RASONABE, CHIARA MAE P.
BSCS3C

Dictionaries

1. **Creation of New Dictionary**
   - This refers to the process of making a new dictionary in Python. A dictionary is a collection of key-value pairs, and a new dictionary can be created using the syntax `dictionary_name = {}`.

```
#1 - Creation of New Dictionary
my_dict = {"name": "Chiara", "age": 21}
print(my_dict)
```

```
{'name': 'Chiara', 'age': 21}
```

2. **Accessing Items in the Dictionary**
   - This involves retrieving the value of a specific key from a dictionary. You can do this by specifying the name of the dictionary followed by the key in square brackets, like `dictionary_name["key"]`.

```
#2 - Accessing Items in the Dictionary
print(my_dict["name"])
```

```
Chiara
```

3. **Change Values in the Dictionary**
   - This involves modifying the value of a specific key in a dictionary. This can be done by assigning a new value to a specific key, like `dictionary_name["key"] = "new value"`.

```
#3 - Change Values in the Dictionary
my_dict["age"] = 21
print(my_dict)
```

```
{'name': 'Chiara', 'age': 21}
```

4. **Loop Through a Dictionary Values**
    - This involves iterating over all the values in a dictionary. This can be done using a for loop and the `.values()` method, like `for value in dictionary_name.values():`.

```
#4 - Loop Through a Dictionary Values
for key in my_dict:
    print(my_dict[key])
```

```
Chiara
21
```

5. **Check if Key Exists in the Dictionary**
    - This involves verifying whether a specific key exists within a dictionary. This can be done using the `in` keyword, like `"key" in dictionary_name`.

```
#5 - Check if Key Exists in the Dictionary
if "name" in my_dict:
    print("Name exists in the dictionary")
```

```
Name exists in the dictionary
```

6. **Checking for Dictionary Length**
    - This refers to finding out how many key-value pairs are in a dictionary. This can be done using the `len()` function, like `len(dictionary_name)`.

```
#6 - Checking for Dictionary Length
print(len(my_dict))
```

```
2
```

7. **Adding Items in the Dictionary**
   - This involves adding a new key-value pair to a dictionary. This can be done by assigning a value to a new key, like `dictionary_name["new key"] = "new value"`.

```
#7 - Adding Items in the Dictionary
my_dict["email"] = "rasonabechiaramae_bscs@plmun.edu.ph"
print(my_dict)
```

```
{'name': 'Chiara', 'age': 21, 'email': 'rasonabechiaramae_bscs@plmun.edu.ph'}
```

8. **Removing Items in the Dictionary**
   - This involves deleting a key-value pair from a dictionary. This can be done using the `del` keyword, like `del dictionary_name["key"]`.

```
#8 - Removing Items in the Dictionary
my_dict.pop("age")
print(my_dict)
```

```
{'name': 'Chiara', 'email': 'rasonabechiaramae_bscs@plmun.edu.ph'}
```

9. **Remove an Item Using del Statement**
   - Similar to the previous point, this involves deleting a key-value pair from a dictionary using the `del` keyword.

```
#9 - Remove an Item Using del Statement
del my_dict["name"]
print(my_dict)
```

```
{'email': 'rasonabechiaramae bscs@plmun.edu.ph'}
```

10. The dict() Constructor

- This refers to creating a new dictionary using the `dict()` function in Python. This function can take in key-value pairs as arguments and return a new dictionary.

```
#10 - The dict() Constructor
new_dict = dict(brand="LEGO", model="Bricks", year=1932)
print(new_dict)
```

```
{'brand': 'LEGO', 'model': 'Bricks', 'year': 1932}
```

11. Dictionary Methods

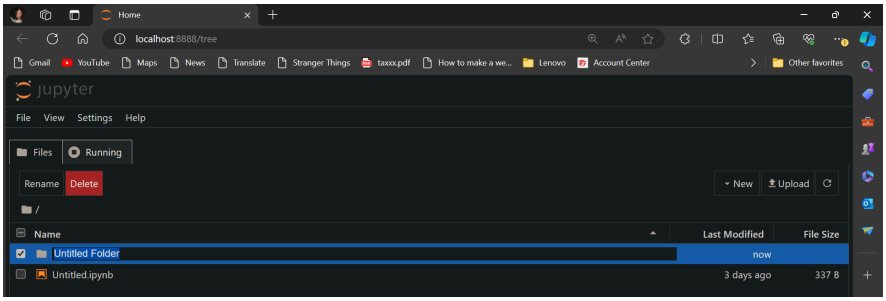
- These are built-in functions in Python that can be used to perform various operations on dictionaries. Some examples include `.keys()`, `.values()`, and `.items()`.

```
#11 - Dictionary Methods
print(my_dict.get("name", "Default Name"))
```

```
Default Name
```

Jupyter notebook

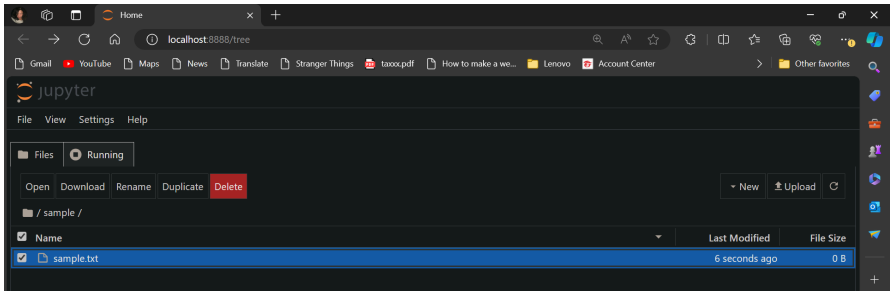
- **Adding Folder**
    - In Jupyter Notebook, folders are usually managed outside the interface, in your system's file explorer. However, you can organize your work by creating folders to hold your notebooks and data files.



- **Adding Text file**

  You can create a text file directly in Jupyter Notebook:

  1. Navigate to the location inside Jupyter where you want the file.
  2. Click on the "New" button at the top right and select "Text File".
  3. This creates a new text file where you can write your content. Rename the file as necessary.

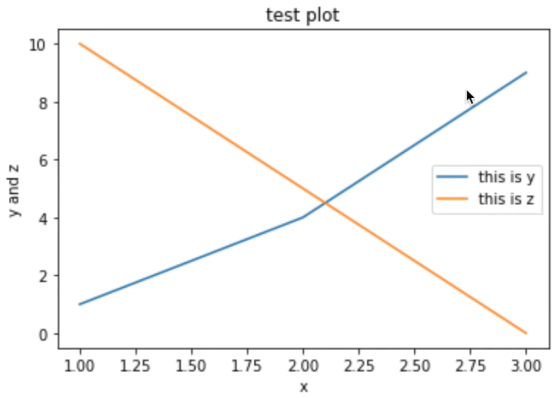


- **CSV file for data analysis and visualization**
    - CSV files are commonly used for data analysis. You can add an existing CSV file to your project folder or create a new CSV file manually or through scripts.

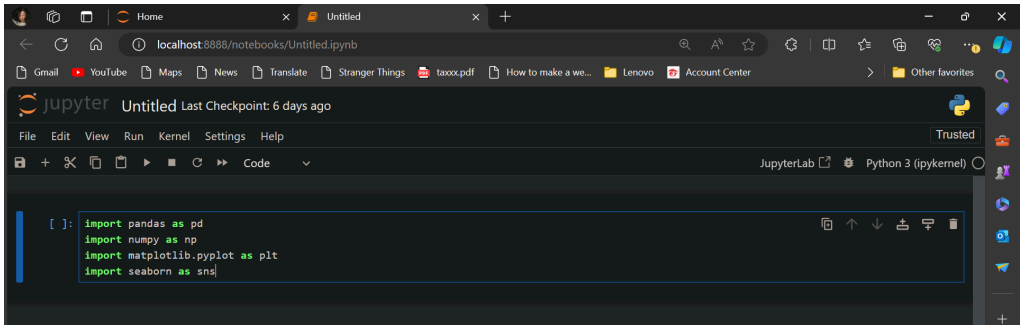
```
In [1]: import pandas as pd

In [2]: from matplotlib import pyplot as plt
```

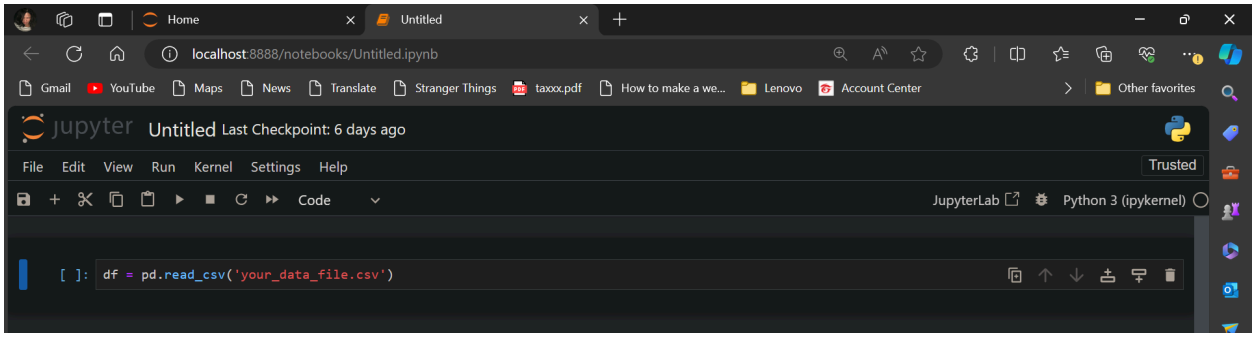
```
In [5]: x = [1, 2, 3]
        y = [1, 4, 9]
        z = [10, 5, 0]
        plt.plot(x, y)
        plt.plot(x, z)
        plt.title("test plot")
        plt.xlabel("x")
        plt.ylabel("y and z")
        plt.legend(["this is y", "this is z"])
        plt.show()
```



- **Import libraries**
    - To perform data analysis and visualization in Jupyter Notebook, you need to import the necessary libraries. The most common libraries include:



- **Finding data**
    - Finding the right data for your analysis depends on your project. Data can be sourced from public datasets (e.g., Kaggle, UCI Machine Learning Repository), APIs, or collected through surveys and experiments.
- **Importing data**
    - To import data into Jupyter Notebook, ensure the data file (e.g., CSV) is in your project folder. Use pandas to read the file:




```
In [7]: sample_data = pd.read_csv('sample_data.csv')

In [8]: sample_data

Out[8]:
```

	column_a	column_b	column_c
0	1	1	10
1	2	4	8
2	3	9	6
3	4	16	4
4	5	25	2

- **Data attributes**

Once your data is imported, you can explore its attributes to understand its structure, size, and type of data it contains:

- **View the first few rows:** 'df.head()'
- **Data shape:** 'df.shape' to see the number of rows and columns.
- **Data types:** 'df.dtypes' to check the data type of each column.
- **Summary statistics:** 'df.describe()' for a statistical summary of numerical columns.
- **Column names:** 'df.columns' to get a list of all column names.
- **Check for missing values:** 'df.isnull().sum()' to see if there are missing values in the data.