Name: Katricia Lopez
Sec:CS3C

Dictionaries

Creation of New Dictionary:

Use curly braces `{}` to create an empty dictionary or use key-value pairs within the braces to initialize a dictionary.

Example:

```
# Creation of New Dictionary
my_dict = {"apple": 5, "banana": 3, "orange": 2}
```

Accessing Items in the Dictionary:

Use square brackets `[]` with the key to access the value associated with that key.

Example:

```
# Accessing Items in the Dictionary

print(my_dict["apple"])
```

Change Values in the Dictionary:

Simply assign a new value to the key.

Example:

```
# Change Values in the Dictionary

my_dict["banana"] = 6
```

Loop Through a Dictionary Values:

Use a loop like `for key, value in dictionary.items():` to iterate through key-value pairs.

Example:

```
# Loop Through a Dictionary Values
for fruit, quantity in my_dict.items():
    print(f"There are {quantity} {fruit}s")
```

Check if Key Exists in the Dictionary:

Use the `in` keyword to check if a key exists in the dictionary.

Example:

```
# Check if Key Exists in the Dictionary
if "apple" in my_dict:
    print("Yes, 'apple' is a key in the dictionary")
```

Checking for Dictionary Length:

Use the `len()` function to get the number of items in the dictionary.

Example:

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

Adding Items in the Dictionary:

Simply assign a value to a new key.

Example:

```
# Adding Items in the Dictionary
my_dict["grape"] = 4
```

Removing Items in the Dictionary:

Use the `pop()` method to remove an item with a specific key.

Example:

```
# Removing Items in the Dictionary

my_dict.pop("orange")
```

Remove an Item Using del Statement:

Use the `del` statement followed by the key to remove an item.

Example:

```
# Remove an Item Using del Statement

del my_dict["banana"]
```

The `dict()` Constructor:

You can create a dictionary using the `dict()` constructor and pass key-value pairs as arguments.

Example:

```
# The dict() Constructor

new_dict = dict(apple=5, banana=3, orange=2)
```

Jupyter Notebook:

Adding Folder:

Use the file explorer interface in Jupyter Notebook to create a new folder.

Example:

In Jupyter, you can add a folder by clicking on the "New" button in the file explorer and selecting "Folder".

Adding Text file:

Similarly, use the file explorer interface to create a new text file.

Example:

Similarly, you can add a text file by clicking on "New" and selecting "Text File".

CSV file for data analysis and visualization:

You can add a CSV file by uploading it through the file explorer interface or using the `pandas` library to read CSV files directly.

Example:

You can import a CSV file containing data for analysis and visualization.

Import libraries:

Use `import` statements to import necessary libraries like `pandas`, `matplotlib`, etc.

Example:

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

Finding data:

Locate the dataset you want to analyze and make sure it's in a format that can be read by Python (e.g., CSV, Excel, etc.).

Example:

You can find data from various sources like Kaggle, UCI Machine Learning Repository, government databases, etc.

Importing data:

Use library functions like `pandas.read_csv()` to import data into your Jupyter Notebook environment.

Example:

```
# Assuming 'data.csv' is the name of your CSV file
data = pd.read_csv('data.csv')
```

Data attributes:

Once data is imported, explore its attributes using methods and attributes specific to the data type (e.g., DataFrame in the case of pandas). You can use methods like `info()`, `head()`, `describe()` to get an overview of the data.

Example:

```
# Print the first few rows of the dataset
print(data.head())

# Print column names
print(data.columns)

# Summary statistics
print(data.describe())
```