Junior Knights Python I - Week 13

Dictionaries

Agenda

Creating Dictionaries

Accessing Values

Modifying Dictionaries

Creating Dictionaries

- A dictionary is a built-in data type that stores data in key-value pairs.
- Keys are unique and immutable (e.g., strings, numbers, tuples).
- Values can be of any type

- Creating a dictionary:
  - Using curly braces:
    - `my_dict = {"name": "Tom", "age": 25, "city": "New York"}`
  - Using dict() constructor:
    - `my_dict = dict(name="Tom", age="25", city="New York")`

Accessing Values

- Accessing like a list: dictionary[key]
  - ```
    my_dict = {"name": "Tom", "age": 25, "city": "New York"}
    print(my_dict["name"])
    ```
- Using get(key) to avoid KeyError, which happens when you try to access a key in a dictionary that doesn't exist
  - ```
    print(my_dict.get("email", "Not Found"))
    ```
- keys() returns a list of all the keys
  - ```
    print(my_dict.keys())
    ```
- values() returns a list of all the values
  - ```
    print(my_dict.values())
    ```
- items() returns tuples of all (key, value) pairs
  - ```
    print(my_dict.items())
    ```
    ```

Accessing Values

- Accessing like a list: dictionary[key]
  - ```
    my_dict = {"name": "Tom", "age": 25, "city": "New York"}
    print(my_dict["name"])      # Output: Tom
    ```
- Using get(key) to avoid KeyError, which happens when you try to access a key in a dictionary that doesn't exist

  ```
  print(my_dict.get("email", "Not Found"))  # Output: Not Found
  ```
- keys() returns a list of all the keys
  - ```
    print(my_dict.keys())                  # Output: ("name", "age",
    "city")
    ```
- values() returns a list of all the values
  - ```
    print(my_dict.values())     # Output: ("Tom", 25, "New York")
    ```
- items() returns tuples of all (key, value) pairs
  - ```
    print(my_dict.items())                 # Output: dict_items([('name',
    'Tom'),
    ```

Accessing Values

- Accessing Keys and Values

```
my_dict = {"name": "Tom", "age": 25,"city": "New York"}

for key, value in my_dict.items():
        print(f"Key: {key}, Value: {value}")
```

- Accessing Keys only

```
for key in my_dict:
    print(f"Key: {key}")
```

- Accessing Values only

```
for value in my_dict.values():
    print(f"Value: {value}")
```

Checking for Items

- Checking for keys

```
my_dict = {"name": "Tom", "age": 25,"city": "New York"}

"name" in my_dict                    # True
"age" in my_dict                     # True
"email" in my_dict                   # False
```

- Checking for values

```
"Tom" in my_dict.values()  # True
"33" in my_dict.values()   # False
```

Modifying Dictionaries

- Adding a new key
  - `my_dict["email"] = "tom@gmail.com"  # Adds new key email & value`

- Updating an existing value
  - `my_dict["age"] = 26  # Updates existing key age`

Removing Items

- Popping: deleting and returning the item
  - `my_dict.pop("email")  # Removes key "email"`
- Deleting a key
  - `del my_dict['age']  # Removes key "age"`
- Removing all items
  - `my_dict.clear()  # Removes all items`
- Deleting the dictionary
  - `del my_dict  # Deletes dictionary`

Methods Review

Function	Purpose
`dict.get(key)`	Return value or None if key is not found
`dict.keys()`	Returns a view of all keys
`dict.values()`	Returns a view of all values
`dict.items()`	Returns a view of (key, value) pairs
`dict["email"] = "tom@gmail.com"`	Adds key or updates value of key “email”
`dict.pop("email")`	Removes key “email”
`dict['age'`	Removes key “age”
`dict.clear()`	Removes all items
`del my_dict`	Deletes dictionary

Live Coding