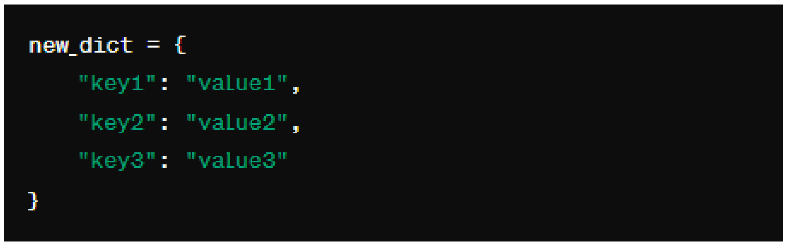
Ramila Jane B. Abobo

CS3C

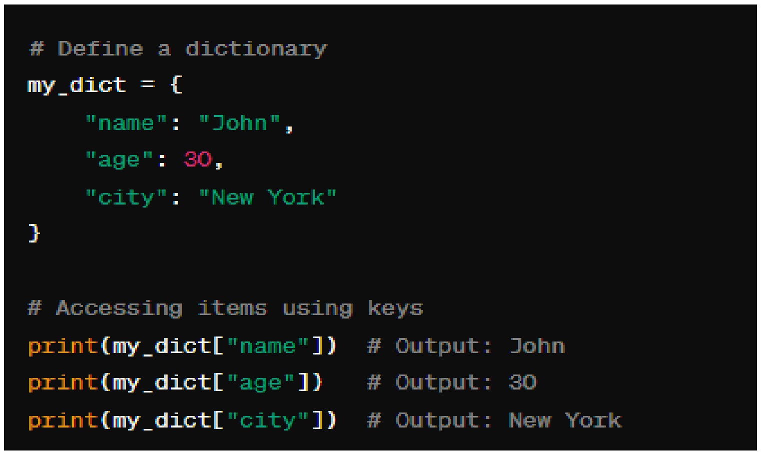
Dictionaries

- Creation of New Dictionary
  - ✓ In Python, you may define a new dictionary by defining it using curly brackets {} and providing key-value pairs.



```python
new_dict = {
    "key1": "value1",
    "key2": "value2",
    "key3": "value3"
}
```

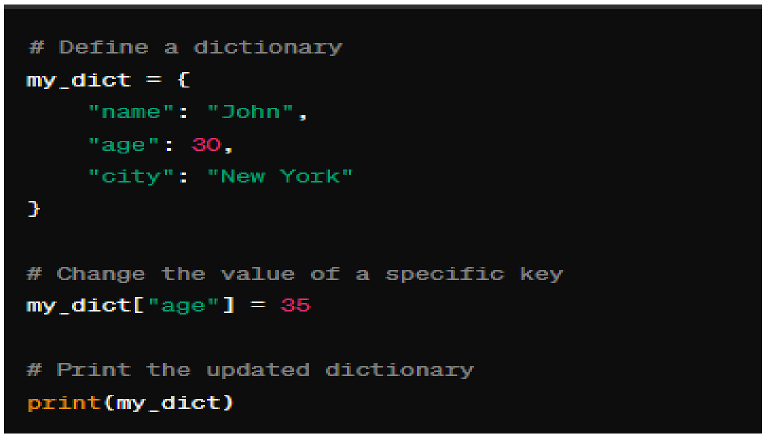
- Accessing Items in the Dictionary
  - ✓ Python keys are used to access things in dictionaries.



```python
# Define a dictionary
my_dict = {
    "name": "John",
    "age": 30,
    "city": "New York"
}

# Accessing items using keys
print(my_dict["name"])  # Output: John
print(my_dict["age"])   # Output: 30
print(my_dict["city"])  # Output: New York
```

- Change Values in the Dictionary
  - ✓ In Python, all you have to do to update the values in a dictionary is retrieve the key and set a new value for it.



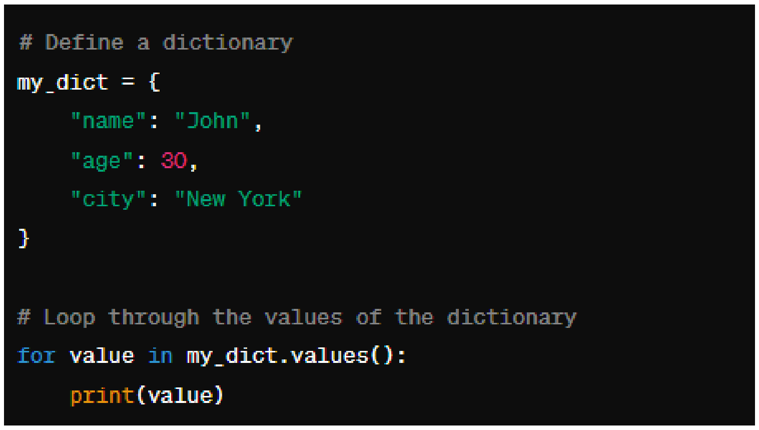
```python
# Define a dictionary
my_dict = {
    "name": "John",
    "age": 30,
    "city": "New York"
}

# Change the value of a specific key
my_dict["age"] = 35

# Print the updated dictionary
print(my_dict)
```

● Loop Through a Dictionary Values

✓ In Python, a for loop can be used to iterate through a dictionary's values.



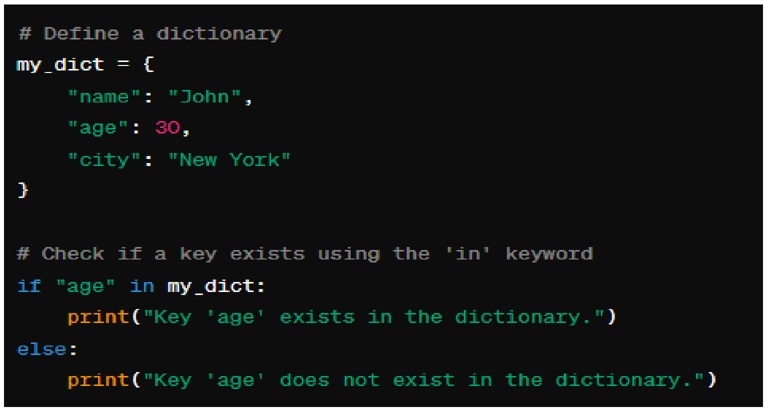
```python
# Define a dictionary
my_dict = {
    "name": "John",
    "age": 30,
    "city": "New York"
}

# Loop through the values of the dictionary
for value in my_dict.values():
    print(value)
```

● Check if Key Exists in the Dictionary

✓ In Python, you can use the get() function or the in keyword to see if a key is present in the dictionary.

‘in’

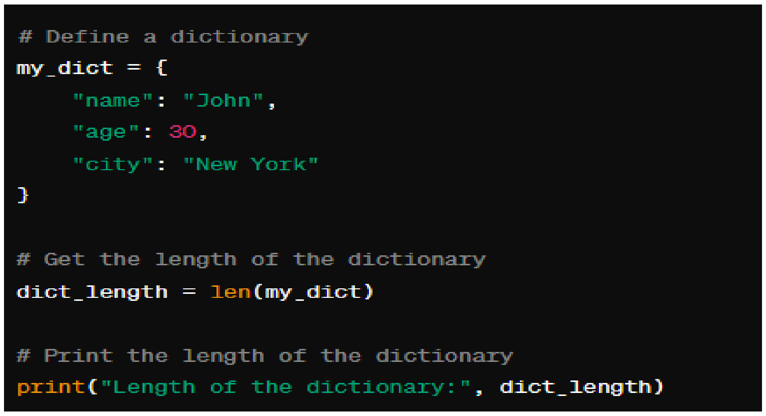


```python
# Define a dictionary
my_dict = {
    "name": "John",
    "age": 30,
    "city": "New York"
}

# Check if a key exists using the 'in' keyword
if "age" in my_dict:
    print("Key 'age' exists in the dictionary.")
else:
    print("Key 'age' does not exist in the dictionary.")
```

● Checking for Dictionary Length

✓ The built-in len() method in Python can be used to find the length of a dictionary. The number of key-value pairs in the dictionary is returned by this function.

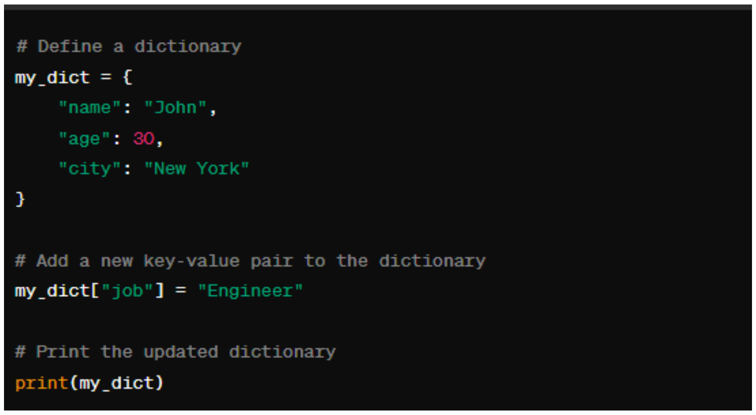


```python
# Define a dictionary
my_dict = {
    "name": "John",
    "age": 30,
    "city": "New York"
}

# Get the length of the dictionary
dict_length = len(my_dict)

# Print the length of the dictionary
print("Length of the dictionary:", dict_length)
```

- Adding Items in the Dictionary
  - ✓ In Python, adding items to a dictionary is as easy as updating an existing key with a new value or giving a new key a value.

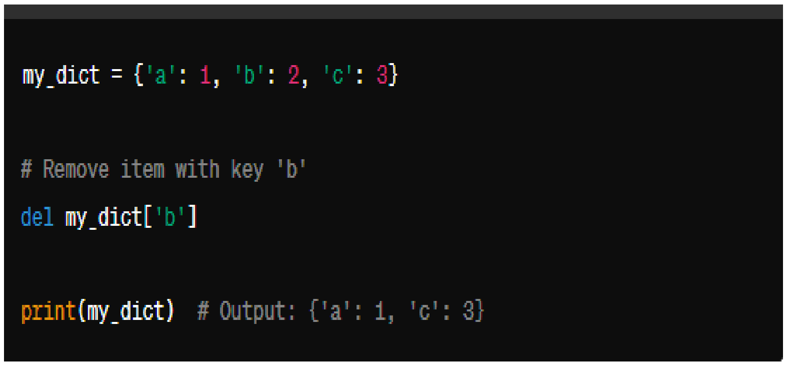


```python
# Define a dictionary
my_dict = {
    "name": "John",
    "age": 30,
    "city": "New York"
}

# Add a new key-value pair to the dictionary
my_dict["job"] = "Engineer"

# Print the updated dictionary
print(my_dict)
```

- Removing Items in the Dictionary
  - ✓ In Python, the del keyword can be used to remove entries from a dictionary.

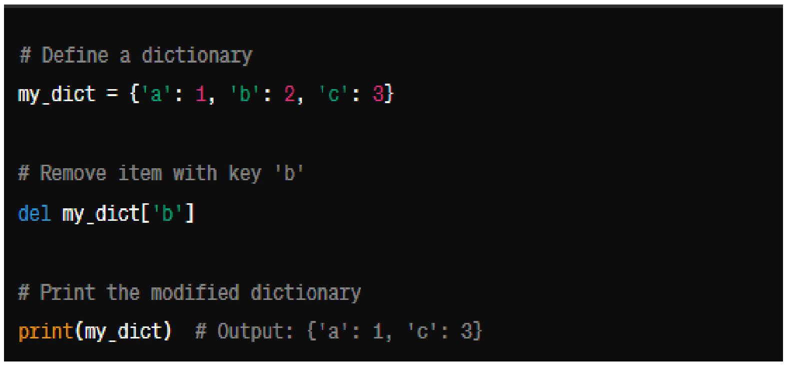


```python
my_dict = {'a': 1, 'b': 2, 'c': 3}

# Remove item with key 'b'
del my_dict['b']

print(my_dict)  # Output: {'a': 1, 'c': 3}
```

- Remove an Item Using del Statement
  - ✓ Python's del statement requires users to supply the item's key in order to remove an entry from a dictionary.



```python
# Define a dictionary
my_dict = {'a': 1, 'b': 2, 'c': 3}

# Remove item with key 'b'
del my_dict['b']

# Print the modified dictionary
print(my_dict)  # Output: {'a': 1, 'c': 3}
```

● The dict() Constructor

✓ The dict() constructor in Python is used to create a new dictionary. You can pass it various types of arguments to initialize the dictionary.
✓ Creating an empty dictionary:
empty_dict = dict()
✓ Creating a dictionary from key-value pairs:
my_dict = dict({'a': 1, 'b': 2, 'c': 3})

● Dictionary Methods

✓ There are numerous ways to carry out common operations on Python dictionaries.

✓ **dict.keys()**: Returns a view object that displays a list of all the keys in the dictionary.
✓ **dict.values()**: Returns a view object that displays a list of all the values in the dictionary.
✓ **dict.items()**: Returns a view object that displays a list of key-value tuple pairs.
✓ **dict.keys():** Returns a view object that displays a list of all the keys in the dictionary.
✓ **dict.values():** Returns a view object that displays a list of all the values in the dictionary.
✓ **dict.items():** Returns a view object that displays a list of key-value tuple pairs.

Jupyter notebook

● Adding Folder

✓ The Jupyter interface is usually used to add a folder in Jupyter Notebook. Whether you are using JupyterLab or the traditional Jupyter Notebook interface.
- JupyterLab:

✓ Open JupyterLab in your web browser.
✓ Locate the folder where you want to add a new folder.
✓ Right-click on the folder or use the "File" menu.
✓ Choose "New Folder" from the context menu.
✓ Enter the name of the new folder when prompted.
✓ Press Enter to create the folder.

● Adding Text file

✓ Navigate to the desired directory
✓ Click on "New"
✓ Rename the file
✓ Edit the file
✓ Save the file

● CSV file for data analysis and visualization

✓ Step 1: Creating a CSV File
✓ Step 2: Loading CSV Data in Jupyter Notebook
✓ Step 3: Data Analysis and Visualization
✓ Step 4: Interpret Results

- Import libraries
  - ✓ Import the libraries that are required for data analysis and visualization first. For data manipulation, Pandas is a frequently used library; for data visualization, Matplotlib or Seaborn are recommended.
- Finding data
  - ✓ Data can be found through a variety of sources, including self-collection, internet repositories, and APIs. Popular websites for finding datasets are Kaggle, UCI Machine Learning Repository, and government data portals.
- Importing data
  - ✓ You must import your data into your Jupyter Notebook after you've located it. Use Pandas' read_csv() function if your data is in a CSV file. If it's in a different format, Pandas can handle a wide range of formats, including Excel, JSON, SQL, and more.
- Data attributes
  - ✓ You can use several Pandas methods to investigate the attributes of your data after it has been imported.
  - ✓ Shape: To get the dimensions of the DataFrame (number of rows and columns).
  - ✓ Columns: To get the column names.
  - ✓ Data Types: To get the data types of each column.