

Recursion

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

Class outline:

- Recursive functions
- Recursion in environment diagrams
- Mutual recursion
- Recursion vs iteration

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

Recursive functions

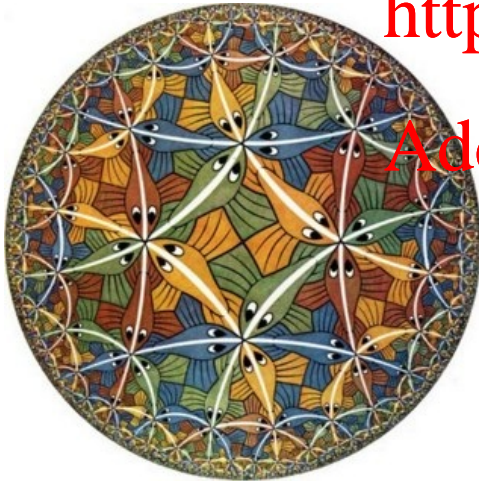
A function is **recursive** if the body of that function calls itself, either directly or indirectly.

Recursive functions often operate on increasingly smaller instances of a problem.

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder



Circle Limit, by M.C. Escher

Summing digits

$$2 + 0 + 2 + 1 = 5$$

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

Summing digits

$$2 + 0 + 2 + 1 = 5$$

Fun fact: The sum of the digits in a multiple of 9 is also divisible by 9.

$$9 * 82 = 738$$

$$7 + 3 + 8 = 18$$

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

The problems within the problem

The sum of the digits of 6 is simply 6.

Generally: the sum of any one-digit non-negative number is that number.

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

The problems within the problem

The sum of the digits of 6 is simply 6.

Generally: the sum of any one-digit non-negative number is that number.

The sum of the digits of 2021 is:

2 0 2 1

Sum of these digits + This digit

<https://powcoder.com>

Add WeChat powcoder

Generally: the sum of a number is the sum of the first digits (`number // 10`), plus the last digit (`number % 10`).

Summing digits without a loop

```
def sum_digits(n):  
    """Return the sum of the digits of positive integer n.  
>>> sum_digits(6)  
6  
>>> sum_digits(2021)  
5  
"""
```

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

Summing digits without a loop

```
def sum_digits(n):  
    """Return the sum of the digits of positive integer n.  
    >>> sum_digits(6)  
    6  
    >>> sum_digits(2021)  
    5  
    """  
    if n < 10:  
        return n  
    else:  
        all_but_last = n // 10  
        last = n % 10  
        return sum_digits(all_but_last) + last
```

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

Anatomy of a recursive function

- **Base case:** Evaluated without a recursive call (the smallest subproblem).
- **Recursive case:** Evaluated with a recursive call (breaking down the problem further)
- **Conditional statement** to decide if it's a base case

```
def sum_digits(n):  
    if n < 10:  
        return n  
    else:  
        all_but_last = n // 10  
        last = n % 10  
        return sum_digits(all_but_last) + last
```

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

Anatomy of a recursive function

- **Base case:** Evaluated without a recursive call (the smallest subproblem).
- **Recursive case:** Evaluated with a recursive call (breaking down the problem further).
- **Conditional statement** to decide if it's a base case

```
def sum_digits(n):  
    if n < 10: # BASE CASE  
        return n  
    else:  
        all_but_last = n // 10  
        last = n % 10  
        return sum_digits(all_but_last) + last
```

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

Anatomy of a recursive function

- **Base case:** Evaluated without a recursive call (the smallest subproblem).
- **Recursive case:** Evaluated with a recursive call (breaking down the problem further)
- **Conditional statement** to decide if it's a base case

<https://powcoder.com>

```
def sum_digits(n):  
    if n < 10: # BASE CASE  
        return n  
    else:      # RECURSIVE CASE  
        all_but_last = n // 10  
        last = n % 10  
        return sum_digits(all_but_last) + last
```

Add WeChat powcoder

Visualizing recursion

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

Recursive factorial

The factorial of a number is defined as:

$$n! = \begin{cases} 1 & \text{if } n = 0 \\ n \cdot (n-1)! & \text{otherwise} \end{cases}$$

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

```
def fact(n):  
    """  
    >>> fact(0)  
    1  
    >>> fact(4)  
    24  
    """
```

Recursive factorial

The factorial of a number is defined as:

$$n! = \begin{cases} 1 & \text{if } n = 0 \\ n \cdot (n-1)! & \text{otherwise} \end{cases}$$

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

```
def fact(n):  
    """  
    >>> fact(0)  
    1  
    >>> fact(4)  
    24  
    """  
    if n == 0:  
        return 1  
    else:  
        return n * fact(n-1)
```

Recursive call visualization

```
def fact(n):  
    if n == 0:  
        return 1  
    else:  
        return n * fact(n-1)  
  
fact(3)
```

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder


```
fact(3)
```

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

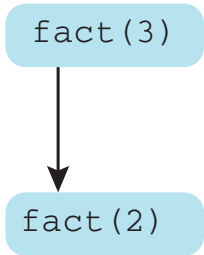
Recursive call visualization

```
def fact(n):  
    if n == 0:  
        return 1  
    else:  
        return n * fact(n-1)  
  
fact(3)
```

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder



Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

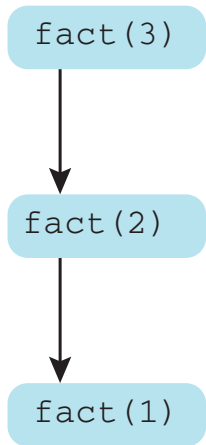
Recursive call visualization

```
def fact(n):  
    if n == 0:  
        return 1  
    else:  
        return n * fact(n-1)  
  
fact(3)
```

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder



Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

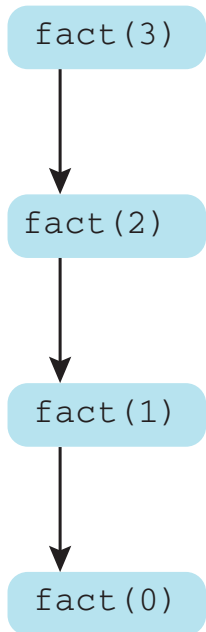
Recursive call visualization

```
def fact(n):  
    if n == 0:  
        return 1  
    else:  
        return n * fact(n-1)  
  
fact(3)
```

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder



Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

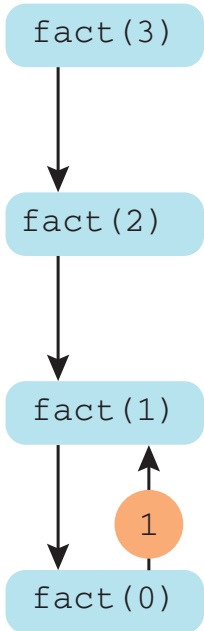
Recursive call visualization

```
def fact(n):  
    if n == 0:  
        return 1  
    else:  
        return n * fact(n-1)  
  
fact(3)
```

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder



Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

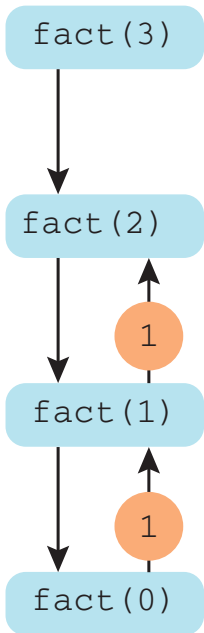
Recursive call visualization

```
def fact(n):  
    if n == 0:  
        return 1  
    else:  
        return n * fact(n-1)  
  
fact(3)
```

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder



Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

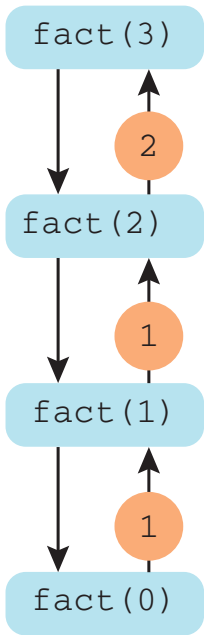
Recursive call visualization

```
def fact(n):  
    if n == 0:  
        return 1  
    else:  
        return n * fact(n-1)  
  
fact(3)
```

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder



Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

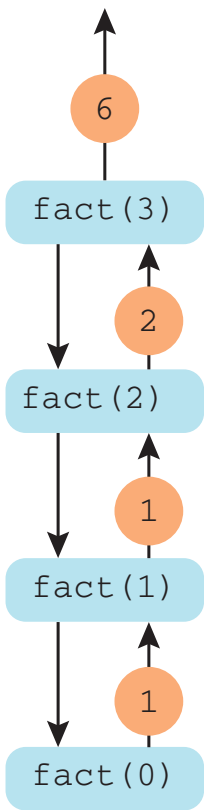
Recursive call visualization

```
def fact(n):  
    if n == 0:  
        return 1  
    else:  
        return n * fact(n-1)  
  
fact(3)
```

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder



Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

Recursion in environment diagrams

```
def fact(n):  
    if n == 0:  
        return 1  
    else:  
        return n * fact(n-1)
```

fact(3)

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

- The same function `fact` is called multiple times
- Different frames keep track of the different arguments in each call
- What `n` evaluates to depends upon the current environment
- Each call to `fact` solves a simpler problem than the last: smaller

`n`

Global frame

fact → func fact[parent=Global]

f1:

Return

value |

f2:

Return
value |

f3:

Return
value |

f4:

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

Recursion in environment diagrams

```
def fact(n):  
    if n == 0:  
        return 1  
    else:  
        return n * fact(n-1)
```

fact(3)

Assignment Project Exam Help

<https://powcoder.com>

- The same function `fact` is called multiple times
- Different frames keep track of the different arguments in each call
- What `n` evaluates to depends upon the current environment
- Each call to `fact` solves a simpler problem than the last: smaller

`n`

Global frame

fact → func fact[parent=Global]

f1: fact[parent=Global]

n | 3

Return | 6

value | _____

f2: fact[parent=Global]

n | 2

Return | 2
value | _____

f3: fact[parent=Global]

n | 1

Return | 1
value | _____

f4: fact[parent=Global]

n | 0

Return | 1
value | _____

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

Verifying recursive functions

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

Falling dominoes

If a million dominoes are equally spaced out and we tip the first one, will they all fall?

1. Verify that one domino will fall, if tipped
2. Assume that any given domino falling over will tip the next one over
3. Verify that tipping the first domino tips over the next one

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

Falling dominoes

If a million dominoes are equally spaced out and we tip the first one, will they all fall?

1. Verify that one domino will fall, if tipped
2. Assume that any given domino falling over will tip the next one over
3. Verify that tipping the first domino tips over the next one

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder



The recursive leap of faith

```
def fact(n):  
    """Returns the factorial of N."""  
    if n == 0:  
        return 1  
    else:  
        return n * fact(n-1)
```

Is `fact` implemented correctly?

1. Verify the base case
2. Treat `fact` as a functional abstraction!
3. Assume that `fact(n-1)` is correct (\leftarrow the leap!)
4. Verify that `fact(n)` is correct

The recursive elf's promise

Imagine we're trying to compute $5!$

We ask ourselves, "If I somehow knew how to compute $4!$, could I compute $5!$?"

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

Credit: FuschiaKnight, r/compsci

The recursive elf's promise

Imagine we're trying to compute $5!$

We ask ourselves, "If I somehow knew how to compute $4!$, could I compute $5!$?"

Yep, $5! = 5 * 4!$

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

Credit: FuschiaKnight, r/compsci

The recursive elf's promise

Imagine we're trying to compute $5!$

We ask ourselves, "If I somehow knew how to compute $4!$, could I compute $5!$?"

Yep, $5! = 5 * 4!$

♀ The `fact()` function promises, "hey friend, tell you what, while you're working hard on $5!$, I'll compute $4!$ for you, and you can finish it off!"

Credit: FuschiaKnight, r/compsci

Mutual recursion

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

The Luhn algorithm

Used to verify that a credit card numbers is valid.

1. From the rightmost digit, which is the check digit, moving left, double the value of every second digit; if product of this doubling operation is greater than 9 (e.g., $7 * 2 = 14$), then sum the digits of that product (e.g., $14: 1 + 4 = 5$)
2. Take the sum of all the digits

Original	1	3	8	7	4	3
Processed						

The Luhn algorithm

Used to verify that a credit card numbers is valid.

1. From the rightmost digit, which is the check digit, moving left, double the value of every second digit; if product of this doubling operation is greater than 9 (e.g., $7 * 2 = 14$), then sum the digits of that product (e.g., $14: 1 + 4 = 5$)
2. Take the sum of all the digits

Original	1	3	8	7	4	3
Processed						3

The Luhn algorithm

Used to verify that a credit card numbers is valid.

1. From the rightmost digit, which is the check digit, moving left, double the value of every second digit; if product of this doubling operation is greater than 9 (e.g., $7 * 2 = 14$), then sum the digits of that product (e.g., $14: 1 + 4 = 5$)
2. Take the sum of all the digits

Original	1	3	8	7	4	3
Processed					8	3

The Luhn algorithm

Used to verify that a credit card numbers is valid.

1. From the rightmost digit, which is the check digit, moving left, double the value of every second digit; if product of this doubling operation is greater than 9 (e.g., $7 * 2 = 14$), then sum the digits of that product (e.g., $14: 1 + 4 = 5$)
2. Take the sum of all the digits

Original	1	3	8	7	4	3
Processed				7	8	3

The Luhn algorithm

Used to verify that a credit card numbers is valid.

1. From the rightmost digit, which is the check digit, moving left, double the value of every second digit; if product of this doubling operation is greater than 9 (e.g., $7 * 2 = 14$), then sum the digits of that product (e.g., $14: 1 + 4 = 5$)
2. Take the sum of all the digits

Original	1	3	8	7	4	3
Processed			1+6=7	7	8	3

The Luhn algorithm

Used to verify that a credit card numbers is valid.

1. From the rightmost digit, which is the check digit, moving left, double the value of every second digit; if product of this doubling operation is greater than 9 (e.g., $7 * 2 = 14$), then sum the digits of that product (e.g., $14: 1 + 4 = 5$)
2. Take the sum of all the digits

Original	1	3	8	7	4	3
Processed		3	1+6=7	7	8	3

The Luhn algorithm

Used to verify that a credit card numbers is valid.

1. From the rightmost digit, which is the check digit, moving left, double the value of every second digit; if product of this doubling operation is greater than 9 (e.g., $7 * 2 = 14$), then sum the digits of that product (e.g., $14: 1 + 4 = 5$)
2. Take the sum of all the digits

Original	1	3	8	7	4	3
Processed	2	3	1+6=7	7	8	3

The Luhn algorithm

Used to verify that a credit card numbers is valid.

1. From the rightmost digit, which is the check digit, moving left, double the value of every second digit; if product of this doubling operation is greater than 9 (e.g., $7 * 2 = 14$), then sum the digits of that product (e.g., $14: 1 + 4 = 5$)
2. Take the sum of all the digits

Original	1	3	8	7	4	3	
Processed	2	3	1+6=7	7	8	3	= 30

The Luhn sum of a valid credit card number is a multiple of 10

Calculating the Luhn sum

Let's start with...

```
def sum_digits(n):  
    if n < 10:  
        return n  
    else:  
        last = n % 10  
        all_but_last = n // 10  
        return last + sum_digits(all_but_last)  
  
def luhn_sum(n):  
    """Returns the Luhn sum for the positive number N.  
>>> luhn_sum(2)  
2  
>>> luhn_sum(32)  
8  
>>> luhn_sum(5105105105105100)  
20  
"""
```

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

Luhn sum with mutual recursion

```
def luhn_sum(n):  
    if n < 10:  
        return n  
    else:  
        last = n % 10  
        all_but_last = n // 10  
        return last + luhn_sum_double(all_but_last)
```

```
def luhn_sum_double(n):  
    last = n % 10  
    all_but_last = n // 10  
    luhn_digit = sum_digits(last * 2)  
    if n < 10:  
        return luhn_digit  
    else:  
        return luhn_digit + luhn_sum(all_but_last)
```

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

Recursion and Iteration

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

Recursion vs. iteration

Using recursion:

```
def fact(n):  
    if n == 0:  
        return 1  
    else:  
        return n * fact(n-1)
```

Using iteration:

```
def fact(n):  
    total = 1  
    k = 1  
    while k <= n:  
        total *= k  
        k += 1  
    return total
```

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

Math:

$$n! = \begin{cases} 1 & \text{if } n = 0 \\ n \cdot (n-1)! & \text{otherwise} \end{cases}$$

$$n! = \prod_{k=1}^n k$$

Names: `fact`, `n`

`fact`, `n`, `total`, `k`

Converting recursion to iteration

Can be tricky: Iteration is a special case of recursion.

Figure out what state must be maintained by the iterative function.

```
def sum_digits(n):  
    if n < 10:  
        return n  
    else:  
        all_but_last = n // 10  
        last = n % 10  
        return sum_digits(all_but_last) + last
```

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

Converting recursion to iteration

Can be tricky: Iteration is a special case of recursion.

Figure out what state must be maintained by the iterative function.

```
def sum_digits(n):  
    if n < 10:  
        return n  
    else:  
        all_but_last = n // 10  
        last = n % 10  
        return sum_digits(all_but_last) + last
```

Assignment Project Exam Help

<https://powcoder.com>

```
def sum_digits(n):  
    digit_sum = 0  
    while n >= 10:  
        last = n % 10  
        n = n // 10  
        digit_sum += last  
    return digit_sum
```

Add WeChat powcoder

Converting iteration to recursion

More formulaic: Iteration is a special case of recursion.

The state of an iteration can be passed as arguments.

```
def sum_digits(n):
```

```
    digit_sum = 0
```

```
    while n >= 10:
```

```
        last = n % 10
```

```
        n = n // 10
```

```
        digit_sum += last
```

```
    return digit_sum
```

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

Converting iteration to recursion

More formulaic: Iteration is a special case of recursion.

The state of an iteration can be passed as arguments.

```
def sum_digits(n):
```

```
    digit_sum = 0
```

```
    while n >= 10:
```

```
        last = n % 10
```

```
        n = n // 10
```

```
        digit_sum += last
```

```
    return digit_sum
```

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

```
def sum_digits(n, digit_sum):
```

```
    if n == 0:
```

```
        return digit_sum
```

```
    else:
```

```
        last = n % 10
```

```
        all_but_last = n // 10
```

```
        return sum_digits(all_but_last, digit_sum + last)
```