Newton School of Technology

4

Time and Space complexity

Newton School of Technology

2

Problem – Compute a^b

Brute Force Solution

Newton School of Technology

- To calculate a^b all we need to do is multiply a b times.
- We also need to calculate the ans as modulo $10^9 + 7$:
  - One way to do it is to modulo in the end.

```
ans = 1
mod = 10**9 + 7
for i in range(b):
    ans = ans * a

print(ans%mod)
```

Brute Force Solution

- To calculate a^b all we need to do is multiply a b times.
- We also need to calculate the ans as modulo $10^9 + 7$:
  - One way to do it is to modulo in the end.

```
ans = 1
mod = 10**9 + 7
for i in range(b):
    ans = ans * a

print(ans%mod)
```

Can we improve brute force?

Newton School of Technology

Brute Force Solution

Newton School of Technology

- Instead of doing modulo in the end we can do modulo after each multiplication so that numbers don't become huge and multiplication doesn't take much time.

```python
ans = 1
mod = 10**9 + 7
for i in range(b):
    ans = ans * a
    ans = ans % mod

print(ans)
```

Can we do exponentially better?

Hint

- You need to print “I am awesome” 500 times, each in new line. But you are not allowed to use loops, repetition operator, defining new function or variables or using any other built in function apart from print.

SOLUTION

1. Write one line then copy paste it.
2. Now you have 2 lines, copy whole thing again and paste it.
3. Now you have 4 lines, copy whole thing again and paste it.
4. Now you have 8 lines, copy whole thing again and paste it.
5. Now you have 16 lines, copy whole thing again and paste it.
6. Now you have 32 lines, copy whole thing again and paste it.
7. Now you have 64 lines, copy whole thing again and paste it.
8. Now you have 128 lines, copy whole thing again and paste it.
9. Now you have 256 lines, copy whole thing again and paste it.
10. Now you have 512 lines, copy whole thing again and paste it.
11. Remove the last 12 lines.

SOLUTION

What will be the number of copy paste required to print n lines?

#Copy-Paste	#Lines
1	2
2	4
3	8
…………..	…………….
…………..	…………….
n	2^n

Let’s say we want to print x lines and for that we require y copy paste. Then we have:

$2^y = x$

$y = \log_2^x$

Power of this solution

N	$\log_2^N$
2	1
100	7
1000000000	30
1000000000000000000	60
....	....
107150860718626732094842504906000181056140481170553360744375038837035105112493612249319837881569585812759467291755314682518714528569231404359845775746985748039345677748242309854210746050623711418779541821530464749835819412673987675591655439460770629145711964776865421676604298316526243868372056680693 76	1000

Can you use similar kind of method for a^b?

Let's say a = 7, b = 32:

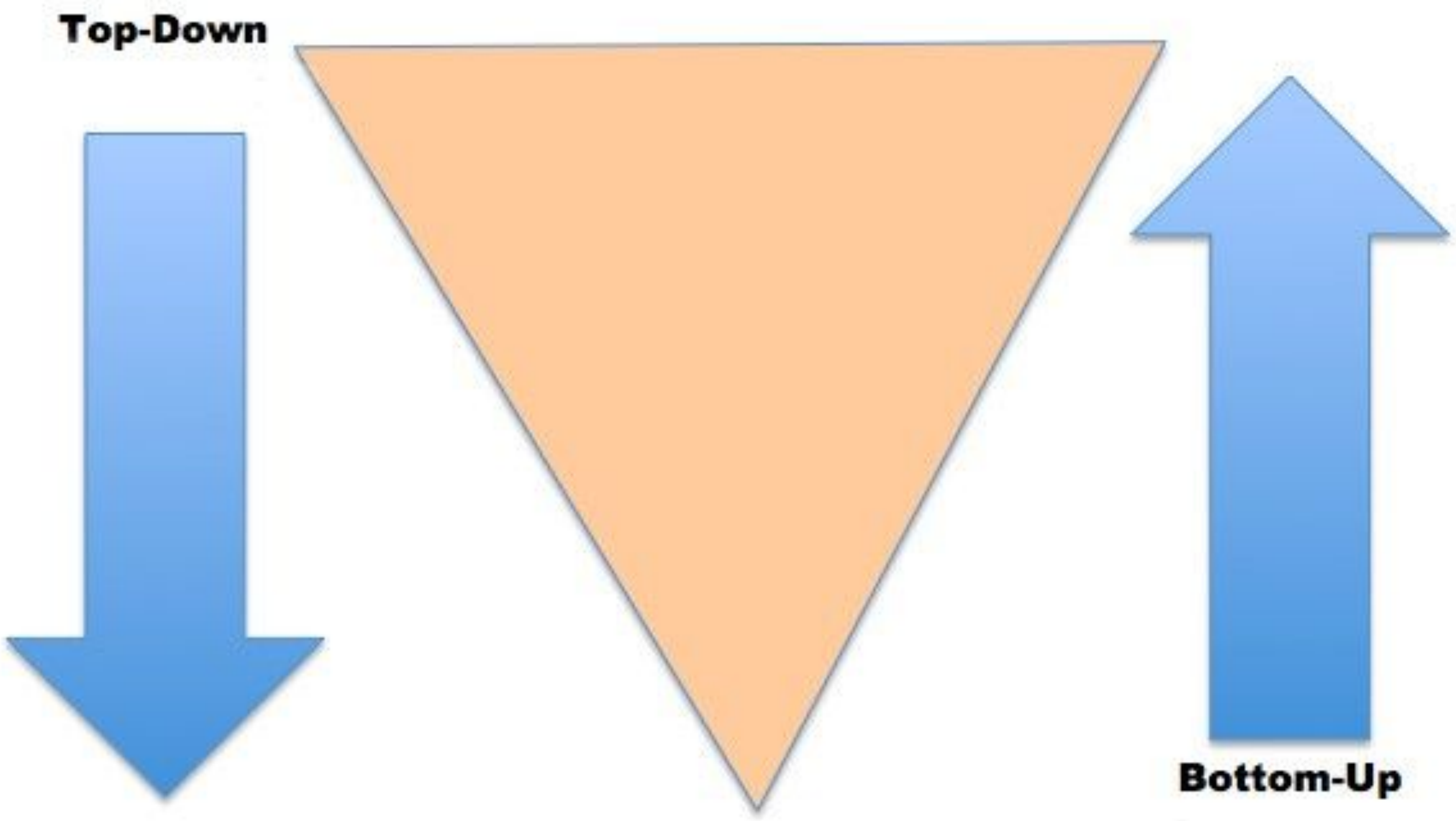
1. $7^2 = 7 * 7$
2. $7^4 = 7^2 * 7^2$
3. $7^8 = 7^4 * 7^4$
4. $7^{16} = 7^8 * 7^8$
5. $7^{32} = 7^{16} * 7^{16}$

What if b is not odd or not a power of 2?

Let's say a = 7, b = 41:

1. $7^2 = 7 * 7$
2. $7^4 = 7^2 * 7^2$
3. $7^8 = 7^4 * 7^4$
4. $7^{16} = 7^8 * 7^8$
5. $7^{32} = 7^{16} * 7^{16}$
6. $7^{41} = 7^{32} * 7^8 * 7$

Can you think of a more cleaner solution?



Binary Exponentiation

Let’s say a = 7, b = 41:

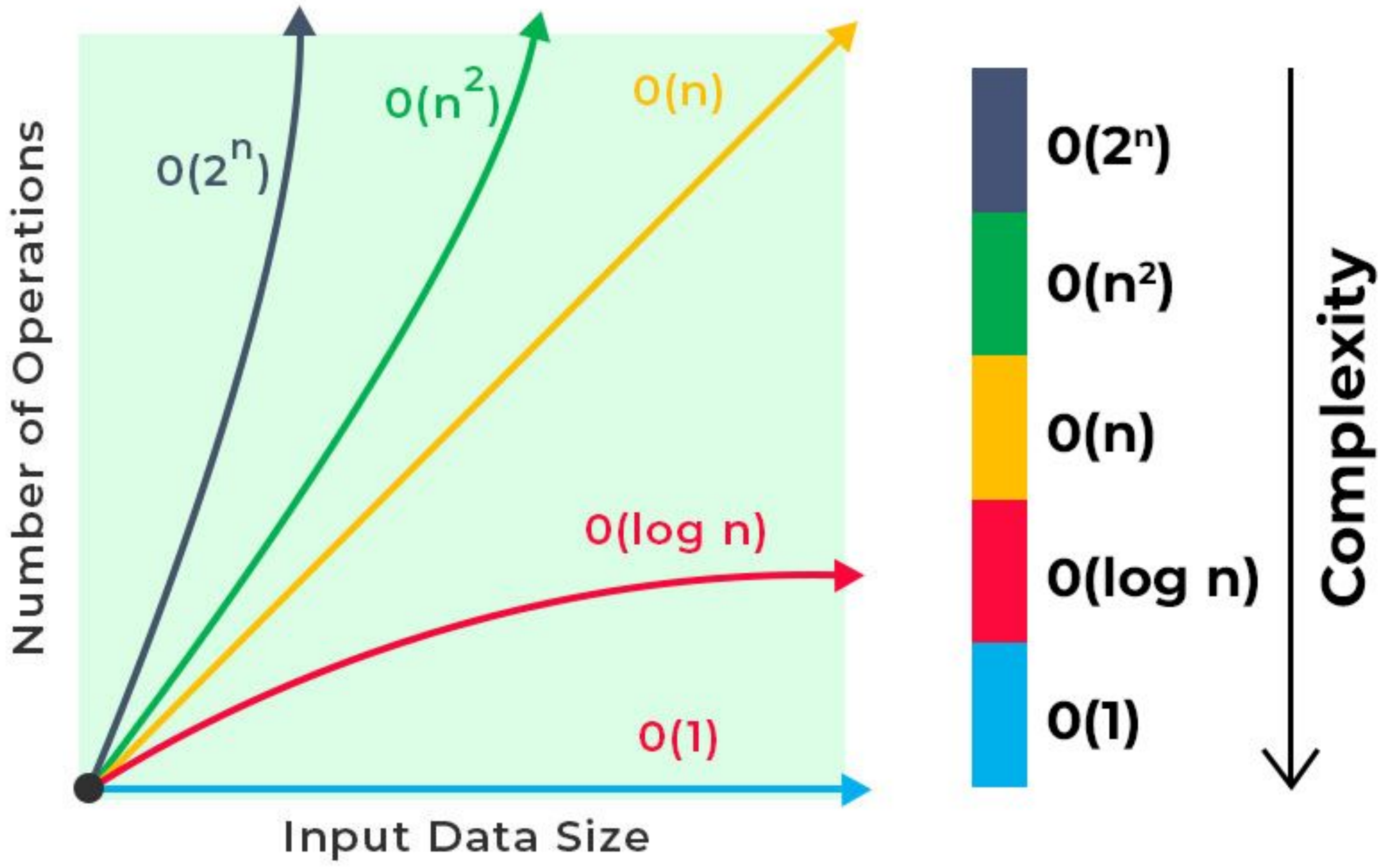
1. $7^{41} = 7 * 7^{20} * 7^{20}$
2. $7^{20} = 7^{10} * 7^{10}$
3. $7^{10} = 7^{5} * 7^{5}$
4. $7^{5} = 7 * 7^{2} * 7^{2}$
5. $7^{2} = 7 * 7$

Binary Exponentiation
For Competitive Programming

$$x^n = \begin{cases} x * (x^2)^{\frac{n-1}{2}} & \text{, if n is odd} \\ (x^2)^{\frac{n}{2}} & \text{, if n is even} \end{cases}$$

Time complexities graph

Newton School of Technology



Newton School of Technology

17

Space Complexity

Why space complexity?

Newton School of Technology

Imagine you're packing for a vacation:

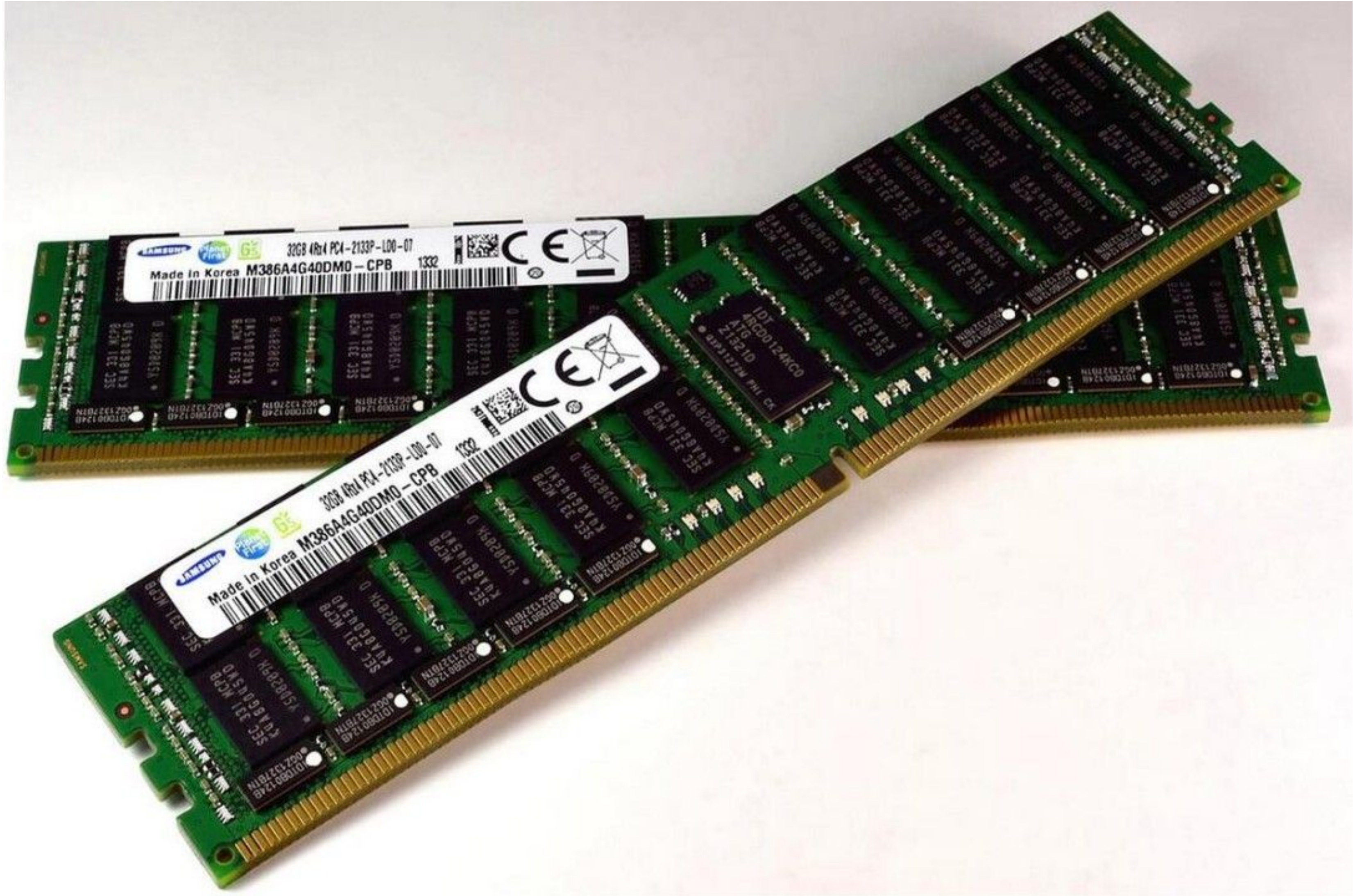
Big suitcase: If you have a huge suitcase, you don't worry much about folding clothes or choosing essentials. You can just throw everything in. But big suitcase is **cumbersome to handle** and is more **expensive**.

Small suitcase: If you have a compact suitcase, you must carefully fold your clothes, **pack only what's necessary, and leave out non-essentials**.

Why space complexity?

Newton School of Technology

In programming, the suitcase is like your program's memory. Optimizing space complexity is like learning to pack smarter, ensuring that everything fits without wasting space or leaving out important items.



What is space complexity?

Newton School of Technology

Space complexity is a measure of the amount of memory an algorithm uses during its execution. It includes both:

Fixed Space Requirements: Memory needed for constants, program instructions. This part **remains the same regardless of the size of the input**.

Variable Space Requirements: Memory needed for input data, temporary storage. **This part depends on the size of the input**.

Auxiliary vs Total space complexity

Newton School of Technology

Aspect	Auxiliary Space Complexity	Total Space Complexity
Definition	Memory used **in addition** to the input size.	**All memory** used by the algorithm.
Includes Input Size?	No	Yes
Focus	Measures how efficiently extra memory is used.	Measures the algorithm's total memory usage.
Examples	Temporary variables, recursion stack.	Input data, output storage, auxiliary space.

Big-O Notation for Space Complexity

Newton School of Technology

Just like time complexity, space complexity is expressed in terms of Big-O Notation to describe how the memory usage grows relative to the input size n. And just like in time complexity we are concerned about the worst case scenario. Some examples:

O(1): Constant space (e.g., iterating over an array without storing additional data).
O(n): Linear space (e.g., storing elements of an array or a list).
O(n^2): Quadratic space (e.g., storing a 2D matrix).

Quiz: What is the space complexity?

What is the space complexity?

Newton School of Technology

```
a = 1
```

Auxiliary Space Complexity: O(1)
Total Space Complexity: O(1)

What is the space complexity?

Newton School of Technology

```
a = 10
b = 20
c = a+b

print(c)
```

Auxiliary Space Complexity: O(1)
Total Space Complexity: O(1)

What is the space complexity?

Newton School of Technology

```python
def find_sum(arr):
    total = 0
    for num in arr:
        total += num
    return total
```

Auxiliary Space Complexity: O(1)
Total Space Complexity: O(1)

What is the space complexity?

Newton School of Technology

```python
# Take input from the user in a single line
arr = list(map(int, input("Enter the elements of the array").split()))

# Calculate the sum of the array
sum_of_elements = sum(arr)

# Print the result
print("Array:", arr)
print("Sum of elements:", sum_of_elements)
```

Auxiliary Space Complexity: O(1)
Total Space Complexity: O(n)
TIme complexity : O(n)

What is the space complexity?

```python
# Take input from the user
arr = list(map(int, input().split()))

# Create an extra array to store the reversed elements
reversed_arr = []

# Add elements to the extra array in reverse order
for i in range(len(arr) - 1, -1, -1):  # Loop from the last index to the first
    reversed_arr.append(arr[i])

# Print the original and reversed arrays
print("Original Array:", arr)
print("Reversed Array:", reversed_arr)
```

Auxiliary Space Complexity: O(n)
Total Space Complexity: O(n) for input + O(n) for auxiliary => O(2*n) => O(n)
TIme complexity : O(n)

What is the space complexity?

```
def create_matrix(n):
    # Creates an n x n matrix
    matrix = [[0] * n for _ in range(n)]
    return matrix
```

Auxiliary Space Complexity: $O(n^2)$
Total Space Complexity: $O(n^2)$

What is the space complexity?

```python
def find_max(arr):
    max_val = arr[0]
    for num in arr:
        if num > max_val:
            max_val = num
    return max_val

arr = list(map(int, input().split()))
print(find_max(arr))
```

Auxiliary Space Complexity: O(1)
Total Space Complexity: O(n)

What is the space complexity?

Newton School of Technology

```python
def find_sum(n):
    total = 0
    for i in range(1, n + 1):
        total += i
    return total
```

Auxiliary Space Complexity: O(1)
Total Space Complexity: O(1)

What is the space complexity?

```python
def fib(n):
    dp = [0] * (n + 1)
    dp[1] = 1
    for i in range(2, n + 1):
        dp[i] = dp[i - 1] + dp[i - 2]
    return dp[n]
```

Auxiliary Space Complexity: O(n)
Total Space Complexity: O(n)

What is the space complexity?

```
def find_pairs(arr):
    n = len(arr)
    result = []
    for i in range(n):
        for j in range(i + 1, n):
            if arr[i] + arr[j] > 10:
                result.append((arr[i], arr[j]))
    return result
```

Auxiliary Space Complexity: $O(n^2)$
Total Space Complexity: $O(n^2)$

What is the space complexity?

Newton School of Technology

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

What is the space complexity?

Newton School of Technology

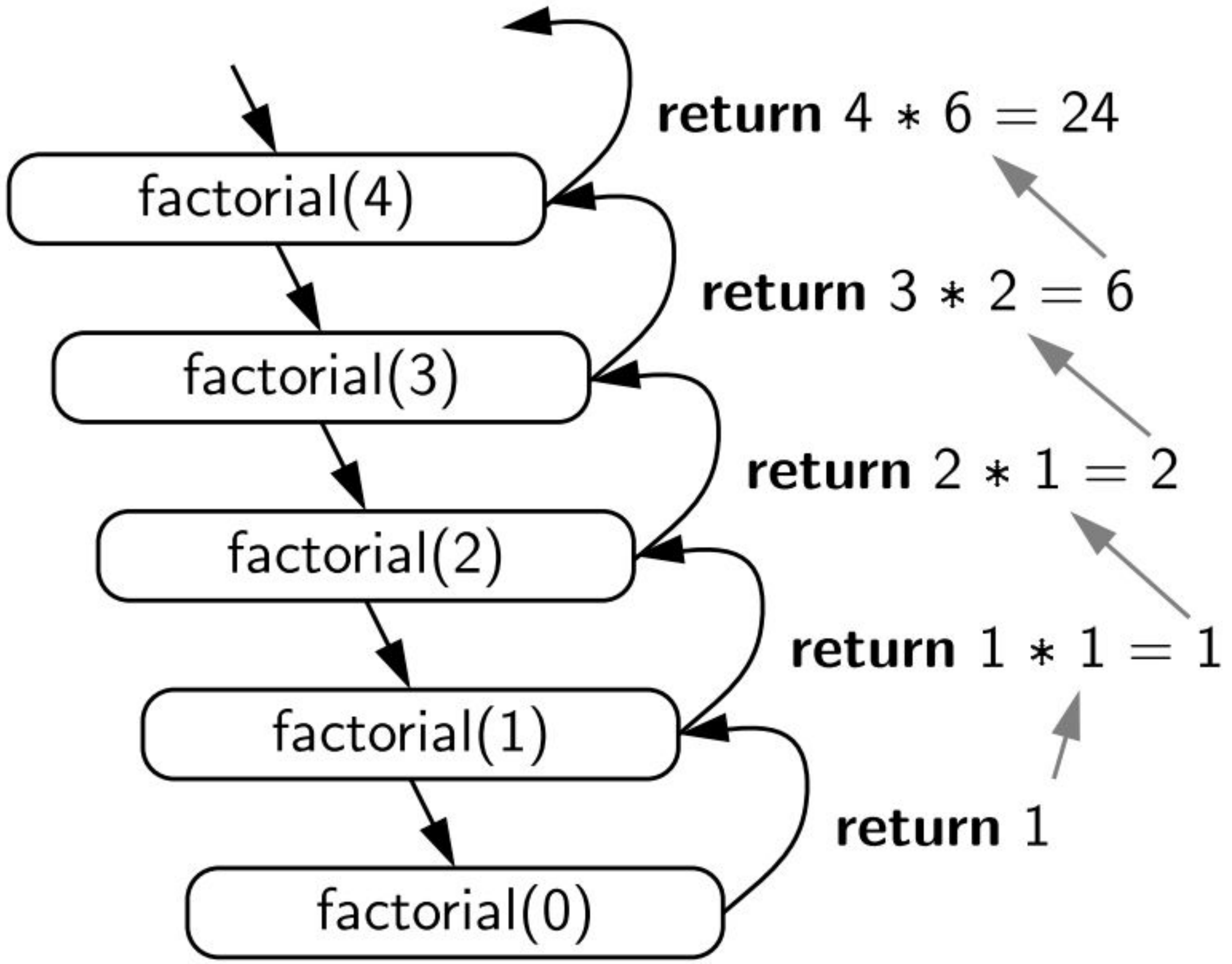
```python
def factorial(n):
    if n == 0:
        return 1
    return n * factorial(n - 1)
```

O(1) → ✗ → But why?

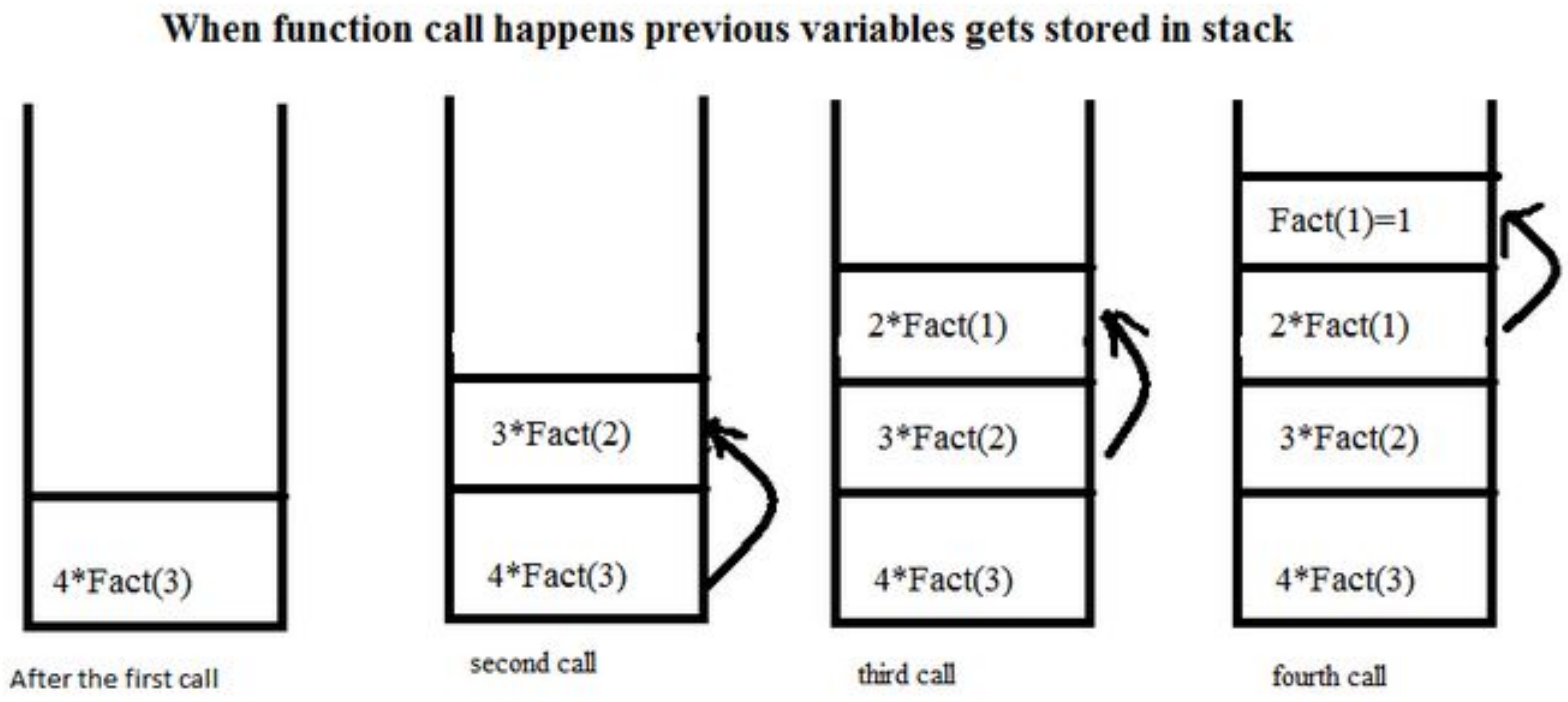
Newton School of Technology

Space Complexity in Recursion

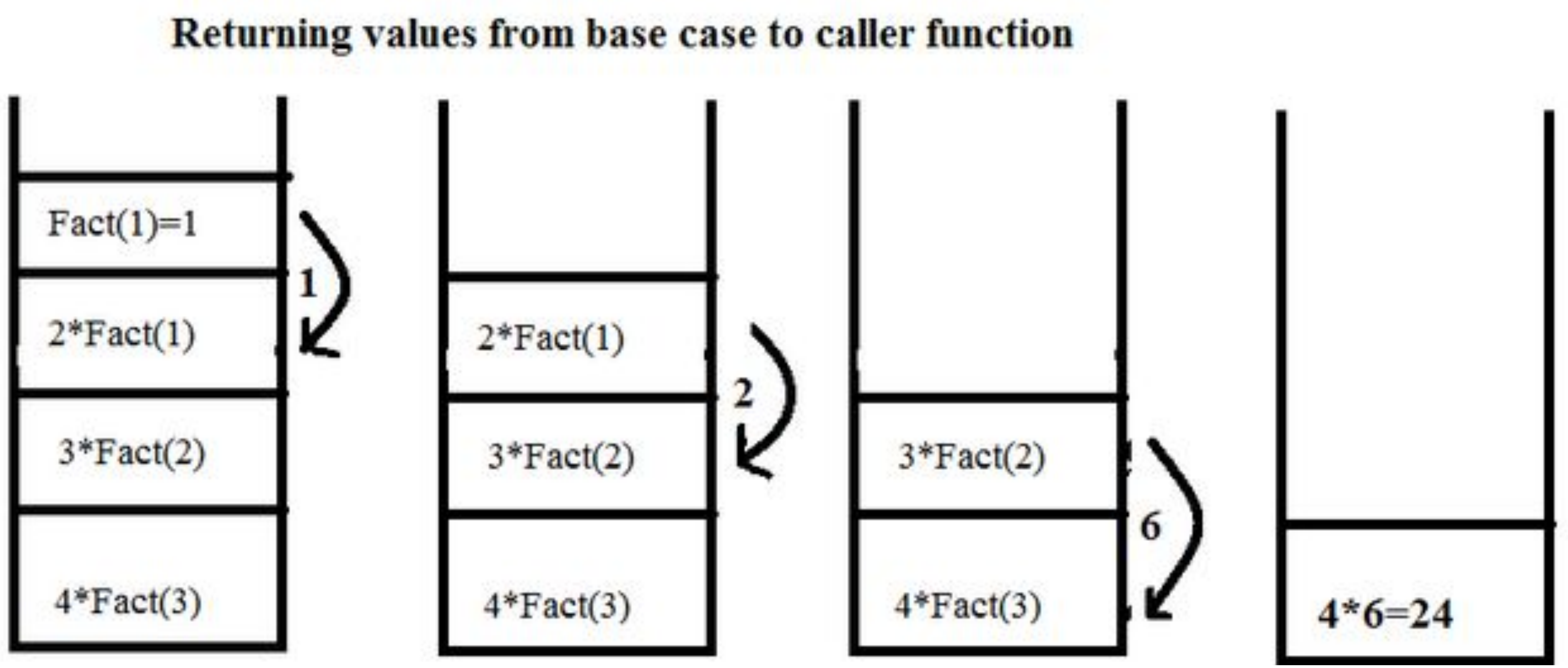
Recursive calls



Space taken by stack



Space taken by stack



Approximating memory required in a python program

Memory taken by variable in python

Standardizing memory usage for Python data types is difficult because memory is influenced by:

- The dynamic and flexible nature of Python objects.
- Platform and implementation-specific factors.
- Internal optimizations and runtime behavior.
- Metadata storage for each object.

So rather than looking to come up with an exact amount we should look to approximate.

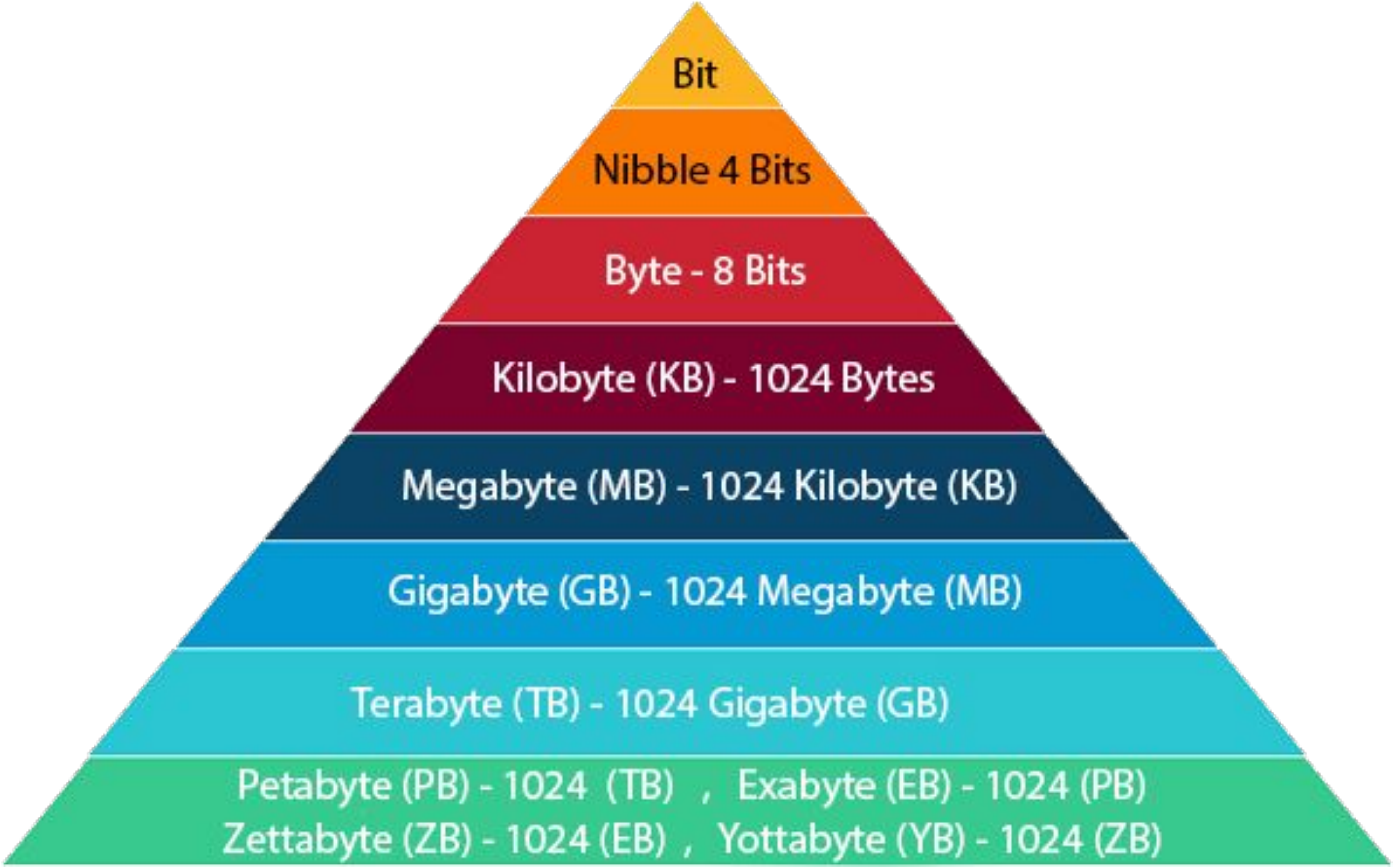
Memory taken by variable in python

But we can do an reasonable approximation by assuming:

- **Overhead** for every new object (Integer/String/List): **30-100 bytes**
- **Additional** space for each number: **8 bytes**
- **Additional** space for each character in string: **1 byte**

Memory units

Newton School of Technology



Summary:

Newton School of Technology

- **Binary exponentiation:** For efficient computation of powers.
- **Space complexity:** Why, What and How?
    - Total Space complexity and Auxiliary Space Complexity
- **Recursive calls and stack space**
- **Approximation** for memory usage in Python

Thank You!