Cmpe 150 Lab 10: Sets

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. A set can contain tuples and strings, as they are immutable and hashable, but it cannot contain dictionaries or lists, as they are mutable and unhashable.

- 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_sets.asp