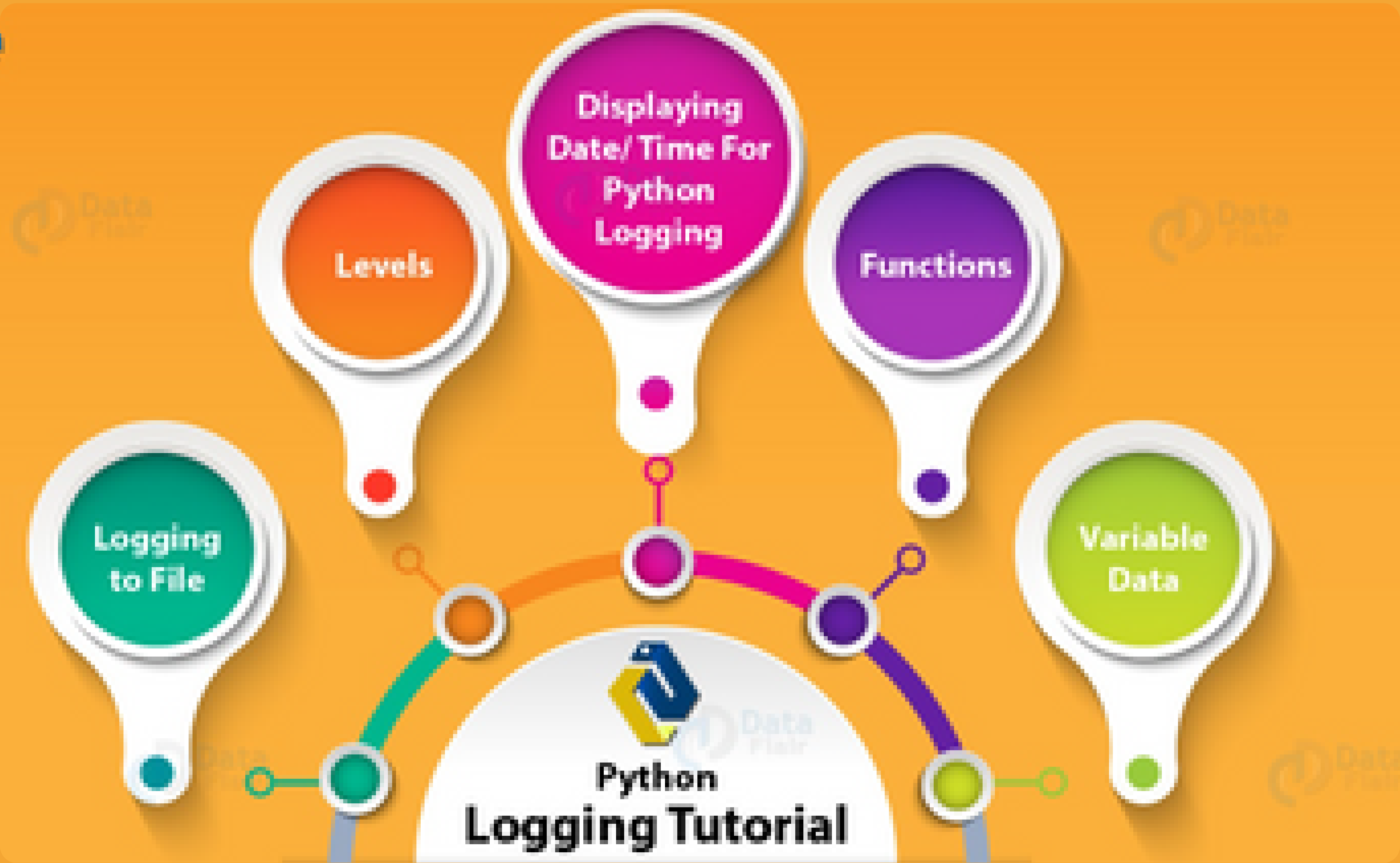
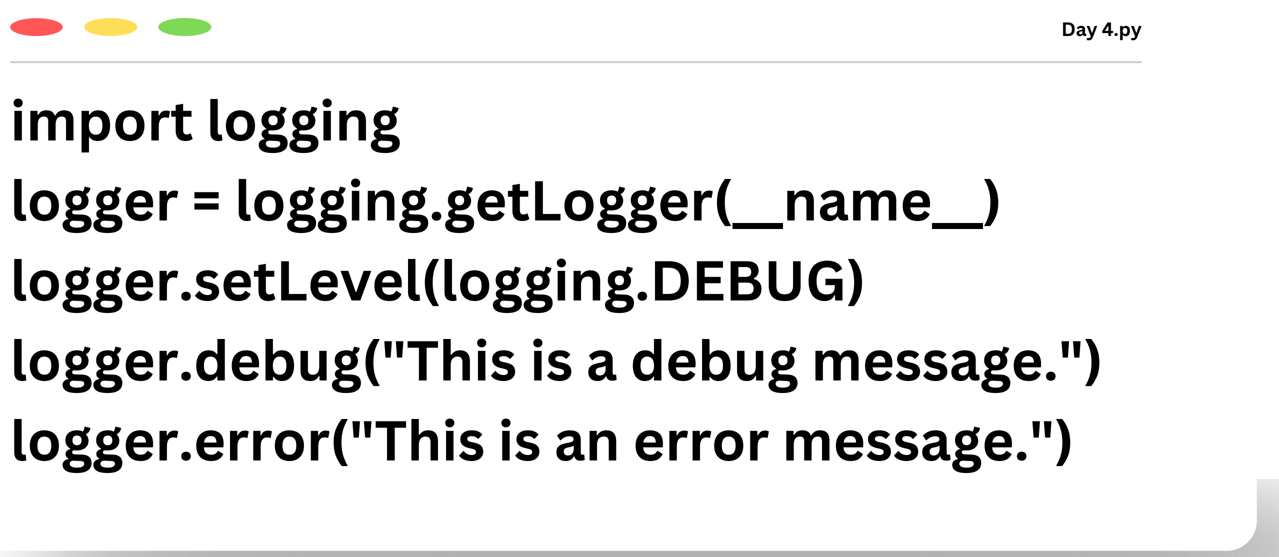
#DAY 22

LOGGING IN PYTHON



LOGGING IN PYTHON

The logging module in Python is a standard library module that provides a flexible event logging and tracking system for Python applications. It allows you to track events that occur while your program is running, and you can use it to record information about errors, warnings, and other events that occur during program execution.



```
import logging
logger = logging.getLogger(__name__)
logger.setLevel(logging.DEBUG)
logger.debug("This is a debug message.")
logger.error("This is an error message.")
```

EXAMPLE

Day 4.py

```
l1_int = []
l2_str = []
for i in l :
  logging.info("we are iterating throuhg our list and our local var is
{}".format(l ))
  if type(i ) == list :
    logging.info("i am inside if statement and i am trying to check list
type" + str(i))
    for j in i :
      logging.info("i am in anothe for loop for list inside list element "+
str(j))
      if type(j) == int :
        logging.info("i am inside if statement")
        l1_int.append(j)
  elif type(i) == int :
    l1_int.append(i)
  else :
    if type(i) == str :
      l2_str.append(i)
logging.info("my final result for int is {l1} and str is {l2}".format(l1
=l1_int , l2 = l2_str ))
```

LOGGING LEVELS

DEBUG:

- This level is used for detailed information, typically of interest only when diagnosing problems.

INFO:

- This level is used to confirm that things are working as expected.

WARNING:

- This level is used as an indication that something unexpected happened, or is indicative of some problem in the near future.

ERROR:

- This level is used to indicate that an error occurred, but the program can continue running.

CRITICAL:

- This level is used to indicate that a critical error occurred, and the program can no longer continue running.

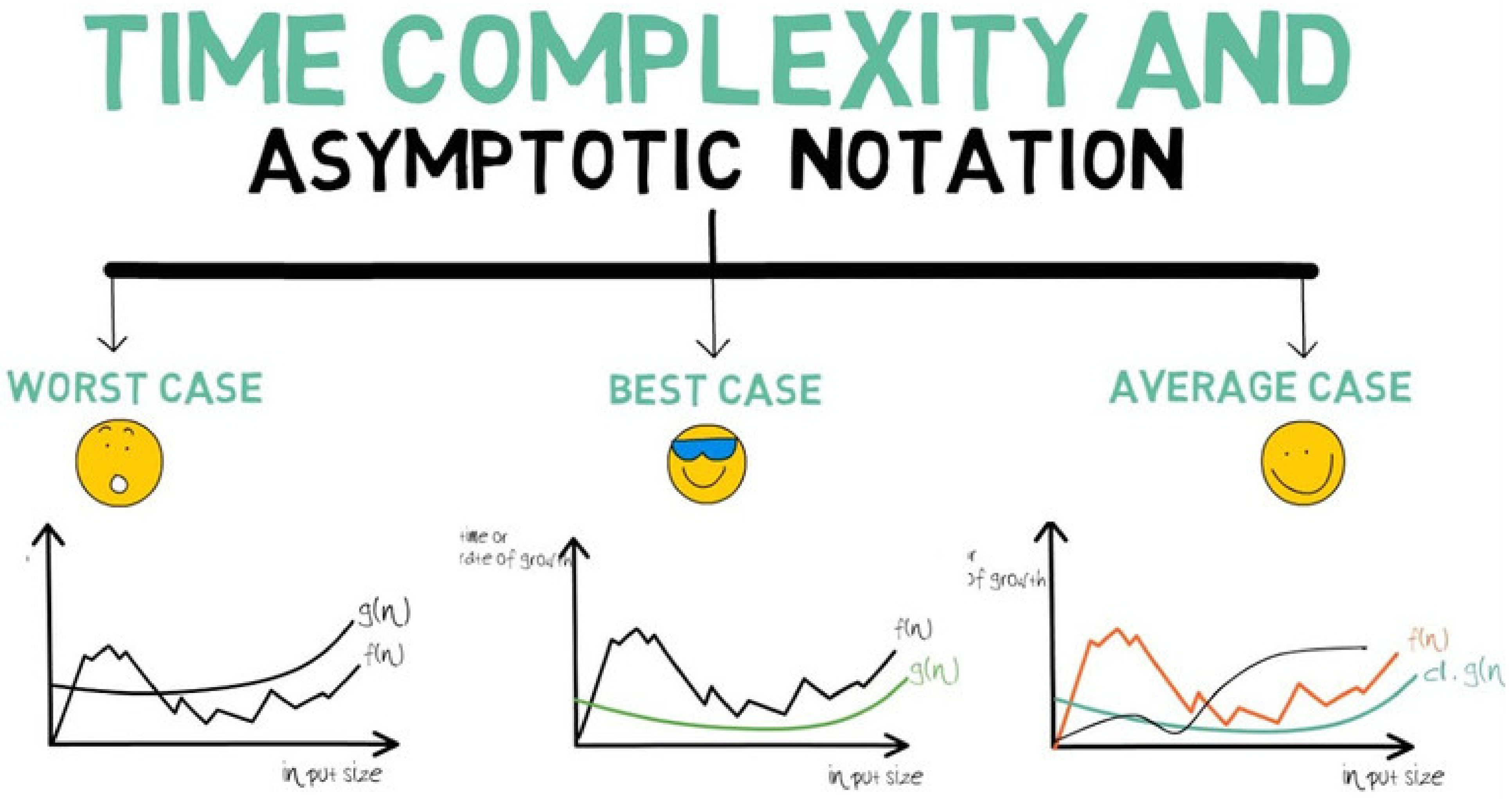
EXAMPLES

Day 4.py

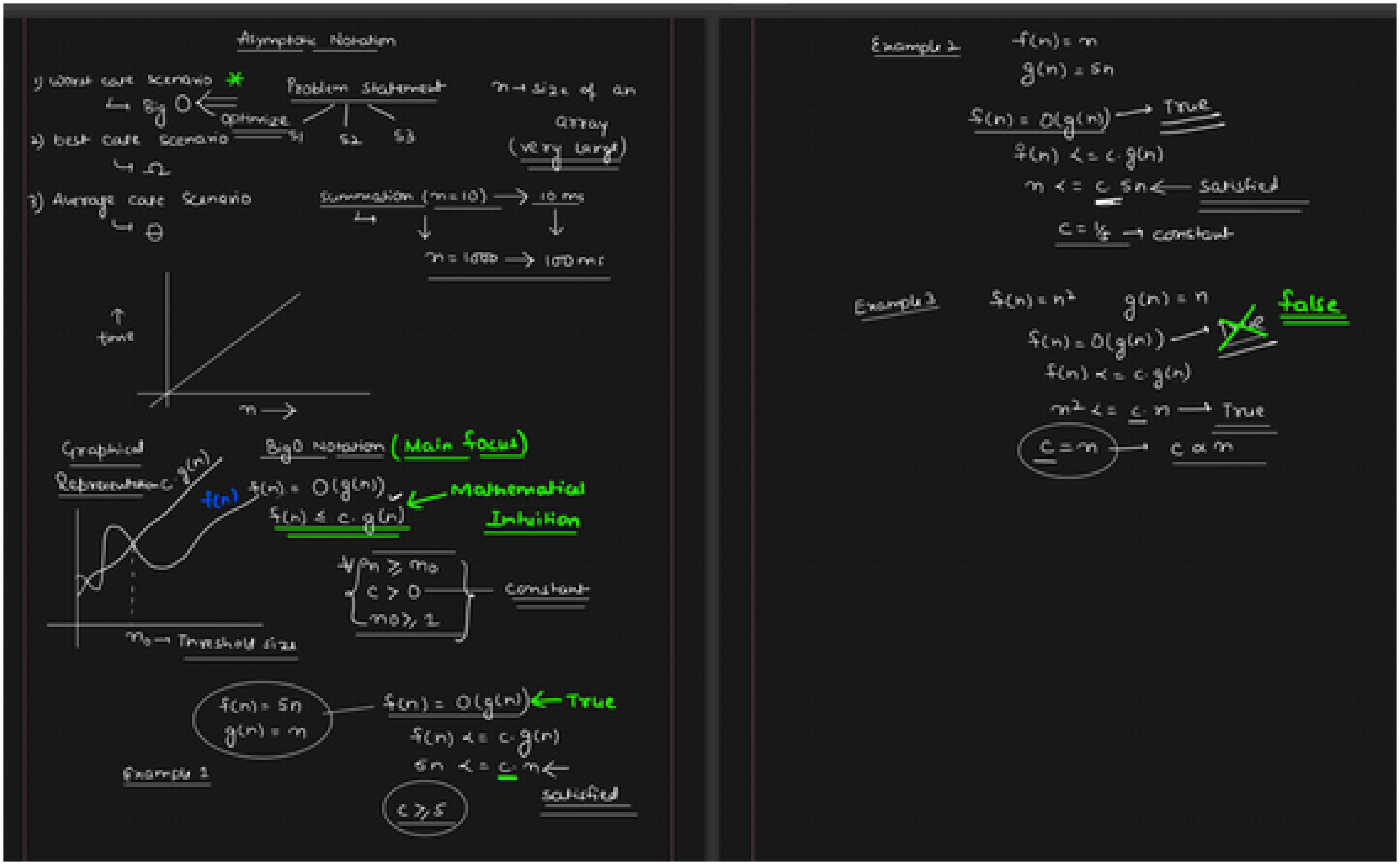
```
import logging
logging.basicConfig(filename = "test.log" ,level = logging.INFO)
logging.info("log this line of executation" )
logging.debug("this is my msg")
logging.warning("this is my warning msg")
logging.error("this is my error")
logging.critical("this is my critical msg")
logging.shutdown()
```

#DAY 4

ASYMPTOTIC NOTATION



BIG-O NOTATION



Asymptotic Notation

1) Worst case scenario *
↳ Big O ⟸ optimize

2) best case scenario
↳ Ω

3) Average case scenario
↳ Θ

Problem Statement → S1, S2, S3

n → size of an array (very large)

Summation (n = 10) ⟶ 10 ms
↓
n = 1000 ⟶ 1000 ms

Graphical Representation

n_0 → Threshold size

Big O Notation (Main focus)

$f(n) = O(g(n))$ ✓ ⟵ Mathematical Intuition

$f(n) \le c \cdot g(n)$

$$\left\{\begin{array}{l} \forall\, n \geq n_0 \\ c > 0 \\ n_0 \geq 1 \end{array}\right\} \text{— constant}$$

Example 1

$f(n) = 5n$
$g(n) = n$

$f(n) = O(g(n))$ ⟵ True

$f(n) <= c \cdot g(n)$

$5n <= c \cdot n$ ⟵ satisfied

$c \geq 5$

Example 2

$f(n) = n$
$g(n) = 5n$

$f(n) = O(g(n)) \longrightarrow$ True

$f(n) <= c \cdot g(n)$

$n <= c\ 5n$ ⟵ satisfied

$c = 1/5 \rightarrow$ constant

Example 3

$f(n) = n^2$ $g(n) = n$

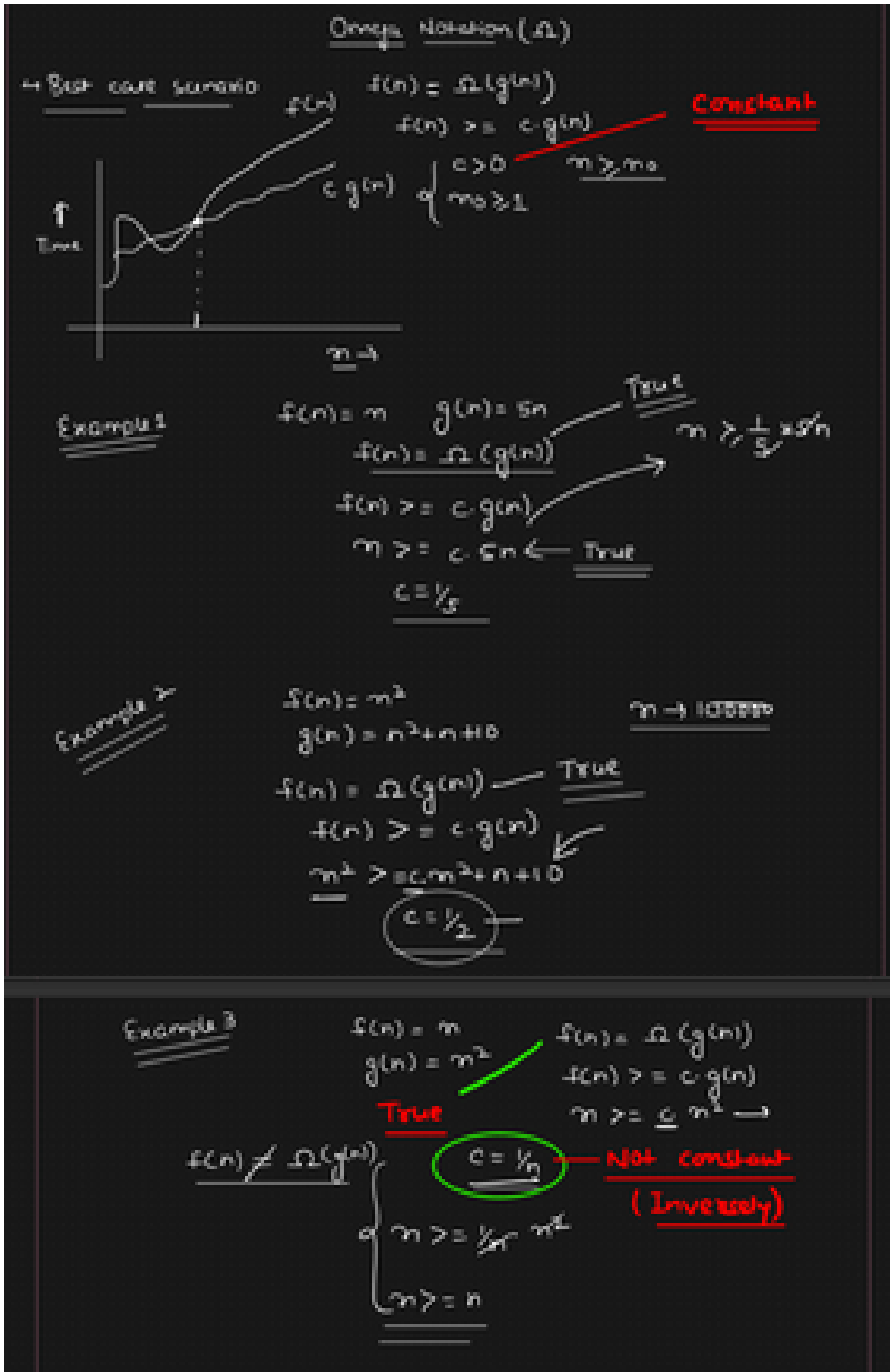
$f(n) = O(g(n)) \longrightarrow$ ~~True~~ false

$f(n) <= c \cdot g(n)$

$n^2 <= c\ n \longrightarrow$ True

$c = n \longrightarrow c \propto n$

OMEGA NOTATION



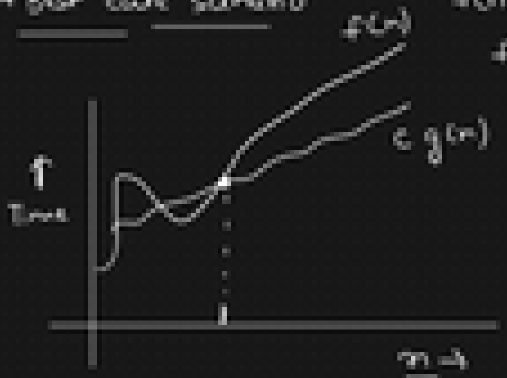
Omega Notation (Ω)

→ Best case scenario

$f(n) = \Omega(g(n))$

$f(n) >= c \cdot g(n)$ — Constant

$\begin{cases} c > 0 \\ n_0 \geq 1 \end{cases}$ $n \geq n_0$



Example 1

$f(n) = n$ $g(n) = 5n$ — True

$f(n) = \Omega(g(n))$

$n \geq \frac{1}{5} \times 5n$

$f(n) >= c \cdot g(n)$

$n >= c \cdot 5n$ ← True

$c = 1/5$

Example 2

$f(n) = n^2$

$g(n) = n^2+n+10$

$n \to 100000$

$f(n) = \Omega(g(n))$ — True

$f(n) >= c \cdot g(n)$

$n^2 >= c \, n^2+n+10$

$c = 1/2$

Example 3

$f(n) = n$

$g(n) = n^2$

True

$f(n) = \Omega(g(n))$

$f(n) >= c \cdot g(n)$

$n >= \underline{c} \, n^2$ →

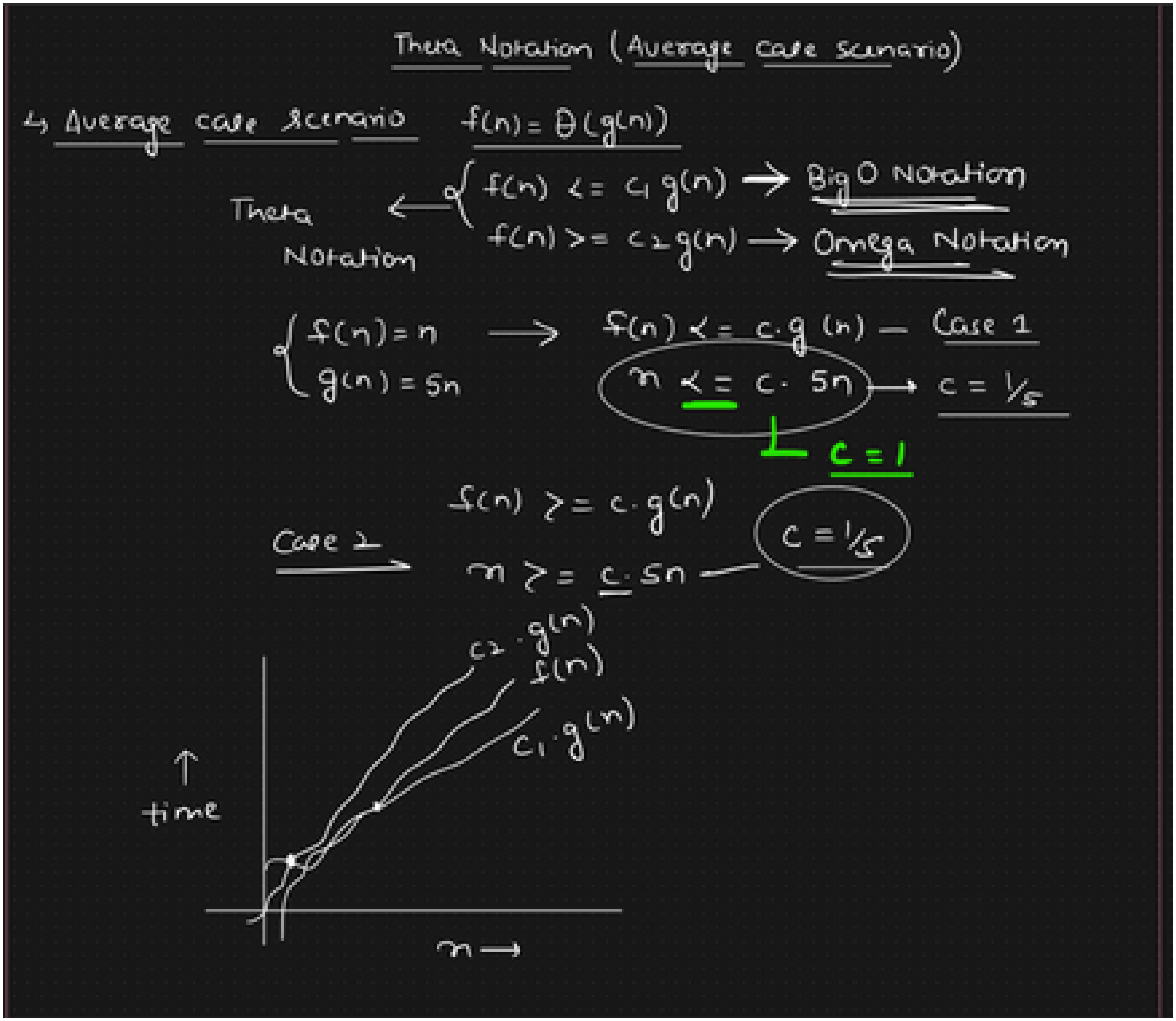
$f(n) \neq \Omega(g(n))$

$c = 1/n$ — Not constant (Inversely)

$n >= \frac{1}{n} n^2$

$n >= n$

THETA NOTATION



Theta Notation (Average case scenario)

↳ Average case scenario $f(n) = \Theta(g(n))$

Theta Notation ← $\begin{cases} f(n) <= c_1\, g(n) \rightarrow \text{Big O Notation} \\ f(n) >= c_2\, g(n) \rightarrow \text{Omega Notation} \end{cases}$

$\begin{cases} f(n) = n \\ g(n) = 5n \end{cases} \longrightarrow f(n) <= c.g(n)$ — Case 1

$n <= c \cdot 5n \longrightarrow c = 1/5$

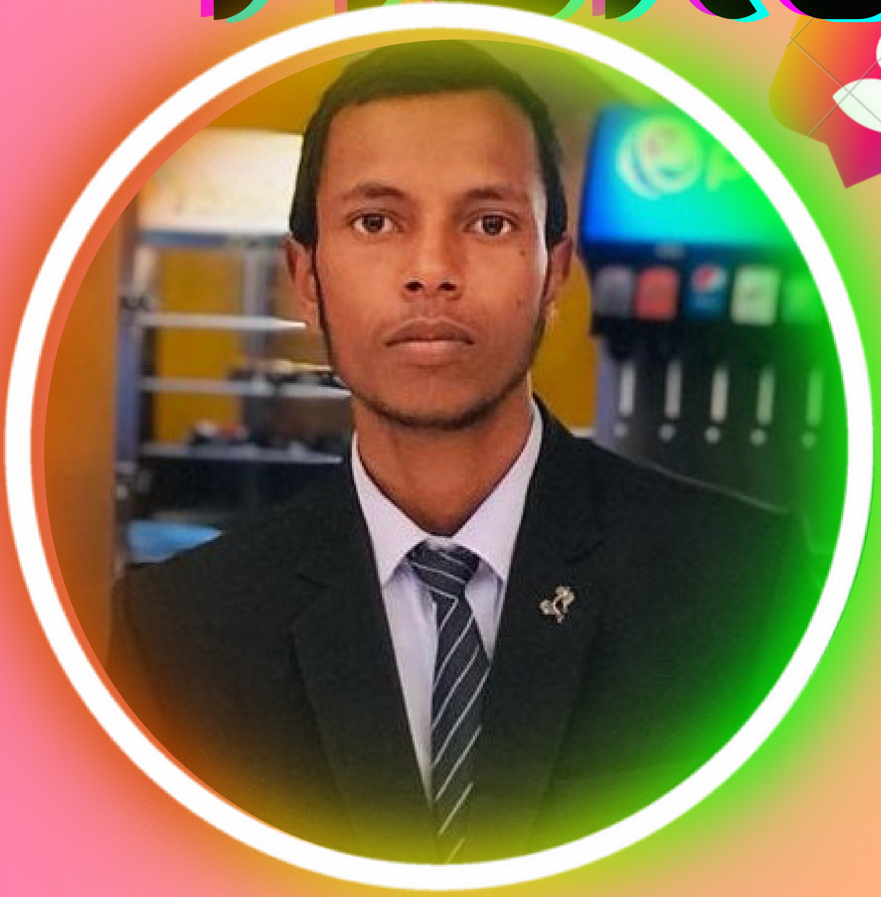
$c = 1$

Case 2 $\longrightarrow$ $f(n) >= c.g(n)$

$n >= c.5n$ — $c = 1/5$

$c_2 \cdot g(n)$
$f(n)$
$c_1 \cdot g(n)$
↑ time
$n \rightarrow$

FOLLOW FOR MORE



FOLLOW ME