Cmpe 150 Lab 9: Dictionaries and Sets

Dictionaries

- They are similar to lists, yet we want to store several pairs as (key, value)

- For example, the name of the people and the number of books they have.

Python Dictionaries

- empty_dict = {}

- non_empty_dict = {'Ali': 15, 'Hasan': 13, 'Osman': 15}

Python Dictionaries (Cont.)

- The type of key and value can be anything, string and int combination is only an example.

- my_key in my_dict to learn if the given item exists as a key in the dictionary.

Access or Change the Value of a Key

- print(my_dict[my_key]) -> Be careful since it gives an error if the key is not in the dictionary, so using “in” before it might be better.

- my_dict[my_key] = new_val -> If my_key is not in the dictionary, it will define it.

Delete a Pair from the Dictionary

- my_dict.pop(key)
- Before: my_dict = {1: 2, 2: 4, 3: 6}
- After my_dict.pop(2) -> {1: 2, 3:6}

Keys and Values

- my_keys_list = list(my_dict.keys())
- my_values_list = list(my_dict.values())

Items

- Returns all the existing information as a list of tuples (key, value)

- my_items = my_dict.items()

Items (Cont.)

- Returns all the existing information as a list of tuples (key, value)

- We can use sorted function to do interesting stuff.

Using a Loop Over Dictionaries

- for key in my_dict:

  print(key, my_dict[key])

- for value in my_dict.values():

  print(value)

Using a Loop Over Dictionaries (Cont.)

- Using items function is also possible.

- for k, v in my_dict.items():

  print(k, v)

Sets

- Almost identical to their use in Math.

- Very similar to lists, yet we do not care about the order or repetition. The only concern is if an element is in the set or not.

Python Sets

- example_set = set()

- Be careful x = {} refers to the dictionary.

Python Sets (Cont.)

- non_empty_set = {4, "Long String", True} -> len(non_empty_set): 3

- Use in to check if an element is in the set.

  item in my_set -> Boolean

Python Sets (Cont.)

- my_set.add(new_val) and my_set.remove(existing_val)

- Use in to check if an element is in the set. It may be helpful before remove since remove gives an error if the item is not in the set.

  item in my_set -> Boolean

issubset

- Checks if a set is a subset of another one.

- set1.issubset(set2)

Basic Set Operations

- union_of_two_sets = set1.union(set2)
- intersection_of_two_sets = set1.intersection(set2)
- difference_of_two_sets = set1.difference(set2)

Iterating over a Set

- Set is unordered, so do NOT use

  ```
  for i in range(len(my_set)):
      print(my_set[i])
  ```

- Instead, the following would work

  ```
  for item in my_set:
      print(item)
  ```

Set to List and List to Set

- set_representation_of_list = set(my_list)
- list_representation_of_set = list(my_set)

Removing Duplicates in a List

- A simple way is to do type conversions: Be careful since the order is not preserved.

- non_repetitive_list = list(set(my_list))

As Always

- Nested structures are possible. The elements of a set can be a list and a dictionary and …

- We can obtain complicated data structures, yet it all depends on what is needed in the specific problem we want to solve.

Thanks

Any questions?

References

1. https://www.w3schools.com/python/python_dictionaries.asp
2. https://www.w3schools.com/python/python_sets.asp