FUNCTIONS: RECURSION

Overview:

- compare recursive and non-recursive functions

Recursive Functions

- functions call other functions
- recursive - can call iself
- example: factorial function

$$n! = 1 \cdot 2 \cdots (n-1) \cdot n$$

- recursive computation:

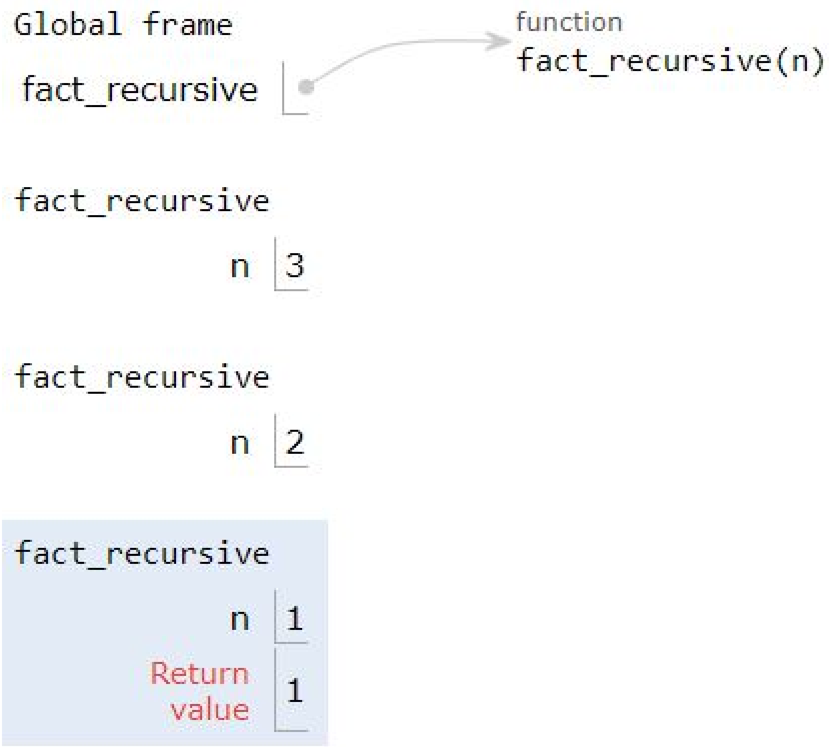
$$\begin{aligned} 0! &= 1 \quad \text{(base case)} \\ n! &= 1 \cdot 2 \cdots (n-1) \cdot n \\ &= (n-1)! \cdot n \end{aligned}$$

- 1!=1, 2!=2, 3!=6, 4!=24

Recursive Factorial

```
def fact_recursive(n):
    if n <= 1:
        return 1
    else:
        return n * fact_recursive(n-1)

y = fact_recursive(3)
```



Global frame

fact_recursive

function
fact_recursive(n)

fact_recursive
n 3

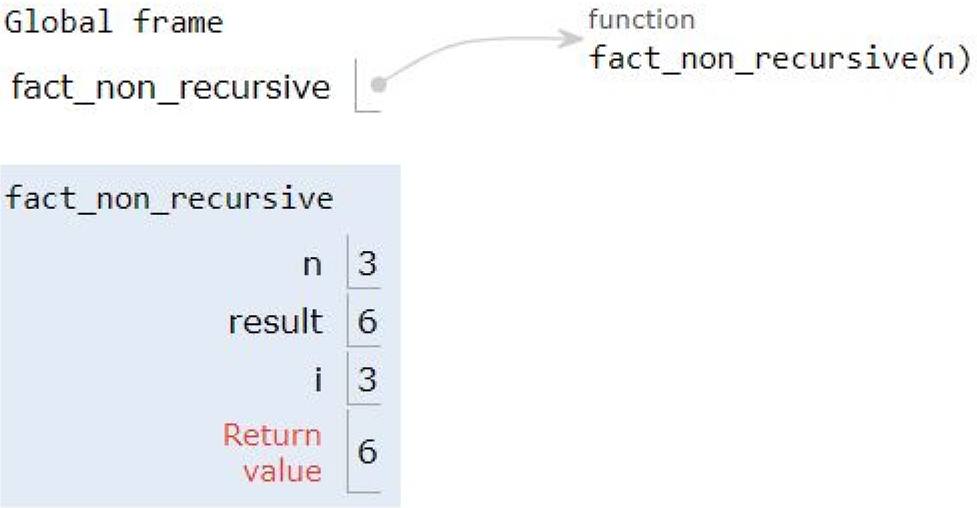
fact_recursive
n 2

fact_recursive
n 1
Return value 1

Non-Recursive Factorial

```
def fact_non_recursive(n):
    result = 1
    for i in range(1, n + 1):
        result = result * i
    return result

z = fact_non_recursive(3)
```



Global frame

fact_non_recursive

function
fact_non_recursive(n)

fact_non_recursive

n 3
result 6
i 3
Return value 6

Recursion Trade-offs

```
def fact_recursive(n):
    if n <= 1:
        return 1
    else:
        return n * fact_recursive(n-1)

def fact_non_recursive(n):
    result = 1
    for i in range(1, n + 1):
        result = result * i
    return result
```

- recursive: more execution time
- non-recursive: larger code

Example: Fibonacci Numbers

- each number is sum of previous two
- 1, 1, 2, 3, 5, 8, …
- recursive

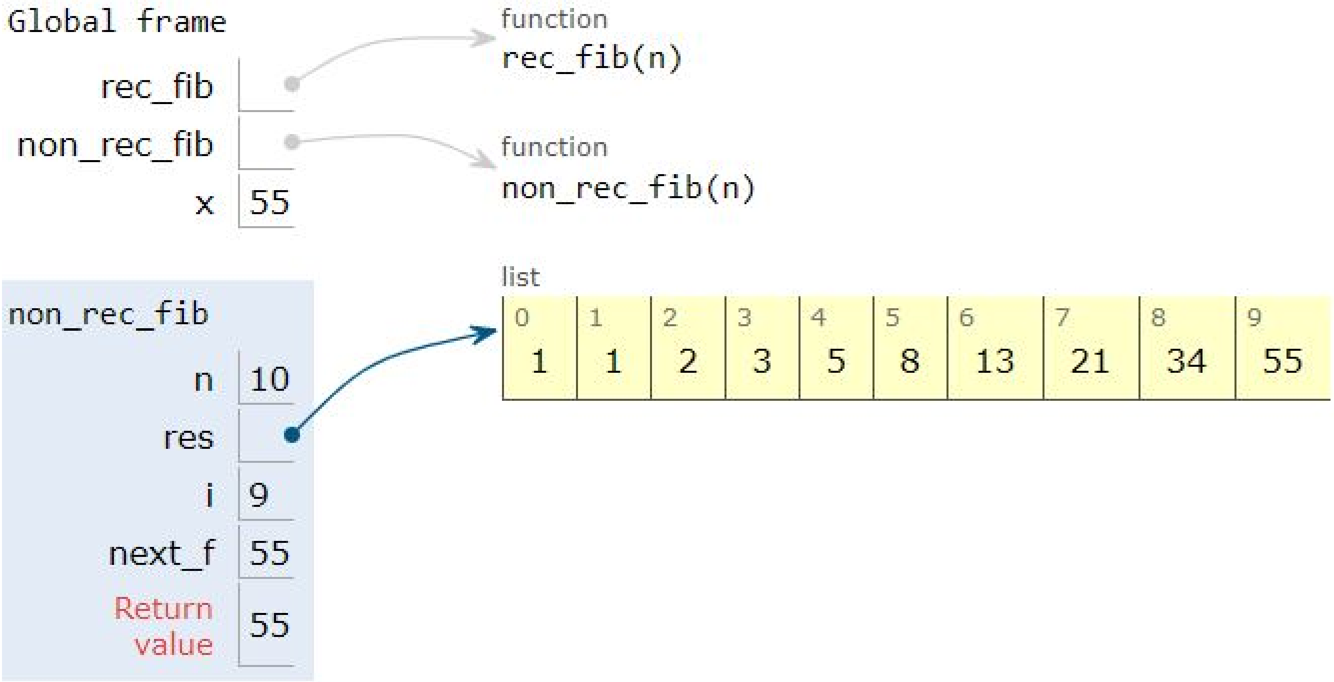
  $f(n) = f(n-1) + f(n-2)$
- non-recursive: need to store results (trade time for space)
- dynamic programming

Example: Fibonacci Numbers

```
# recursive computation
def rec_fib(n):
    if n <= 2:
        return 1
    else:
        return rec_fib(n-1) + rec_fib(n-2)

# non-recursive computation
# store intermediate results in a list
# example of dynamic programming
def non_rec_fib(n):
    if n <= 2:
        return 1
    else:
        res = [1, 1]
        for i in range(2, n):
            next_f = res[i-2] + res[i-1]
            res.append(next_f)
        return res[-1]
```

Example: Fibonacci Numbers



Global frame

rec_fib	
non_rec_fib	
x	55

function
rec_fib(n)

function
non_rec_fib(n)

non_rec_fib

n	10
res	
i	9
next_f	55
Return value	55

list

0	1	2	3	4	5	6	7	8	9
1	1	2	3	5	8	13	21	34	55

- dynamic programming

Exercise(s):

- write both recursive and non-recursive (iterative) versions of function to compute the sum of the first n terms in arithmetic progression $A(a, d)$:

  $a, a+d, a+2d, \ldots, a+(n-1)d$

Exercise(s):

- write both recursive and non-recursive (iterative) versions of function to compute the sum of the first n terms in geometric progression $G(b, q)$:

$$b, bq, bq^2, \ldots, bq^{n-1}$$