Dans le premier atelier pratique enseigner par le professeur M. Abdel Bacet Mhamdi, on a commencé à débuter l’environnement pluto.jl et le language de programmation Julia, **par définition :** Pluto est une interface de bloc-notes basée sur un navigateur pour le langage Julia. Il permet de présenter le code Julia et les résultats de calcul de manière étroitement liée.

On a testé d’abord quelque instruction de base dans le language de programmation Julia (appelation de language, mise à jour des packs installé], exécution des fonctions, méthode help ? shell ! …). Pour être précis, Julia est un langage de programmation dynamique de haut niveau, conçu pour donner aux utilisateurs la vitesse de C./C++ tout en restant aussi facile à utiliser que Python. Cela signifie que les développeurs peuvent résoudre les problèmes plus rapidement et plus efficacement. On a appris dans le cours que julia ce caractérise comme un language **dynamique** est **composable** et qui permet la création de solutions informatiques hautement performantes. Pour cela, elle repose sur des techniques de calcul numériques et statistiques ainsi que sur une syntaxe très simple.

Pour l’environnement pluto.jl, on a commencé par des exercices de compréhension de l’interface toute en commençant par la création des titre en forme précise puis a l’ajout et le traitement de texte jusqu’à la résolution d’un petit problème (pizza) mathématique qui sert a ajouter un slider qui définit une quantité et de l’afficher en message, en conclusion c’est toute un problème successive d’instruction.

Ce que j’ai appris de pluto.jl c’est qu’il est un environnement **réactif** car lors de la modification d'une fonction ou d'une variable, Pluto met automatiquement à jour toutes les cellules concernées, **léger** car Pluto est écrit en pur Julia et est facile à installer.

Le plus important pour moi c’est qu’un cahier Pluto est composé de petits blocs de code Julia (cellules) et ensemble, ils forment un cahier **réactif**. Lorsque on modifie une variable, Pluto ré exécute automatiquement les cellules qui s'y réfèrent. Les cellules peuvent même être placées dans un ordre arbitraire – **par** **définition** : une analyse syntaxique intelligente détermine les dépendances entre elles et prend en charge l'exécution.