

# Recursion

---

## Announcements

## Self-Reference

## Returning a Function Using Its Own Name

```

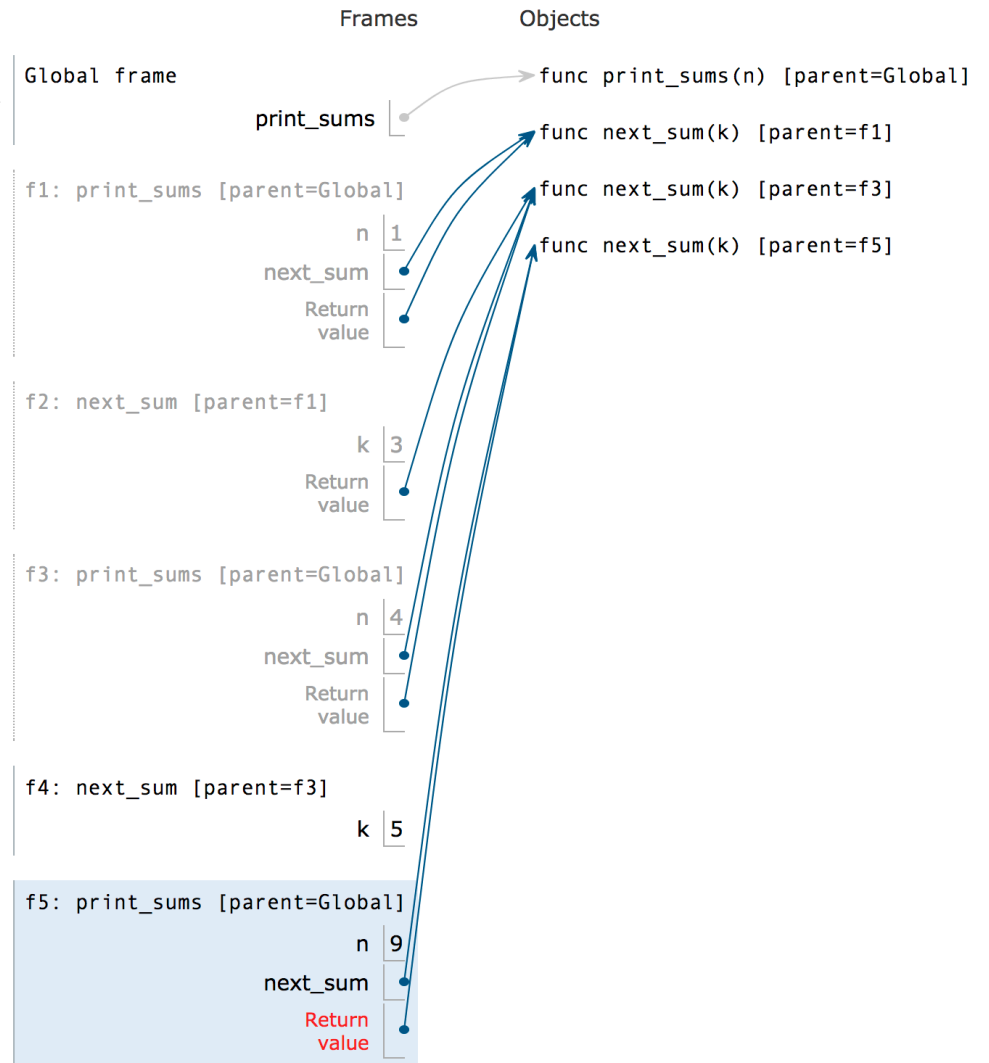
1  def print_sums(n):
2      print(n)
3      def next_sum(k):
4          return print_sums(n+k)
5      return next_sum
6
7  print_sums(1)(3)(5)

```

`print_sums(1)(3)(5)` prints:

$$\begin{array}{l} 1 \\ 4 \quad (1 + 3) \\ 9 \quad (1 + 3 + 5) \end{array}$$

`print_sums(3)(4)(5)(6)` prints:

$$\begin{array}{l} 3 \\ 7 \quad (3 + 4) \\ 12 \quad (3 + 4 + 5) \\ 18 \quad (3 + 4 + 5 + 6) \end{array}$$


## Example: Mutual Recursion

---

Mutual recursive functions call one another.  
What is the last line that will be printed out?

```
1 def add_next(n):  
2     print(n)  
→ 3     return lambda f: subtract_next(n + f)  
4  
5 def subtract_next(n):  
6     print(n)  
7     return lambda f: add_next(n - f)  
8  
→ 9 add_next(2500)(500)(1000)(24)
```

# Recursive Functions

( Demo )

## From Iteration to Recursion

## Example: Boxes and Pyramids

Here's a simple function `boxes_iter(k)` that iteratively prints out `k` boxes in a line.  
(a pair of square brackets, like: `[]`)  
Can we make it recursive?

```
def boxes_iter(k):  
    """ prints out k boxes.  
    >>> boxes_iter(4)  
    [] [] [] []  
    """  
    while k > 0:  
        print("[]", end="")  
        k -= 1  
    return
```

```
def boxes_r(k):  
    """ prints out k boxes.  
    >>> boxes_r(4)  
    [] [] [] []  
    """  
    if k == 0:  
        return  
    else:  
        print("[]", end="")  
        boxes_r(k-1)  
    return
```

base case

recursive case



Here's a function *pyramid* that prints out a *pyramid* of *k* height using one of the boxes functions we wrote on the last slide.

How would we flip the pyramid upside-down?

[illegible]

## Discussion Question: Play Twenty-One

Rewrite play as a recursive function without a while statement.

- Do you need to define a new inner function? Why or why not? If so, what are its arguments?
- What is the base case and what is returned for the base case?

```
def play(strategy0, strategy1, goal=21):  
    """Play twenty-one and return the winner.
```

```
>>> play(two_strat, two_strat)
```

```
1
```

```
"""
```

```
n = 0
```

```
who = 0 # Player 0 goes first
```

```
while n < goal:
```

```
    if who == 0:
```

```
        n = n + strategy0(n)
```

```
        who = 1
```

```
    elif who == 1:
```

```
        n = n + strategy1(n)
```

```
        who = 0
```

```
return who
```

```
def play(strategy0, strategy1, goal=21):  
    """Play twenty-one and return the winner.
```

```
>>> play(two_strat, two_strat)
```

```
1
```

```
"""
```

```
def f(n, who):
```

```
    if n >= goal:
```

```
        return who
```

```
    if who == 0:
```

```
        n = n + strategy0(n)
```

```
        who = 1
```

```
    elif who == 1:
```

```
        n = n + strategy1(n)
```

```
        who = 0
```

```
    return f(n, who)
```

```
return f(0, 0)
```