Lecture 05: Data Structures: Dictionaries and Tuples

February 20, 2023

1 Python Data Structures: Dictionaries & Tuples

Ahmed Moustafa

2 Dictionaries

- A dictionary is an unordered collection of key-value pairs.
- Each key is unique and maps to a value.
- The syntax for creating a dictionary is:

[2]:
```
my_dict = {"key1": "value1", "key2": "value2", "key3": "value3"}
my_dict
```

[2]:
```
{'key1': 'value1', 'key2': 'value2', 'key3': 'value3'}
```

You can access values in a dictionary using their keys:

[3]:
```
print(my_dict["key1"])
```

```
value1
```

3 Example

[4]:
```
states = {}                    # Create empty dict
states['ca'] = 'California'    # 1. Set key/value pairs into dict
states['ok'] = 'Oklahoma'
states['nj'] = 'New Jersey'
```

```
states['tx'] = 'Texas'
states
```

[4]:
```
{'ca': 'California', 'ok': 'Oklahoma', 'nj': 'New Jersey', 'tx': 'Texas'}
```

[5]:
```
type(states)
```

[5]:
```
dict
```

```
<IPython.core.display.Image object>
```

Source: Nick Parlante's Python guide

4 Dictionary commonly used functions

- keys(): Returns a list of all the keys in the dictionary
- values(): Returns a list of all the values in the dictionary
- items(): Returns a list of all the key-value pairs in the dictionary

[7]:
```
phonebook = {"Alice": "555-1234", "Bob": "555-5678", "Charlie": "555-9012"}
phonebook
```

[7]:
```
{'Alice': '555-1234', 'Bob': '555-5678', 'Charlie': '555-9012'}
```

[8]:
```
print(phonebook["Alice"])
```

```
555-1234
```

[9]:
```
print(phonebook.keys())
```

```
dict_keys(['Alice', 'Bob', 'Charlie'])
```

[10]:
```
print(phonebook.values())
```

```
dict_values(['555-1234', '555-5678', '555-9012'])
```

[11]:
```
print(phonebook.items())
```

```
dict_items([('Alice', '555-1234'), ('Bob', '555-5678'), ('Charlie',
'555-9012')])
```

[12]:
```
phonebook["John"]
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
Cell In[12], line 1
----> 1 phonebook["John"]
```

```
KeyError: 'John'
```

5 Tuples

- A tuple is an ordered collection of values.
- Tuples are immutable, meaning you can't modify their values once they're created.
- The syntax for creating a tuple is:

[13]:
```
my_tuple = ("value1", "value2", "value3")
my_tuple
```

[13]:
```
('value1', 'value2', 'value3')
```

You can access values in a tuple using their index:

[14]:
```
print(my_tuple[0])
```

```
value1
```

6 Tupples commonly used functions:

- `count()`: Returns the number of times a value appears in the tuple
- `index()`: Returns the index of the first occurrence of a value in the tuple

[15]:
```
days_of_week = ("Monday", "Tuesday", "Wednesday", "Thursday", "Friday",␣
↪"Saturday", "Sunday")
days_of_week
```

[15]:
```
('Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday', 'Saturday', 'Sunday')
```

[16]:
```
print(days_of_week[0])
```

```
Monday
```

[17]:
```
print(days_of_week.count("Monday"))
```

```
1
```

[18]:
```
print(days_of_week.index("Wednesday"))
```

```
2
```

7 Differences between Dictionaries and Tuples

- Dictionaries and tuples are used for different purposes.
- Dictionaries are used to *map* **unique** keys to values, while tuples are used to *store* collections of values.
- Dictionaries are **mutable**, while tuples are **immutable**.
- Dictionaries are **unordered**, while tuples are **ordered**.

8 More Example

Storing information about a person, such as their name, address, and phone number:

[19]:
```
person = {"name": "Jane Doe", "address": "123 Main St", "phone": "555-1234"}
person
```

[19]:
```
{'name': 'Jane Doe', 'address': '123 Main St', 'phone': '555-1234'}
```

Storing a dictionary of translations between different languages:

[20]:
```
translations = {"hello": {"spanish": "hola", "french": "bonjour"}}
translations
```

[20]:
```
{'hello': {'spanish': 'hola', 'french': 'bonjour'}}
```

Representing a point in two-dimensional space:

[21]:
```
point = (3, 5)
point
```

[21]:
```
(3, 5)
```

Storing the RGB color values for a particular color:

[22]:
```
color = (255, 128, 0)
color
```

[22]:
```
(255, 128, 0)
```

9 Exercise: Factorial

Factorial is a mathematical function that is represented by an exclamation mark (!). It is used to find the product of all positive integers from 1 to a given number. For example, the factorial of 5 (represented as 5!) is calculated as 5 x 4 x 3 x 2 x 1, which equals 120. In other words, factorial is the multiplication of all positive integers up to a given number.

Write a program that calculates the factorial of 5.

[23]:
```
n = 5

factorial = 1

for i in range(1, n+1):
    factorial *= i

print(factorial)
```

```
120
```

10 Exercise: Bubble Sort

Bubble Sort is a simple sorting algorithm that repeatedly steps through a list of elements, compares adjacent elements, and swaps them if they are in the wrong order. The algorithm gets its name from the way smaller elements "bubble" to the top of the list as the algorithm executes.

The algorithm works by comparing each pair of adjacent elements in the list from left to right, and swapping them if they are in the wrong order. This process is repeated multiple times until the list is fully sorted.

Here is the basic algorithm: 1. Starting at the beginning of the list, compare each pair of adjacent elements. 2. If the elements are in the wrong order (i.e., the first element is greater than the second), swap them. 3. Continue stepping through the list and comparing adjacent elements, until the end of the list is reached.

Repeat steps 1-3 until no more swaps are needed, indicating that the list is sorted.

```
<IPython.core.display.Image object>
```

Write a Python program to perform a bubble sort on the following list, and display the sorted list as the output

```
items = [5, 3, 8, 6, 7, 2]
```

[25]:
```
items = [5, 3, 8, 6, 7, 2]

n = len(items)
swap = True
while swap:
    swap = False
    for i in range(1, n):
        if items[i-1] > items[i]:
            items[i-1], items[i] = items[i], items[i-1]
            swap = True

# Print the sorted list
print(items)
```

```
[2, 3, 5, 6, 7, 8]
```