Database Systems Lab
Architecture

Christian Rauch

(changed: September 13, 2024)

Lists and Sets

Collections (also called containers) accept elements. In Python, most collections are heterogeneous (i.e., accept different element data types) and mutable (i.e., can be modified after creation).

Lists are indexable and ordered collections.

```
my_list = [1, 4, ’b’, 4, 5]
my_list.append(2)
print(my_list)       # [1, 4, ’b’, 4, 5, 2]
print(my_list[1])    # 4
print(my_list[-2])   # 5
print(my_list[1:4])  # [4, ’b’, 4]
```

Sets contain each element at most once (and guarantee no order).

```
my_set = { 2, 3, "a", 4, 5}
my_set.add(6)
print(my_set)  # {3, ’a’, 2, 4, 5, 6}
my_set.add(3)  # add an existing element
print(my_set)  # {’a’, 4, 2, 3, 5, 6}
```

Dictionaries

Dictionaries are unordered collections of key-value pairs. Each key is unique, and keys are used to access the associated values.

Dictionaries are indexed by keys, not by position.

```
my_dict = {’a’: 1, ’b’: 2}
my_dict[’d’] = 4    # Adding a key-value pair

print(my_dict)      # {’a’: 1, ’b’: 2, ’d’: 4}
print(my_dict[’b’]) # 2

my_dict[’a’] = 10   # Updating value for a key
print(my_dict)      # {’a’: 10, ’b’: 2, ’d’: 4}

del my_dict[’b’]    # Removing a key
print(my_dict)      # {’a’: 10, ’d’: 4}
```

Sorting

Sorting with simple sortable types.

```python
my_list = [5, 2, 9, 1, 5]
my_list.sort()
print(my_list)  # [1, 2, 5, 5, 9]
```

Sorting based on a custom key.

```python
team = [
    {’name’: ’Alice’, ’age’: 22},
    {’name’: ’Bob’, ’age’: 20},
    {’name’: ’Charlie’, ’age’: 23}
]

team = sorted(team, key=lambda x: x[’age’])
print(team)
#  [{’name’: ’Bob’, ’age’: 20},
#   {’name’: ’Alice’, ’age’: 22},
#   {’name’: ’Charlie’, ’age’: 23}]
```

Projection

For complex types you may want to query specific attributes.

```
team = [
    {’name’: ’Alice’, ’age’: 22},
    {’name’: ’Bob’, ’age’: 20},
    {’name’: ’Charlie’, ’age’: 23}
]

names = [s[’name’] for s in team]
print(names) # [’Alice’, ’Bob’, ’Charlie’]
```

Projections may involve transformations.

```
years_as_adult = [(s[’age’]-18) for s in team]
print(years_as_adult)  # [4, 2, 5]
```

Selection

Selecting elements (filtering based on condition).

```
my_list = [1, 4, 6, 7, 9]
evens = [x for x in my_list if x % 2 == 0]
print(evens)  # [4, 6]
```

Alternative approach using filter().

```
my_list = [1, 4, 6, 7, 9]
evens = list(filter(lambda x: x % 2 == 0,
    my_list))
print(evens)  # [4, 6]
```

Combination

Adding another list (concatenation).

```python
list1 = [1, 2, 3]
list2 = [4, 5, 6]
combined = list1 + list2
print(combined)  # [1, 2, 3, 4, 5, 6]
```

Intersecting with another list.

```python
list1 = [1, 2, 3, 4]
list2 = [3, 4, 5, 6]
intersection = [x for x in list1 if x in list2]
print(intersection)  # [3, 4]
```

CSV files

CSV files describe tabular objects.

```
x;y;z
5;3;7
3;2;2
9;1;3
```

Read data from CSV files.

```python
import csv
def read_csv(file_name):
    x = []
    y = []
    with open(file_name, ’r’) as csvfile:
        csvreader = csv.reader(csvfile,
            delimiter=’;’)
        next(csvreader)  # Skip the header
        for row in csvreader:
            x.append(float(row[0]))
            y.append(float(row[1]))
    return x, y
```

JSON files

JSON files can describe complex objects.

```
[
    {"x": 1, "y": 2, "z": [9, 7]},
    {"x": 2, "y": 4, "z": [5, 3, [1, 2]]},
    {"x": 3, "y": 6, "z": [3, "a"]}
]
```

Read data from JSON files.

```
import json
def read_json(file_name):
    x = []
    y = []
    with open(file_name, ’r’) as jsonfile:
        data = json.load(jsonfile)
        for item in data:
            x.append(float(item[’x’]))
            y.append(float(item[’y’]))
    return x, y
```

Plotting

Generate plots using matplotlib.

```python
import matplotlib.pyplot as plt

def plot_data(x, y):
    plt.plot(x, y, marker='o', linestyle='-',
        color='b')
    plt.title('Example')
    plt.xlabel('X-axis')
    plt.ylabel('Y-axis')
    plt.grid(True)

    # Save the figure
    plt.savefig(file_name)

    # Show the figure
    plt.show()
```

Conclusion

Thank you for your attention!