

"Python Recursion & Me" Adventure



Recursion

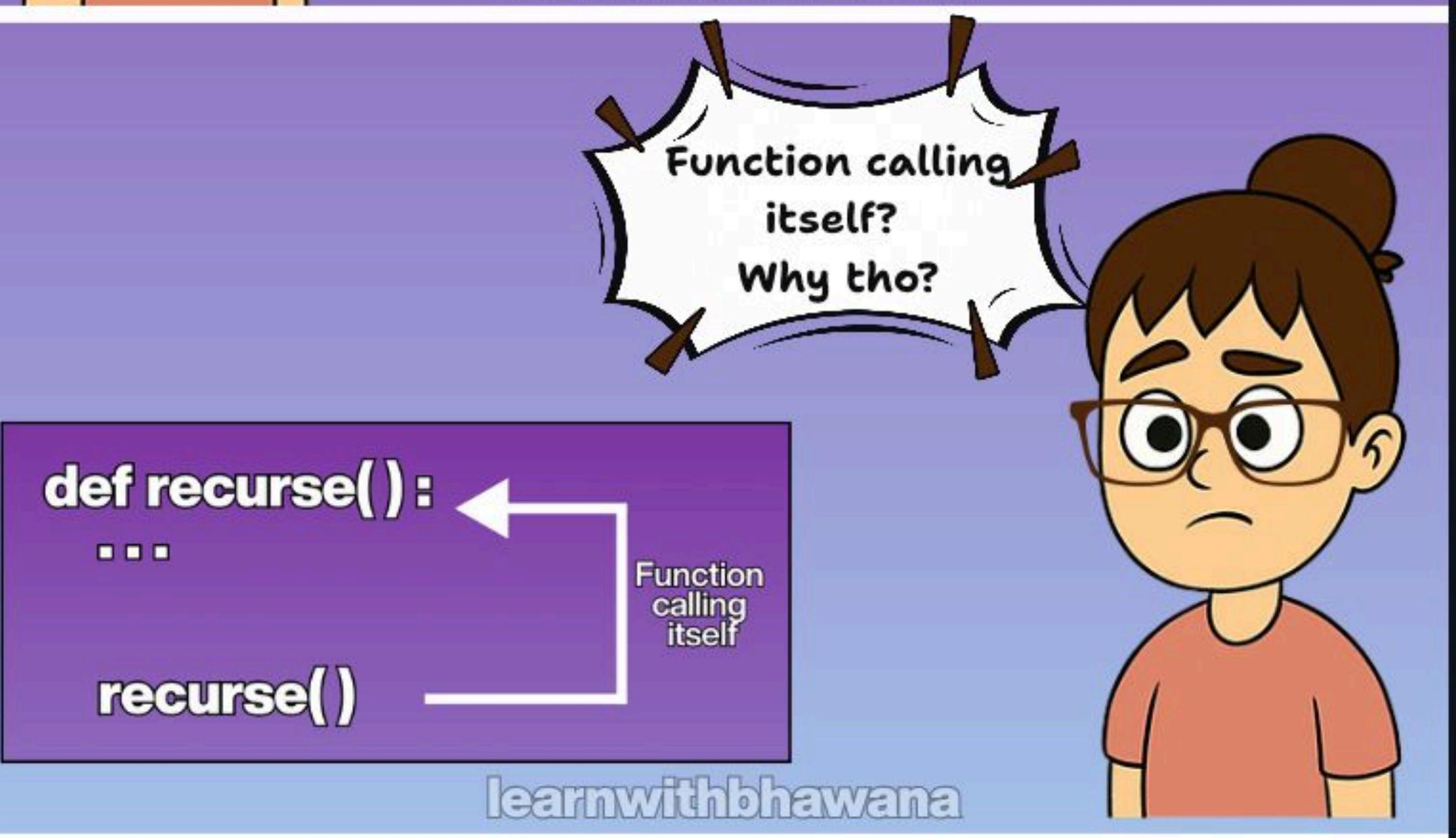


LearnwithBhawana

learnwithbhawana



learnwithbhawana



learnwithbhawana

learnwithbhawana



learnwithbhawana

Recursion in Python

Recursion is when a function calls itself to solve a smaller version of a problem

learnwithbhawana



```
recursion.py > ...
1 def greet(x) :
2     if x ==0:
3         print("Stop")
4     else:
5         print("Hello")
6         greet(x-1)
7
8 #--Function call --
9 greet(6)
```

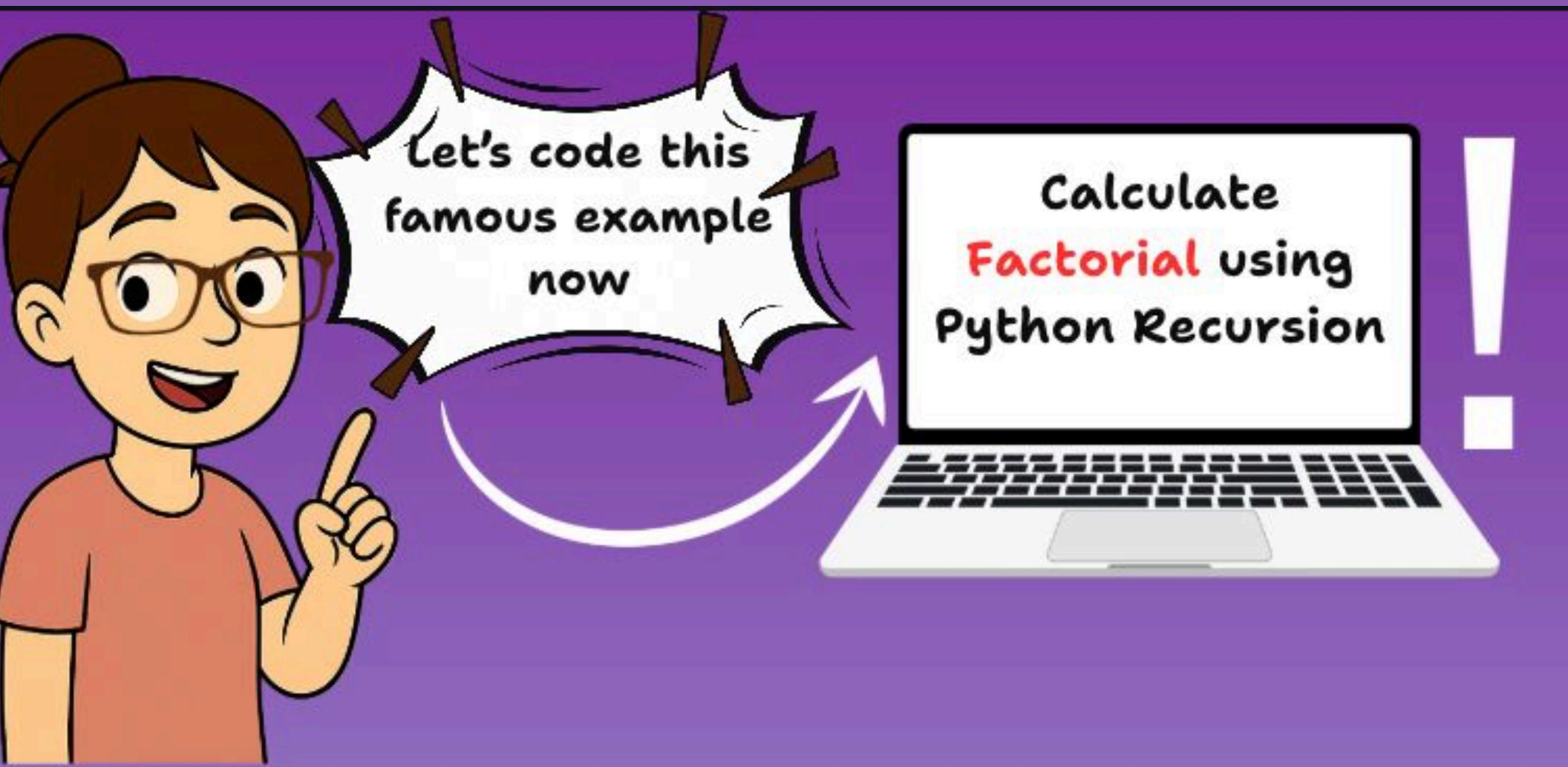
learnwithbhawana

Output:

```
Hello
Hello
Hello
Hello
Hello
Hello
Hello
Stop
```



learnwithbhawana

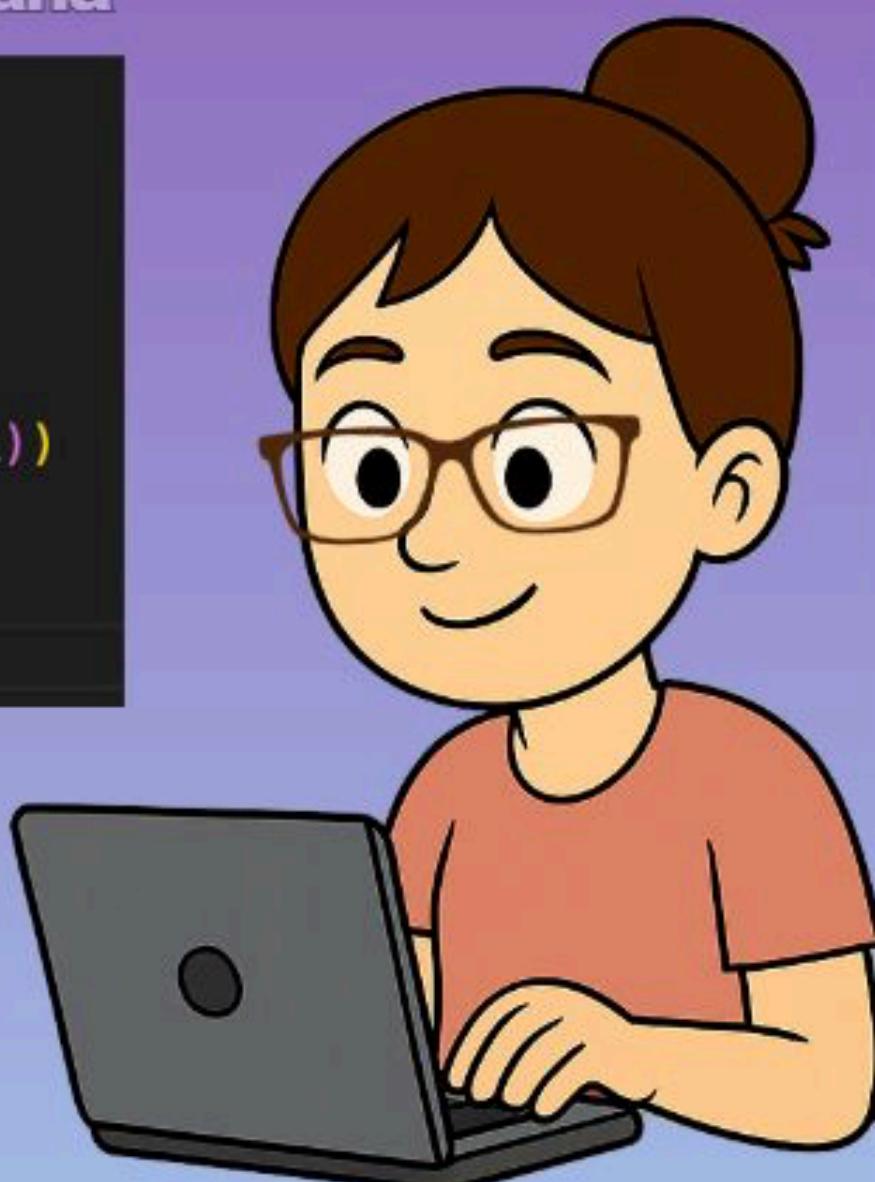


learnwithbhawana

```
1 #Factorial using recursion
2 def factorial(x) :
3     if x ==0:
4         return 1
5     else:
6         return (x * factorial(x-1))
7
8 #--Function call --
9 print(factorial(3))
```

Output:

6



learnwithbhawana

- In this example, I call **factorial(3)** .
- It says: $3 \times \text{factorial}(2)$
- Which is: $3 \times 2 \times \text{factorial}(1)$
- Which is: $3 \times 2 \times 1 \times \text{factorial}(0) \Rightarrow \text{return } 1$

★ Final Answer: $3 \times 2 \times 1 \times 1 = 6$

Code Explanation

```
1 #Factorial using recursion
2 def factorial(x) :
3     if x ==0:
4         return 1
5     else:
6         return (x * factorial(x-1))
7
8 #--Function call --
9 print(factorial(3))
```

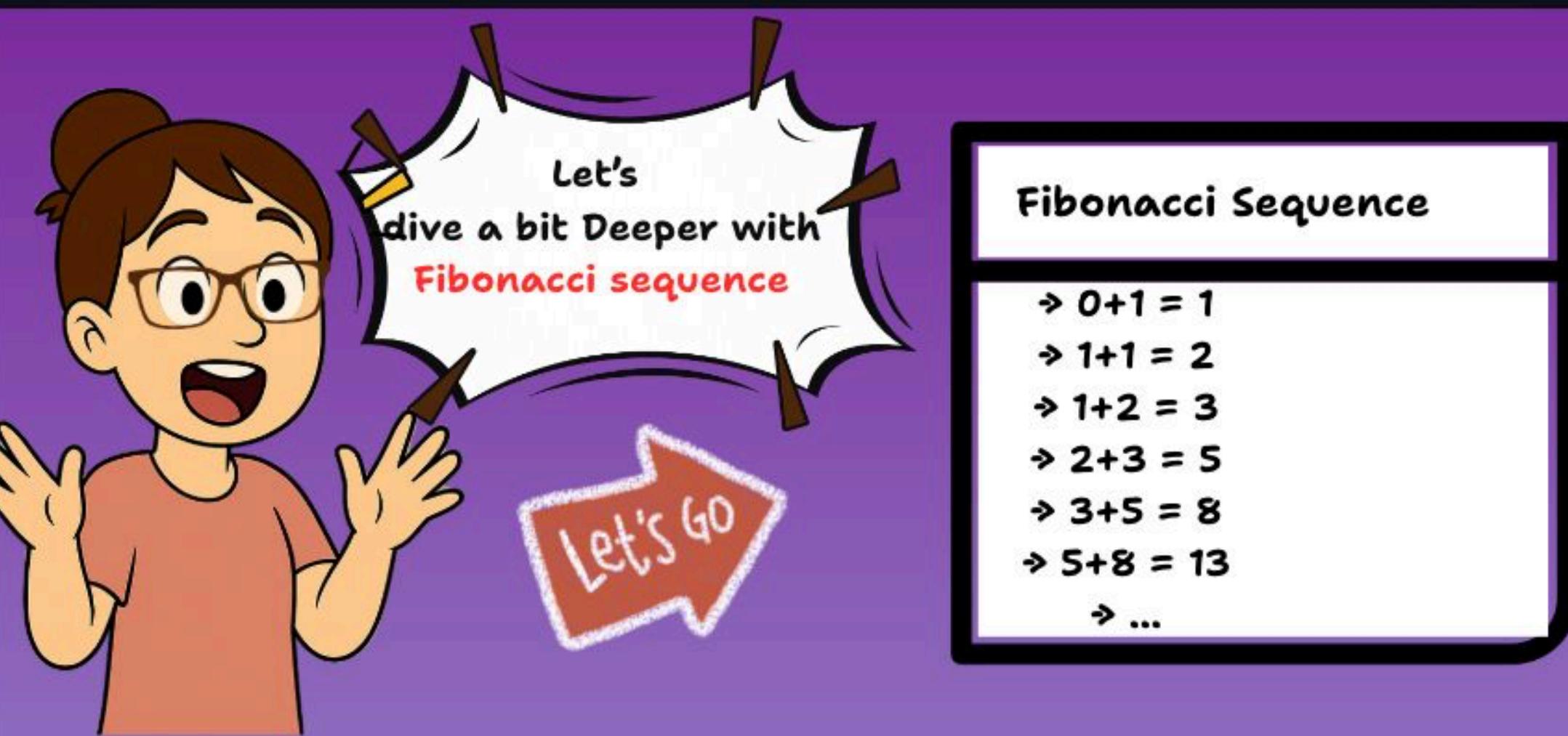
Output : 6

learnwithbhawana

- So **Recursion** is like a loop
- But Instead of Repeating, it calls itself **again and again** until done.

RECUSION

learnwithbhawana



Fibonacci Sequence

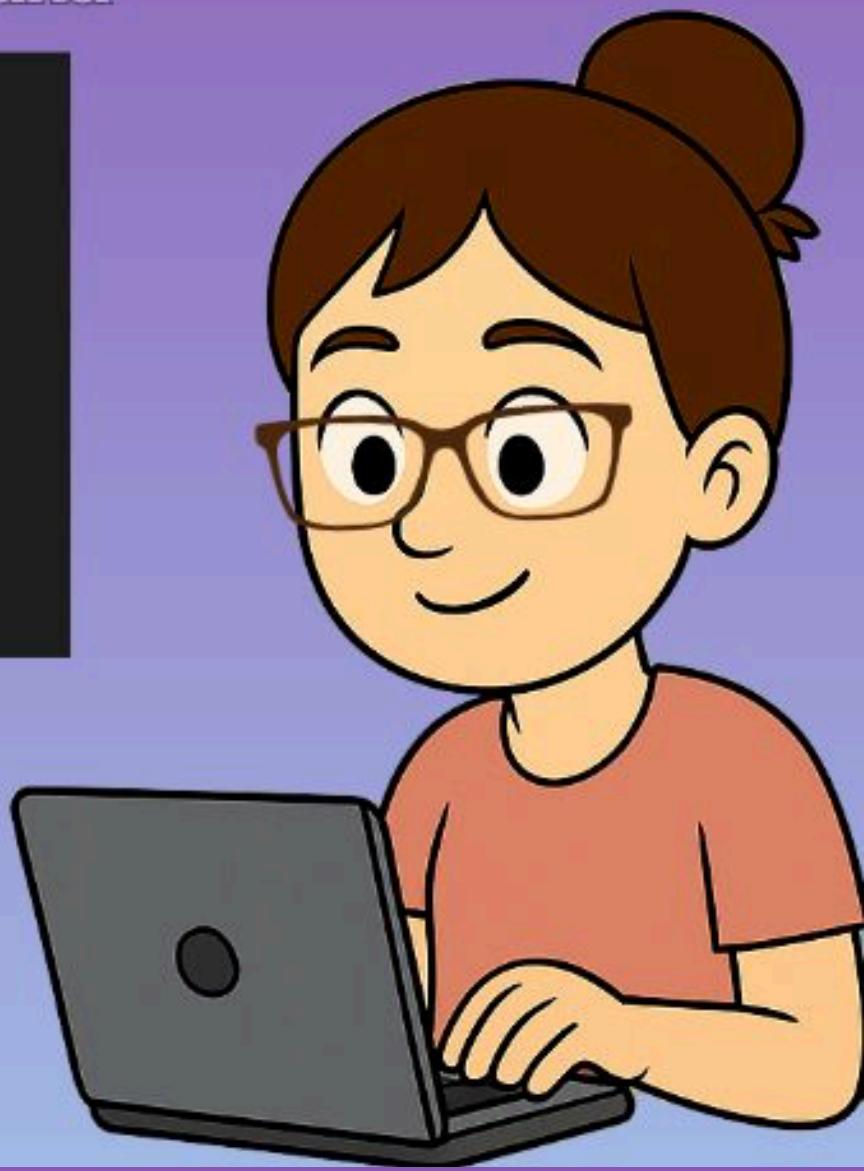
→ $0+1 = 1$
→ $1+1 = 2$
→ $1+2 = 3$
→ $2+3 = 5$
→ $3+5 = 8$
→ $5+8 = 13$
→ ...

learnwithbhawana

```
1 #Fibonacci using recursion
2 def fib(n):
3     if n <= 1:
4         return n
5     return fib(n-1) + fib(n-2)
6
7 #--Function call --
8 print(fib(4)) #Output : 3
9
```

Output:

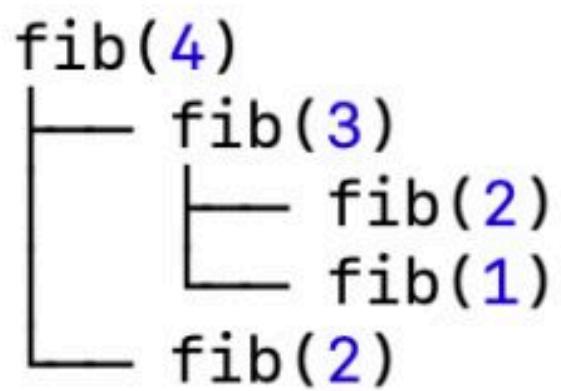
3



learnwithbhawana



Recursion Tree (for fib(4)):



learnwithbhawana

So, Recursion is **Powerful**.
Just remember these rules:

- 1 Always define a Base Case.
- 2 Function should call itself with a smaller input.
- 3 Trust the function to solve the smaller problem!



learnwithbhawana

Stack Overflow risk

Function that calls itself



`def recurse():`

...

`recurse()`

learnwithbhawana

So many calls!
My head is spinning!

Can be slow;
stack overflow risk

fact(30000)
fact(3000)
fact(300)
fact(30)
fact(3)
fact(0)

RecursionError

learnwithbhawana

Memoization

Let's remember past answers!



Memoization stores answers of past calls



```
from functools import lru_cache
@lru_cache(maxsize=None)
def fib(n):
    if n <= 1:
        return n
    return fib(n-1) + fib(n-2)
```

fib(6) (calls **fib(5)**, **fib(4)**, **fib(3)** only once each)

Result: **fib(6) = 8** #fast and efficient!

Now: If **fib(3)** is called again, Python gets it from memory, not recalculates.

fib(3) → saved in cache → reused

@lru_cache remembers previous answers, so the function doesn't repeat the same work.

