

---

**Main Title**

**Class subtitle**

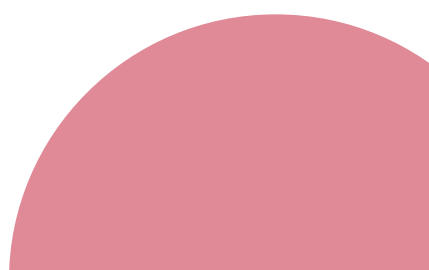
07/09/2025 - 18/09/2025

---

**Tom Planche**

2025-2026

*Class name*



## Table des matières

I - Main title .....	3
I. I - Maths .....	3
I. I. I - #definition .....	3
I. I. II - #example .....	3
I. I. III - #theorem .....	3
I. I. III. 1 - With title .....	3
I. I. III. 2 - Without title .....	3
I. I. IV - ar .....	4
I. II - Subtitle .....	4
I. II. I - Subsubtitle .....	4

# I - Main title

## I. I - Maths

For my maths class, I made these things:

### I. I .I - #definition

**Definition 1.1.** (Linéarité):

On dit que  $\varphi$  est linéaire (homomorphisme) si:

$$\varphi(\lambda_1 X_1 + \lambda_2 X_2 + \dots + \lambda_n X_n) = \lambda_1 \varphi(X_1) + \lambda_2 \varphi(X_2) + \dots + \lambda_n \varphi(X_n) \quad (1.1.1.1)$$

### I. I .II - #example

**Example 1.1.** (Example title): Basic text.

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua quaerat.

$$\begin{aligned} \varphi(0, 0, 0) &= (0, 0) = 0_{\mathbb{R}^2} \\ \varphi(\alpha X_1 + \beta X_2) &\stackrel{?}{=} \alpha \varphi(X_1) + \beta \varphi(X_2) \end{aligned} \quad (1.1.2.2)$$

### I. I .III - #theorem

#### I. I. III. 1 - With title

**Theorem 1.1.** (Théorème de Stokes):

Soit  $M$  une variété différentielle à bord, orientée de dimension  $n$ , et  $\omega$  une  $(n-1)$ -forme différentielle à support compact sur  $M$  de classe  $C_1$ .

Alors, on a :

$$\int_M d\omega = \int_{\{\partial M\}} i^* \omega \quad (1.1.3.3)$$

où  $d$  désigne la dérivée extérieure,  $\partial M$  le bord de  $M$ , muni de l'orientation induite, et  $i^* \omega = \omega|_{\{\partial M\}}$  la restriction de  $\omega$  à  $\partial M$ .

#### I. I. III. 2 - Without title

**Theorem 1.2.**

Soit  $E$  un espace vectoriel de dimension finie,  $F$  un sous-espace vectoriel de  $E$  et  $B = (X_1, X_2, \dots, X_n)$  une base de  $F$ .

Alors, il existe une base  $(X_1, X_2, \dots, X_n, X_{n+1}, \dots, X_m)$  de  $E$  telle que  $(X_1, X_2, \dots, X_n)$  soit une base de  $F$ .

I. I .IV - ar

For vectors, I use ar(X) and it gives  $\vec{X}$ .

I. II - Subtitle

I. II .I - Subsubtitle

Custom Block

Custom Blockquote

Basic inline raw text

This code block uses #code() macro.

```
src/string_utils.rs
1 /// Extension traits and utilities for string manipulation
2 ///
3 /// This module provides additional functionality for working with strings,
4 /// including title case conversion and other string transformations.
5 use std::string::String;
6
7 /// Trait that adds title case functionality to String and &str types
8 pub trait TitleCase {
9     /// Converts the string to title case where each word starts with an uppercase letter
10    /// and the rest are lowercase
11    ///
12    fn to_title_case(&self) -> String;
13 }
14
15 impl TitleCase for str {
16     fn to_title_case(&self) -> String {
17         self.split(|c: char| c.is_whitespace() || c == '_' || c == '-')
18             .filter(|s| !s.is_empty())
19             .map(|word| {
20                 // If the word is all uppercase and longer than 1 character, preserve it
21                 if word.chars().all(|c| c.is_uppercase()) && word.len() > 1 {
22                     word.to_string()
23                 } else {
24                     let mut chars = word.chars();
25                     match chars.next() {
26                         None => String::new(),
27                         Some(first) => {
28                             let first_upper = first.to_uppercase().collect::<String>();
29                             let rest_lower = chars.as_str().to_lowercase();
30                             format!("{}", first_upper, rest_lower)
31                         }
32                     }
33                 }
34             })
35         .collect::<Vec<String>>()
36         .join(" ")
37     }
38 }
```

```
37     }
38 }
39
40 impl TitleCase for String {
41     fn to_title_case(&self) → String {
42         self.as_str().to_title_case()
43     }
44 }
45
46 #[cfg(test)]
47 mod tests {
48     use super::*;
49
50     #[test]
51     fn test_title_case_str() {
52         assert_eq!("hello world".to_title_case(), "Hello World");
53         assert_eq!("HASH_TABLE".to_title_case(), "HASH TABLE");
54         assert_eq!("dynamic-programming".to_title_case(), "Dynamic Programming");
55         assert_eq!("BFS".to_title_case(), "BFS");
56         assert_eq!("two-sum".to_title_case(), "Two Sum");
57         assert_eq!("binary_search_tree".to_title_case(), "Binary Search Tree");
58         assert_eq!(" spaced words ".to_title_case(), "Spaced Words");
59         assert_eq!("".to_title_case(), "");
60     }
61 }
```