# Big Daddy's

**Data Container Quick Chart**

## Sets (set type, mutable but no mutable collections)

| ACTION/EVENT | Key Term/Symbol/Method/Attribute | Example |
|---|---|---|
| Create initially | Setname = set (sequential item collection) ~no duplicates, unordered~ set([1,2,3,4])~ | SPreps = set(LPreps) or myset = "Bannanas are nice" ↳ {'s', ' ', 'n', 'B', 'a', 'r', 'i', 'e', 'c'} |
| | Setname = set() <-**empty;** or ={item,item,...} | Pets = {'dog','cat','fish'} |
| | can be built with set comprehensions | a = {x for x in 'abracadabra' if x not in 'abc'} |
| Combine/merge containers | .union(set2) or .update(set2) ~unique items~ | newset=set1.union(set2) |
| Combine overlap items only | .intersection(set2) | newset=set1.intersection(set2) |
| Combine non-overlap items only | .difference(set2) | newset=set1.difference(set2) |
| Add at first position | N/A - sets are not ordered | |
| Add items at end | N/A - sets are not ordered | |
| Add one item at end | N/A - sets are not ordered | |
| Add at position in container | .add(item) or add values with .union | set1.add('j'); ~ set1 = set1.union(['f','g','h']) |
| Add somewhere inbetween | N/A | |
| Add multiple items | .union(otherset) | newset=set1.union(set2) |
| or simple concatenation... | N/A | |
| Remove a known value or key | .remove(item) or .discard(item) | myset.remove("B") ~KeyError if not present~ |
| Remove item(s) by index | N/A - sets are not ordered | |
| Remove and return the last item | N/A | |
| Remove and return a known item | N/A | |
| Remove and return a random item | .pop() | SPreps.pop() |
| Remove and return item number i | N/A - sets are not ordered | |
| Replace an item/pair or value | (1) create set 2 with items to be removed and use .difference, then (2) create set 3 | |
| Replace a group of items | with the items to be added and use .union to add those back | |
| Retrieve sequential items | N/A - sets are not ordered | |
| Retrieve values, keys, or pairs | | |
| Retrieve value from known key | N/A | |
| Retrieve all keys, values, pairs | N/A | |
| Retrieve index number of first value x | N/A - sets are not ordered | |
| Compare overlap | s1.isdisjoint(s2) ~True if no common elements~ | Spreps.isdisjoint(myset)) |
| Compare subset | s1.issubset(s2) or s1 <= s2 ~s1 contained by s2~ | print(myset2 <= myset1) |
| *compare as true subset(not equal) | s1<s2 ~ both s1<=s2 and s1 !=s2 ~& not equal~ | print(myset2 < myset1) |
| Compare superset | s1.issuperset(s2) or s1 >= s2 | print(myset2 >= myset1) |
| *compare as true superset(not equal) | s1 > s1 | print(myset2 > myset1) |
| Iteration (loop) | | |
| Iteration (iter, next) | iter, next ~item delivery appears random~ | |
| Return number of items/pairs | len(setname) | print(str(len(SPreps))) |
| Find count of x values | can only hold unique values | |
| Find maximum value | N/A | |
| Find minimum value | N/A | |
| Determine membership | in , not in | print(str("B" in myset)) ↳ True |
| - | | |
| Copy | .copy() | newset=oldset.copy() |
| Sort | N/A - sets are not ordered | |
| Reverse items | N/A - sets are not ordered | |
| Clear all | .clear() | someset.clear() |
| Delete the object | del set ~after which attempted access->error~ | del someset |
| Convert | list to set: | myset=set(mylist) |
| Other: setdefault | | |