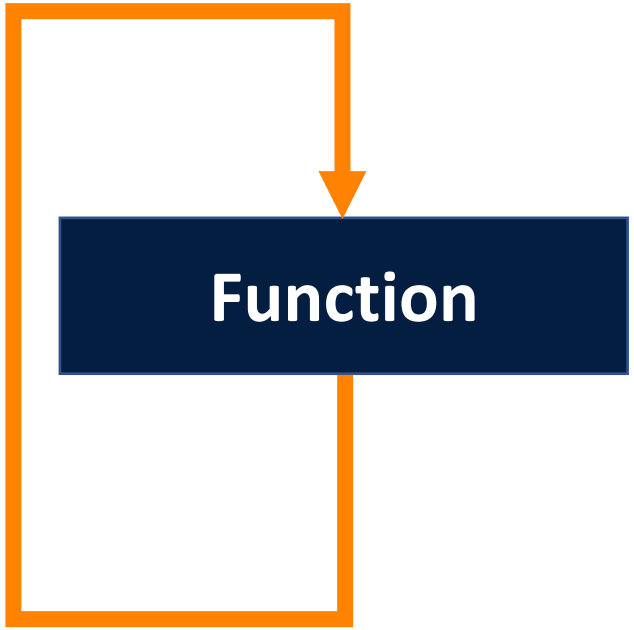
THE UNIVERSITY OF TEXAS AT EL PASO

Recursion

Sayed Reza
PhD Candidate
Computer Science
Website: https://smreza.com/

What is Recursion?

- Recursion is fundamental technique in Computer Science and can be applied to tasks that are **repetitive**.
- Recursion is a function that **calls itself**.
- Recursion is generally used when a problem can be **divided into smaller part**.



Which tasks we can apply recursion?

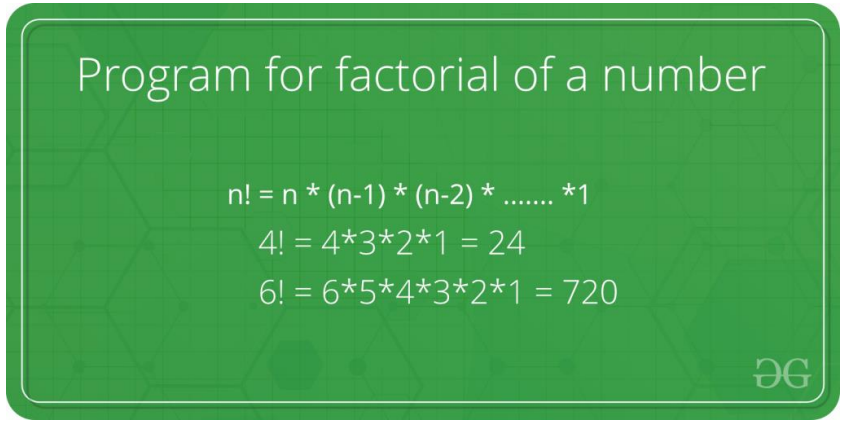
1. Repetitive Tasks

2. Divided into smaller part

Any Ideas from students?

0:30

Timer



Mathematical Factorial

Robot Put Away Plates

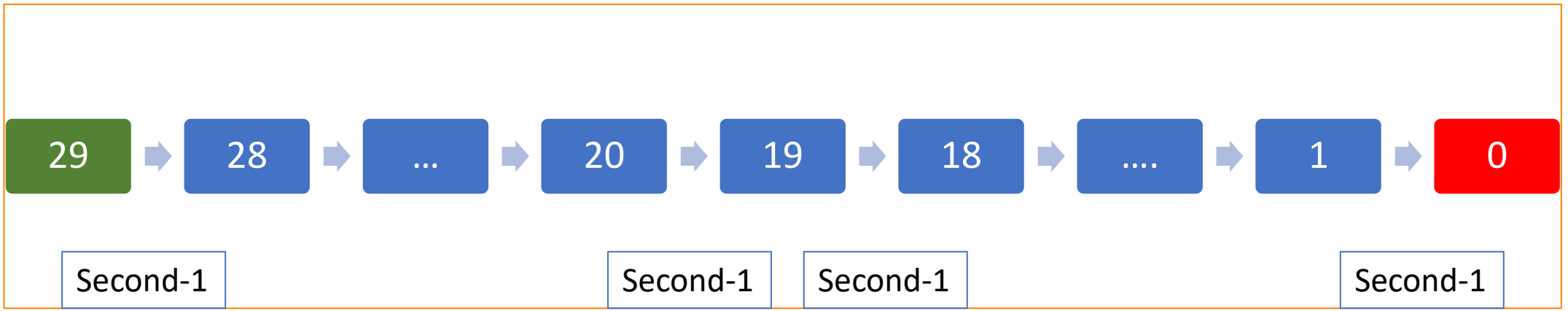
Timer – How can we divided into subtask?

0:30

Timer

Pseudocode:

1. Print (second -1) after each second passes
2. Stop the timer when it reaches 0



Recursive Solutions

When creating a recursive solution, there are a few things we want to keep in mind:

1. We need to break the problem into **smaller pieces** of itself
2. We need to define a **“base case”** to stop at
3. The smaller problems we **break down** into need to eventually reach the base case

How Recursion Function is Developed

```
def timerRecursive(seconds):
```

Base Case

Recursion ends and returns a value/exit

Recursive Case

Function call itself

Sample Recursion code Run: https://onlinegdb.com/LV76WYXxc

Recursion Terminology

```
def func(n):
  if n == 0:
      return 0

  print(n)
  return func(n - 1)
```

Base case

Recursive case

Task: Find func(5)

func(5)	func(4)	func(3)	func(2)	func(1)	func(0)
Print(5) return func (4)	Print(4) return func (3)	Print(3) return func (2)	Print(2) return func (1)	Print(1) return func (0)	return 0

Recursion Issue

```
def func(n):
      if n == 0:
            return 0

      print (n)
      return func (n + 1)

Find func(5)
```

We have a base case and a recursive case.

What's wrong?

Recursion Issue

```
def func(n):
      if n == 0:
            return 0

      print (n)
      return func (n + 1)

Find func(5)
```

func(5)	func(6)	func(7)	func(8)	. . .	func(1000)
Print(5) return func (6)	Print(6) return func (7)	Print(7) return func (8)	Print(8) return func (9)		Print(1000) return func (1001)

Recursion Issue

```
def func(n):
      if n == 0:
            return 0

      print (n)
      return func (n + 1)

Find func(5)
```

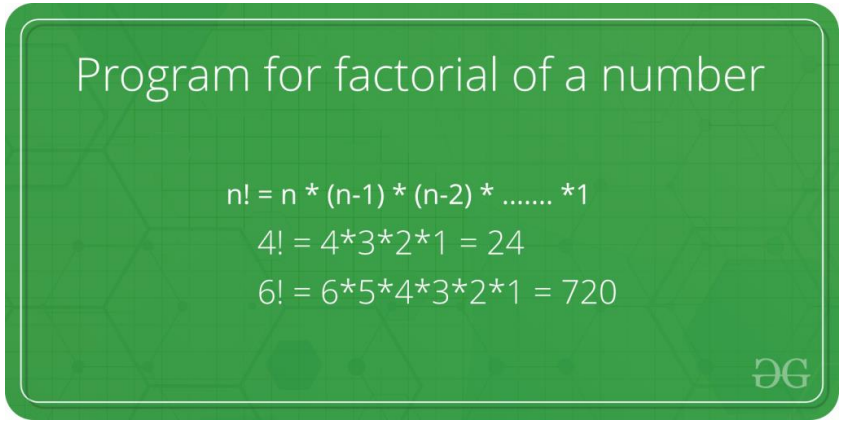
```
def func(n):
      if n == 0:
            return 0

      print (n)
      return func (n - 1)

Find func(5)
```

Important when you code

1. Make Sure **base case** is called at some point
2. Try to avoid forever loop
3. **Recursive cases** will end up with **base case** at some point

Which tasks we can apply recursion?

Timer



Mathematical Factorial

Robot Put Away Plates

Mathematical Factorial

- Does anyone know the **value of 9!** ?
- 362,880
- Does anyone know the **value of 10!** ?
- How did you know?

Mathematical Factorial

$9! = 9 \times 8 \times 7 \times 6 \times 5 \times 4 \times 3 \times 2 \times 1$

$10! = 10 \times 9 \times 8 \times 7 \times 6 \times 5 \times 4 \times 3 \times 2 \times 1$

$10! = 10 \times 9!$

$n! = n \times (n - 1)!$

That's a recursive definition!

Mathematical Factorial

Step 1:

$$n! = n \times (n - 1)!$$

Step 2:

```
def fact(n):
    return n * fact(n - 1)
```

Step 3:

```
fact(3)
3 * fact(2)
3 * 2 * fact(1)
3 * 2 * 1 * fact(0)
...
```

Mathematical Factorial

What did we do wrong?

Missing base case for recursion

```
def fact(n):
      return n*fact(n-1)

Find fact(10)
```

```
def fact(n):
      if n == 1:
            return 1

      return n*fact(n-1)

Find fact(10)
```

Which tasks we can apply recursion?

0:30

Timer

Program for factorial of a number

n! = n * (n-1) * (n-2) * *1

4! = 4*3*2*1 = 24

6! = 6*5*4*3*2*1 = 720

Mathematical Factorial

Robot Put Away Plates

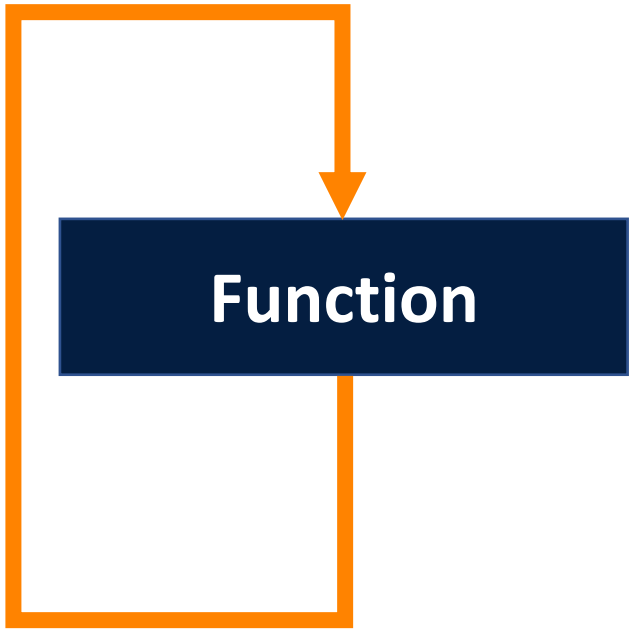
Robot Put Away Recursion Code

```
def putAwayPlate(numberPlates):

      if numberPlates == 0:
            print ("Finished")
            return 0;

      movePlate()
      putAwayPlate(numberPlates - 1)
```

Robot Put Away Plates

Summary

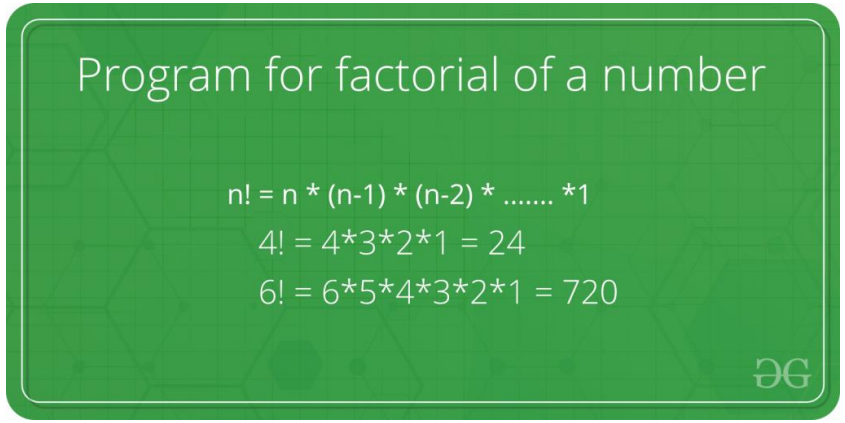
- Recursion is fundamental technique in Computer Science and can be applied to tasks that are **repetitive**.
- Recursion is a function that **calls itself**.
- Recursion has 2 cases
  - Recursive cases
  - Stop/base case
- Make Sure **base case** is called at some point
- Try to avoid forever loop



Any Questions

0:30

Timer



Mathematical Factorial

Robot Put Away Plates