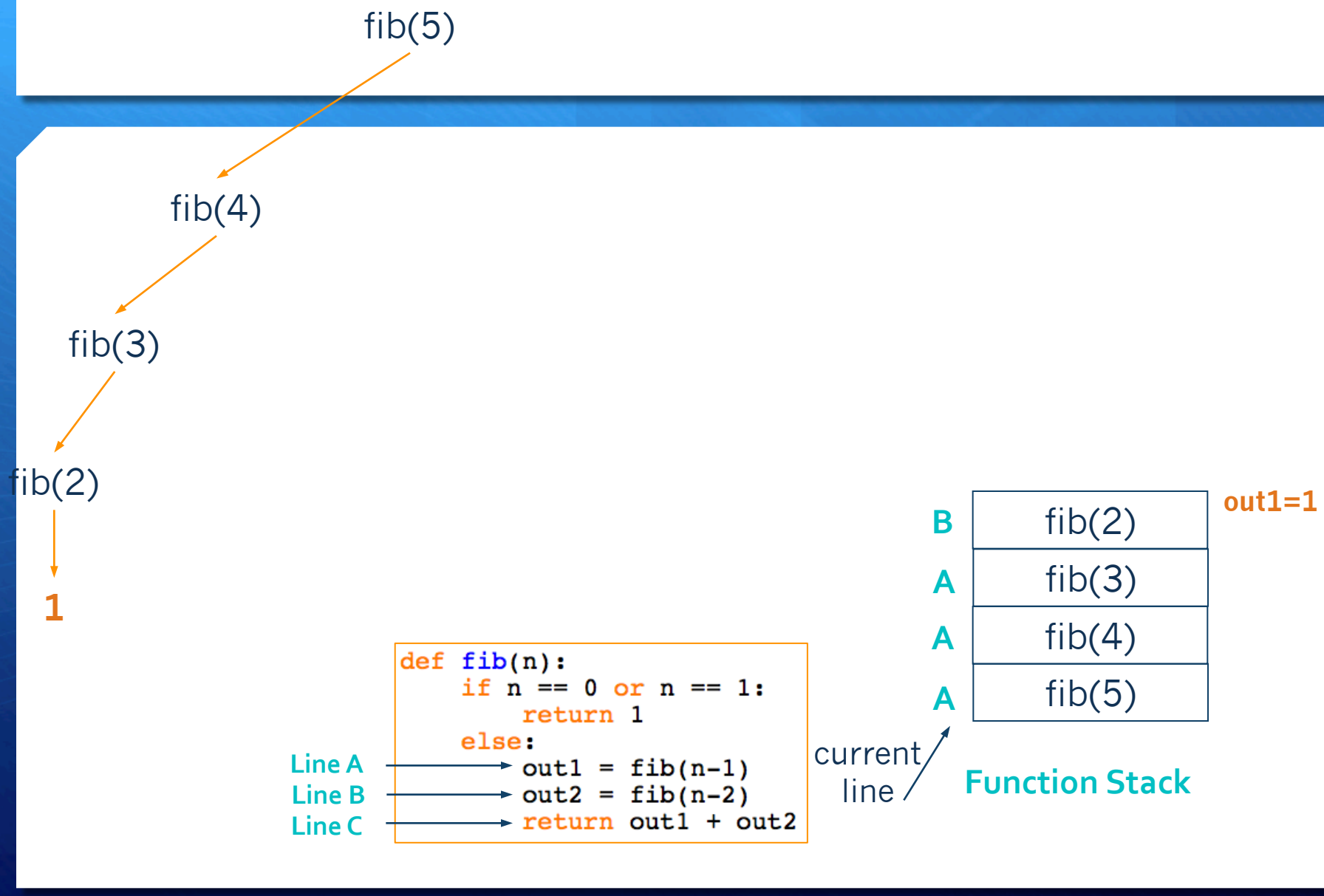
Fibonacci Function Stack



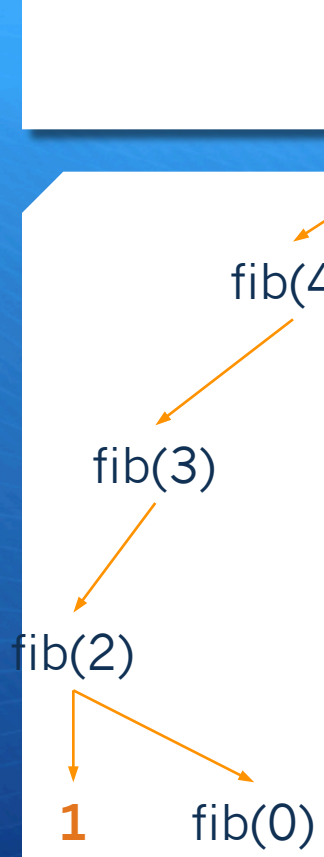
Fibonacci Function Stack



Fibonacci Function Stack



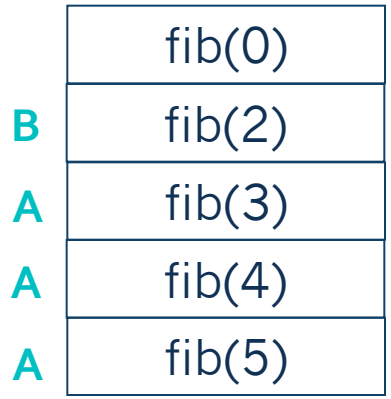
Fibonacci Function Stack




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

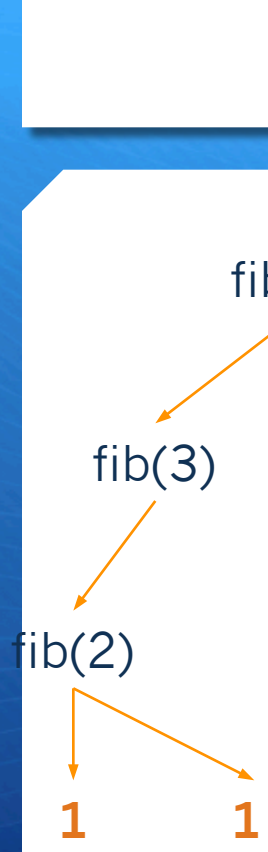
Line A
Line B
Line C

current line



Function Stack

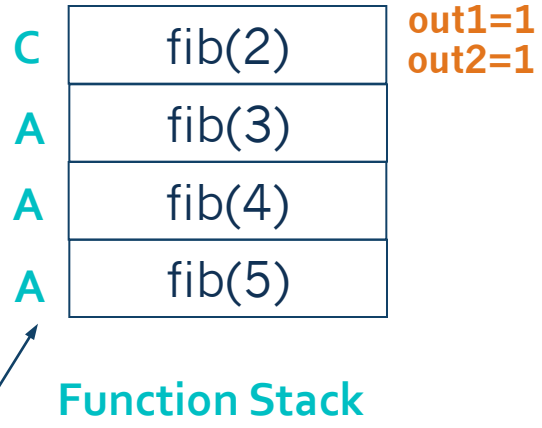
Fibonacci Function Stack




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

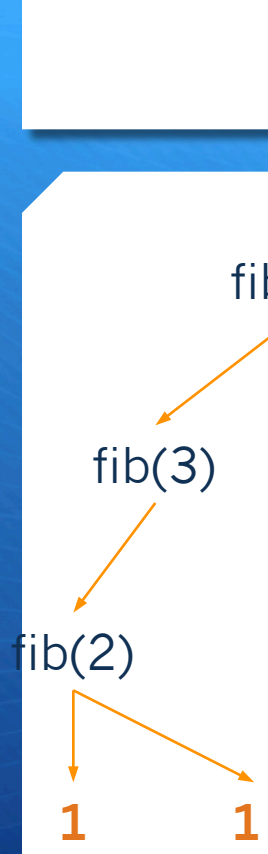
Line A
Line B
Line C

current line



Function Stack

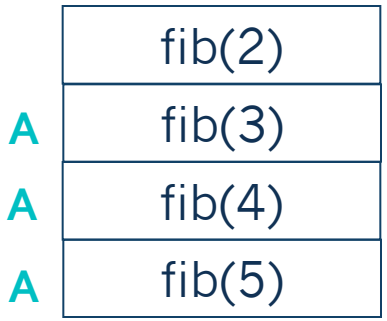
Fibonacci Function Stack




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

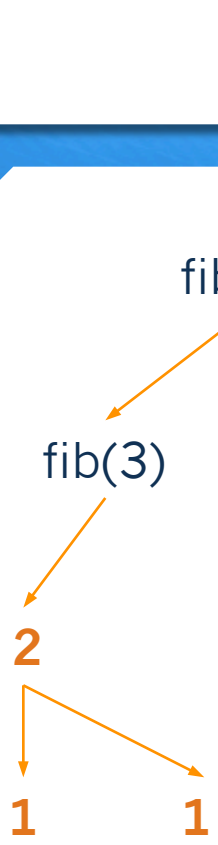
Line A
Line B
Line C

current line



Function Stack

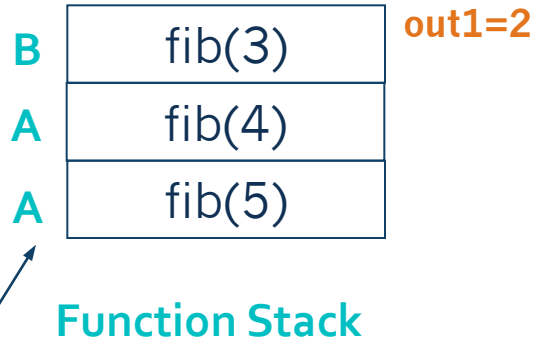
Fibonacci Function Stack



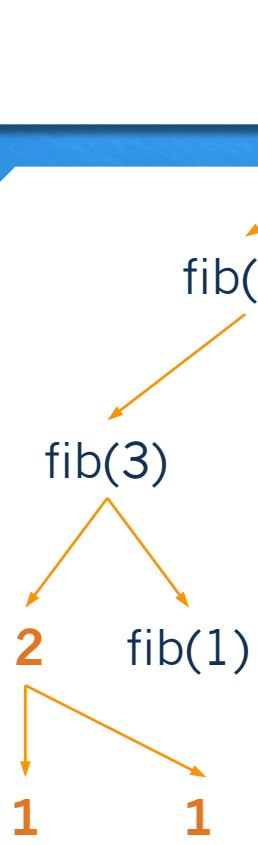

```
def fib(n):
    if n == 0 or n == 1:
        return 1
    else:
        out1 = fib(n-1)
        out2 = fib(n-2)
        return out1 + out2
```

Line A
Line B
Line C

current line



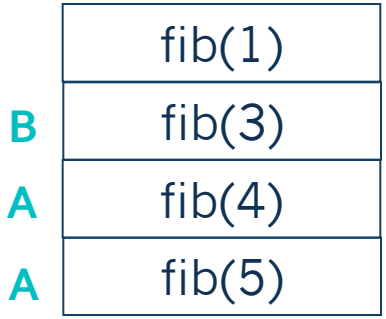
Fibonacci Function Stack




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

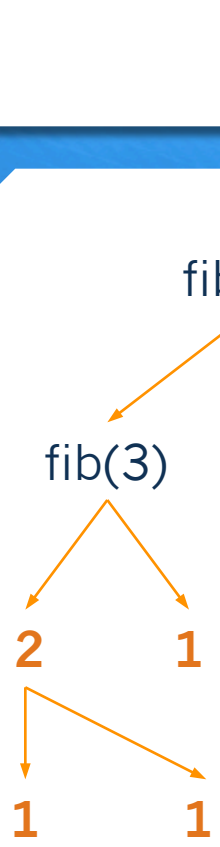
Line A
Line B
Line C

current line



Function Stack

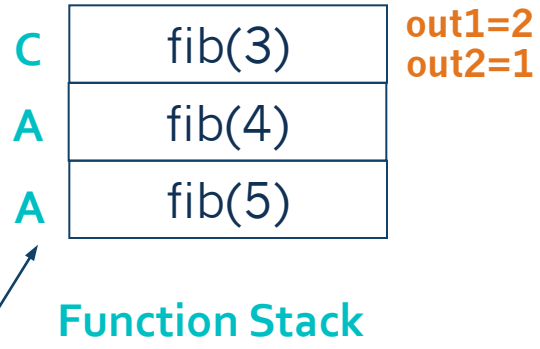
Fibonacci Function Stack



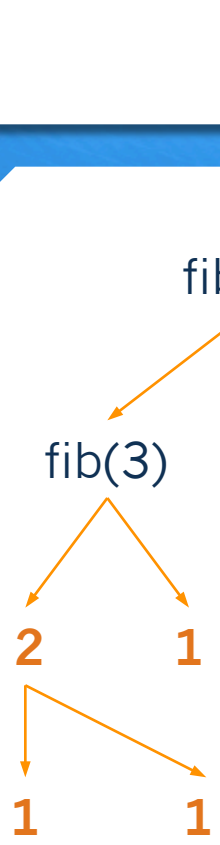

```
def fib(n):
    if n == 0 or n == 1:
        return 1
    else:
        out1 = fib(n-1)
        out2 = fib(n-2)
        return out1 + out2
```

Line A
Line B
Line C

current line



Fibonacci Function Stack



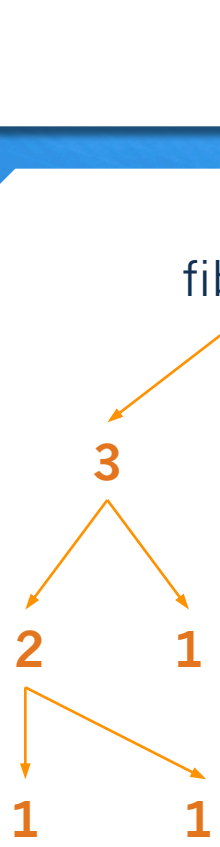

```
def fib(n):
    if n == 0 or n == 1:
        return 1
    else:
        out1 = fib(n-1)
        out2 = fib(n-2)
        return out1 + out2
```

Line A
Line B
Line C

current line

Function Stack

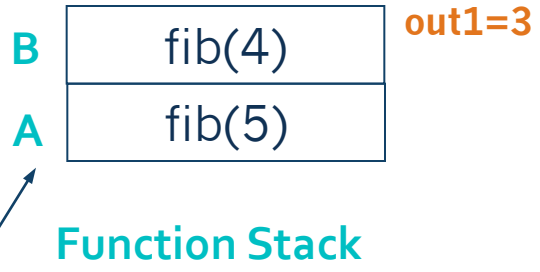
Fibonacci Function Stack



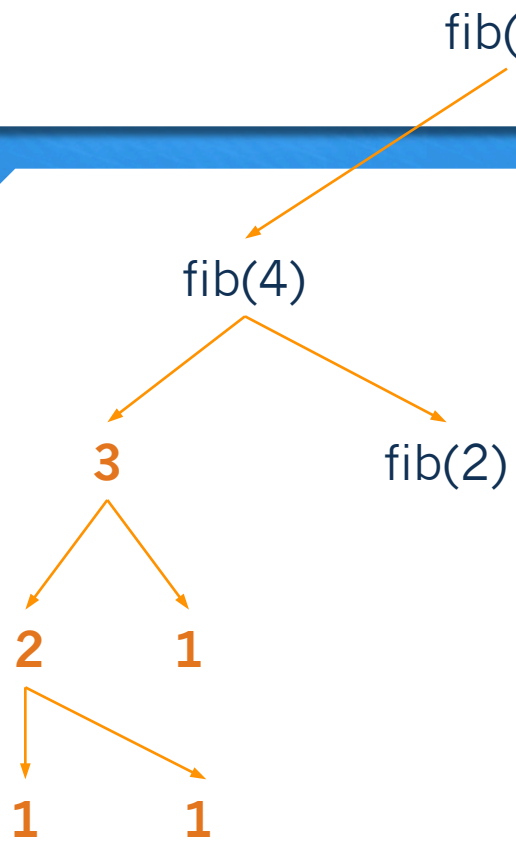

```
def fib(n):
    if n == 0 or n == 1:
        return 1
    else:
        out1 = fib(n-1)
        out2 = fib(n-2)
        return out1 + out2
```

Line A
Line B
Line C

current line



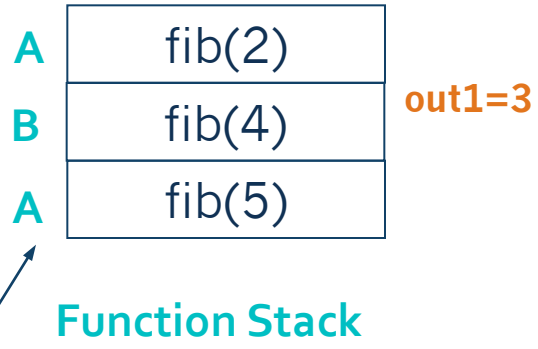
Fibonacci Function Stack



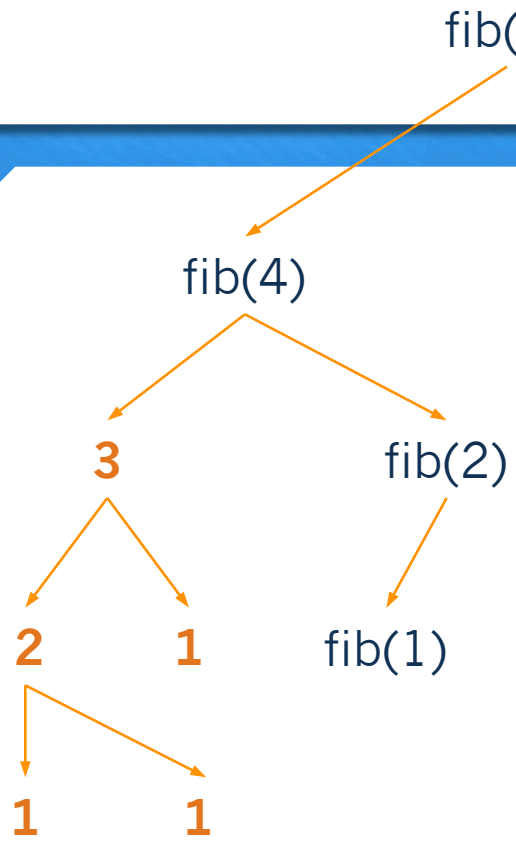

```
def fib(n):
    if n == 0 or n == 1:
        return 1
    else:
        out1 = fib(n-1)
        out2 = fib(n-2)
        return out1 + out2
```

Line A
Line B
Line C

current line



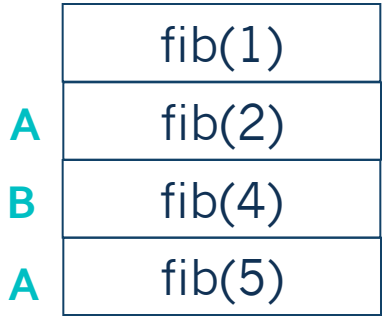
Fibonacci Function Stack




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

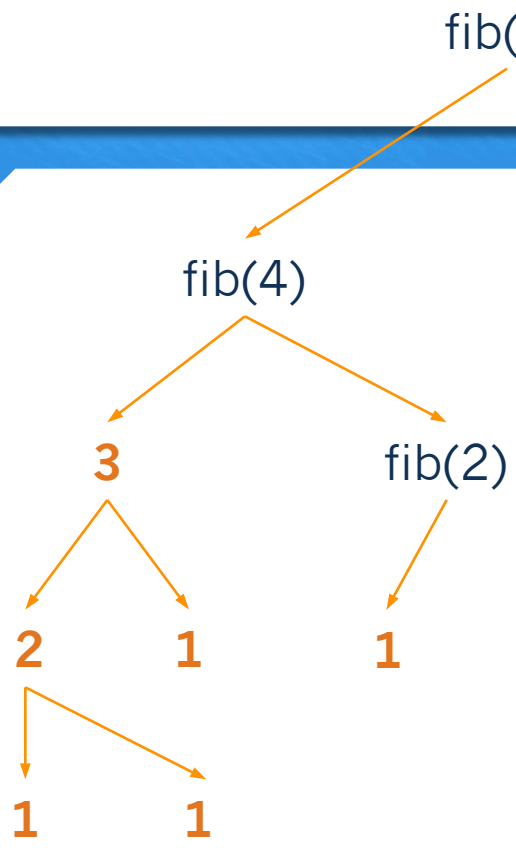
Line A
Line B
Line C

current line



Function Stack

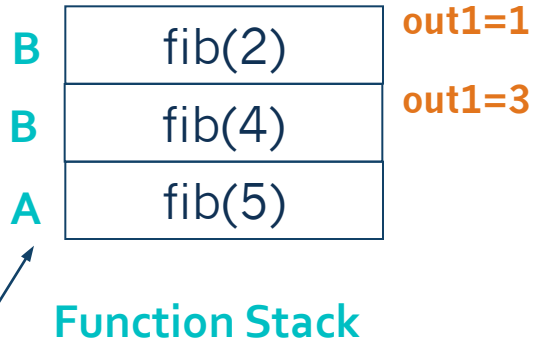
Fibonacci Function Stack



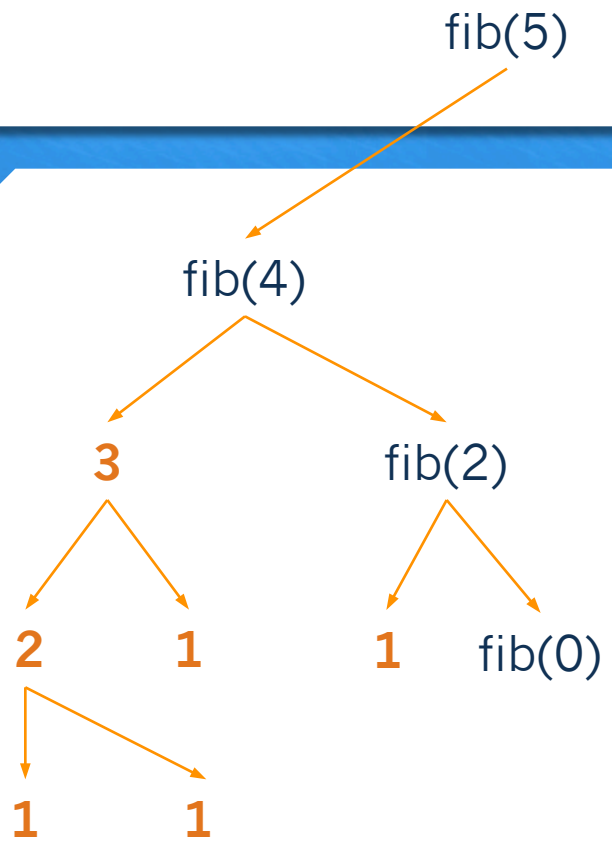

```
def fib(n):
    if n == 0 or n == 1:
        return 1
    else:
        out1 = fib(n-1)
        out2 = fib(n-2)
        return out1 + out2
```

Line A
Line B
Line C

current line



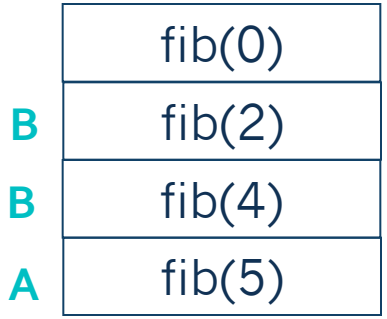
Fibonacci Function Stack




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

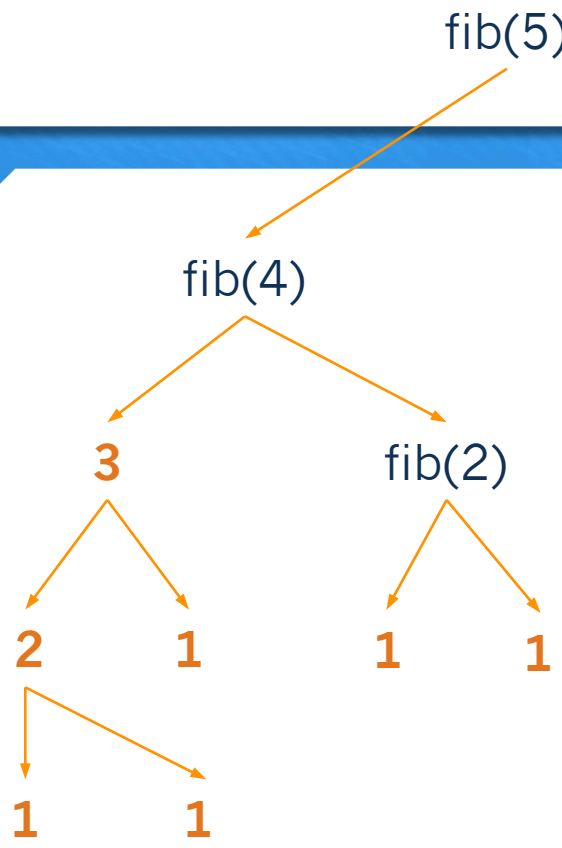
Line A
Line B
Line C

current line



Function Stack

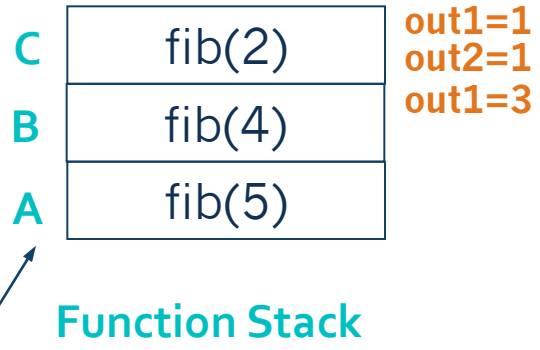
Fibonacci Function Stack




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

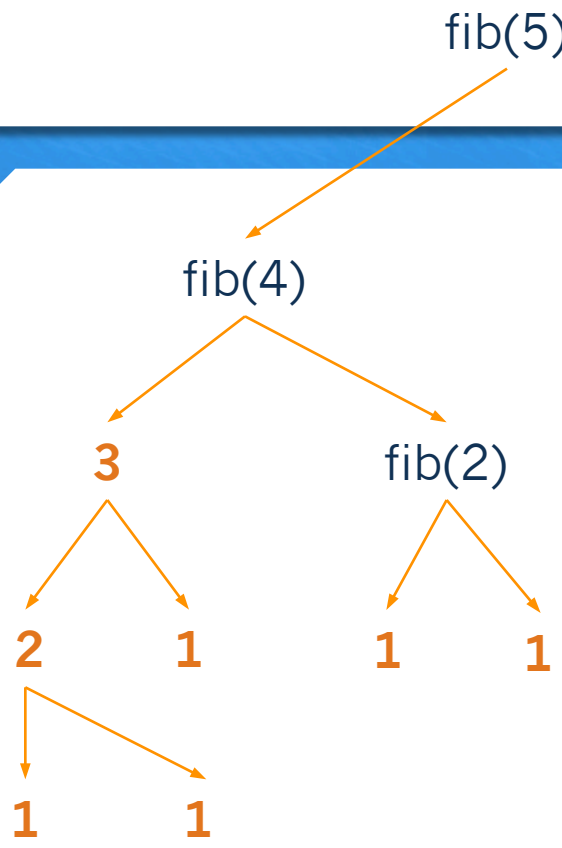
Line A
Line B
Line C

current line



Function Stack

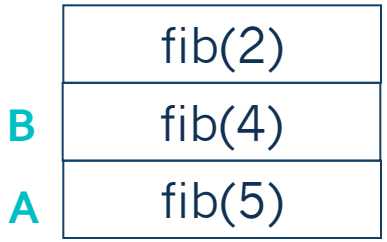
Fibonacci Function Stack




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

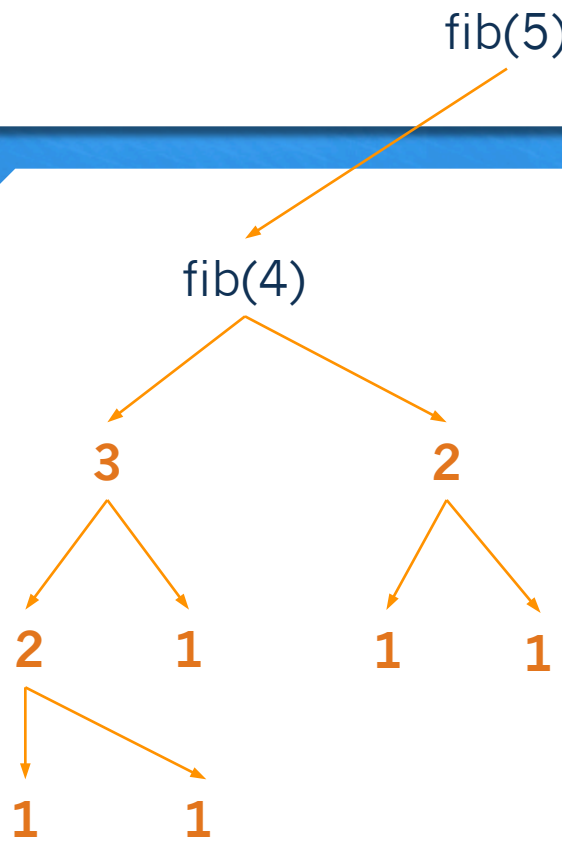
Line A
Line B
Line C

current line



Function Stack

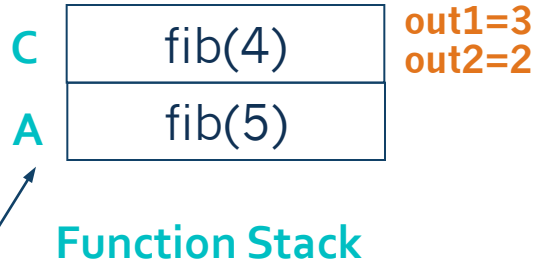
Fibonacci Function Stack



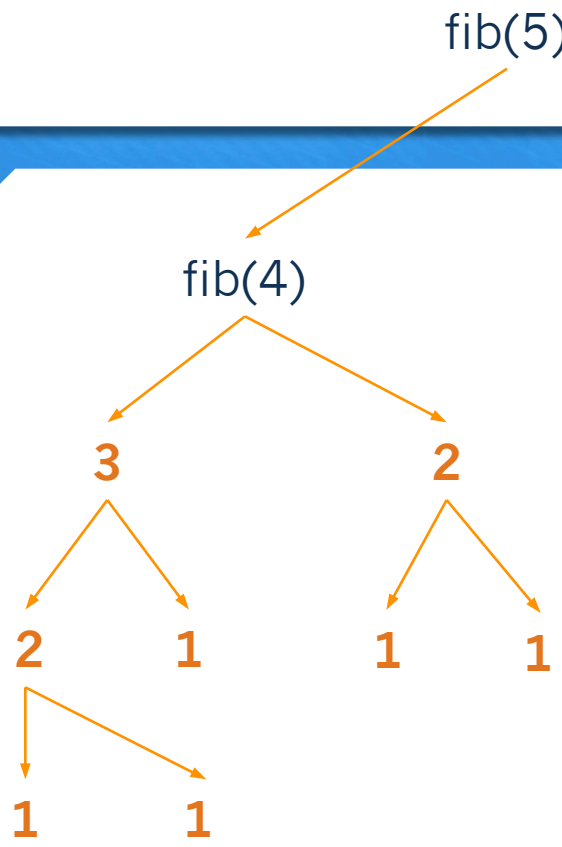

```
def fib(n):
    if n == 0 or n == 1:
        return 1
    else:
        out1 = fib(n-1)
        out2 = fib(n-2)
        return out1 + out2
```

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

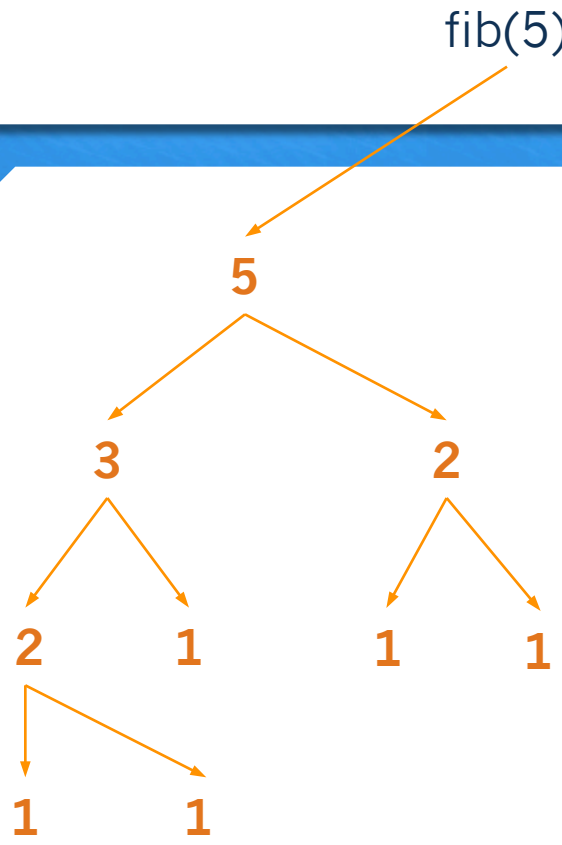
A

fib(4)
fib(5)

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

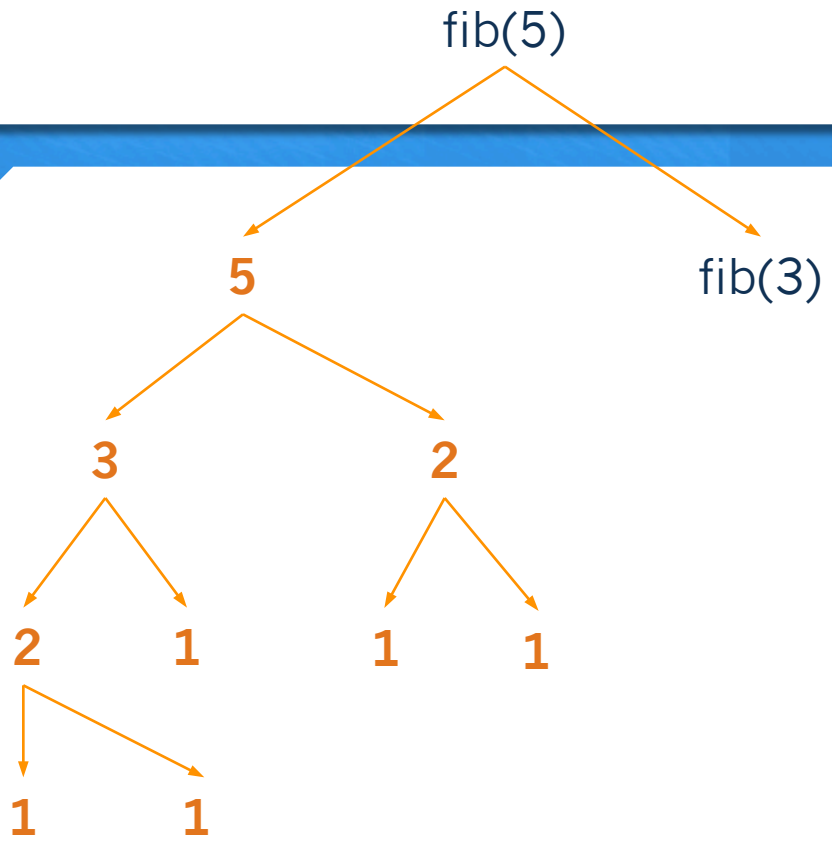
B

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

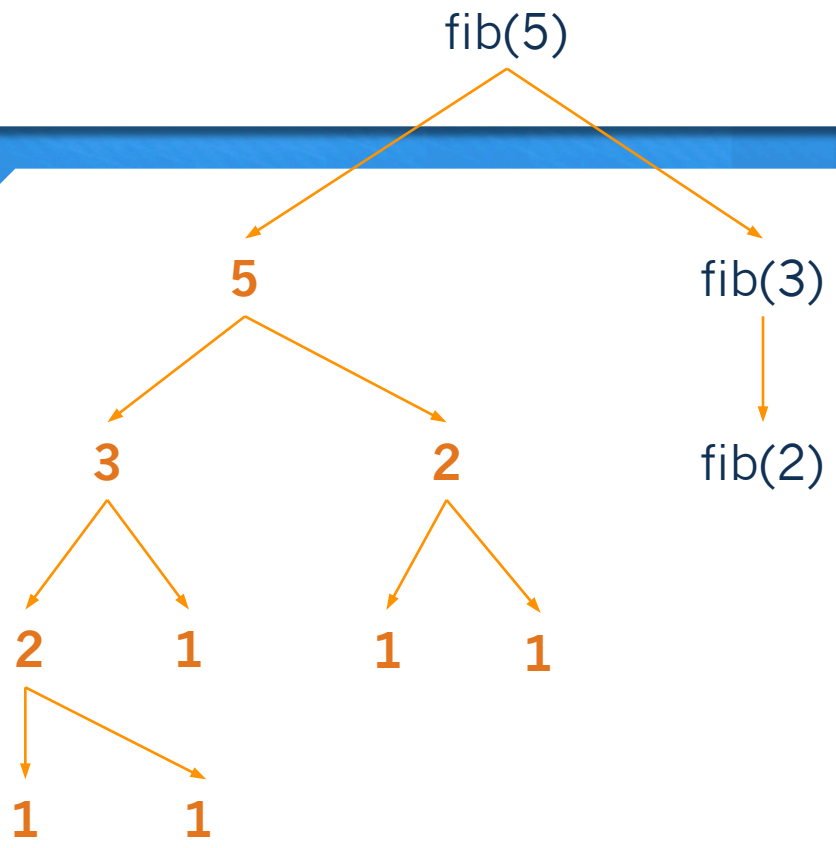
current line

A fib(3)
B fib(5)

out1=5

Function Stack

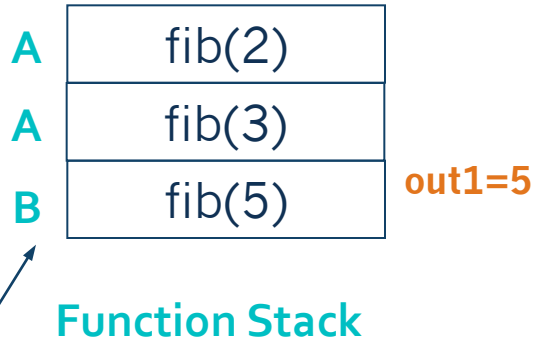
Fibonacci Function Stack



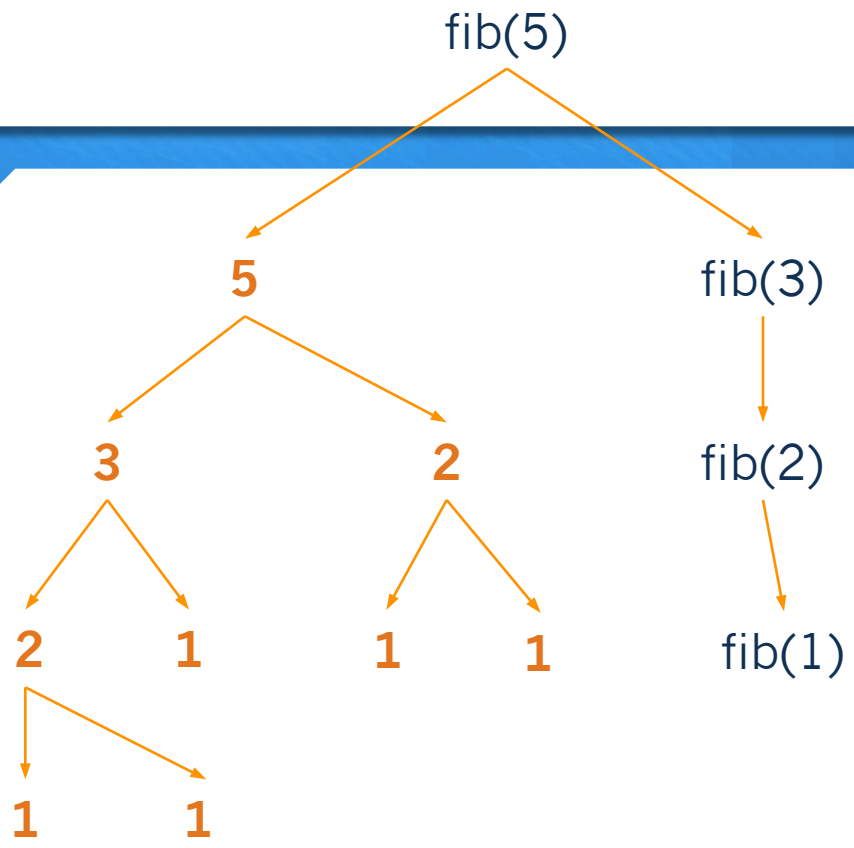

```
def fib(n):
    if n == 0 or n == 1:
        return 1
    else:
        out1 = fib(n-1)
        out2 = fib(n-2)
        return out1 + out2
```

Line A
Line B
Line C

current line



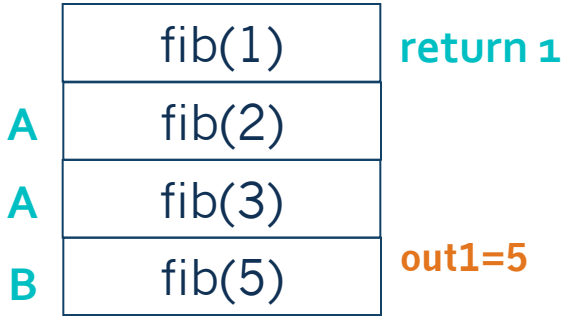
Fibonacci Function Stack




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

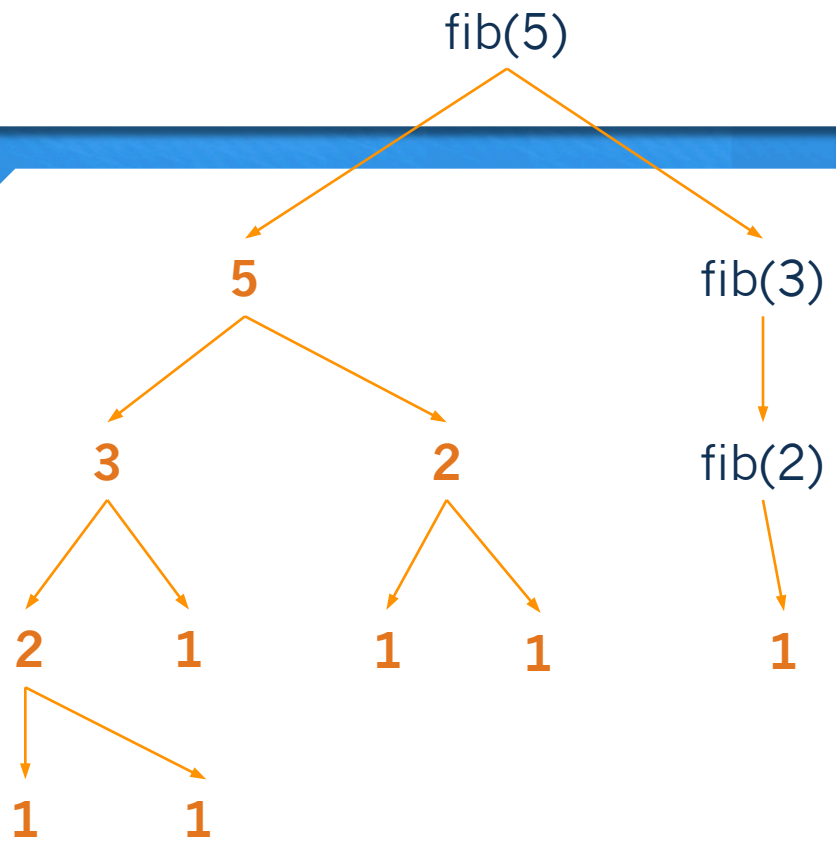
Line A
Line B
Line C

current line



Function Stack

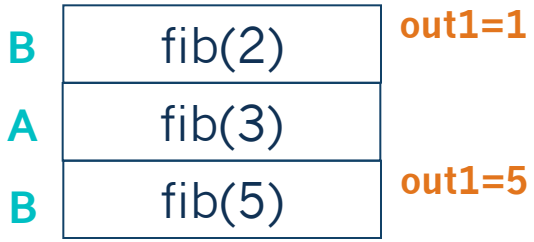
Fibonacci Function Stack




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

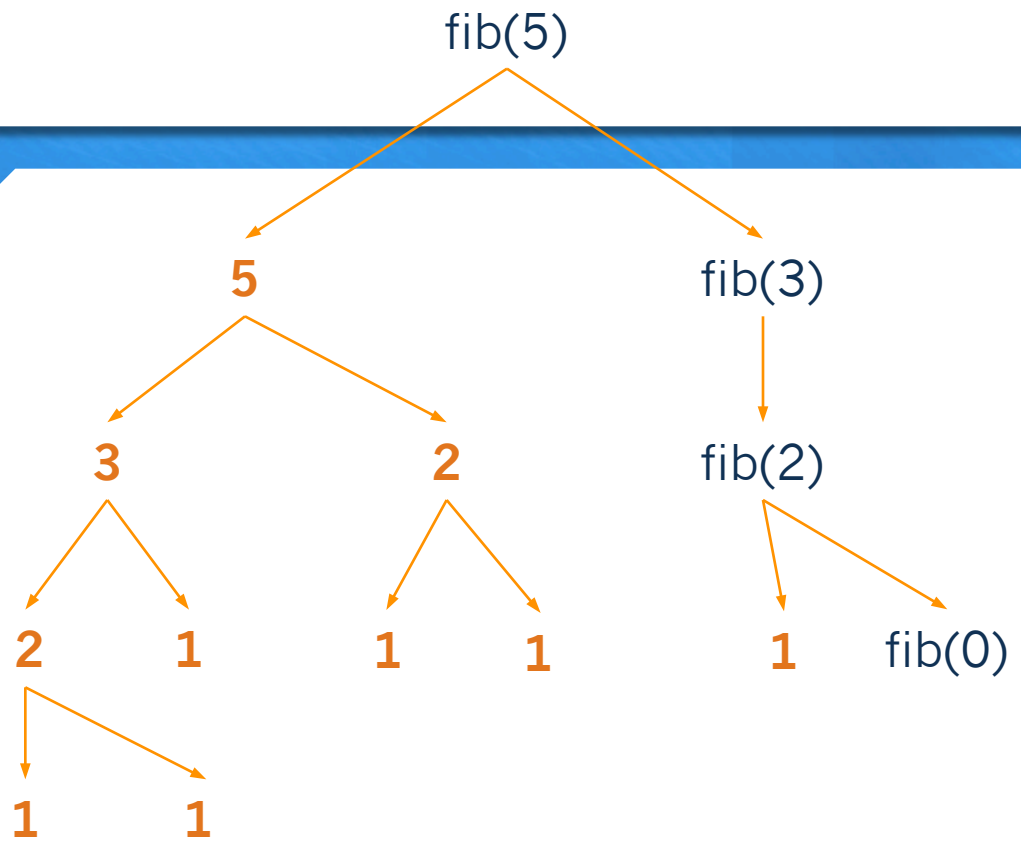
Line A
Line B
Line C

current line



Function Stack

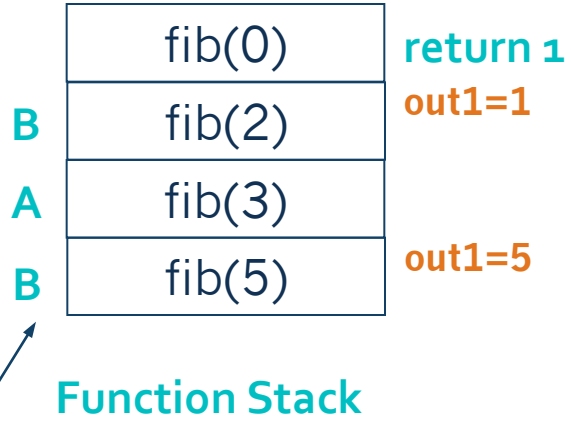
Fibonacci Function Stack




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

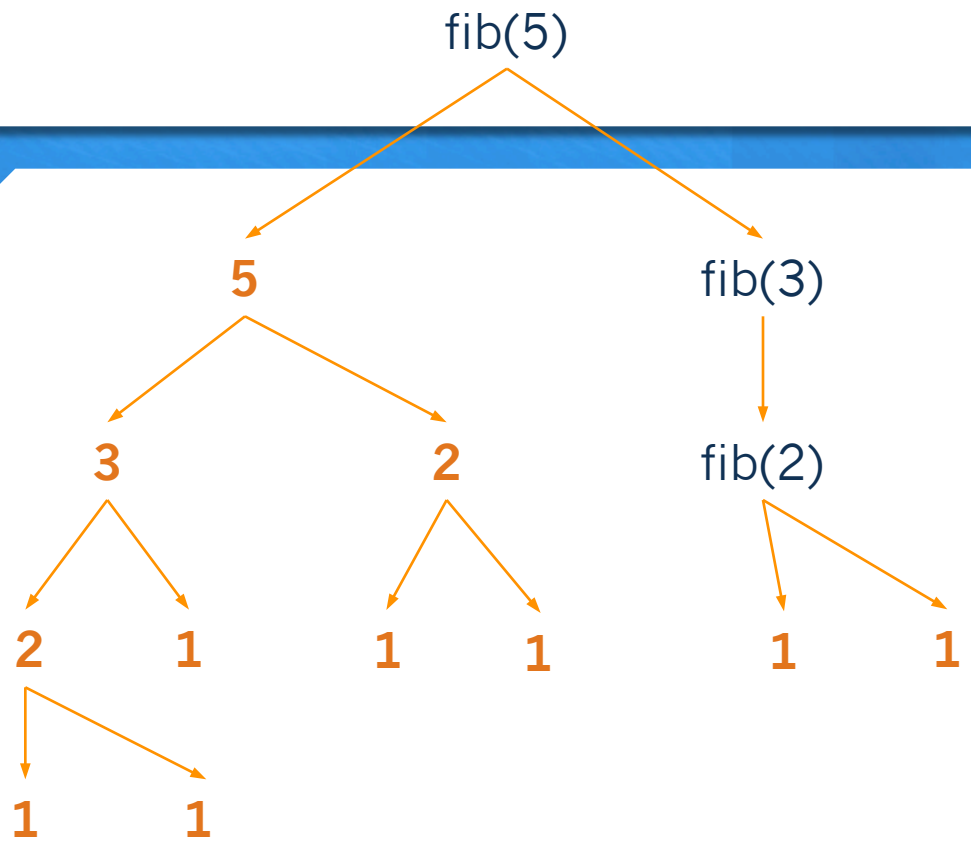
Line A
Line B
Line C

current line



Function Stack

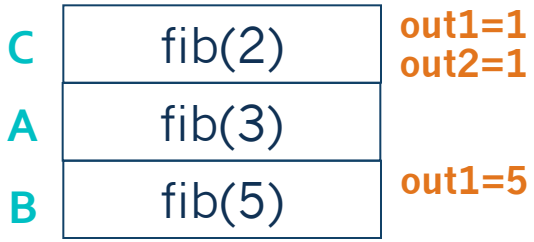
Fibonacci Function Stack




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

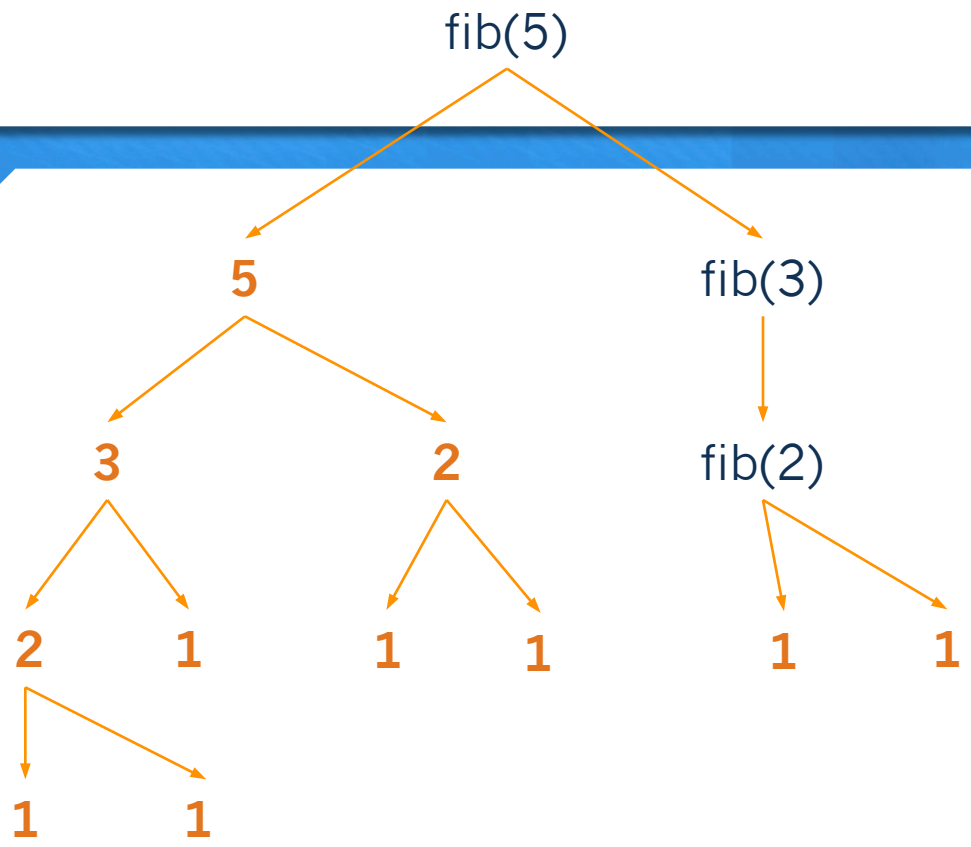
Line A
Line B
Line C

current line



Function Stack

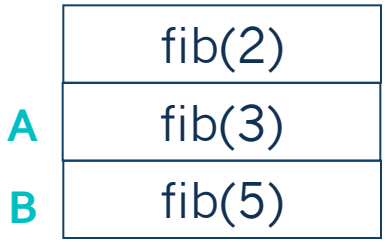
Fibonacci Function Stack




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

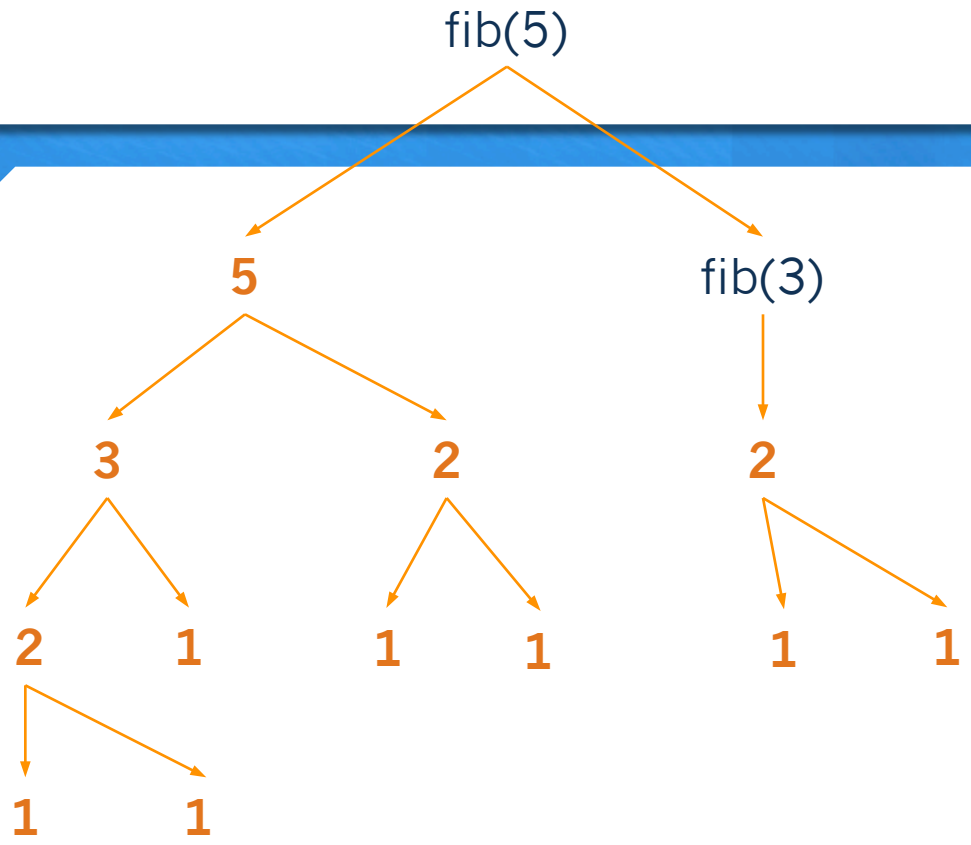
Line A
Line B
Line C

current line



Function Stack

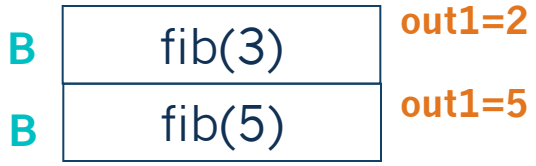
Fibonacci Function Stack




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

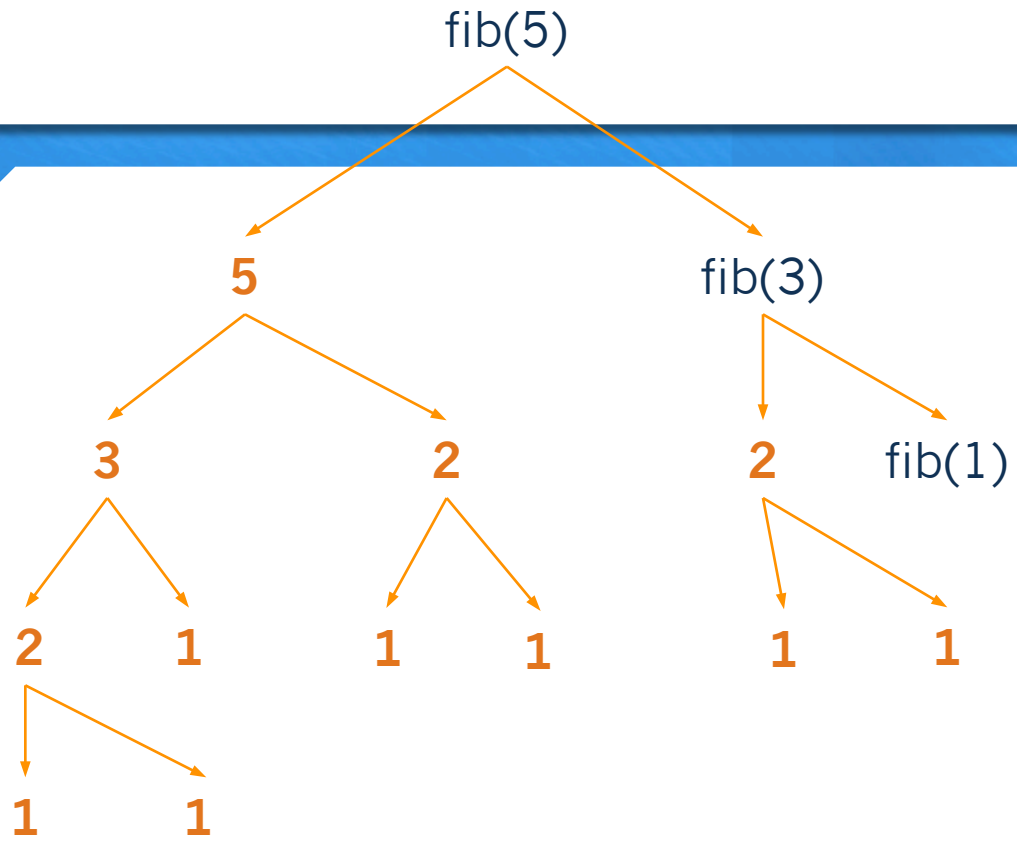
Line A
Line B
Line C

current line



Function Stack

Fibonacci Function Stack



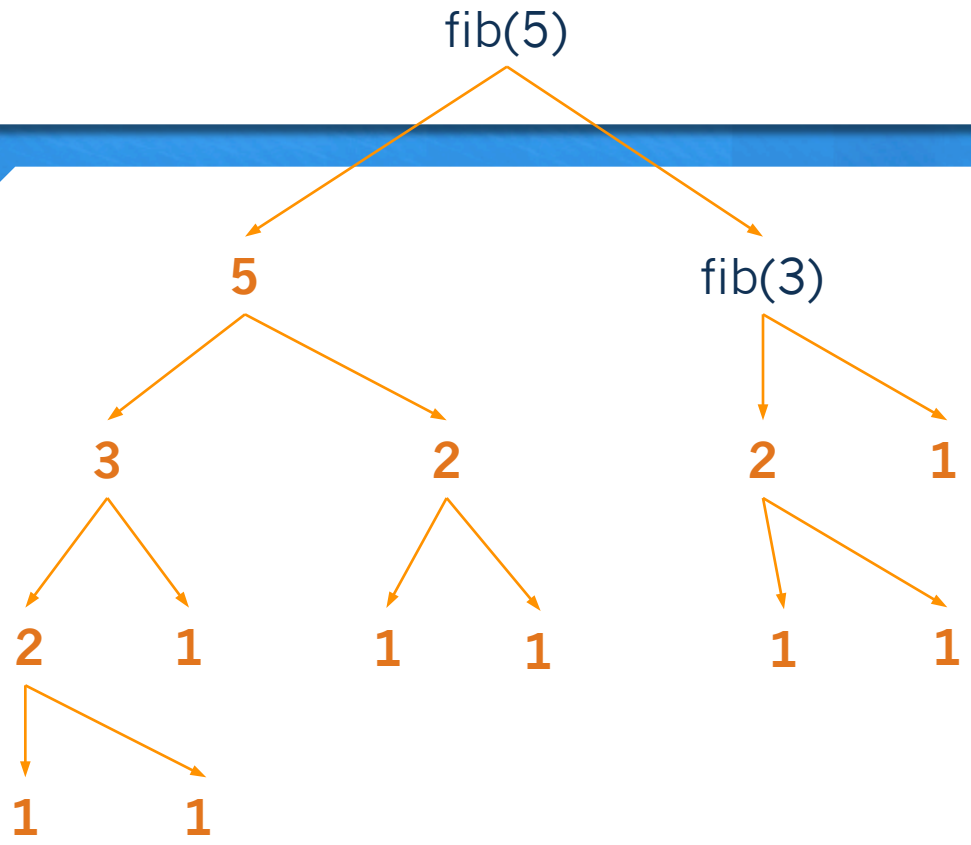

```
def fib(n):
    if n == 0 or n == 1:
        return 1
    else:
        out1 = fib(n-1)
        out2 = fib(n-2)
        return out1 + out2
```

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

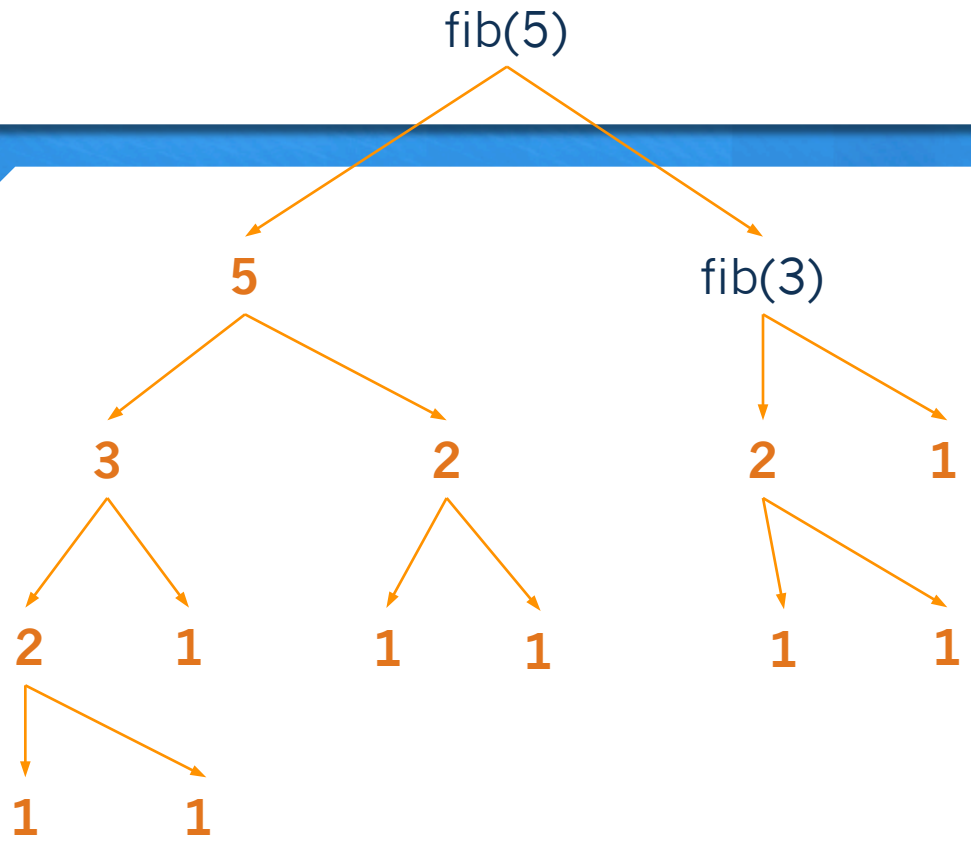
current line

C fib(3) out1=2 out2=1

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

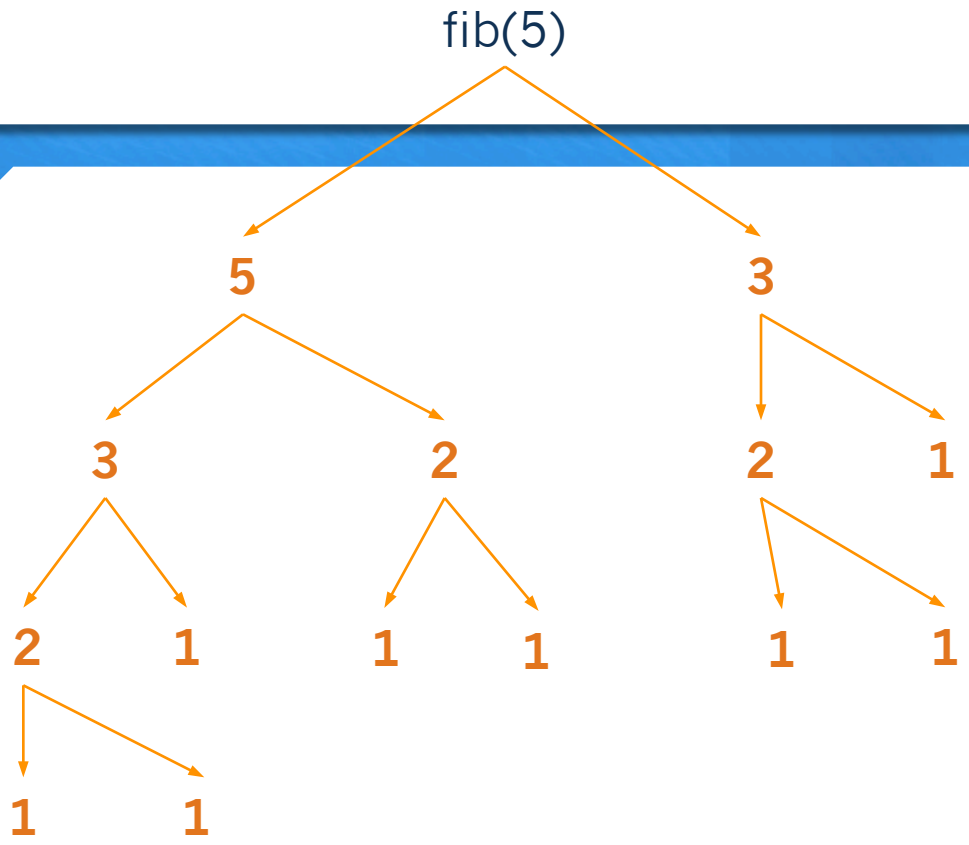
current line

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
```

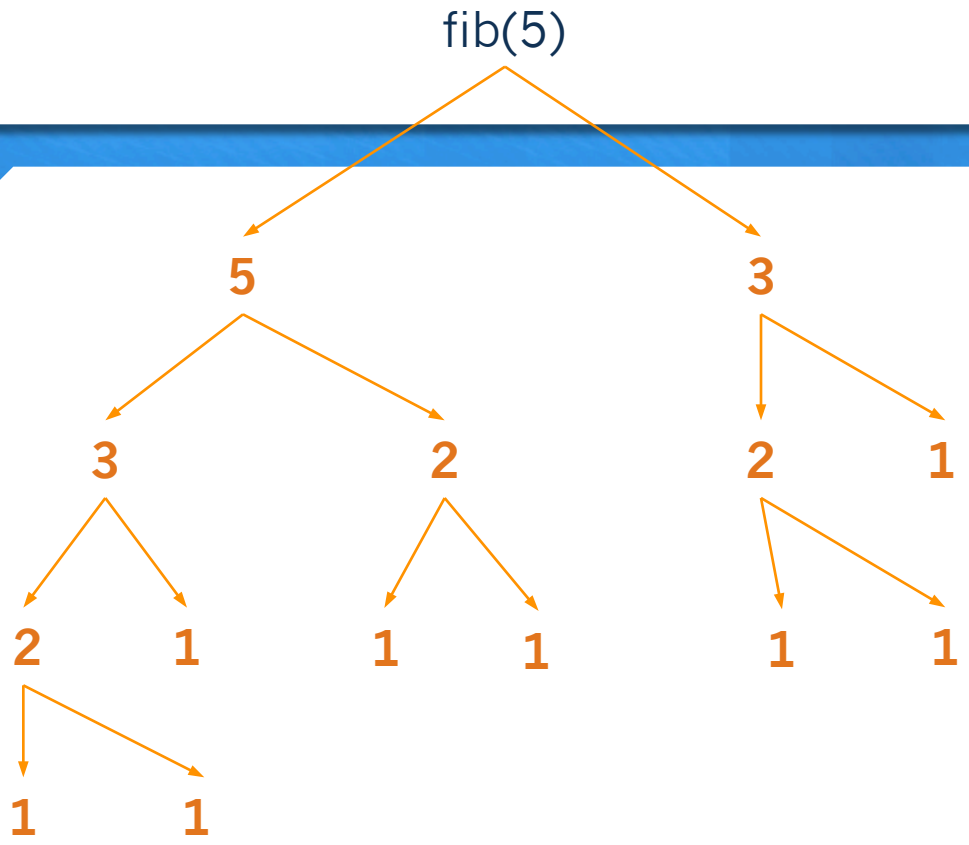
Line A
Line B
Line C

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
```

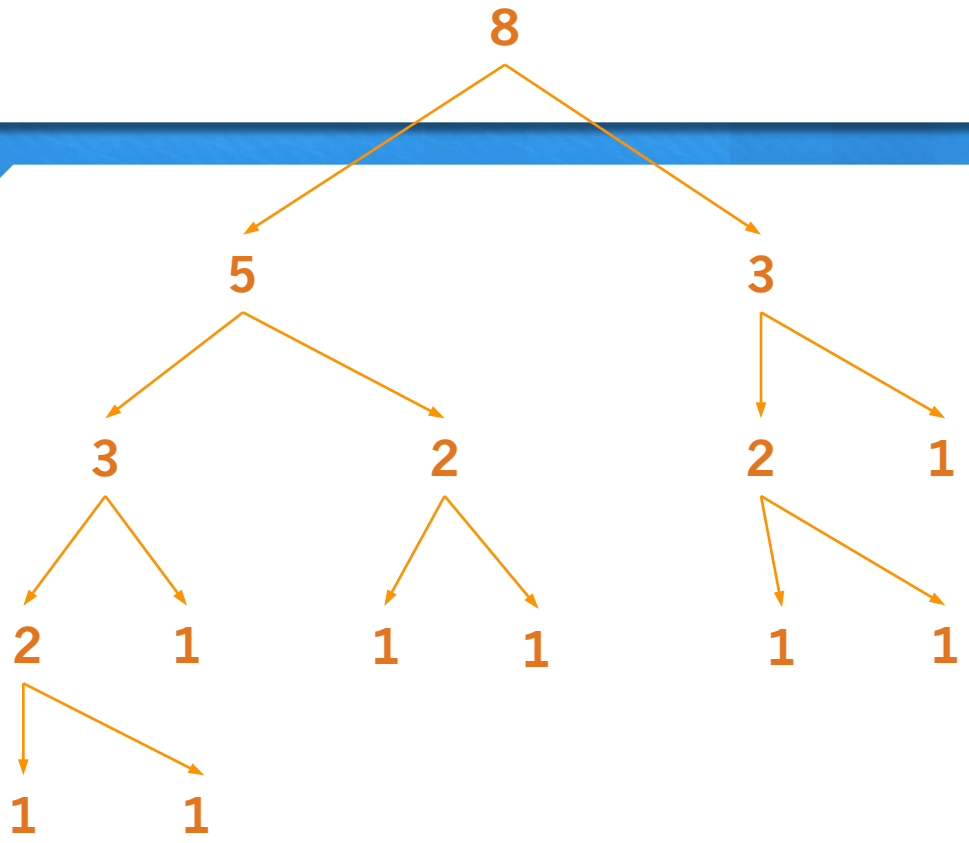
Line A
Line B
Line C

current line

fib(5) return 8

Function Stack

Fibonacci Function Stack



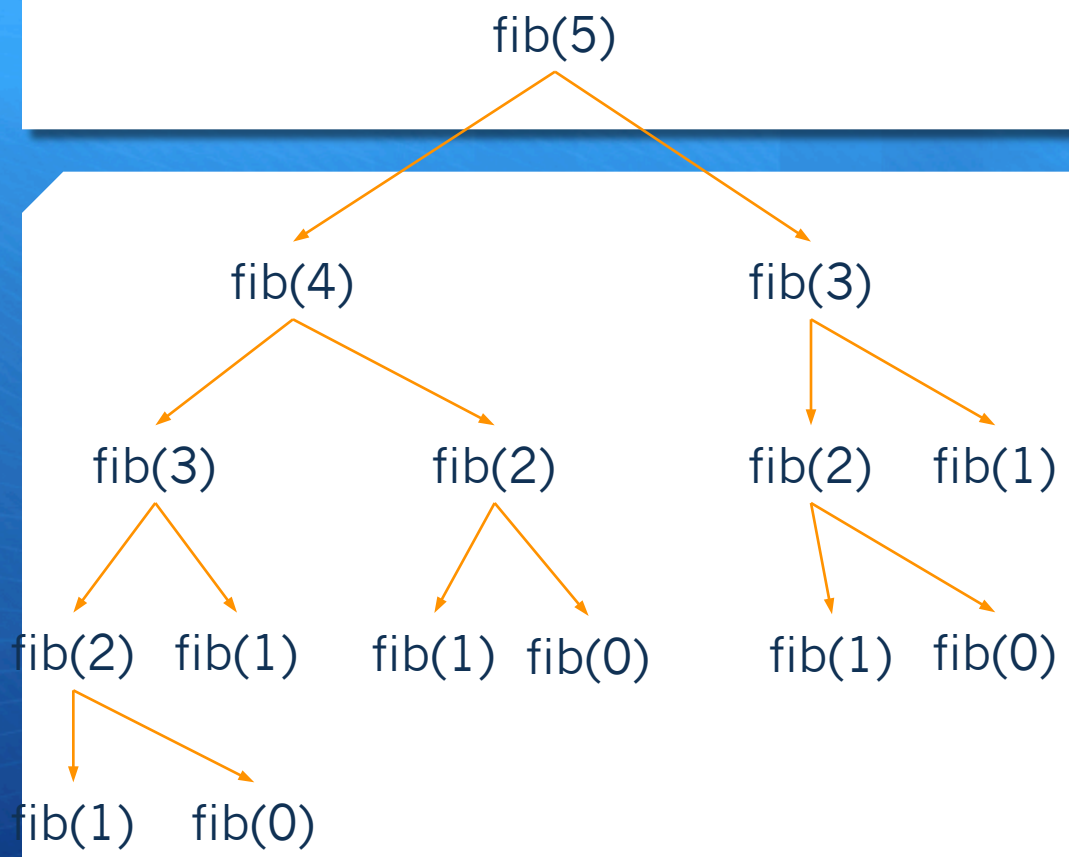

```
def fib(n):
    if n == 0 or n == 1:
        return 1
    else:
        out1 = fib(n-1)
        out2 = fib(n-2)
        return out1 + out2
```

Line A → `out1 = fib(n-1)`
Line B → `out2 = fib(n-2)`
Line C → `return out1 + out2`

empty!

Function Stack

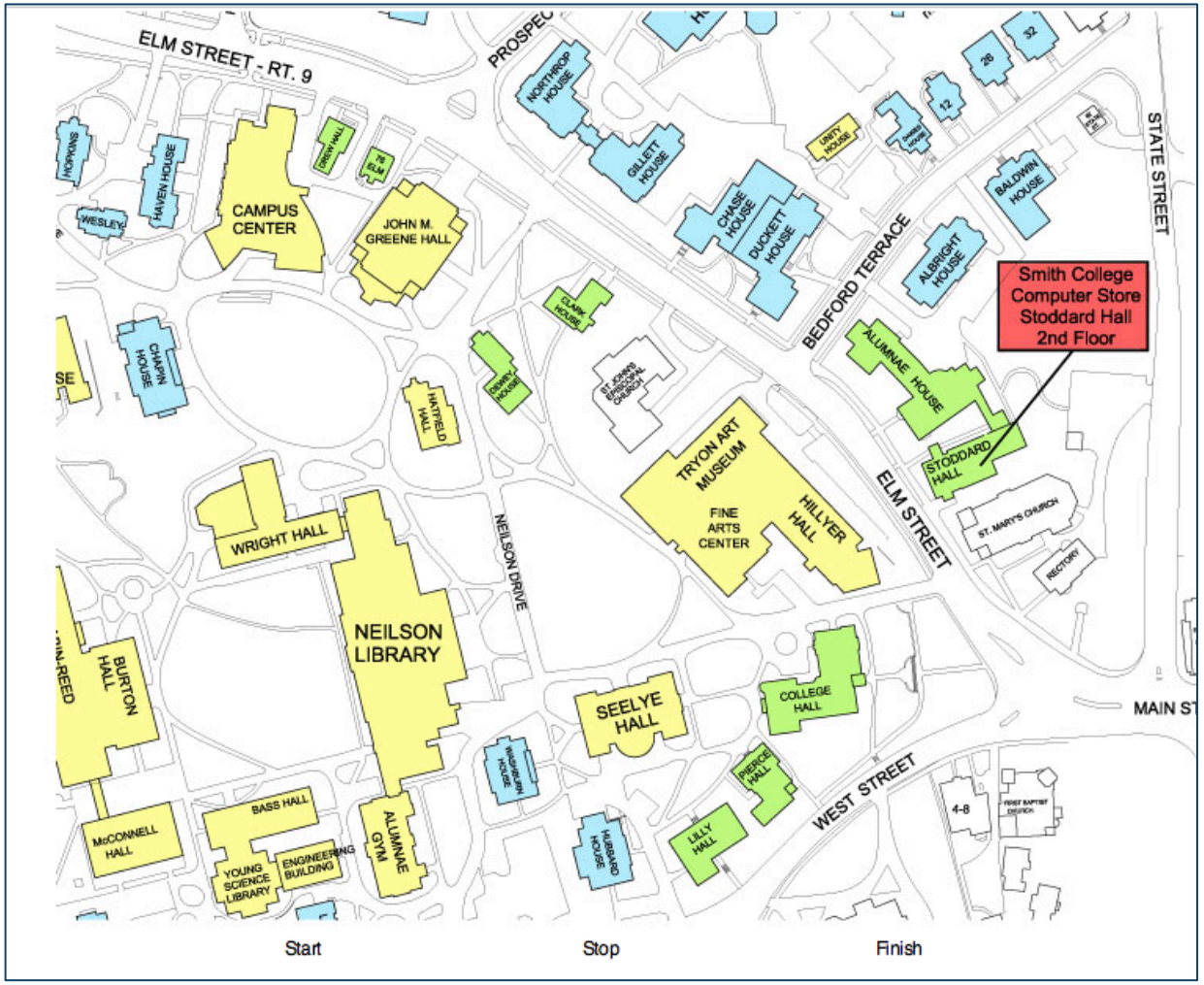
Fibonacci Tree with Function Calls



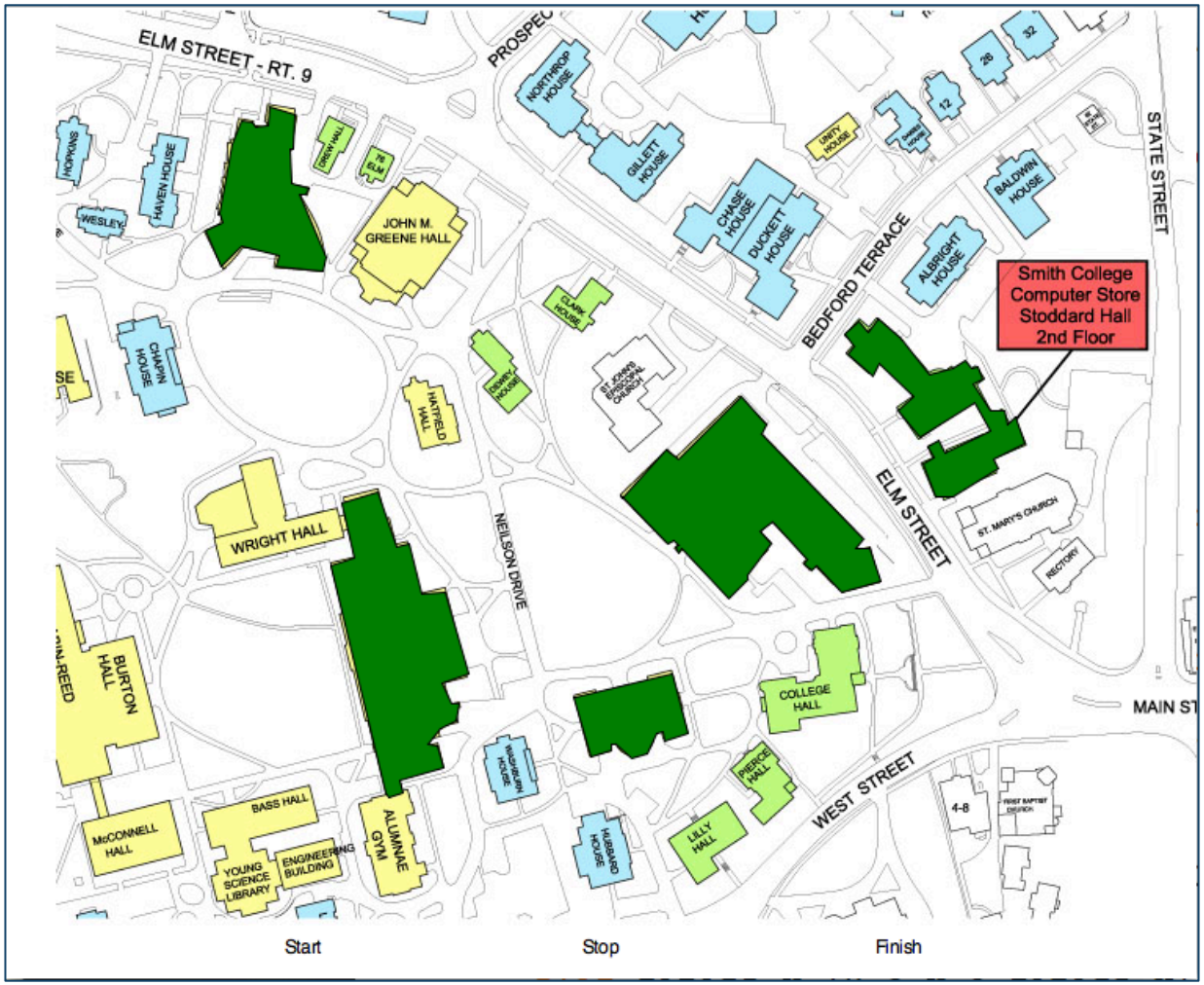

```
def fib(n):
    if n == 0 or n == 1:
        return 1
    else:
        out1 = fib(n-1)      # Line A
        out2 = fib(n-2)      # Line B
        return out1 + out2   # Line C
```

Homework 8 and Lab 8

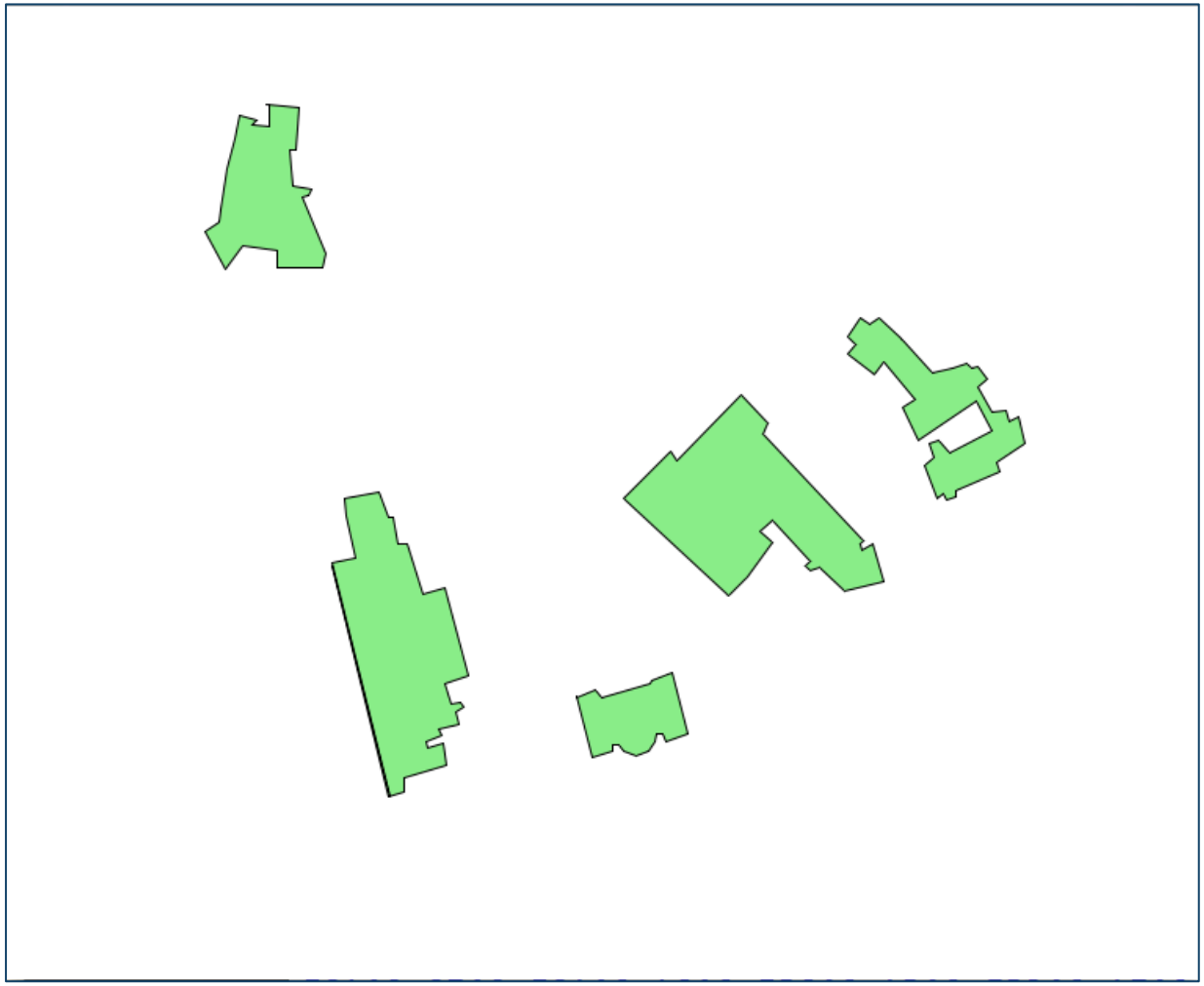
Homework 8: Maps



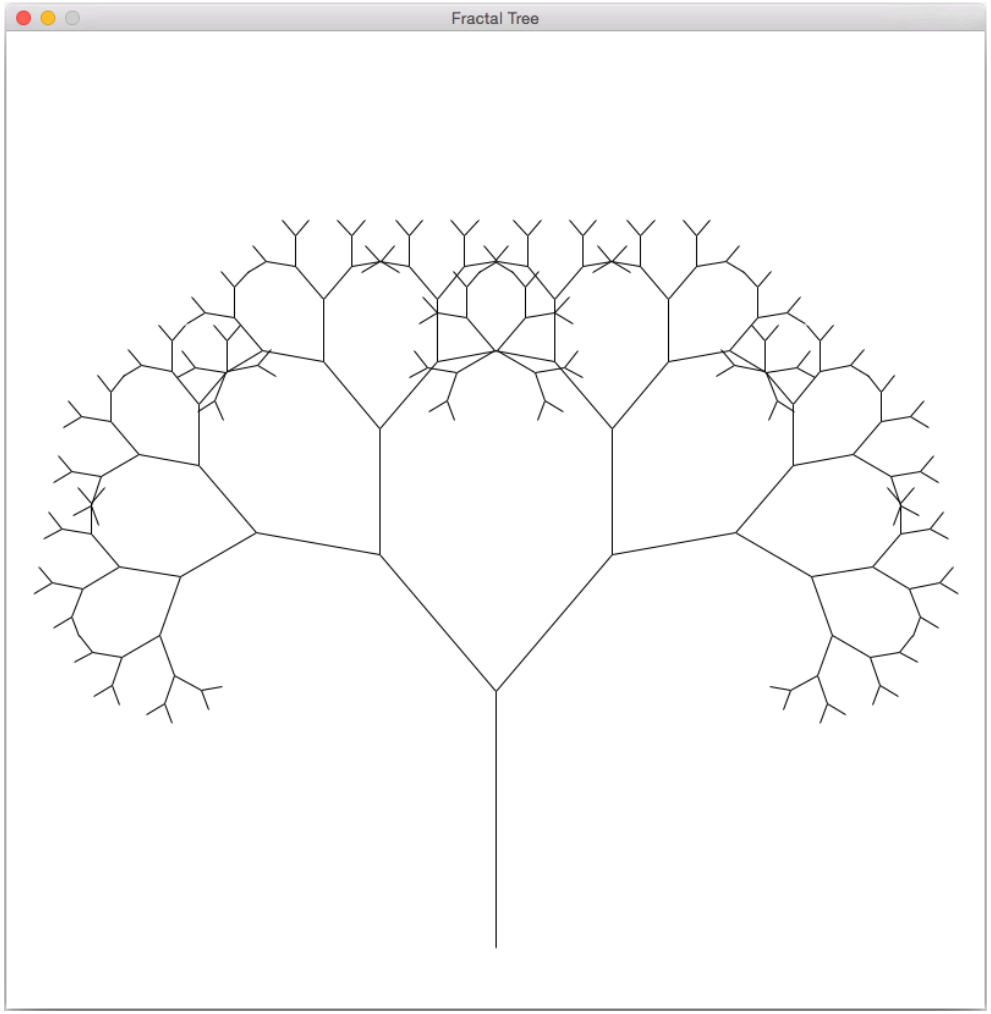
Homework 8: Maps



Homework 8: Maps



Lab 8: Fractal Trees using Recursion



Lab 8: Fractal Trees using Recursion