Pair_exercise_sets

April 10, 2025

Pair Exercise: Sets

Beija Richardson 4/10/2025

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 {}

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

[3]:
```
{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

[6]:
```
#we can iterate a set, but the order may be random

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

```
1
3
5
7
9
```

[8]:
```
dir(a)
```

[8]:
```
['__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']
```

[10]:
```
#adding to a set
a.add(11)
a
```

[10]: {1, 3, 5, 7, 9, 11}

[12]:
```
# there is a pop function, it removes an arbitrary element
z=a.pop()
print(a)
print(z)
```

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

[14]:
```
# add z back!
a.add(z)
```

[16]:
```
#merging sets
b={2,4,6,8}
a.update(b)
a
```

[16]: {1, 2, 3, 4, 5, 6, 7, 8, 9, 11}

[18]:
```
# adding an alias


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

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

[20]:
```
#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}
```

[22]:
```
#find the length of a set
```

```
len(a)
```

[22]: 10

Set Operations

[25]:
```
# testing for membership

y=21
y in a
```

[25]: False

[27]:
```
y not in a
```

[27]: True

[29]:
```
#classic set operations

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

#union
t|c
```

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

[31]:
```
#intersection
t&c
```

[31]: {'orange'}

[33]:
```
#set differences
t-c
```

[33]: {'apple', 'banana'}

[35]:
```
c-t
```

[35]: {'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

0.1 Its like the order of subratction example: 3-2=1 but 2-3=-1

0.2 The symmetric difference of two sets returns a new set with elements that are in either of the sets but not in both.

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