Pair Exercise: Sets

HD Sheets July 2024

For DSE5002

Sources https://docs.python.org/2/library/sets.html

Sets are unordered collections of unique objects

They are mutable

sets are defined within curly brackets {}

In [1]:
```
a={1,3,5,7,9,1,1}
a
```

Out[1]:
```
{1, 3, 5, 7, 9}
```

I entered 1 several times, but it only appears once in the set

Values are in the set or not, there is no need to list them more than once

In [2]:
```
#we can iterate a set, but the order may be random

for val in a:
    print(val)
```

```
1
3
5
7
9
```

In [3]:
```
dir(a)
```

Out[3]:
```
['__and__',
 '__class__',
 '__class_getitem__',
 '__contains__',
 '__delattr__',
 '__dir__',
 '__doc__',
 '__eq__',
 '__format__',
 '__ge__',
 '__getattribute__',
 '__getstate__',
 '__gt__',
 '__hash__',
 '__iand__',
 '__init__',
 '__init_subclass__',
 '__ior__',
 '__isub__',
 '__iter__',
 '__ixor__',
 '__le__',
 '__len__',
 '__lt__',
 '__ne__',
 '__new__',
 '__or__',
 '__rand__',
 '__reduce__',
 '__reduce_ex__',
 '__repr__',
 '__ror__',
 '__rsub__',
 '__rxor__',
 '__setattr__',
 '__sizeof__',
 '__str__',
 '__sub__',
 '__subclasshook__',
 '__xor__',
 'add',
 'clear',
 'copy',
 'difference',
 'difference_update',
 'discard',
 'intersection',
 'intersection_update',
 'isdisjoint',
 'issubset',
 'issuperset',
 'pop',
 'remove',
 'symmetric_difference',
 'symmetric_difference_update',
```

```
 'union',
 'update']
```

In [4]:
```
#adding to a set
a.add(11)
a
```

Out[4]:
```
{1, 3, 5, 7, 9, 11}
```

In [5]:
```
# there is a pop function, it removes an arbitrary element
z=a.pop()
print(a)
print(z)
```

```
{3, 5, 7, 9, 11}
1
```

In [6]:
```
# add z back!
a.add(z)
```

In [7]:
```
#merging sets
b={2,4,6,8}
a.update(b)
a
```

Out[7]:
```
{1, 2, 3, 4, 5, 6, 7, 8, 9, 11}
```

In [8]:
```
# adding an alias

c=b
c.pop()
print(b)
print(c)
```

```
{2, 4, 6}
{2, 4, 6}
```

In [9]:
```
#making a copy
d=a.copy()
d.pop()
print(d)
print(a)
```

```
{2, 3, 4, 5, 6, 7, 8, 9, 11}
{1, 2, 3, 4, 5, 6, 7, 8, 9, 11}
```

In [10]:
```
#find the length of a set

len(a)
```

Out[10]:
```
10
```

Set Operations

In [11]:
```
# testing for membership
```

```
y=21
y in a
```

Out[11]: False

In [12]:
```
y not in a
```

Out[12]: True

In [13]:
```
#classic set operations

t={"apple","orange","banana"}
c={"red","green","orange"}

#union
t|c
```

Out[13]: {'apple', 'banana', 'green', 'orange', 'red'}

In [14]:
```
#intersection
t&c
```

Out[14]: {'orange'}

In [15]:
```
#set differences
t-c
```

Out[15]: {'apple', 'banana'}

In [16]:
```
c-t
```

Out[16]: {'green', 'red'}

Why is c-t not equal to c-t? What is going on here?

look up symmetric _difference for python sets

Add a markdown cell and explain this

c-t is not equal to t-c because the set difference operator '-' returns items that are present in the first set, but not the second. Any duplicate values between the first and second set will be dropped as well.

Symmetric differences are the values in two sets that are not common between the two

What are sets good for?

Testing for membership

Sets work much faster for the "in" test, since sets are hashed.

the "in" test can be done using a list, but this is not hashed and thus 50 to 100 times slowere

Not a big deal for a small project, a huge deal for a bit data set that is repeatedly re-analyzed