Dictionaries

Key Concepts:

- Dictionary Data Structure
- Creating and Using Dictionaries
- Modifying Dictionaries
- Traversing Dictionaries
- Copying Dictionaries

Dictionary Data Structure

Introduction:

- Like lists, dictionaries allow storing multiple values of various types.
- Unlike lists, dictionary values are not stored in any particular order.

Creating and Using Dictionaries

Syntax:

```
D = { key1: value1, key2: value2, ..., keyN: valueN }
```

Example:

```
notes = { 'quentin': 15.5, 'nathan': 12.0 }
```

- The variable `notes` is a dictionary containing the grades of two students.
- The strings `'nathan'` and `'quentin'` are the keys, while `12.0` and `15.5` are the values.
- Dictionary elements are unordered.

```
print(notes)  # Output: {'nathan': 12.0, 'quentin': 15.5}
```

Accessing Values:

- Access a value using its key.

```
notes = { 'nathan': 12.0, 'quentin': 15.5 }
```

```
print(notes['quentin'])  # Output: 15.5
```

- Dictionaries are also known as "associative arrays" because they associate each key with a value of any type.

Modifying Dictionaries

Adding New Entries:

- Use the assignment operator to add a new key-value pair to an existing dictionary.

```
D = {}  # creates an empty dictionary
D['a'] = 1  # adds a new entry
print(D)  # Output: {'a': 1}
```

- If the key already exists, it updates the value.

```
D['a'] = 3
print(D)  # Output: {'a': 3}
```

Removing Entries:

- Use the `del` operator to remove a key-value pair.

```
D = {'a': 1, 'b': 2, 'c': 3}
del D['a']
print(D)  # Output: {'b': 2, 'c': 3}
```

Checking for Existence:

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

```
prices = {'asus': 450, 'alienware': 1200, 'lenovo': 680}
print('asus' in prices)  # Output: True
print('toshiba' in prices)  # Output: False
```

- Note: `in` checks for the presence of a key, not a value.

```
print(1200 in prices)  # Output: False
```

Handling Missing Keys:

- Accessing a non-existent key raises a `KeyError`.

```
letters = {'a': 103, 'b': 8, 'e': 150}
# print(letters['k'])  # Raises KeyError
```

- Always check if the key exists before accessing it.

```
if 'u' in letters:
    letters['u'] = letters['u'] + 1
else:
    letters['u'] = 1
```

Traversing a Dictionary

- Use a `for` loop to iterate over all keys in a dictionary.

```
for key in D:
    print('The key', key, 'has the value:', D[key])
```

Example:

```
birthdays = {'ingrid': [12, 6, 1995], 'marc': [27, 8, 1996], 'brice':
[11, 10, 1995]}

for name in birthdays:
    date = birthdays[name]
    print(name, 'will celebrate their birthday on', date[0], '/',
date[1], '/2017')
```

Output:

```
ingrid will celebrate their birthday on 12 / 6 / 2017
marc will celebrate their birthday on 27 / 8 / 2017
brice will celebrate their birthday on 11 / 10 / 2017
```

Key and Value Types

- Values in a dictionary can be of any type, including other dictionaries.

```
my_pc = {
    'ram': 16,
    'cpu': 3.5,
    'portable': False,
    'os': 'windows',
    'ports': ['usb3.0', 'jack', 'ethernet', 'hdmi'],
    'graphics_card': {
        'vram': 4,
        'name': 'gtx970',
        'bus': 256
    }
}
```

- Only certain types can be used as keys. In this course, we limit ourselves to integers and strings.

Copying Dictionaries

- Assigning a dictionary to another variable references the same dictionary.

```
D = {1: 10, 2: 20, 3: 30}
E = D
E[5] = 50
print(E)  # Output: {1: 10, 2: 20, 3: 30, 5: 50}
print(D)  # Output: {1: 10, 2: 20, 3: 30, 5: 50}
```

- To create a copy, use `dict()`:

```
F = dict(D)
F[6] = 60
print(F)  # Output: {1: 10, 2: 20, 3: 30, 5: 50, 6: 60}
print(D)  # Output: {1: 10, 2: 20, 3: 30, 5: 50}
```