Using Python in Power BI

Dan Vukota

linkedin.com/in/danvukota/

Advantages of using Python in Power BI

Using Python scripts in Power BI can enhance functionality, enable reusable code, automate repetitive tasks, and provide advanced capabilities not natively available within Power BI. Some examples of practical tasks that can be performed by Python in Power BI are:

- Data cleaning
- Handling of missing data
- Data transformation
- Machine learning
- Advanced visualizations
- Custom Data Source Connectivity

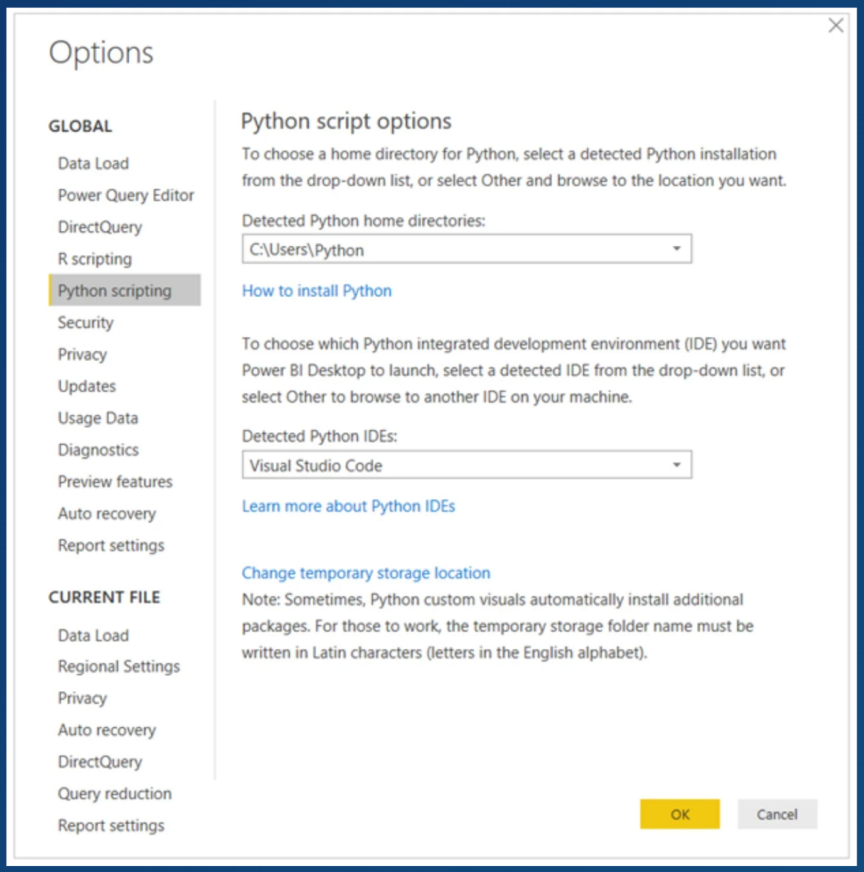
Limitations of using Python in Power BI

- Only Pandas dataframes can be used
- Execution time limits for both Python scripts and Python visuals
- Python scripts and visuals can only be refreshed through a Personal Gateway
- Python visuals are static
- Data sources added with Python must be public or have proper authentication
- A limited number of Python libraries are supported by Power BI Service

https://learn.microsoft.com/en-us/power-bi/connect-data/service-python-packages-support

Minimum Requirements for using Python

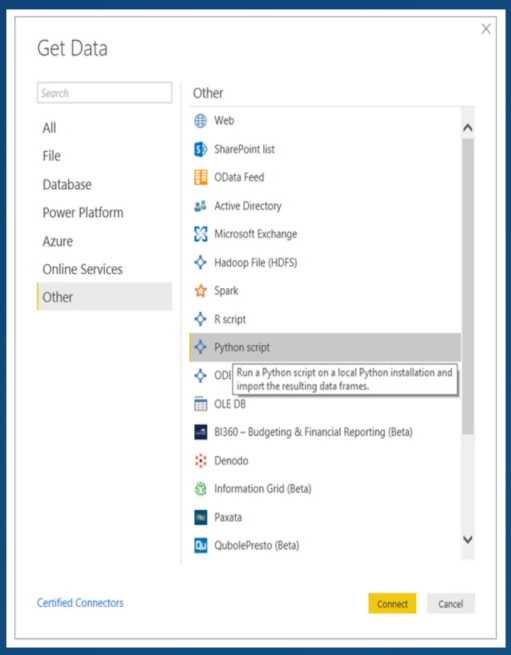
- Python needs to be installed on your local machine
- Installation of the Pandas package
- Python scripting settings in Power BI Options:



https://learn.microsoft.com/en-us/power-bi/connect-data/desktop-python-scripts

Examples of Python script use

- **Importing Python script as a data source**



Use Get Data → Other → Python script to import data from Python script.

Power BI only supports implicit global variables called dataset or data in the script, which holds the current state of the data in the pipeline, represented as a Pandas dataframe. Only dataframes can be used as a data source.

The following Python script example will create an **employees** dataframe with **Employee**, **Department** and Age columns along with values:

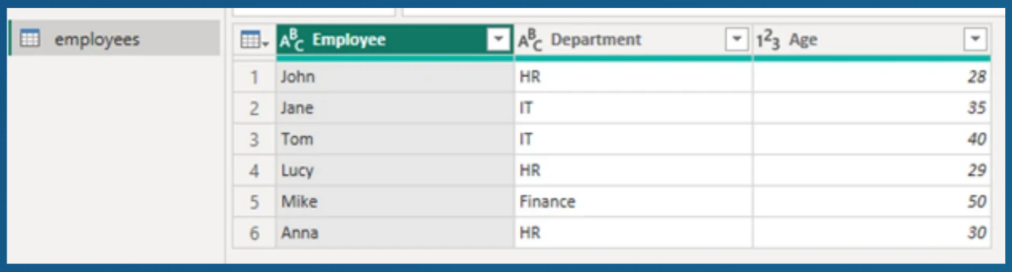
```
import pandas as pd

# Sample data: employee, department, and age
dataset = {
    'Employee': ['John', 'Jane', 'Tom', 'Lucy', 'Mike', 'Anna'],
    'Department': ['HR', 'IT', 'IT', 'HR', 'Finance', 'HR'],
    'Age': [28, 35, 40, 29, 50, 30]
}

# Creating a DataFrame
employees = pd.DataFrame(dataset)
```

Examples of Python script use

- **Embedding Python script in Power Query**

Use Transform Data → Transform → Run Python Script to run a Python script on a data source.

We will use the same employees Python data source we imported for this example.



employees

	Employee	Department	Age
1	John	HR	28
2	Jane	IT	35
3	Tom	IT	40
4	Lucy	HR	29
5	Mike	Finance	50
6	Anna	HR	30

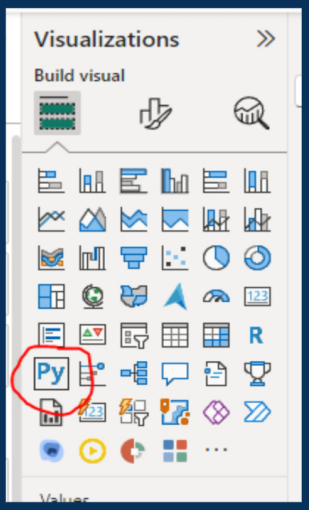
The following Python script example will use the imported dataset and create a new dataset by calculating the average age by **Department**. The **Age** column is renamed, the original dataset is dropped, and the new **dept_avg_age** dataset is returned as output.

```
import pandas as pd

# Group by 'Department' and calculate the average age
dept_avg_age = dataset.groupby('Department')['Age'].mean().reset_index()

# Rename the 'Age' column to 'Avg Age'
dept_avg_age.rename(columns={'Age': 'Avg_Age'}, inplace=True)

# drop original dataset table
del dataset

# Output dept_avg_age table
dept_avg_age
```

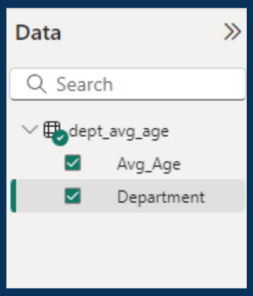
Examples of Python script use

- **Adding a Python Visual to the Report**

In the Report View, choose the Python Visual in the Visualizations Pane.

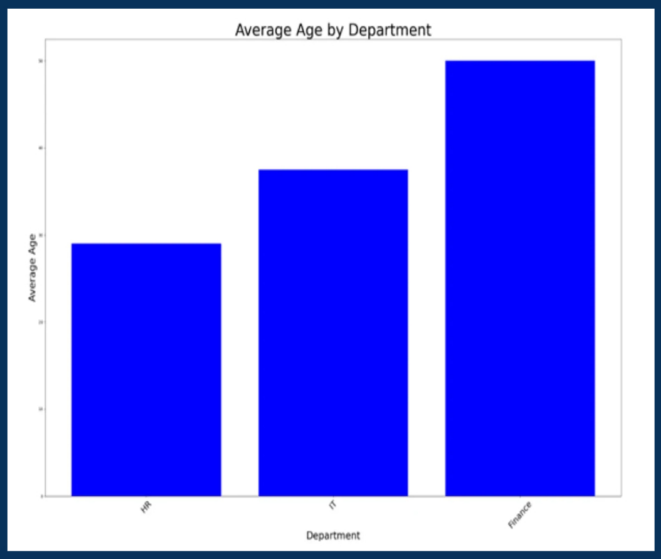


We can select any of the columns in **dept_avg_age** data source in the Data Pane that we previously created for this example.



The following Python script example using the Python script editor will use the imported dataset and create a bar chart of the Average Age by Department using the matplotlib package. Clicking on the run button in the right corner will generate the following bar chart:

```
import pandas as pd
import matplotlib.pyplot as plt

# Create a bar chart
plt.figure(figsize=(30,20))
plt.bar(dataset['Department'],
    dataset['Avg_Age'], color='blue')
plt.xlabel('Department', fontsize=26)
plt.ylabel('Average Age', fontsize=26)
plt.title('Average Age by Department',
     fontsize=40)
plt.xticks(fontsize=22)
plt.xticks(rotation=45)
plt.show()
```



Thank You

Comment

Repost

Save

linkedin.com/in/danvukota/

Dan Vukota