**Empowering Analytics: A Beginner’s Guide to Integrating Python Scripts in Power BI**

Python, being one of the most popular languages in the Data Science community, offers a robust toolkit that facilitates various stages of a project’s lifecycle. From data acquisition and cleaning, exploration, and feature engineering, to predictive modeling and result visualization, Python has become an integral tool.

On the other hand, Power BI stands out as a data analysis tool that enables the intuitive creation of highly effective interactive dashboards. Despite its steep learning curve, Power BI streamlines the development of elaborate and complex dashboards in a shorter timeframe.

When we delve into the integration of Python scripts in Power BI, we take a significant step to maximize the efficiency and potential of these tools. This synergy not only speeds up dashboard creation through more agile methods than manual clicks but also automates data download directly. Essentially, we are enabling a more fluid and powerful workflow, allowing users to harness the full data analysis capabilities of Python within the intuitive environment of Power BI.

**1. Essential Prerequisites**

Let’s get started with the prerequisites.

To run Python scripts in Power BI Desktop, you need to have installed Python on your local machine. You can download Python from the [Python website](https://www.python.org/downloads/).

To integrate Python scripts into Power BI, you need to have available and installed the following Python packages:

* [Pandas](https://pandas.pydata.org/): A software library for dataset manipulation and analytics. It is by far the most widely used toolset for working with data, it has tools for working with different data structures, and it ranges from downloading, cleaning, and manipulating almost any type of data. **To create tables in Power BI from a Python Script, we must create them as [DataFrames](https://www.w3schools.com/python/pandas/pandas_dataframes.asp" \t "_blank)** (which is one of the main data structures that Pandas works with). You can use the following command directly in your console:

pip install pandas

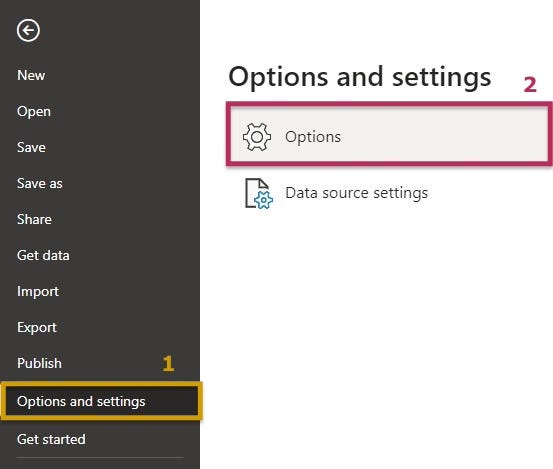
* [Matplotlib](https://matplotlib.org/): A plotting library for Python, which is a tool for creating static, animated, and interactive visualizations. It is one of the most widely used plotting libraries. You can use the following command directly in your console to install it:

pip install matplotlib

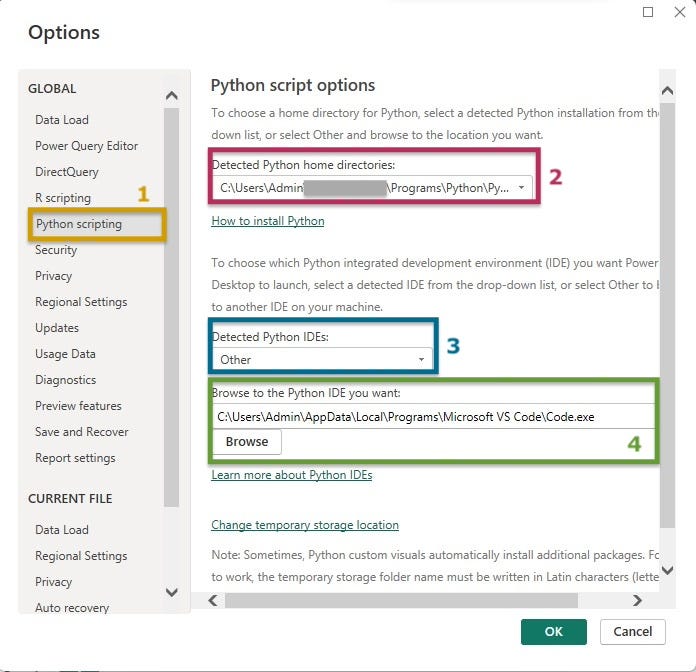
Once installed, you are ready for the next step!

**2. Enabling Python Scripting in Power BI**

In Power Bi Desktop you should go to **Files > Options and Settings > Options**. Then you should see a window like this



Here we are going to select Python scripting and the Python script options page appears. Here, you can see up to in the “Detected Python home directories” and you are going to find your local Python installation path if it is not already given you can just click on the arrow and select the path that leads to your Python home directory. Make sure the path is for the local Python installation that you want Power BI Desktop to use.



At the bottom, you have your “Detected Python IDEs”. Since the Power BI could not select the Python IDE preferred, you click on “other” and browse for the Python IDE that you wanted, which in my case is the Visual Studio code.

One way to identify the IDE path is to find the executable of your IDE, right-click on the icon, and open the file location.

Once you select your Python Directory and IDE you are ready to begin running Python scripts in Power BI, so let's move on to importing data with python by clicking the OK button.

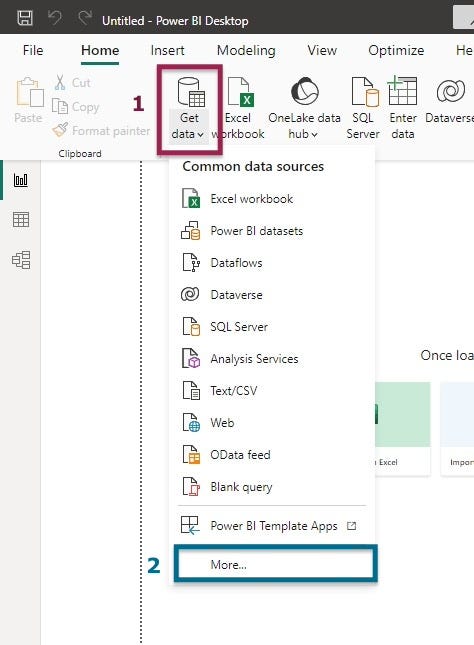
**3. Using Python Scripts to Import Data**

Python Scripts are useful in importing data. They can be used to automate the process of importing data from various sources into a Python program. With the help of these scripts, data can be imported from various formats such as CSV, Excel, and SQL databases. Python Scripts are easy to write, and they can be easily customized to suit specific needs. By using Python Scripts, the process of importing data can be made more efficient and less time-consuming, saving valuable time and resources.

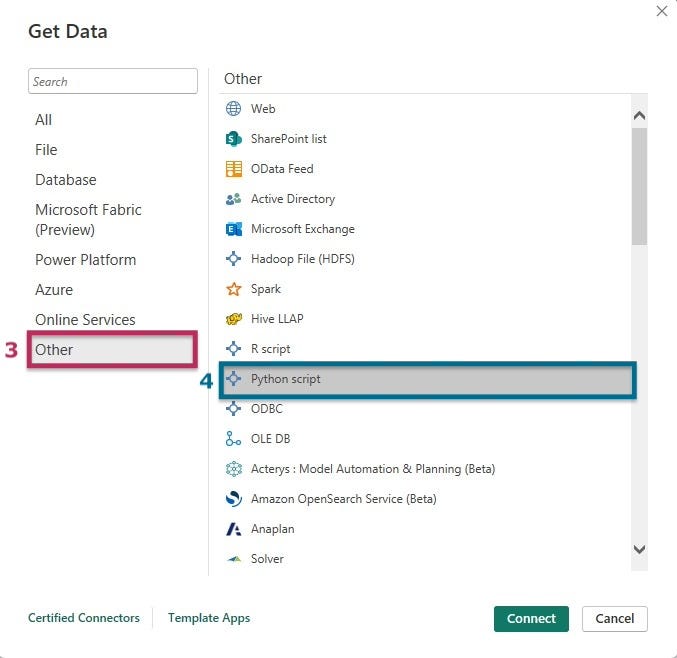
First of all, we need to prepare a Python script, so we should create a script on the local Python environment and make sure it runs successfully. In this article, I’ll be using a basic python script as follows:

from sklearn.datasets import load\_wine  
import pandas as pd  
  
# Load the Wine dataset  
wine = load\_wine()  
  
# Create a DataFrame with the data  
wine\_df = pd.DataFrame(data=wine.data, columns=wine.feature\_names)  
  
# Add the target variable (cultivar) to the DataFrame  
wine\_df['cultivar'] = wine.target

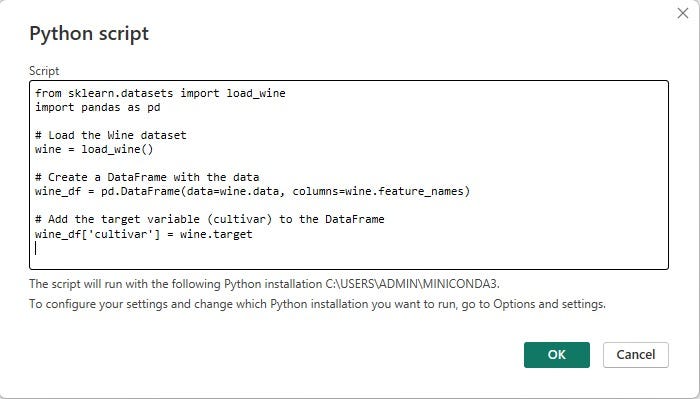
In the Home tab of the ribbon, select ‘Get data’ to bring up the full list of data connections.



Select the ‘Other’ category and find ‘Python script’ on the list.



An input box will appear, prompting you to input your Python code. In this illustration, we’ll employ the Scikit-Learn library to load the Wine dataset, a dataset we’ll continue working with in subsequent sections. Keep in mind that Power BI exclusively supports the import of Pandas DataFrames, so we need to do the conversion of the dataset using ‘pd.DataFrame.’



When you click ‘OK’, another window will open. Make sure to select the DataFrame table which in this case is called ‘wine\_df’ and contains the data. The following steps are as we usually connect to any data source in Power BI.

When you have several DataFrames in the same script you should only repeat these steps once and in the final step just make sure to select all the DataFrames that you wish to import.

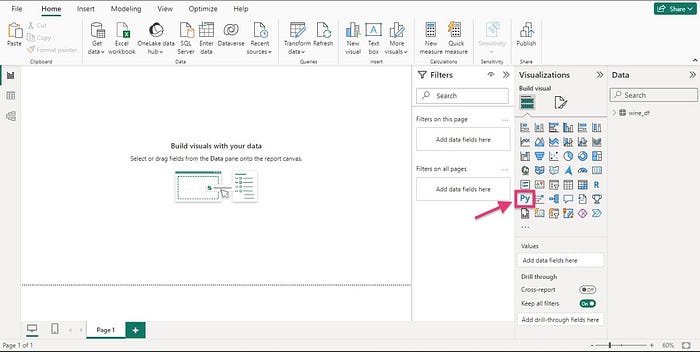
Keep in mind, that when you click on refresh in the home menu, which usually updates data tables using the connections, in this case, “refresh” will run again the script.

Pretty easy right? Let’s then continue our learning path.

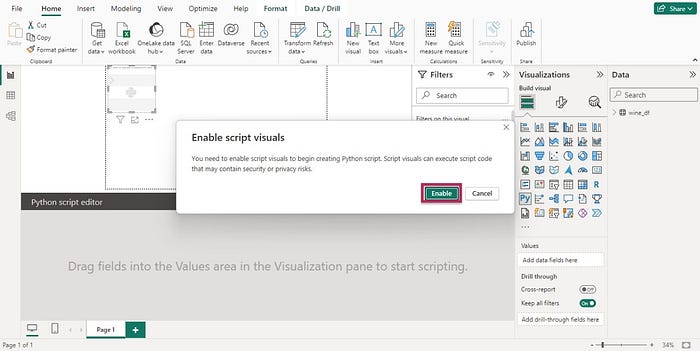
**4. Using Python Scripts to Create Visualizations**

Now that we’ve covered data importation with Python, let’s proceed to construct data visualizations using the previously uploaded data. It’s essential to emphasize that the primary focus of this article is to understand the integration of the Python language within Power BI. The aim is not to develop intricate scripts (as this is not a programming tutorial); instead, we’ll maintain simplicity in our approach.

To start, let’s choose the Python visual from the visualization pane, identifiable by the Py icon.

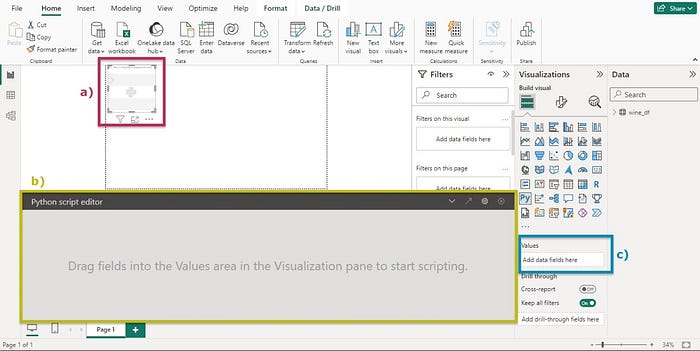


This action will prompt a dialog called “Enable script visuals,” which appears the first time you use this feature in each Power BI session, so we are going to click enable.

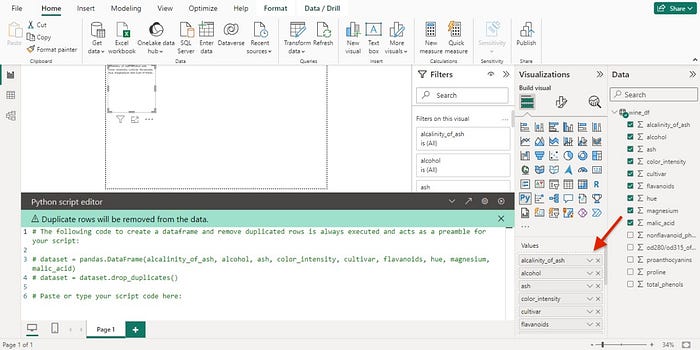


When we add a Python visual to a report we can expect two things to happen:

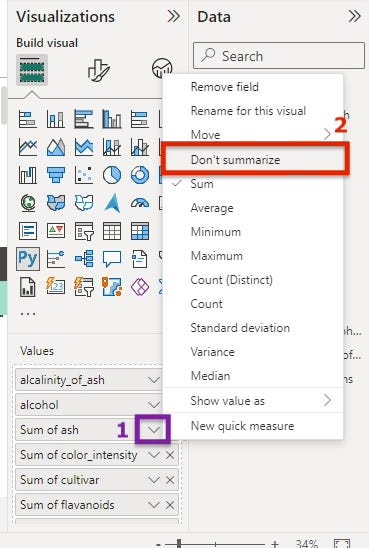
1. We will see our placeholder Python visual image, which appears on our report canvas (we can identify it on the image below as labeled “a)”).
2. Then we will see the Python Script Editor which appears at the bottom (we can find it labeled with “b)”).



We should take into account that our Python Script can only take in fields added to the values section (labeled as “c)” inside the image). For this reason, we are going to drag some data variables to the values section.



Ensure that the variables within the values section are not summarized. Verify this by clicking on the downward arrow next to the value, and confirm that “Don’t summarize” is selected.



Now, you can utilize the selected data for plotting. As you add or remove data values, the supporting code in the Python script editor will automatically generate or remove accordingly. As you already noticed, the Python script editor generates a binding code, which creates a data frame with all the fields that we have added.

Similar to table visuals,**fields are grouped and duplicate rows appear just once**. If you don’t want them to be grouped, you can prevent this by adding an index column, which makes sure that none of the rows is repeated, and as a consequence, the data isn’t grouped.

With the DataFrame automatically generated, including the selected fields, we are prepared to proceed and craft a Python script to construct a plot. Once the script is complete, you can click the run button.

It’s important to note that Power BI replots a visual under two conditions:

1. When selecting “run” from the Python script editor bar.
2. Whenever a data change occurs, such as during data refresh or filtering.

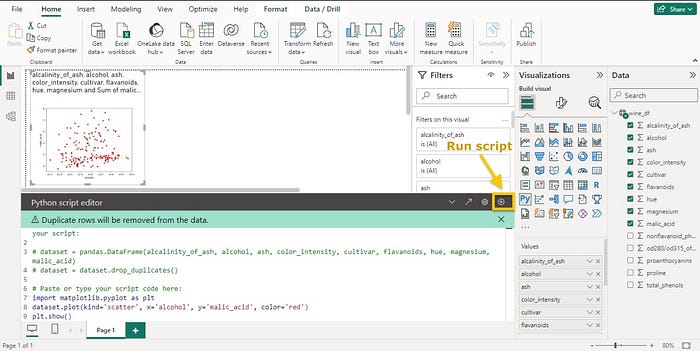
Therefore, if an error occurs in the script, the graph may not appear.

**Let’s create some visuals**

1. Suppose we want to build a correlation plot to explore if there is a correlation between alcohol and malic acid.

***Scatterplot***

import matplotlib.pyplot as plt  
dataset.plot(kind='scatter', x='alcohol', y='malic\_acid', color='red')  
plt.show()

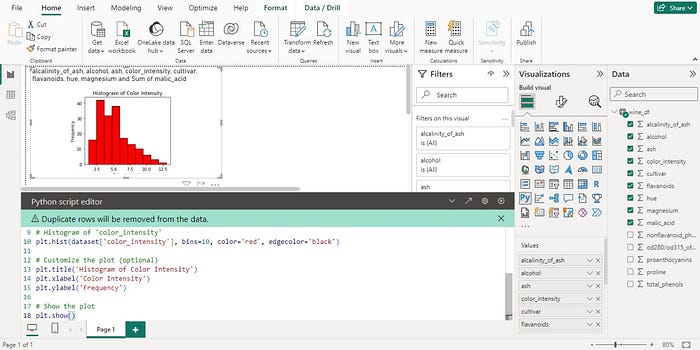


As you can see, our visualization has appeared.

2. Suppose now that we want to create a histogram and a boxplot to explore the distribution of color intensity. Let’s use the following code to build this new visualization.

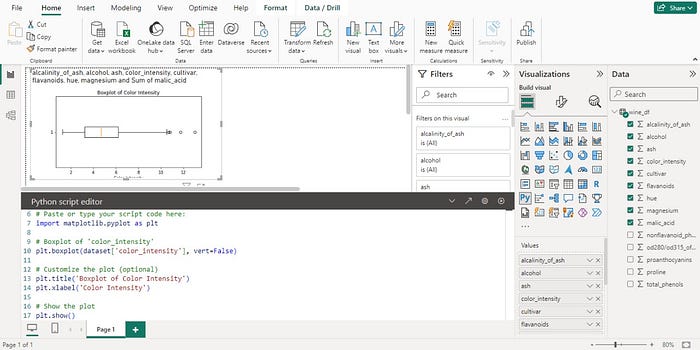
***Histogram***

import matplotlib.pyplot as plt  
  
# Histogram of 'color\_intensity'  
plt.hist(dataset['color\_intensity'], bins=10, color='red', edgecolor='black')  
  
# Customize the plot (optional)  
plt.title('Histogram of Color Intensity')  
plt.xlabel('Color Intensity')  
plt.ylabel('Frequency')  
  
# Show the plot  
plt.show()



***Boxplot***

import matplotlib.pyplot as plt  
  
# Boxplot of 'color\_intensity'  
plt.boxplot(dataset['color\_intensity'], vert=False)  
  
# Customize the plot (optional)  
plt.title('Boxplot of Color Intensity')  
plt.xlabel('Color Intensity')  
  
# Show the plot  
plt.show()



**5. Drawbacks: Considerations and Limitations**

**Considerations and Limitations when Using Python in Power BI:**

1. **Public Access for Data Sources:** Data sources added with Python must be set to public, ensuring accessibility for collaboration and sharing within Power BI.
2. **Limited to Pandas DataFrames:** Only Pandas DataFrames can be imported into Power BI using Python, limiting the supported data structures.
3. **Timeout for Long-Running Scripts and Visuals:**Scripts exceeding 30 minutes and Python visuals surpassing 5 minutes in runtime will encounter timeouts, impacting execution in Power BI.
4. Interactive calls in Python script such as waiting for user input halting the script's execution and, when setting the working directory within the Python script you must define a full path to the working directory rather than a relative path.

*Nested tables are currently not supported.*

1. **Restricted Python Libraries:**Power BI Service supports a restricted set of Python libraries, imposing limitations on the available libraries for data manipulation and analysis.
2. **Refresh via Personal Gateway:** Reports using Python can only be refreshed in Power BI Service through a personal gateway, necessitating this intermediary for efficient data updates.
3. **Limited Cross-Filtering in Python Visuals:** Python visuals do not support cross-filtering, meaning selections within a Python visual won’t propagate to filter other visuals, reducing interactivity in Power BI reports.

**Additional Considerations:**

* Power BI Service imposes restrictions on Python libraries; however, these limitations don’t apply when building reports in Power BI Desktop.
* Power BI Service currently supports Python 3.7.7 runtime. Ensure compatibility of your Python scripts, as earlier versions may lead to runtime issues.

**Supported Libraries in Power BI Service:**

* Matplotlib
* NumPy
* Pandas
* Scikit-learn
* Scipy
* Seaborn
* Statsmodels
* XGBoost

Stay mindful of these considerations to optimize your experience when integrating Python with Power BI.

*References:*[*https://learn.microsoft.com/en-us/power-bi/connect-data/desktop-python-scripts*](https://learn.microsoft.com/en-us/power-bi/connect-data/desktop-python-scripts)

[*https://www.datacamp.com/tutorial/running-python-scripts-in-power-bi-tutorial*](https://www.datacamp.com/tutorial/running-python-scripts-in-power-bi-tutorial)