

CS PROFESSIONAL ELECTIVE

DICTIONARIES:

Creation of a New Dictionary:

- You can create an empty dictionary using curly braces {}. - You can also create a dictionary with key-value pairs separated by colons, all enclosed in curly braces.

Here's an example:

```
my_dict = {"apple": "Fruit", "banana": "Fruit"}
```

Accessing Items in the Dictionary:

- Use square brackets [] to access an item by its key. The key must exist in the dictionary, otherwise you'll get a KeyError.

```
fruit = my_dict["apple"] # fruit will now contain "Fruit"
```

Changing Values in the Dictionary:

- Use the same square bracket notation to modify an existing key's value.

```
my_dict["apple"] = "Red Fruit"
```

Looping Through Dictionary Values: - You can use a for loop to iterate over the values in the dictionary.

```
for value in my_dict.values(): print(value)
```

Checking if a Key Exists: - Use the in operator to check if a key exists in the dictionary.

```
if "apple" in my_dict: print("apple key exists")
```

CS PROFESSIONAL ELECTIVE

DICTIONARIES:

Checking Dictionary Length:

- Use the len() function to get the number of key-value pairs in the dictionary.

```
print(len(my_dict))
```

Adding Items to the Dictionary: - Use square brackets with a new key and its value to add an item.

```
my_dict["banana"] = "Yellow Fruit" print(my_dict)
```

Removing Items from the Dictionary: - The pop() method removes a key-value pair and returns the value. If the key doesn't exist, it throws a KeyError. You can provide a default value to avoid the error.

```
my_dict.pop("banana") # Removes the banana key-value pair
```

Removing Items Using del:

- The del statement removes a key-value pair.

```
del my_dict["apple"]
```

The dict() Constructor: - An alternative way to create dictionaries is using the built-in dict() constructor. You can pass keyword arguments or another dictionary-like object.

```
my_dict = dict(name="John", age=30)
```

CS PROFESSIONAL ELECTIVE

DICTIONARIES:

Dictionary Methods: - Dictionaries have several methods for working with them. Here are a few common ones:

- `get(key, default)`: Returns the value for the key, or the default value if the key doesn't exist.
- `keys()`: Returns a view of all the keys in the dictionary.
- `items()`: Returns a view of all the key-value pairs in the dictionary as tuples.

Adding Folders and Files: - You can use the file explorer in Jupyter Notebook to add folders and files to your project.

CSV Files for Data Analysis and Visualization:* - Jupyter Notebook is great for working with data. You can import libraries like Pandas to read CSV files, analyze the data, and create visualizations.

Importing Libraries:* - Use the import statement to import libraries like Pandas or NumPy for data manipulation and analysis.

Finding Data: - You can search for publicly available datasets online or use your own data files.

Importing Data:* - Once you have your data, use Pandas functions like `pd.read_csv()` to import the data into a DataFrame object for further analysis.

Data Attributes:* - After importing data, you can explore its attributes like the number of rows and columns, data types, etc.