Jupyter Notebook Extensions for Teaching and Learning

By: Andrea Mitchell, Jenny Sun, Derek Huang

CISC 499 Undergraduate Research Project

Final Report

By: Derek Huang

Content:

Part I. Jupyter Notebook Software Requirement Specification

Abstract

Jupyter Notebook is an open-source web application that allows users to write notebooks which support a mixture of runnable code, markup text, multimedia, graphs and more. In this report, we will be discussing all aspects of Jupyter Notebook which are fundamental to the understanding of the system which we are developing improvements for. We will first describe an overview of the system, discussing the purpose, scope, stakeholders, as well as a context diagram. We will furthermore discuss the Architecture of the software from the perspective of a development view. Finally, we will discuss our improvements and in what way they will benefit the user experience.

Introduction

Jupyter Notebook is a web-based interactive development environment for Jupyter notebooks, code, and data. Jupyter is flexible: configure and arrange the user interface to support a wide range of workflows in data science, scientific computing, and machine learning. Jupyter is extensible and modular: write plugins that add new components and integrate with existing ones. (Jupyter, https://jupyter.org/)

Jupyter Notebook is a great platform for university students to use as a tool for learning and programming purposes in computing science courses involving programming. In this paper, we will be analyzing Jupyter Notebook's software architecture, including its requirements and specifications.

Purpose

The major purpose of Jupyter Notebooks is to provide a web platform for users to code and visualize their data in an online learning environment. Instead of building the complex and heavyweight software systems, users can build small programs based on this platform. In addition, users can create and build their own features based on the requirements of their needs. As an open-source platform, users can choose other plugin or extension features they need in the community. Once the extension is settled, people all around the world will be able to access it.

Scope

The notebooks may be used for various tasks. They provide a means to write code with a quick and convenient compilation and output loop. The code can be mixed with diagrams, graphs, multimedia, explanatory text, and mathematical equations. It can be used as a tool in data science, machine learning, statistics, numerical simulation and more. Therefore, the user base is broad, ranging from high level tech companies, to high school students learning to code.

Stakeholders

Stakeholders are a person, group or entity with an interest in or concern about the realization of the architecture with various expectations and needs. Jupyter Notebook is open source and is free to download and install for any kind of use.

There are many stakeholders involved in this project. The most important stakeholders in Jupyter Notebooks are demonstrated below:

Developers
Construct and deploy Jupyter Notebooks according to specifications or lead the teams that do this. In particular, extension developers are part of this role, as they are responsible for developing certain extensions or plugins for the Notebooks.

Maintainers
Manage the evolution of Jupyter Notebook once it is operational. Jupyter Notebook is maintained by the core developers. They make sure that there are enough developers for the Jupyter Notebook project and invest money from Jupyter Notebook partner services worldwide in making sure the Jupyter Notebook project stays alive.

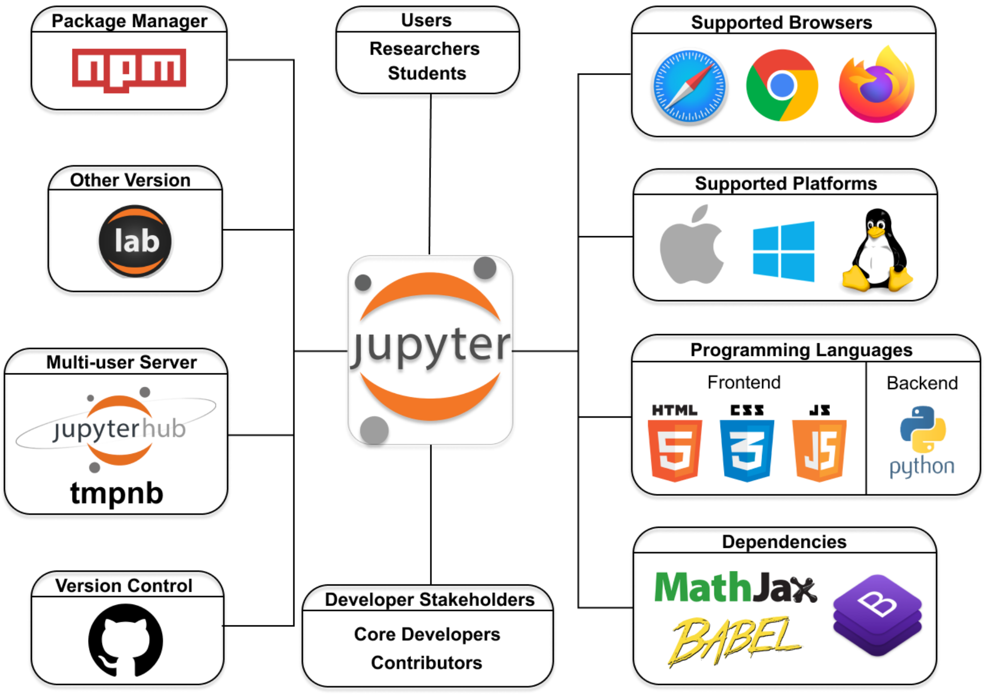
Testers
Test Jupyter Notebook to ensure it is suitable to use. The core team involved is the software quality assurance team. Testers look at all the issues in the testing queue, trying each fix and feature to make sure that it does fix the problem it was supposed to, and that there are no regressions each week. Users can also report bugs to the issue tracker in Jupyter Notebook.

Students
Consists mainly of students who need to attend online courses, finish coding programs, analyze and visualize training data in an online environment.

Site Admins
Consists mainly of site administrators who need to change site appearance, manage accounts and permissions and check performance.

Context View

The context view is a description of the interactions that the software has with its environment. It includes the relationship with stakeholders, dependencies with systems, and anything else the software interacts with. In this section, we show the context view of Jupyter Notebooks as a diagram.

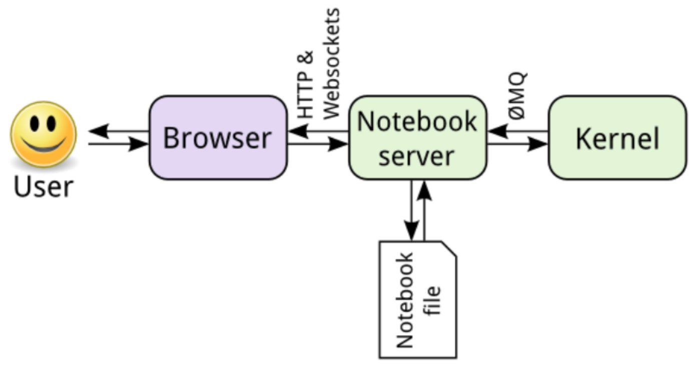


Overview of Architecture

When installing Jupyter notebooks (using PIP or Conda) you install open source software onto your computer. While Jupyter notebook is a browser based application, Jupyter is actually running on your own computer which is acting as a server. The notebooks are being saved in a file in the Jupyter directory. Jupyter notebook has a **plug-in style architecture**. The plug-in architecture style consists of a core system and plug in modules. The core system provides a baseline structure for the system, and the plug in modules are customizable and add specific features which can be used to perform a wide variety of tasks. This style provides flexibility and the ability to extend the software. Since the scope of Jupyter is broad, this style of architecture allows it to be so widely used.

Development Viewpoint of Architecture

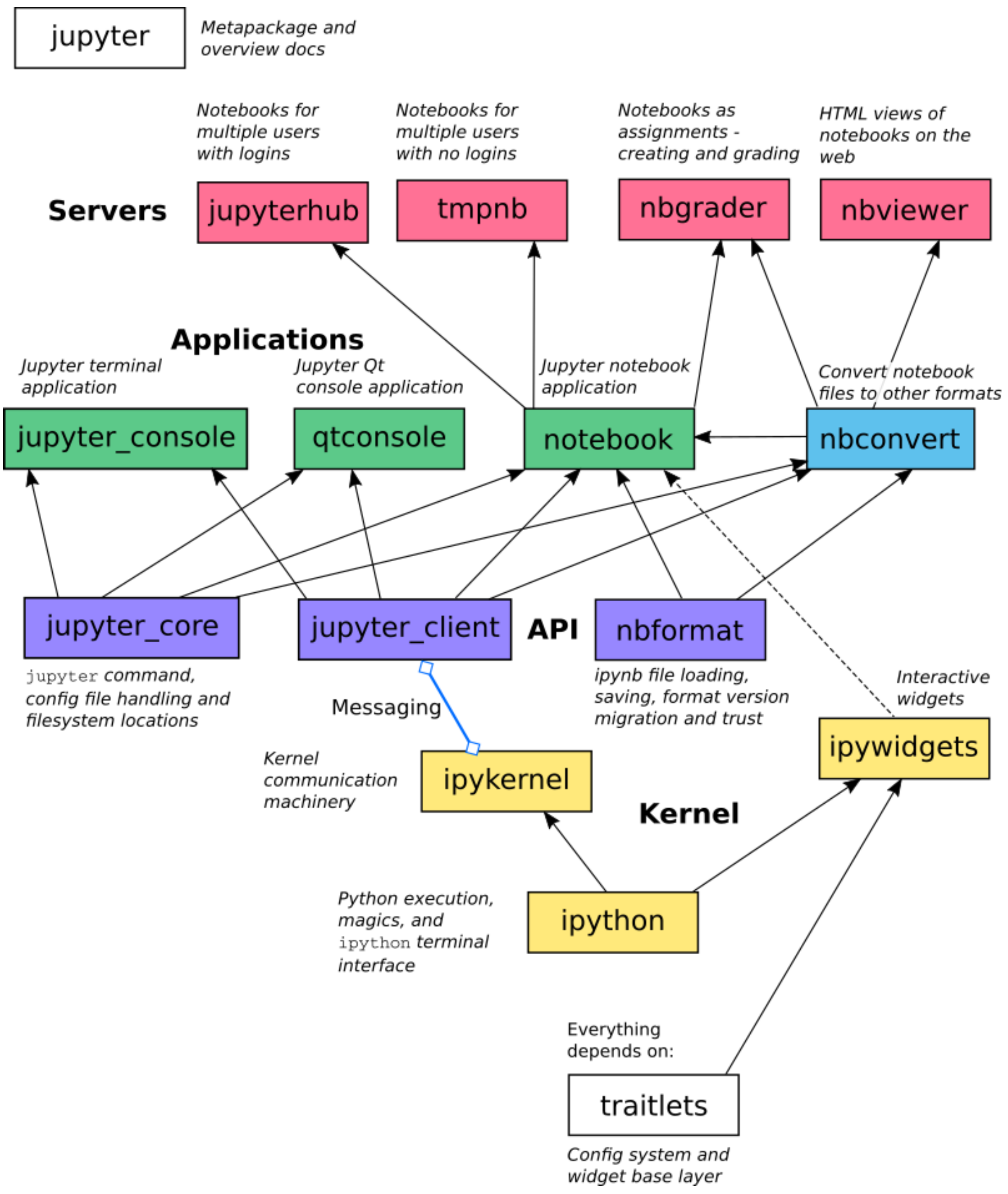
Jupyter notebook uses IPython Kernel to do all compilations and run code. The front end communicates with the Kernel by sending JSON messages to the Kernel API. The model is called a REPL (Read-Evaluate-Print-Loop). The Kernel can be connected to multiple frontends which has provided multi-language support. Alternative languages are compatible when a wrapper kernel is installed. A wrapper Kernel can be easily created for languages that have Python bindings.



Not only does Jupiter run code, it also stores code and markdown notes in a document which they call a “notebook”. These notebooks can be saved, which saves it on disk with the extension .ipynb which has a JSON file structure. The notebook server is separate from the kernel, and it saves and loads notebooks. The kernel acts separately, only activating when the code is being run.

Therefore, since the notebook server is separate from the kernel, you can edit and save languages that are not supported by the kernel. To run that code, you need to install a wrapper kernel which supports the language.

In the diagram seen below, we can analyze the system from a development view. The **ipython kernel** is responsible for the computations in python and potentially other languages using wrapper kernels. It communicates with the **Jupyter Client**. The Jupyter client package provides an API for starting, managing and communicating with the kernel. The Jupyter Client interacts with the notebook and consoles to read code and return outputs. The **Jupyter Core** is a fundamental base package on which Jupyter projects rely. **Nbformat** refers to notebook Format which is the format of the notebooks, it is a JSON file. The nbformat module is used to read or write notebook files as particular nbformat versions. It provides python API’s for working with the notebooks. **Nbconvert** is used to convert these notebook JSON files into other formats such as HTML, LaTeX, PDF, Markdown and other file types. The applications **Jupyter Console, Qtconsole,** and **Notebook** are the direct interfaces for the users to use the notebooks. The Notebook is most commonly recognized as the Jupyter interface. The servers described at the top are a deployment aspect of jupyter which provide different ways of interacting with the notebooks.


jupyter
Metapackage and overview docs
Servers
Notebooks for multiple users with logins
jupyterhub
Notebooks for multiple users with no logins
tmpnb
Notebooks as assignments - creating and grading
nbgrader
HTML views of notebooks on the web
nbviewer
Applications
Jupyter terminal application
jupyter_console
Jupyter Qt console application
qtconsole
Jupyter notebook application
notebook
Convert notebook files to other formats
nbconvert
jupyter_core
jupyter command, config file handling and filesystem locations
jupyter_client
API
nbformat
ipynb file loading, saving, format version migration and trust
Messaging
Kernel communication machinery
ipykernel
Kernel
Interactive widgets
ipywidgets
Python execution, magics, and ipython terminal interface
ipython
Everything depends on:
traitlets
Config system and widget base layer

Architecture of Extensions

There are 3 parts to a Jupyter notebook extension: A YAML file, main.js, and a README.md markdown description. These three files are to be stored in a folder which needs to be located in the nbextensions folder. When your extension is stored in this location, Jupyter will recognize it as an extension, and it will appear on the **Nbextensions** tab in the Jupyter notebook header. The **YAML** file is a human readable file that describes the extension. It is rendered on to the Nbextensions tab so the users can see a description of the extension. It will have a name, description, and compatibility listing. The **main.js** file is the javascript file that contains the code of the extension. CSS files can also be linked for style purposes. The **README.md** file is a markdown file that explains to the users how to use the extensions. Similarly to the YAML file, it is also viewed in the Nbextensions tab.

Our Proposed Improvements

Please see our UI Design feature list for a complete list of our proposed extensions. Here, we will discuss why they will be useful for the Jupyter notebook.

- **Highlighting** and changing **font colour** is an aspect of jupyter which is lacking. Currently a highlighting extension exists, however the colour choices are limited, and the user experience is not ideal. We plan to make a system which is easier to use, and includes highlighting as well as text colour changing. This addition will be useful for all users.
- We plan to create a system for **multiple choice questions**. As a large portion of the user base includes students, we feel that this feature would be beneficial for creating a hands-on learning environment within the notebook itself.
- Students will also be able to check the **hidden correct answer** after answering **short answer questions**, which helps students practice and consolidate the knowledge they learnt.
- **Template cells** can be a helpful tool for both instructors and students to save time from repetitive content like notebook headers.
- **Annotating text** can be a useful option for users to use, because it provides users with the option to make a short note about a portion of texts in a cell that are of particular interest to them.
- **Creating the button that helps users move to the top/bottom cell** can make users more convenient to move to the very top, or the very bottom, which can help users reduce the time and efforts to scroll over a long page.

Conclusion

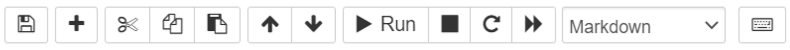
This part summarizes the analysis of Jupyter Notebook focused on its software architecture, helping the reader to be able to understand and contribute to the project. The overall architecture is easy to change and build for Jupyter Notebook because it can be modified and expanded using a variety of plugins, which makes it a highly flexible software. The whole analysis of Jupyter Notebook on its interesting software architecture shows that projects like this largely benefit from the open-source environment and its flexible architecture. A group of passionate developers and contributors have made Jupyter Notebook evolving even further years. In conclusion, Jupyter Notebook is a well-built software system for education. The analysis process of this system is useful to comprehend the software architecture. The proposed enhancement of our design feature lists allows students to be more involved and engaged in the use of the Jupyter Notebook.

Part II. New Extensions - UI Design Feature List

The UI Design Feature List shows the user interface design of the 6 new extensions that our team was to develop, along with the team member responsible for each extension.

Highlight text (Andrea)

- User will select text
- An option of 'Highlight' will be located at the top bar after these icons:



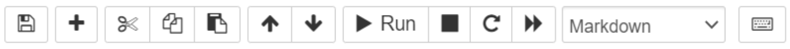
- Selected text becomes highlighted once user clicks the highlight icon
- We will have several colour options for highlighting
- Highlighting is simply done by using the background-colour CSS setting
    - Highlight yellow: <span style="background-color:yellow">Selected Text</span>
- *Relevant stakeholder:* students for highlighting important text in their notes

Change text colour (Andrea)

- Similar process to highlighting text, adding an extra button to the top navigation bar.
- This button will have several text colour options, and once the user selects text, they can click the colour button, and choose their desired colour.
- Text colour done simply using CSS colour setting:
    - Colour red: <span style="color:red">Selected Text</span>
- Great way to personalize for ideal user experience
- *Relevant stakeholder:* instructors and students to create more interesting looking notes

Annotate text (Derek)

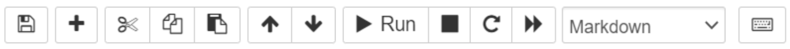
- User will select text
- An option of 'Annotate' will be located at the top bar after these icons:



- A small pop-up will pop out and the user can write some annotations for the selected text in the tiny pop-up window
- *Relevant stakeholder:* Students and instructors. For things like plotted graphs, the instructors may wish to highlight aspects of the graph with the annotation ability. Students may wish to annotate notes with extra add ons to keep track of important aspects.

Hide answer cell for Short Answer Question (Jenny)

- A button will be added to the navigation toolbar:

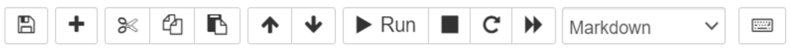


- Users can click the button to set the current cell into a question cell, then it will insert an answer cell and a correct answer cell after the current cell automatically.

- Every time a notebook that contains question cells opens, the correct answer cells will be hidden defaultly, and then can be revealed by clicking the ‘See Correct Answer’ button. (or maybe the answer cells can only be revealed after typing in the answer cell, something similar to the multiple question)
- *Relevant stakeholder:* Instructors, to make notes more condense and hide unimportant cells

Create a template cell (Jenny)

- A button will be added to the navigation toolbar:



- When users click the button, it will insert and run a cell containing a pre-set template after the current cell. If it is a new notebook without any cell, it will add and run a template cell at the top of the notebook.
- *Relevant stakeholder:* Instructors, to save time when making notes with repeating headers

Multiple choice questions (Andrea)

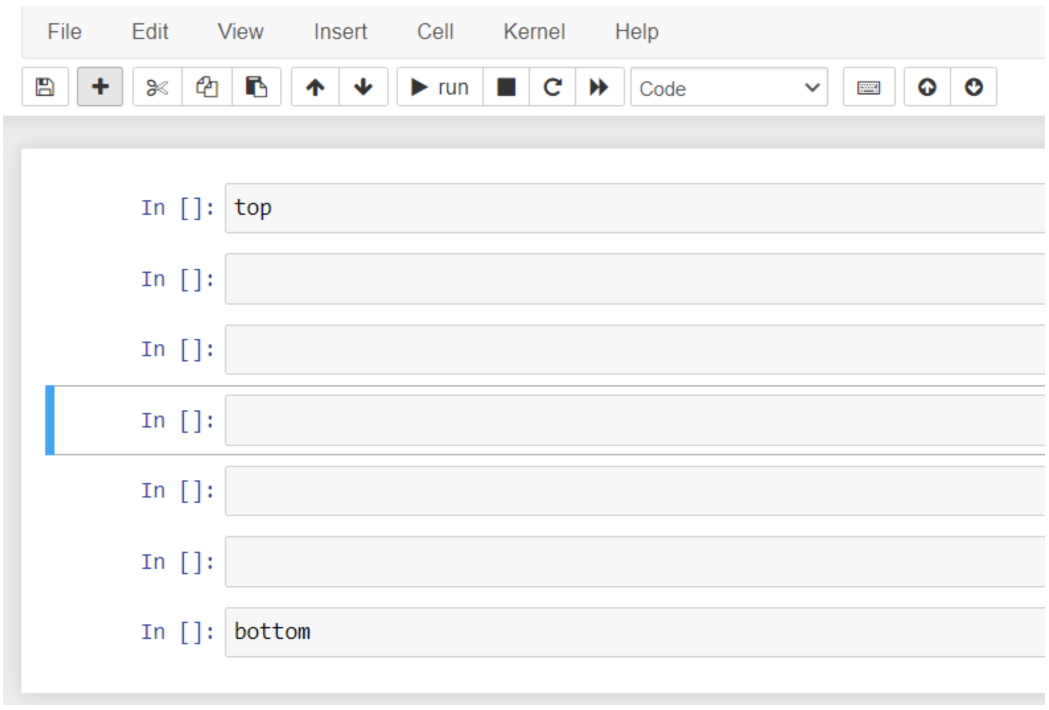
- We will create a function called create_question that can be used for multiple choice questions, with inputs (question: type string, options: type list for [a, b, c, d], correct result: char value)
- It will output the question with options a,b,c,d using a toggle button. The correct value will be outputted once the student clicks the toggle
- We will use our hidden cell technique to hide the cell with the function create_question to keep the documents concise
- *Relevant stakeholder:* Instructors and students to add an interactive “learn as you go” technique to improve the quality of learning

Button that helps users navigate cell to the top/bottom (Derek)

- We will create a button located on the top navigation bar that has the option “top” and “bottom”.
- This will direct users to the top or the bottom cell of the document respectively. When documents become long, it would be handy to move to the bottom of the document, as this is where you would start adding text and cells.
- *Relevant stakeholder:* Mostly instructors, when creating notebooks which are long with many cells.

Part III. Understanding 2 New Extensions

1. Move to Top or Bottom Cell



```
In [ ]: top
In [ ]:
In [ ]:
In [ ]:
In [ ]:
In [ ]:
In [ ]: bottom
```

Figure 1: A screenshot of the Move to top or bottom cell extension

1.1 Description

Functionality: Move to top cell or move to bottom cell.

This extension will create two buttons located on the top navigation toolbar. The first button is called "Move to top cell", with the icon ⊙ , whereas the other button is called "Move to bottom cell", with the icon ⊙ .

1.2 Benefits

In computing science courses involving programming, learning has to do with seeing code written, and practicing writing code. Many students or instructor may start using Jupyter Notebook for programming or documenting purposes. A Jupyter Notebook document may consist of an abundance of cells, no matter markdown cells or programming cells.

As Jupyter Notebook documents started to contain a large number of cells, the overall length of the entire document would indeed become long, with possibly hundreds or thousands of cells. In that sense, it would be very handy introduce such kind of extension, where users have the ability to move directly towards the very top position, or the very bottom position of the document, as this is where you would start adding text and cells.

The relevant stakeholders that would benefit from this extension are: Mostly instructors, when creating notebooks which are long with many cells.

1.3 Rules of use

With this extension enabled, two buttons and would appear on the top navigation toolbar.

1. The user should first *select* any one cell, by clicking on that cell.
2. As you *click* button, the notebook will jump to the topmost cell in the document.
3. As you *click* button, the notebook will jump to the bottommost cell in the document.

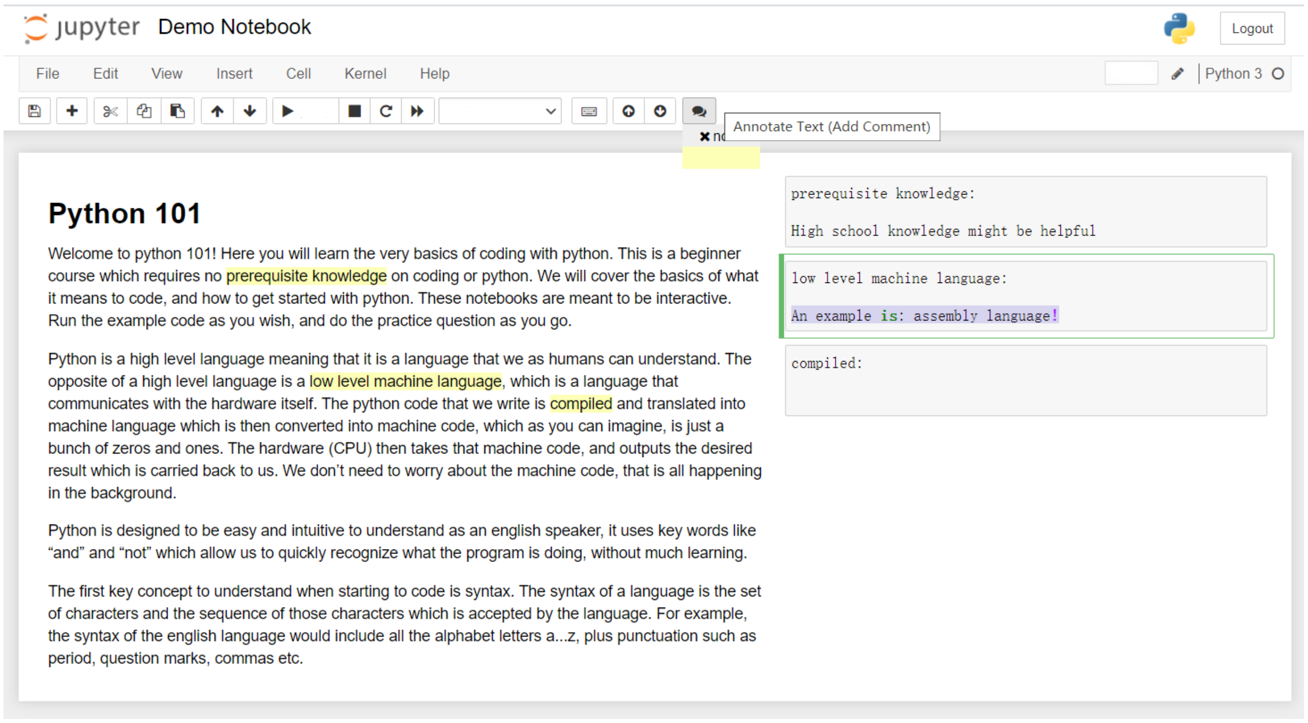
1.4 Limitation and constraint

a. The total number of cells should be no more than 1000 cells

A slight limitation is that the total number of cells needs to be no more than 1000 cells when using this extension. If more than 1000 cells exist in the document, the move button can only work by moving 1000 steps once at a time. In that sense, if more than 1000 cells exist in the document, the user may need to click the move to top/bottom cell button a couple more times, until it reaches the end or beginning position.

This constraint is in place because the extension works by relying on the theory of the time efficiency of algorithms. In practice, the extension works by using a for-loop that keeps reiterating 1,000 times on the pre-established functions that were in place in the Jupyter Notebook framework: *Jupyter.notebook.select_prev()* and *Jupyter.notebook.select_next()*. Re-iterating 1000 times seems to be responding pretty quickly, in the essence of time efficiency in algorithm. If I change this to re-iterate more times, such as 10000 or even 100000 times, the move seems to be pretty logy and slow, that takes a few seconds to complete. Therefore, I keep the iteration of the loop to 1000 times rather than more, but it can be easily modified to more than that, if needed.

2. Annotate Text


jupyter Demo Notebook
Logout
File Edit View Insert Cell Kernel Help
Python 3
Annotate Text (Add Comment)

Python 101

Welcome to python 101! Here you will learn the very basics of coding with python. This is a beginner course which requires no prerequisite knowledge on coding or python. We will cover the basics of what it means to code, and how to get started with python. These notebooks are meant to be interactive. Run the example code as you wish, and do the practice question as you go.

Python is a high level language meaning that it is a language that we as humans can understand. The opposite of a high level language is a low level machine language, which is a language that communicates with the hardware itself. The python code that we write is compiled and translated into machine language which is then converted into machine code, which as you can imagine, is just a bunch of zeros and ones. The hardware (CPU) then takes that machine code, and outputs the desired result which is carried back to us. We don't need to worry about the machine code, that is all happening in the background.

Python is designed to be easy and intuitive to understand as an english speaker, it uses key words like "and" and "not" which allow us to quickly recognize what the program is doing, without much learning.

The first key concept to understand when starting to code is syntax. The syntax of a language is the set of characters and the sequence of those characters which is accepted by the language. For example, the syntax of the english language would include all the alphabet letters a...z, plus punctuation such as period, question marks, commas etc.

```
prerequisite knowledge:

High school knowledge might be helpful
```

```
low level machine language:

An example is: assembly language!
```

```
compiled:
```

Figure 2: A screenshot of the Annotate Text extension

2.1 Description

For annotating purposes, this extension provides a user-friendly system to add comment, after highlighting a portion of words in a markdown cell. The dropdown functionality creates a familiar look and feel and it is easy to use.

This extension will create a button called "*Annotate Text (Add Comment)*" located on the top navigation toolbar, with dropdown options to either select the highlighting color, or remove highlighting in selected cell.

Note: After enabling the extension, the page would automatically be partitioned as a two-column mode, where markdown (text) cells are on the left while code cells are on the right.

2.2 Benefits

In computing science courses involving programming, learning has to do with seeing code written, along with teaching materials associated with codes. Many students or instructor may start using Jupyter Notebook for programming or documenting purposes. A Jupyter Notebook document may consist of a markdown cell that contains teaching materials written in plain English paragraphs.

As Jupyter Notebook documents started to contain paragraphs written in natural language rather than programming codes, people may wish to keep track of important aspects by highlighting words, or even add comments to annotate them. In that sense, it would be very handy introduce such kind of extension, where users have the ability to annotate notes.

2.3 Rules of use

With this extension enabled, a button called "*Annotate Text (Add Comment)*" would appear on the top navigation toolbar.

1. The user should first *select* a portion of texts, in a markdown cell.
2. After you *click* button, *click* on the color yellow button on the dropdown menu. This will highlight the selected text in yellow, and the text would appear on the right column cell.
3. On the right column, you can *type* something to add notes or comments towards the highlighted text.
4. To remove annotated notes, *select* a portion of texts and *click* on the ✖ none button on the dropdown menu, after you *click* button. Finally, delete the annotation cell on the right manually.

2.4 Limitations and constraints

This extension is developed based upon the *highlighting extension* that was previously developed by the other teammate. Hence, apart from the already forementioned constraints from the highlighting extension, the other three main limitations are as follows:

a. Un-highlighting a word wouldn't remove the corresponding annotation cell - users will have to delete the annotation cell manually afterwards.

A slight limitation is that removing the highlighting of a portion of words wouldn't delete the annotation cell. In that sense, users will have to delete the annotation cell manually afterwards.

This constraint is in place because it creates another annotation cell every time as we highlight a portion of texts. In order to programmatically delete a cell, it would not be easy to track which annotation cell belongs to which specific highlighted text after all. Hence, it could be hard to find or track which cell belongs to that word, because all the annotation cells could be out of order after all.

Conversely, this limitation could also be a benefit, working as a safeguard, because users can carefully check and see which annotation cell was really the one they wish to delete, and this can avoid important data being deleted mistakenly. In Jupyter Notebook, one important issue is that it may not be easy to recover mistakenly thrown cells. In that sense, if a cell containing important data was mistakenly deleted by the extension, we may lose a lot of data that could be invaluable.

b. After highlighting a word, the new cell containing annotation will always be added to the end of the document, even if that word is located at an earlier position in the paragraph.

Another limitation is that after highlighting a word, the annotation cell will always be added to the end of the document, even if that word is located at an earlier position in the paragraph.

This constraint is in place because the extension works by inserting a new code cell always at the 1000th position in the Jupyter Notebook document, and that would mostly be the last position of the right column. That way, over the user interface, people can always see a scenario that looks like a new sticky note is being inserted at the end.

c. Only one markdown cell is allowed per document, and there should be no other code cells.

You may create only one markdown cell that includes all the necessary teaching materials. When I was trying out to create a document that includes lots of markdown cells, I noticed that the annotation cell would later be inserted at the bottom on the right column. To avoid this, having only one markdown cell, rather more, would solve this problem.

Also, there should be no other code cells because this extension works by making the user interface look like a two-column view, which is done by floating all text cells on the left and code cells on the right. With the extension here, we assume that all the code cells are created on the right to be treated as sticky notes. If there are other code cells that exist, they will be floated on the right as well.

Part IV. Any Open Problems Left for Future Work

1. Lack of documentations on Jupyter Notebook framework

One of the most important problems is the shortage of documentations on Jupyter Notebook framework available for developing the extensions. As we developed the extensions, we recognized that the framework of Jupyter Notebook is quite different than what we have seen for a natural webpage, and there were not much of other external documentations that are written for Jupyter Notebook, except for those in the official website. Thus, we needed to rely solely on the documentation on the official website. However, they are quite trivial and are not listed together. Many probable functions that people could make use of are not listed or are missing in the website. For future work on this problem, Jupyter Notebook's internal site admins, developers, or maintainers may need to upgrade and replenish their documentations.

2. The limitations and constraints of our extensions

Upon developing those extensions, we realize that this is the first phase of generation of the production. Hence, they might not be perfect, in the sense that there were quite some forementioned limitations associated with them. Therefore, those limitations could be treated as the other problems left for future work. In the future, software developers may work on maintaining the extensions and improving the forementioned limitations.

Part V. Conclusion

In conclusion, this project gives us a good opportunity to get in touch with the internal architectural structure of the Jupyter Notebook. It is a good project to work with because we got to know what extensions would be useful for teaching and learning and we learnt to add missing features to the software by developing multiple Jupyter Notebook extensions.

Among all of the six extensions, each extension has its own advantage. The extension of highlight text and change text color can help students to keep track of important aspects in a teaching material. The Annotate text extension adds further benefits to this aspect by letting teachers or students add annotations on those important parts. The Hide answer cell for Short Answer Question extension and the Multiple choice questions extension can help students verify if they solved the problems correctly, before showing all the answers to them. The Create a template cell extension can help teachers easily insert a pre-set template, such as the test paper title containing the course number, when designing a test in the Notebook. The Move to the top/bottom cell extension helps users reduce the time and efforts to scroll over a long page.

Part VI. Peer Evaluation

The below table and Part II of the report both illustrate the distribution of workload in our team. Each teammate is responsible for about two extensions. Of course, we did discuss and help each other out during the development process.

Extensions	The teammate in charge
Highlight text and change text color	Andrea
Annotate text	Derek
Hide answer cell for Short Answer Question	Jenny
Create a template cell	Jenny
Multiple choice questions	Andrea
Button that helps users move to the top/bottom cell	Derek

Peer evaluation on each group member:

Andrea was a very nice person and worked very well in all aspects. She was very easy to work with and talk with. She did her work very well and on schedule. She did mostly take on a leadership or organizational role. Of course, she was always on time for meeting. She did her highlighting extension very well. Since my annotate text extension was developed based upon hers, she did patiently explain to me how her code programmatically work as well. It was very nice to have her as a teammate. I would give her 5 points.

Jenny was also a nice teammate. She worked on the two extension pretty well. We did keep connecting to each other to see how each other’s extension was going sometimes. She was helpful and cooperative. Of course, she was always on time for all the meetings. Works were done on schedule. I would give her 5 points.

Derek was a nice teammate. I worked on the two extensions for this project. Of course, I was always on time for all the meetings. I did my work on schedule. I sometimes kept connected on Teams to discuss and help each other out. I worked mostly well in all aspects. I award myself 5 points.

References

“The Jupyter Notebook,” *The Jupyter Notebook - Jupyter Notebook 6.2.0 documentation*. [Online]. Available: https://jupyter-notebook.readthedocs.io/en/stable/. [Accessed: 01-Feb-2021].

Jupyter Notebook. (n.d.). *Unofficial Jupyter Notebook Extensions — jupyter_contrib_nbextensions 0.5.0 documentation*. Retrieved April 17, 2021, from https://jupyter-contrib-nbextensions.readthedocs.io/en/latest/