

PoC 2025-2026

## *Preuve de concept*

## *Présentation du module*

Gregory.faraut@ens-paris-saclay.fr

**Gregory Faraut**

kevin.godineau@ens-paris-saclay.fr

**Kevin Godineau**

# Intervenants

## Gregory Faraut

- Professeur des Universités, ENS Paris-Saclay
- Thématique de recherche : Systèmes à événements discrets et systèmes dynamiques hybrides



## Kevin Godineau (*Responsable du module*)

- Maître de Conférences, ENS Paris-Saclay
- Thématique de recherche : Pilotage des machines de fabrication additive LPBF



## Sylvain Lavernhe

- Professeur des Universités, Université Paris-Saclay
- Thématique de recherche : Amélioration des procédés additifs et soustractifs



# Objectifs

## 1. Améliorer vos compétences en programmation (Python et Matlab)

- Reproduire des travaux décrits dans la littérature scientifique

## 2. Rendre compte de vos développements scientifiques

- Analyser de manière pertinente les développements effectués :
  - Au travers de cas d'études judicieusement choisi
  - Au travers de critères (complexité algorithmique, complexité en temps ...)
- Fournir un travail reproductible

## 3. Maîtriser des outils de transmission

- Maîtriser les outils adaptés (à la présentation, à la diffusion, à la reproductibilité) des développements informatiques effectués

Il ne s'agit pas de Génie Logiciel mais plutôt de bonnes pratiques scientifiques associées au développement logiciel

# Déroulement du module (1/2)

## Première partie – mise en place des outils, des méthodes

- Séance 1 : 1h à 2h – GF et KG
  - Présentation du module, des objectifs, des outils et des ressources
  - Présentation des sujets et précisions sur le travail attendu et les modalités de l'évaluation du projet
- Séance 2 et 3 : 4h – GF et KG
  - CSC : Exercices d'applications (Python)
  - IN2P : Exercices d'applications (Matlab)
- Séance 4 : 4h – GF et SL
  - CSC: Programmation Orientée Objet, import, traitement et affichage (Python)
  - IN2P : Manipulation de tenseurs (matrice n-dimensions), calcul symbolique (Matlab)

Travail intermédiaire :  
maîtrise d'outil

# Déroulement du module (2/2)

## Deuxième partie – réalisation d'une preuve de concept

- Séance 5 : 2h – GF et KG
  - Évaluation du projet à mi-parcours
  - Précisions sur le travail attendu et les modalités de l'évaluation du projet
- Séance 6 et 7 : Suivi de projet - 4h – GF et KG
  - Vous nous présentez votre travail de manière informelle, vos difficultés, votre avancement
  - Nous sommes disponibles pour répondre à vos questions
- Séance 8 : Évaluation
  - Rendu sous GitHub de votre projet
  - Code commenté et fonctionnel
  - Rapport écrit en Markdown qui explicite vos travaux
  - Soutenance orale

# L'évaluation

## Suivi de projet - 40 %

- Avancement du projet entre maintenant et la date de rendu
- Développement cohérent et en parallèle du code, des commentaires, des tests et du rapport explicatif.

## Rendu écrit - 30 %

- Clarté du code déposé
- Pertinence du rapport

## Présentation orale - 30 %

- Soutenance
- Réponse aux questions

# Plan

1. Introduction
2. Qu'est-ce que l'on entend par « preuve de concept »
3. Compétences attendues à l'issue du module
4. Présentation des outils utilisés
5. Présentation des ressources
6. Un premier travail !

# Plan

1. Introduction
2. Qu'est-ce que l'on entend par « preuve de concept »
3. Compétences attendues à l'issue du module
4. Présentation des outils utilisés
5. Présentation des ressources
6. Un premier travail !

# Description générale

« La preuve (de concept) semble être une preuve de possibilité dont la réalisation est démontrée dans une pratique expérimentale. Elle prouve que le lien de causalité hypothétique, la structure proposée, la fonction suggérée ou l'approche méthodologique adoptée dans la recherche est vérifiée dans au moins un cas réel - le cas test » [Kendig, 2016]

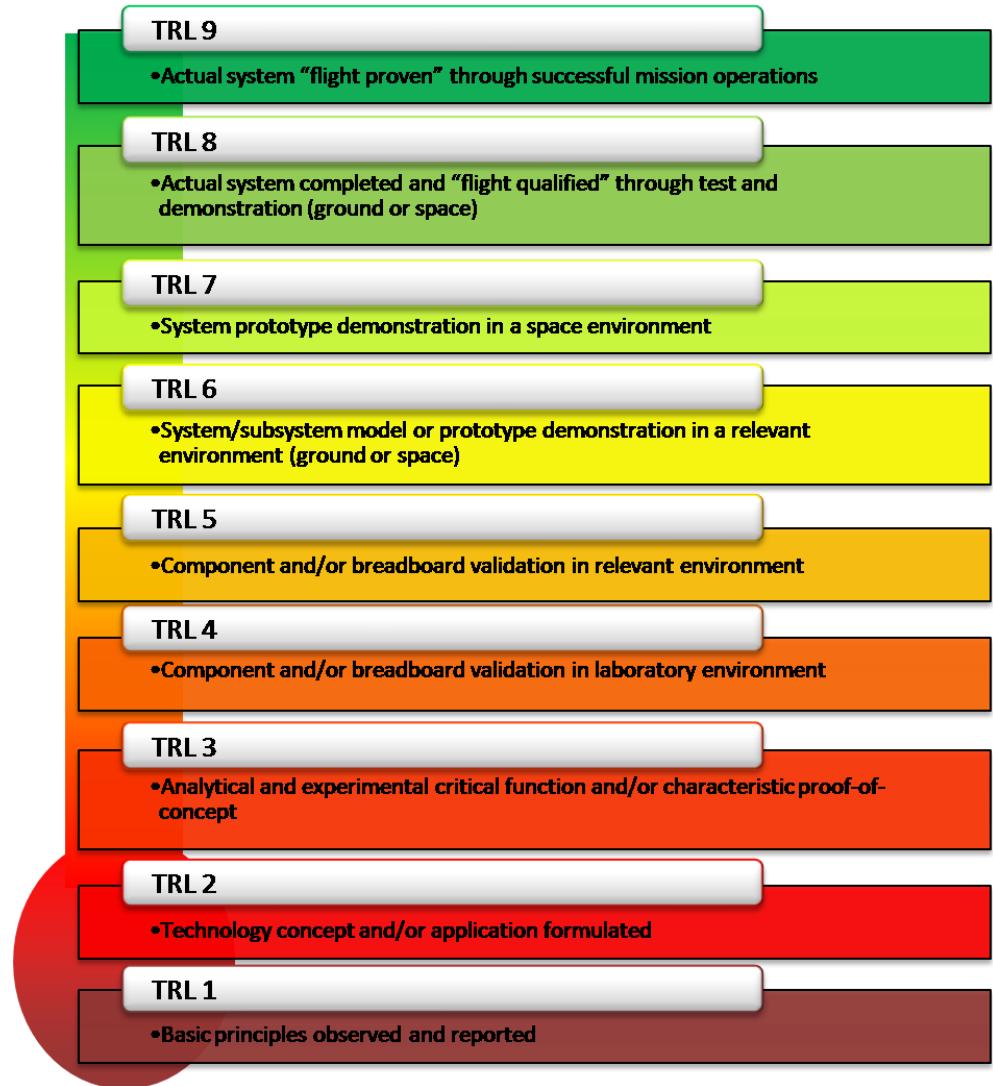
[Kendig, 2016] Kendig, Catherine Elizabeth, 2016. What is Proof of Concept Research and how does it Generate Epistemic and Ethical Categories for Future Scientific Practice? *Science and Engineering Ethics*. June 2016. Vol. 22, no. 3, p. 735–753. doi : [10.1007/s11948-015-9654-0](https://doi.org/10.1007/s11948-015-9654-0)

# Appliquée à nos domaines

En science de l'ingénieur, avec un regard industriel on parle de : notion de TRL (Technology Readiness Level) ou Niveau de maturité technologique

- La preuve de concept correspond ici au TRL 3
- Notion centrée sur le développement d'un « produit, système, procédé »

À l'échelle d'un laboratoire de recherche on ne dépasse que rarement le TRL 4



# Pourquoi enseigner cela

Aujourd'hui, tous les scientifiques doivent développer des modèles logiciels pour valider les résultats de leurs recherches (Proof of Concept). Il s'agit de :

- Modéliser et coder un algorithme issu de vos travaux de recherche et déterminer sa classe de complexité temporelle et spatiale.
- Collecter, générer, stocker et formater des ensembles de données à traiter par l'algorithme
- Faire traiter les ensembles de données par votre algorithme
- Évaluer la qualité des résultats produits par l'algorithme à l'aide d'une évaluation clairement définie.

C'est dans ce contexte informatique que le module d'enseignement est proposé.

- Pour mettre en pratique ces techniques et outils, chaque étudiant travaillera sur une étude de cas. L'étude de cas sera un projet de recherche déjà publié, dont les résultats devront être reproduits, documentés et éventuellement étendus.
- Reproduire un travail existant, c'est à la fois le valider et en comprendre le contenu.

# Plan

1. Introduction
2. Qu'est-ce que l'on entend par « preuve de concept »
- 3. Compétences attendues à l'issue du module**
4. Présentation des outils utilisés
5. Présentation des ressources
6. Un premier travail !

# Les compétences attendues

## À l'issue de la séance 4 (fin octobre)

- Maitrise d'outil et d'un langage de programmation

## À l'issue du module (mi-décembre/début janvier)

- Implémenter, valider et tester un algorithme issu d'un article scientifique
- Mettre en place des jeux de données pertinents
- Valider tester l'algorithme sur ces jeux de données
- Evaluer les performances au travers de critères
- Représenter les résultats
- Analyser, expliciter et retranscrire cela dans un rapport explicitant le code, les développements

# Exemple simple de ce que l'on attend

Présentation de deux exemples disponibles sur GitHub

[\(lien 1\)](#)

- Exemple sur un cas d'application simple scientifiquement.
- Travail fait par Bruno Denis à titre d'exemple
- Sujet : intersection entre un plan et un maillage au format .stl

# Deux langages de programmation pourquoi !

Le master 2 AMSS possède 2 parcours IN2P et CSC. Dans chacun des domaines scientifiques, historiquement des langages différents ont été utilisés :

- **Matlab (IN2P)**
  - Manipulation de matrice
  - Représentation et affichage
  - Intégration avec Simulink (commande des systèmes continu)
- **Python (CSC)**
  - Programmation orientée objet
  - Un langage de programmation open-source
  - Une communauté de développeur conséquente

Pour les IN2P, le projet peut être fait en Python si l'étudiant maîtrise vraiment ce langage de programmation.

# Plan

1. Introduction
2. Qu'est-ce que l'on entend par « preuve de concept »
3. Compétences attendues à l'issue du module
- 4. Présentation des outils utilisés**
5. Présentation des ressources
6. Un premier travail !

# Présentation générale

## Langages informatiques

- Programmation
  - Matlab
  - Python



- Documentation
  - Markdown
  - Latex math



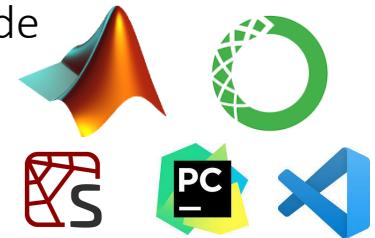
## Système de gestion de version

- Git



## Logiciels informatiques

- Environnement de développement
  - Matlab
  - Anaconda

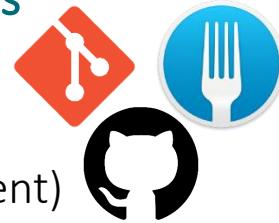


- Editeurs
  - Marktext
  - Or online editor



## Outils de gestion de versions

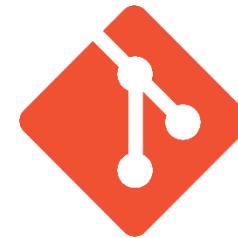
- Git (client mode texte)
- Fork (client graphique)
- Github.com (hébergement)



# Git (1/2)

## Description

- Logiciel de gestion de versions collaboratif.



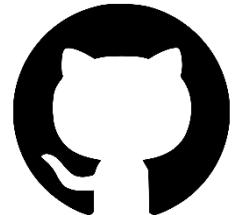
## Interface graphique Fork :

- Une interface graphique pour utiliser Git à la souris. Une parmi beaucoup d'autre
- Avantage : compatibilité Mac et Windows + accès gratuit aux répertoires privés sur GitHub



## Partage et documentation des données via GitHub

- Plateforme web d'hébergement et de gestion de développement de logiciels, utilisant le logiciel de gestion de versions Git.
- Plus importante plateforme hébergeur de code informatique
- On aurait souhaité utiliser le Gitlab Université Paris-Saclay mais Celui-ci ne fonctionne plus suite au piratage de l'Université.



# Git (2/2)

## Pourquoi utiliser Git

- Pour éviter les V1, V2, Vfinale, Vfinalefinale
- Car vous sauvegarder une arborescence et pas uniquement la trace à l'instant t
- Outil utilisé internationalement dans le développement (logiciel, jeux vidéo ...)

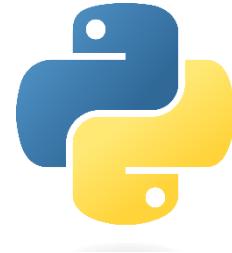
## Présentation d'un exemple

## Ressources ([lien](#))

## Pour aller plus loin : [GitFlow \(lien\)](#)

- Méthodologie permettant de structurer des projets proprement

# Eco-system Python (1/4)



## Programming language

Python is a high-level, general-purpose programming language (v3.x)

## Language implementation

C<sup>Python</sup> the reference implementation written in C and Python

I<sup>Python</sup>, which stands for Interactive Python (notebook interface, interactive data...)

## Libraries

Standard modules such as math, random, statistics, sys, ...

Third-party modules such as NumPy, matplotlib, SciPy, Graphviz, TensorFlow, ...

## Package manager

Pip, Conda

## Environment management system

venv, Conda

## Notebook

Jupyter

## Integrated Development Environment (IDE)

Syder, PyCharm, VsCode, ...



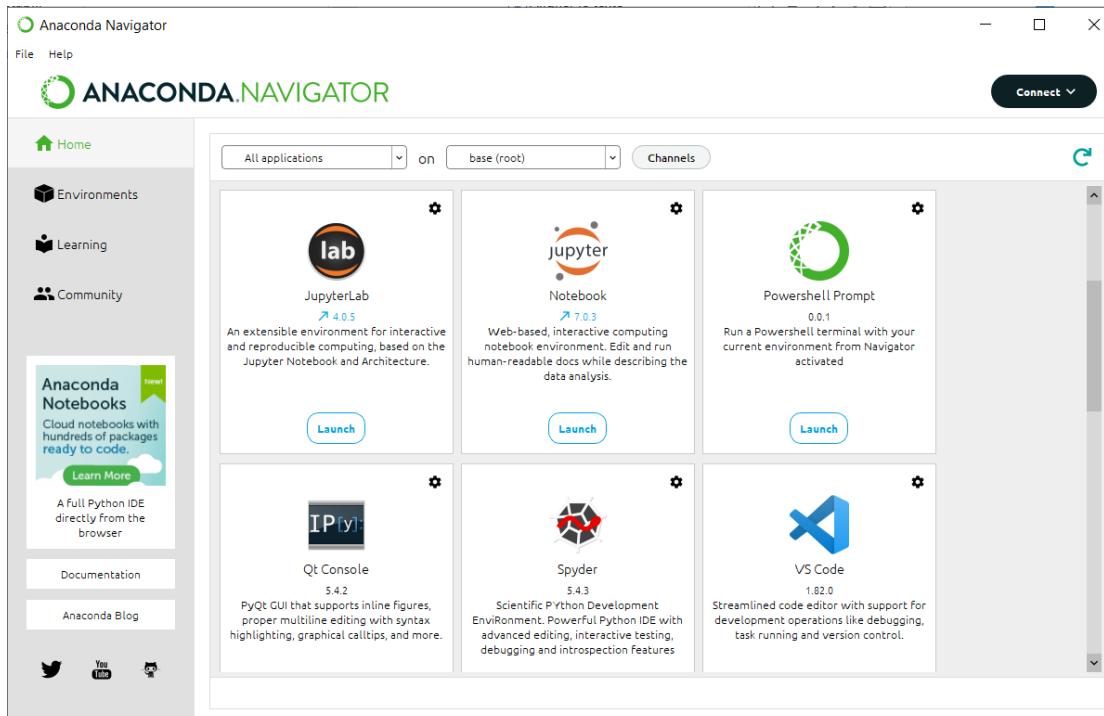


# Eco-system Python (2/4)

## Anaconda

Anaconda is a distribution of the Python including : CPython, Ipython, Standard modules, many modules for scientific computing, IDE (Sypder, Vscode, Pycharm), Conda, Jupyter Notebook

- Download from depuis <https://www.anaconda.com/products/distribution>
- Choose Python 3.X version



## Eco-system Python (3/4)



# IDE example: Spyder

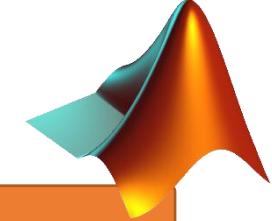
The screenshot shows the Spyder IDE interface with several windows open:

- File Explorer:** Shows the file structure with files like App.py, template.py, plot\_example.py, and plugin.py.
- Code Editor:** The plot\_example.py script is open, containing code to generate a terrain model and a polar plot.
- Variable Explorer:** A table showing variables and their values, including data (ndarray), df (DataFrame), filename (str), i (array), li (list), r (float), t (tuple), and tinylist (list).
- Plots:** Two plots are displayed: a 3D surface plot of a terrain model and a polar plot.
- IPython console:** An IPython console window is visible at the bottom.



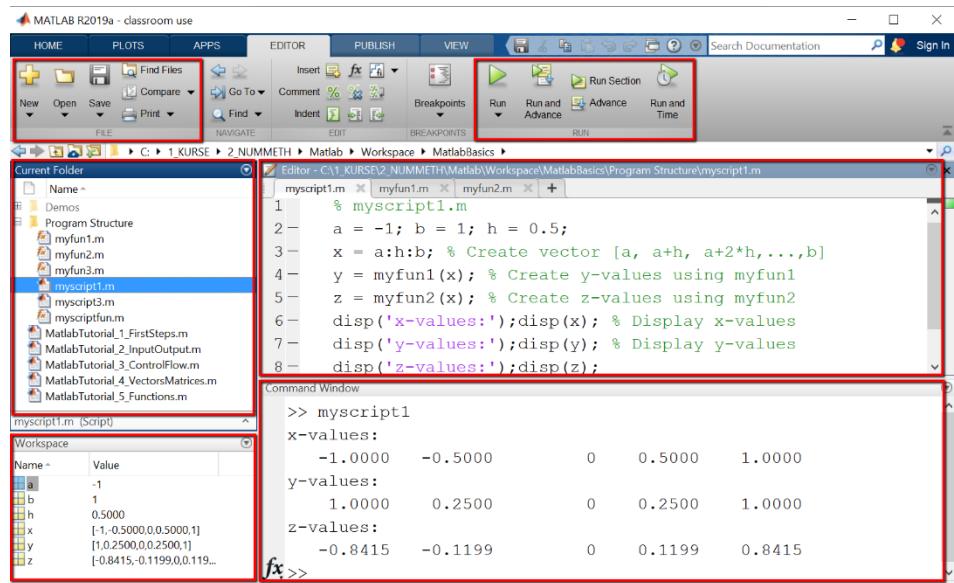
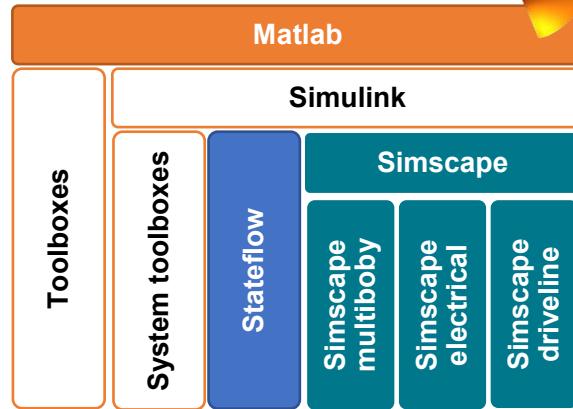
## Installation guide

- Download Anaconda distribution from  
<https://www.anaconda.com/products/distribution>
- Select « Python 3.X » version at the installation beginning



## Présentation rapide de l'environnement

- Matlab est à la fois un langage de programmation et un environnement intégré de développement (IDE)
- L'environnement de Matlab intègre
  - Un **éditeur** de code Matlab
  - Un **interpréteur** de code pour exécuter
  - Un **éditeur graphique** Simulink
  - De nombreuses bibliothèques appelées « **toolbox** »



# Markdown language (1/8)



## What is Markdown?

Markdown is a lightweight markup language that you can use to add formatting elements to plaintext text documents. Created by [John Gruber](#) in 2004, Markdown is now one of the world's most popular markup languages.

Depending on the application you use, you may not be able to preview the formatted document in real time. But that's okay. According to Gruber, Markdown syntax is designed to be readable and unobtrusive, so the text in Markdown files can be read even if it isn't rendered.

For example, to denote a heading, you add a number sign before it

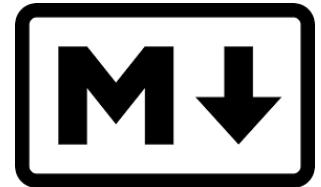
**# Heading One**

Or to make a phrase bold, you add two asterisks before and after it

**\*\*this text is bold\*\*.**

(extract from [www.markdownguide.org](http://www.markdownguide.org))

# Markdown language (2/8)



## Why use it ?

Markdown is used in [GitHub.com](https://GitHub.com) and [GitLab.com](https://GitLab.com) and [Paris-Saclay GitLab](https://Paris-Saclay.GitLab) websites and in [Jupyter](#), [Google Colab](#), [Microsoft Azure](#), [Apache Zeppelin](#) and [Deepnote](#) notebooks. More over it is platform independent and portable.

## Editors and viewers

Online Markdown editors

- Dillinger <https://dillinger.io/>
- StackEdit <https://stackedit.io/>

Open source Software

- MarkText <https://www.marktext.cc/>
- ghostwriter <https://kde.github.io/ghostwriter/>

Extensions available of VScode

# Markdown language (3/8)



## Quick reference guide

Markdown	Preview
<code>**bold text**</code>	<b>bold text</b>
<code>*italicized text*</code> or <code>_italicized text_</code>	<i>italicized text</i>
<code>`Monospace`</code>	Monospace
<code>~~strikethrough~~</code>	<del>strikethrough</del>
<code>[A link](https://www.google.com)</code>	A link
<code>![An image](https://www.google.com/images/rss.png)</code>	

# Markdown language (4/8)



Headings are rendered as titles

```
# Section 1
# Section 2
## Sub-section under Section 2
### Sub-section under the sub-section under Section 2
# Section 3
```

## Section 1

---

## Section 2

---

### Sub-section under Section 2

---

#### Sub-section under the sub-section under Section 2

---

## Section 3

---

# Markdown language (5/8)



## Block code

```
```python  
print("a")  
```
```

```
print("a")
```

## Lists

Ordered lists:

```
1. One  
1. Two  
1. Three
```

1. One  
2. Two  
3. Three

Unordered lists:

```
* One  
* Two  
* Three
```

• One  
• Two  
• Three

# Markdown language (6/8)



## Tables

| First column name | Second column name |
|-------------------|--------------------|
| Row 1, Col 1      | Row 1, Col 2       |
| Row 2, Col 1      | Row 2, Col 2       |

| First column name | Second column name |
|-------------------|--------------------|
| Row 1, Col 1      | Row 1, Col 2       |
| Row 2, Col 1      | Row 2, Col 2       |

## Inline equations

$$y=x^2$$

$$e^{i\pi} + 1 = 0$$

$$e^x = \sum_{i=0}^{\infty} \frac{1}{i!} x^i$$

$$\frac{n!}{k!(n-k)!} = {n \choose k}$$

$$y = x^2$$

$$e^{i\pi} + 1 = 0$$

$$e^x = \sum_{i=0}^{\infty} \frac{1}{i!} x^i$$

$$\frac{n!}{k!(n-k)!} = {n \choose k}$$

LATEX

# Markdown language (7/8)



## Equation blocks

```
$$  
y=x^2  
$$  
  
$$  
e^{i\pi} + 1 = 0  
$$  
  
$$  
e^x=\sum_{i=0}^{\infty} \frac{1}{i!}x^i  
$$  
  
$$  
\frac{n!}{k!(n-k)!} = {n \choose k}  
$$  
  
$$  
A_{m,n} =  
\begin{pmatrix}  
a_{1,1} & a_{1,2} & \cdots & a_{1,n} \\  
a_{2,1} & a_{2,2} & \cdots & a_{2,n} \\  
\vdots & \vdots & \ddots & \vdots \\  
a_{m,1} & a_{m,2} & \cdots & a_{m,n}  
\end{pmatrix}  
$$
```

$$y = x^2$$

$$e^{i\pi} + 1 = 0$$

$$e^x = \sum_{i=0}^{\infty} \frac{1}{i!}x^i$$

$$\frac{n!}{k!(n-k)!} = \binom{n}{k}$$

$$A_{m,n} = \begin{pmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,n} \\ a_{2,1} & a_{2,2} & \cdots & a_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m,1} & a_{m,2} & \cdots & a_{m,n} \end{pmatrix}$$

# Markdown language (8/8)



## Example avec Github view

### Intersection plan-line

Notations

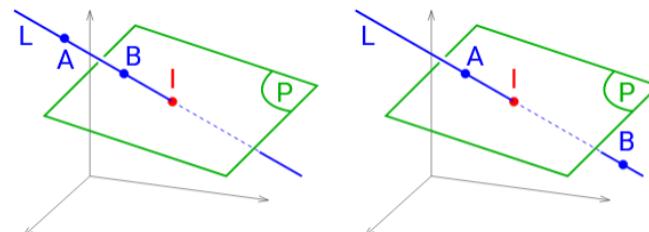
- $A(x_A, y_A, z_A)$  and  $B(x_B, y_B, z_B)$  are two points
- $L = \overleftrightarrow{AB}$  is a line defined by points A and B
- $P$  is a plan defined by the following equation  $M(x, y, z) \in P$  iff  $ax + by + cz = d$
- $I(x_I, y_I, z_I)$  is the point intersection common to line  $L$  and plan  $P$
- $S = \overline{AB}$  is a line segment bounded by points A and B

![Fig](img/plan\_segment.svg)

### Intersection plan-line

Notations

- $A(x_A, y_A, z_A)$  and  $B(x_B, y_B, z_B)$  are two points
- $L = \overleftrightarrow{AB}$  is a line defined by points A and B
- $P$  is a plan defined by the following equation  $M(x, y, z) \in P$  iff  $ax + by + cz = d$
- $I(x_I, y_I, z_I)$  is the point intersection common to line  $L$  and plan  $P$
- $S = \overline{AB}$  is a line segment bounded by points A and B



Preuve de concept



# Jupyter notebook (1/2)

## What is a notebook ?

It is a documents containing

- Code (Python, Matlab, ...)
- Equations (Markdown)
- Visualizations (results of code execution)
- Narrative text (Markdown)

## What is a Jupyter ?

The Jupyter Notebook is an open-source web application for creating and sharing a notebook

The screenshot shows a Jupyter Notebook interface with the title "jupyter spectrogram (autosaved)". The menu bar includes File, Edit, View, Insert, Cell, Kernel, and Help. Below the menu is a toolbar with icons for file operations like save, new, and delete, and navigation buttons. The main area contains a section titled "Simple spectral analysis" with the subtitle "An illustration of the [Discrete Fourier Transform](#)". A mathematical formula for the Discrete Fourier Transform is displayed: 
$$X_k = \sum_{n=0}^{N-1} x_n \exp \frac{-2\pi i}{N} kn \quad k = 0, \dots, N-1$$
. Below this is a code cell labeled "In [2]:" containing Python code to read a wav file: 

```
from scipy.io import wavfile  
rate, x = wavfile.read('test_mono.wav')
```

. Another code cell labeled "In [5]:" shows code to plot the raw audio signal and its spectrogram: 

```
fig, (ax1, ax2) = plt.subplots(1,2,figsize=(16,5))  
ax1.plot(x); ax1.set_title('Raw audio signal')  
ax2.specgram(x); ax2.set_title('Spectrogram');
```

. At the bottom is a plot titled "Raw audio signal" showing a blue line representing the audio waveform over time.



# Jupyter notebook (2/2)

## Installation guide

Jupyter note book is included in Anaconda distribution

- To use Jupyter with Python : no extra installation after Anaconda installation
- To use Jupiter with Matlab
  - Matlab use its own notebook format (included with Matlab installation) but it can not be render by Github.com (proprietary format)
  - Access Matlab directly from Jupyter install a dedicated Python module  
`> python3 -m pip install jupyter-matlab-proxy`  
(see <https://www.mathworks.com/products/reference-architectures/jupyter.html>)

# Plan

1. Introduction
2. Qu'est-ce que l'on entend par « preuve de concept »
3. Compétences attendues à l'issue du module
4. Présentation des outils utilisés
- 5. Présentation des ressources**
6. Un premier travail !

# Ressources à disposition

Toutes les informations seront disponibles sur eCampus ([lien](#))

- Slides
- Liens pour les formations Python, Matlab, Git
- Exercices d'application et solutions
  - Séance 2, 3 et 4
  - Exemple type d'évaluation

## Exemple de projet sur GitHub

- Projet présenté par Bruno Denis et étudié lors de la séance 4
- Projet en libre accès dès la fin de la séance 4

# Formation Python

## Installation guide

- Download Anaconda distribution from  
<https://www.anaconda.com/products/distribution>
- Select « Python 3.X » version at the installation beginning

## Resources

- Tutorials for beginners
  - <https://docs.python.org/3/tutorial/index.html> (free access, many languages)
  - <https://www.learnpython.org/>
    - For Numpy → ([https://www.learnpython.org/en/Numpy\\_Arrays](https://www.learnpython.org/en/Numpy_Arrays))

# Formation Matlab

## Installation

- Matlab version R2023a
- Attention l'installation de Matlab nécessite d'avoir une adresse mail @ens-paris-saclay.fr et donc de s'être inscrit.

## Ressources à disposition

- Formation en ligne (faut un compte matworks et donc une adresse mail ens)
  - <https://matlabacademy.mathworks.com/fr/>
  - [MATLAB Onramp](#), [MATLAB Fundamentals](#), [MATLAB for Data Processing and Visualization](#)
  - Pour aller plus loin [MATLAB Programming Techniques](#)
- Autres
  - Document sur les bonnes pratiques
  - Fichier .mlx présentant quelques bases
  - Site matlab.developpez.com ([lien 1](#)) ([lien 2](#))
  - Matlab Centrale
  - Chaine YouTube de Matlab

# Formation Git

## Installation

- Installer la version d'évaluation de Fork qui est gratuite et à durée illimitée.
- Cette version permet d'avoir accès à un dépôt externe privé gratuitement contrairement à bon nombre d'interfaces graphique Git.

## Ressources à disposition

- Formation git (en ligne de commande) : <https://grafikart.fr/formations/git>
- Documentation git <https://git-scm.com/book/en/v2>

# Plan

1. Introduction
2. Qu'est-ce que l'on entend par « preuve de concept »
3. Compétences attendues à l'issue du module
4. Présentation des outils utilisés
5. Présentation des ressources
6. Un premier travail !

# Un premier travail !

## Créer un Dépôt Github contenant

- Un Readme.md décrivant le contenu du dépôt
- Un fichier .ipynb permettant de présenter les travaux effectués
- Une structure de dossier propre

## Objectif

- Réalisé un code permettant d'importer des données d'un fichier .txt, de calculer le plan des moindres carrée associé et d'afficher tous cela.

## Nous partager le dépôt GitHub en lecture (avant le 17/10/25 – 22h)

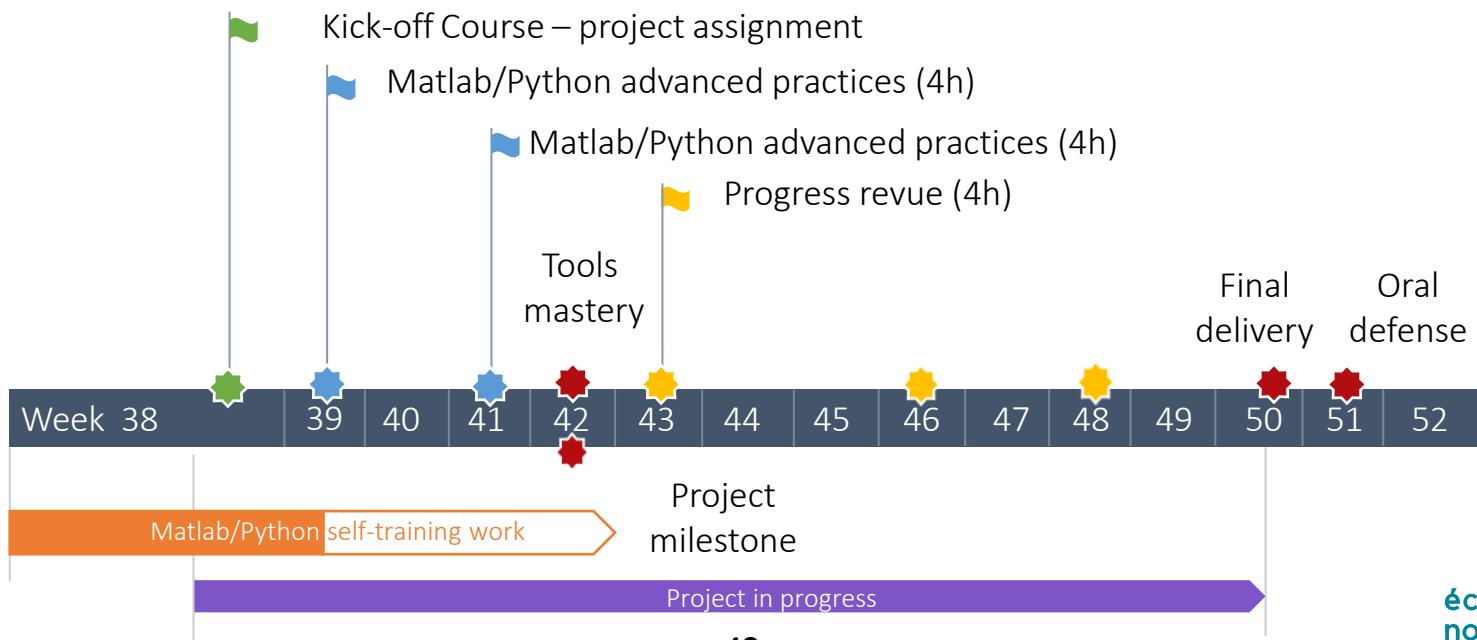
- Identifiant GitHub : kevin-godineau

## Exemple (lien privé ne marche que lors de la présentation)

# Conclusion

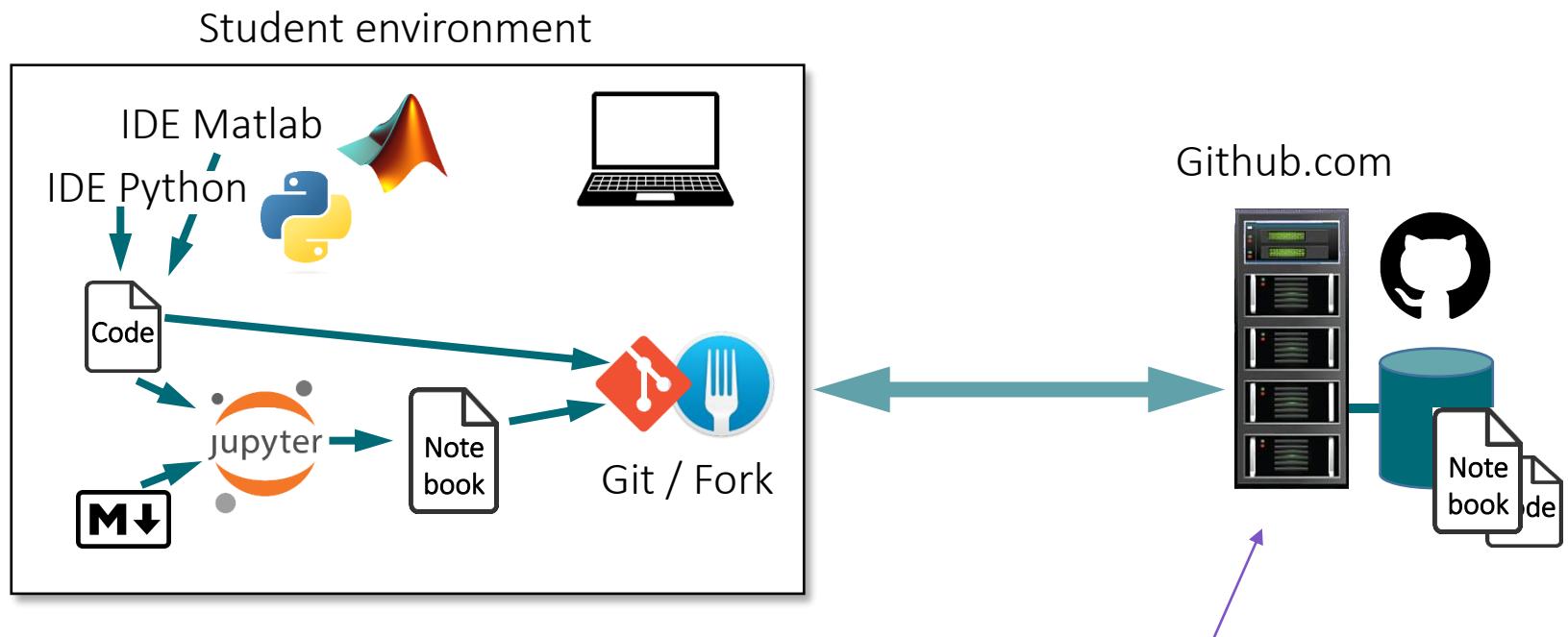
## Schedule

- Kick-off teaching module - Presentation and project assignment
- Matlab/Python advanced practices
- **Tools mastery** (before 17/10/25 – 22h)
- Progress review session
- **Project milestone**
- **Project delivery**
- **Project oral defense**



# Conclusion

## IT organization overview



Il faut donc que vous  
créer un compte GitHub

# Pourquoi tout cela !



## Objectif :

- Ce n'est pas d'apprendre à coder ! (Pour cela vous êtes assez grand pour vous auto-former)
- C'est apprendre à présenter le contenu d'un développement scientifique logiciel et à rendre compte intelligiblement de ces performances.

