Introduction to Programming W11

Introduction to Jupyter Notebook, Pandas

2022/06/17

1 Jupyter notebook

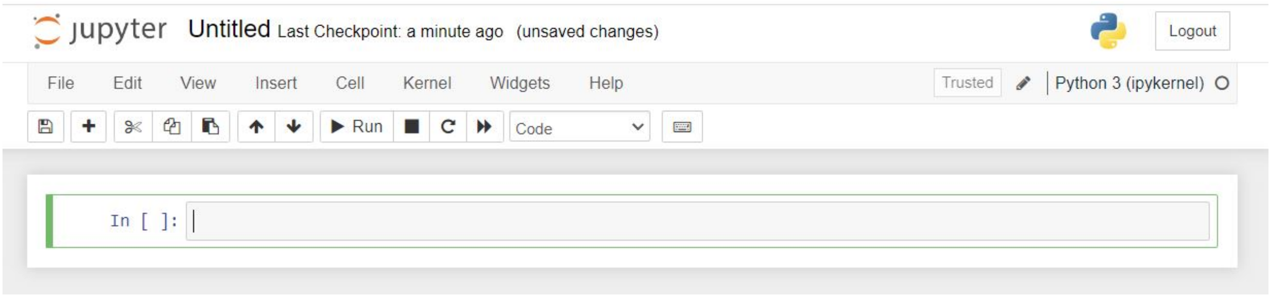
Jupyter Notebook is basically a web application that can be used to create documents including live/interactive code, visualizations, and free text. Jupyter notebook supports many languages, including Python. When looking for code on the Internet, you will often encounter *notebooks* in the .ipynb format. These are basically JSON documents containing text, source code, metadata. Jupyter notebook comes already installed with the Anaconda distribution. If you do not have Jupyter already, it can be installed with the `pip install notebook` command from the terminal (or command prompt where you can run conda/pip commands).
You can start Jupyter Notebook in multiple ways:

- Open a terminal, and run the command `jupyter notebook`

  OR

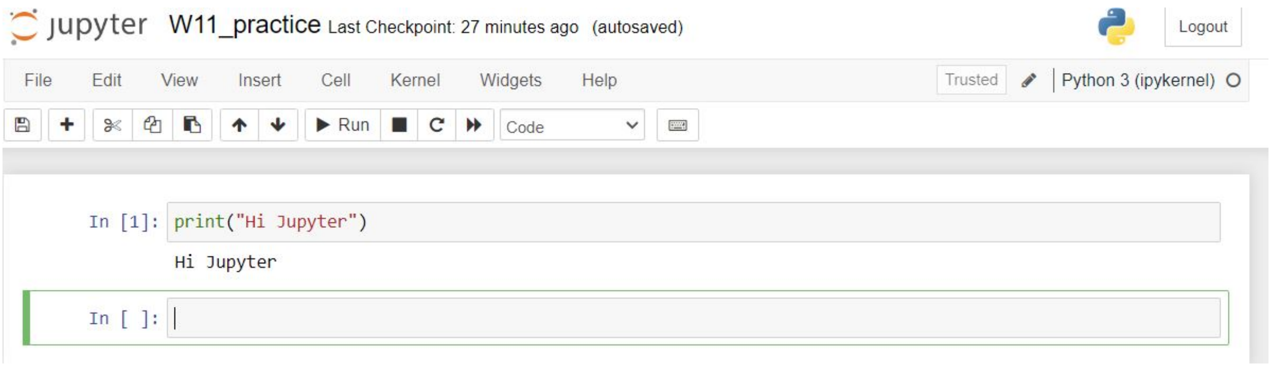
- Open the Anaconda Navigator, and just click on Jupyter Notebook (JupyterLab is an extended version of Jupyter Notebook, but we will cover only the latter in this class. So make sure you click on Notebook, *not* JupyterLab)

You will see that a Jupyter window has popped up in your default browser and your folder structure is shown (the exact location is OS and installation type dependent). Navigate to a folder where you wish to create your notebook. Then, click on "New" in the top right corner, and select Python 3. You will see something like this in your browser:



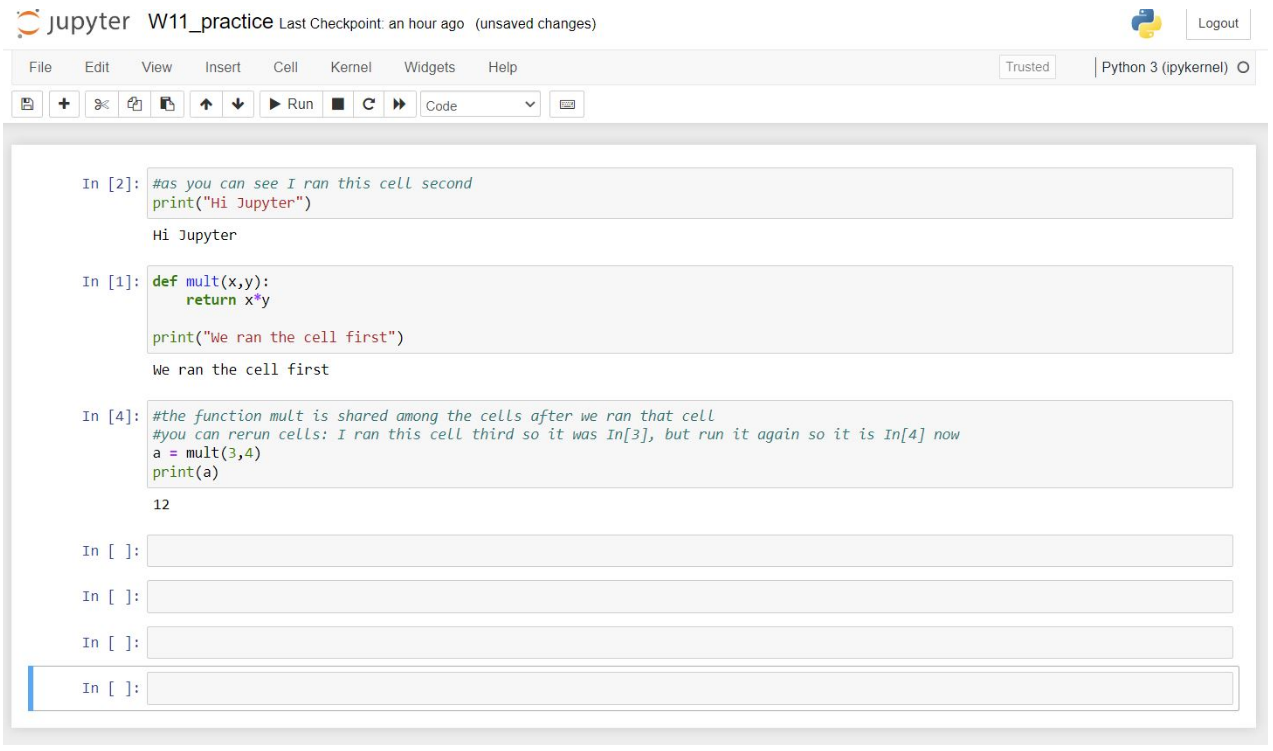
```
In [ ]:
```

As you can see the notebook is still untitled. Your browser tab will also be named "Untitled - Jupyter Notebook", so in the Jupyter tab click File/Rename to rename the file to something that makes sense. You can name this notebook `W11_practice`. Then if you navigate to the folder where the file was created, you will see that the file `W11_practice.ipynb` is there (if your system does not show file extensions, I highly recommend you change it so it shows the extension of every file).
Now you can start writing Python code to the *cells*. You can see you already have one empty cell, so write a simple print statement and execute the cell to make sure everything works. Executing a cell is sometimes called *running* the cell, and you can do this two ways: 1. clicking the *Run* button or it is much easier to just 2. press `Shift` + `Enter`:



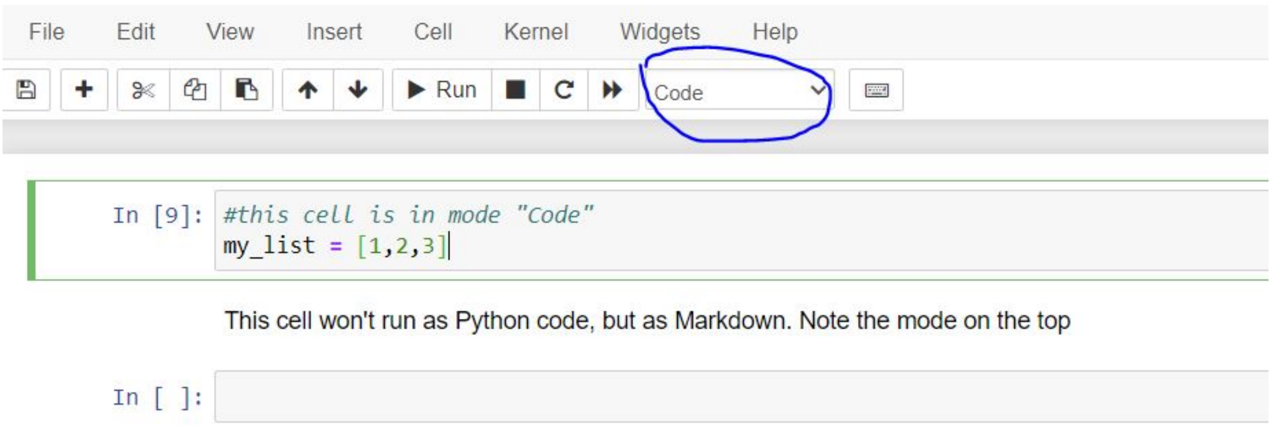
```
In [1]: print("Hi Jupyter")

        Hi Jupyter

In [ ]:
```

One of the convenient things about Jupyter Notebook is that you can run cells in any order you wish, and share the variables, imports, functions etc. across the cells. This means you can experiment code in a fast way, and also makes debugging easier. The `In[]` left to the cells indicate the order the cells ran.


Jupyter W11_practice Last Checkpoint: an hour ago (unsaved changes)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3 (ipykernel)
Run
Code

```
In [2]: #as you can see I ran this cell second
        print("Hi Jupyter")

        Hi Jupyter

In [1]: def mult(x,y):
            return x*y

        print("We ran the cell first")

        We ran the cell first

In [4]: #the function mult is shared among the cells after we ran that cell
        #you can rerun cells: I ran this cell third so it was In[3], but run it again so it is In[4] now
        a = mult(3,4)
        print(a)

        12

In [ ]:

In [ ]:

In [ ]:

In [ ]:
```

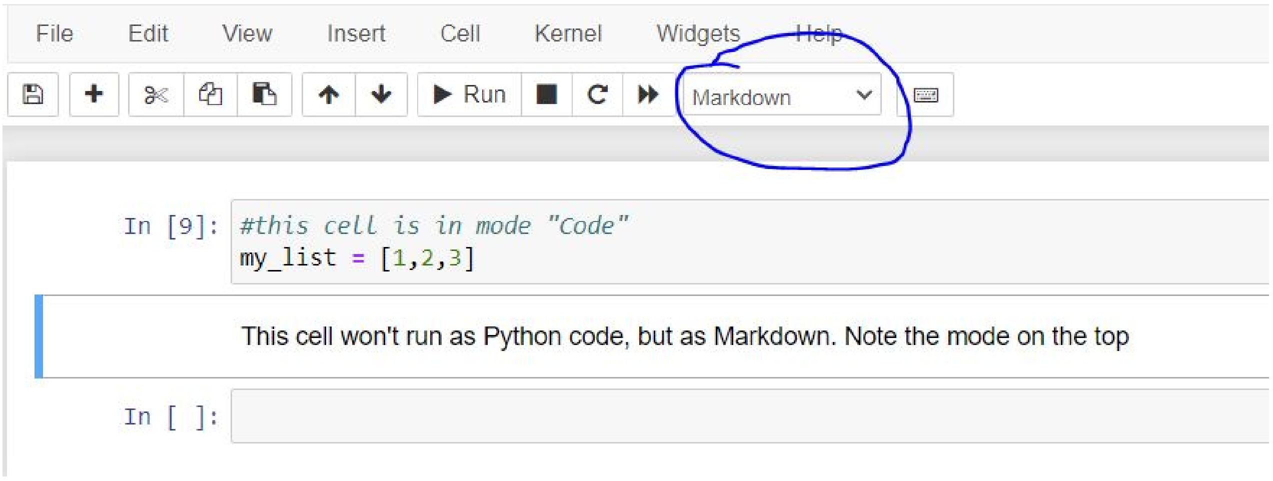
You can manually add new cells below the cell you select. Do this by clicking on the plus sign (+) in the top left corner, or Insert/Cell below (or above). The easiest way to delete a cell is to select it (basically clicking on it) and cut them by the cut symbol on the top left, or Edit/Delete Cells or Cut Cells. If you select Kernel/Restart, the whole session will be restarted (you lose your variables, etc. from the memory, and cell counting will restart, although the code itself is not lost). Delete all cells so you only have the first empty cell.
There are different modes cells can run in, and if you check in the top, so far everything was in "Code" mode. There are other modes, and one very useful (and often encountered in other peoples' notebooks) is "Markdown". Markdown is a markup language for creating formatted text using a plain-text editor. We will not cover Markdown in this class, but the point is that you can write code explanations, etc. into cells in Markdown mode, and those will not be executed as Python code. You can change the mode of the cell by selecting the desired mode from the drop-down menu in the top.



```
In [9]: #this cell is in mode "Code"
        my_list = [1,2,3]
```

This cell won't run as Python code, but as Markdown. Note the mode on the top

```
In [ ]:
```



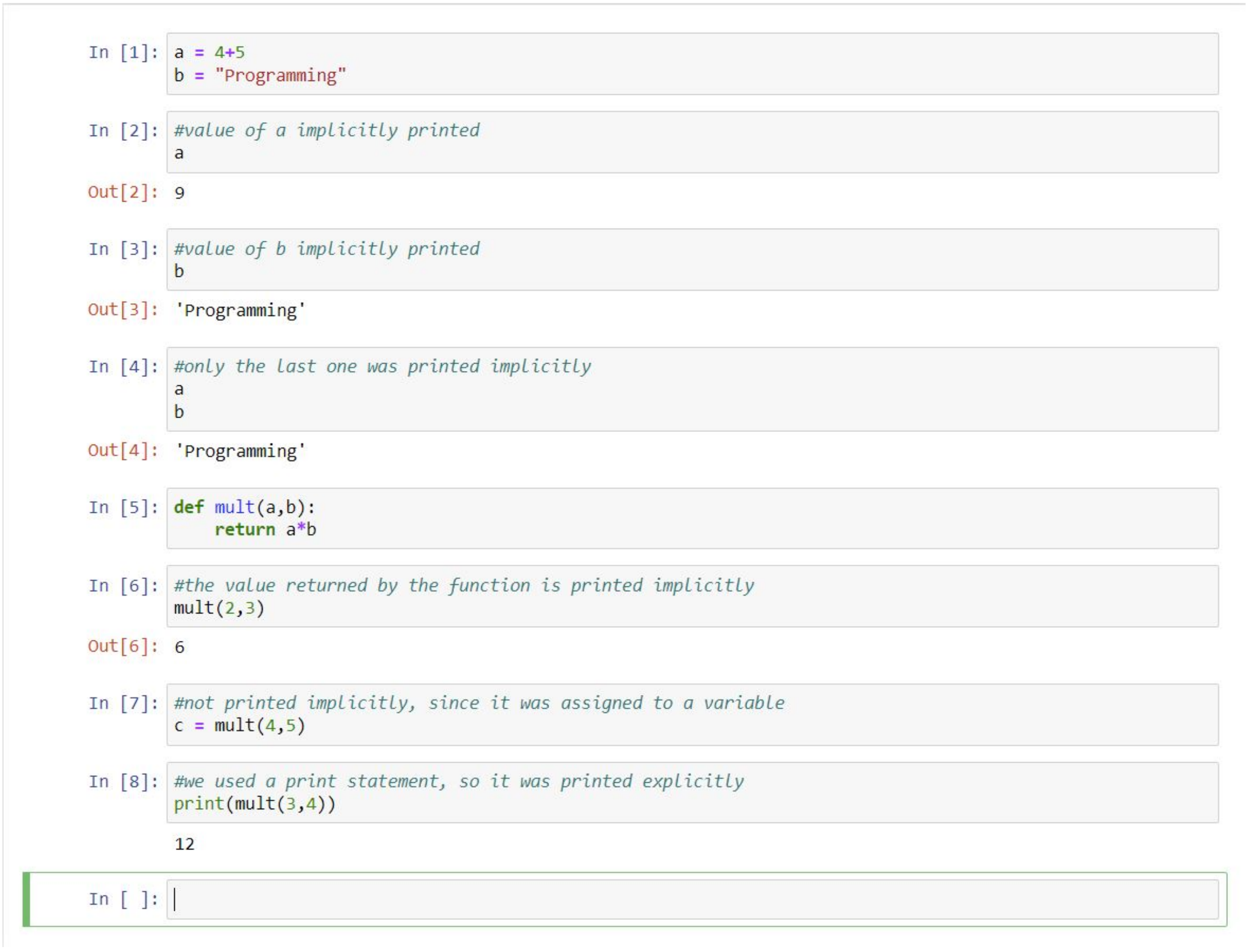
```
In [9]: #this cell is in mode "Code"
        my_list = [1,2,3]
```

This cell won't run as Python code, but as Markdown. Note the mode on the top

```
In [ ]:
```

Restart your kernel, and delete all cells so you only have the first empty cell.

In Jupyter Notebook, the code is executed in a REPL (read-eval-print loop) or in a *interactive language shell*, meaning that it always returns the value printed. This print is implicit, but you can still use the print statement explicitly as before.



```
In [1]: a = 4+5
        b = "Programming"

In [2]: #value of a implicitly printed
        a

Out[2]: 9

In [3]: #value of b implicitly printed
        b

Out[3]: 'Programming'

In [4]: #only the last one was printed implicitly
        a
        b

Out[4]: 'Programming'

In [5]: def mult(a,b):
            return a*b

In [6]: #the value returned by the function is printed implicitly
        mult(2,3)

Out[6]: 6

In [7]: #not printed implicitly, since it was assigned to a variable
        c = mult(4,5)

In [8]: #we used a print statement, so it was printed explicitly
        print(mult(3,4))

        12

In [ ]: |
```

This behavior of Jupyter Notebook can be really useful to experiment with code quickly.
There are many other things in Jupyter notebook that can be useful, and the documentation is available at `https://jupyter-notebook.readthedocs.io/en/stable/`

2 Pandas

Pandas is an essential Python package for Data Scientists. From the documentation:
Pandas is a Python package providing fast, flexible, and expressive data structures designed to make working with

"relational" or "labeled" data both easy and intuitive. It aims to be the fundamental high-level building block for doing practical, real-world data analysis in Python. Pandas is well suited for many different kinds of data:

- Tabular data with heterogeneously-typed columns, as in an SQL table or Excel spreadsheet
- Ordered and unordered (not necessarily fixed-frequency) time series data
- Arbitrary matrix data (homogeneously typed or heterogeneous) with row and column labels
- Any other form of observational/statistical data sets

The two primary data structures of pandas, *Series* (1-dimensional) and *DataFrame* (2-dimensional). Pandas is built on top of NumPy and is intended to integrate well within a scientific computing environment with many other 3rd party libraries. Pandas already comes installed with the Anaconda distribution, but if you don't have it (encounter an error when trying to import it) you can install it by running the `pip install pandas` command in the terminal. It is a good idea to get familiar with pandas, because datasets are often distributed in a tabular format (such as .csv) which is easy to deal with using this package.

Pandas is often used in Jupyter Notebook to work with it interactively. It is customarily imported as `pd`. Make a new notebook in Jupyter (you should still have the main Jupyter tab in your browser where you can see your folder structure), and rename it to something that makes sense. Go trough the "Getting started tutorials" of the pandas documentation. You can skip parts related to data visualization (e.g., "How to create plots in pandas?").
https://pandas.pydata.org/docs/getting_started/intro_tutorials/index.html

3 Homework

No assignments this week, but make sure you went through the "Getting started tutorials" of the pandas documentation. In the next few weeks, you will be required to read in/save and preprocess data. You do not need to memorize the methods/functions, but make sure you can effectively use the pandas documentation. I highly recommend you to make notes or create browser bookmarks, etc. You will be required to use the following:

- object creation (Dataframes and Series)
- view, concatenate-join data
- set, sort, select/slice data
- Boolean data indexing
- applying basic pandas statistics to data
- save and read in data (csv)