# Day 11

## Sets in Python

## Creating a Set

```
In [23]: a = {"animals", "ball", "cat"}
         print(a)
         print(type(a))
```

```
{'ball', 'animals', 'cat'}
<class 'set'>
```

## Set Items are unordered

```
In [8]: a = {"animals", "ball", "cat"}
        print(a[0])
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
~\AppData\Local\Temp\ipykernel_11748\1711286952.py in <module>
      1 a = {"animals", "ball", "cat"}
----> 2 print(a[0])

TypeError: 'set' object is not subscriptable
```

## Duplicates Not Allowed

```
In [10]: a = {"animals", "ball", "cat","ball", "cat", "ball"}
         print(a)
```

```
{'ball', 'animals', 'cat'}
```

## True and 1 is considered the same value:

```
In [11]: a = {"animals", "ball", "cat","ball", "cat", "ball", 1, True, 2,3,4,5}
         print(a)
```

```
{1, 2, 3, 4, 5, 'ball', 'animals', 'cat'}
```

## Get the Length of a Set

```
In [12]: a = {"animals", "ball", "cat","ball", "cat", "ball", 1, True, 2,3,4,5}
         print(len(a))
```

```
8
```

## Set Items - Data Types

```
In [14]: x1 = {1,2,3,4}
         x2 = {1.2,2.3,2.4,2.5}
         x3 = {"a", "b","c"}
         x4 = {True,False,True}
         print(x1,x2,x3,x4)
```

```
{1, 2, 3, 4} {1.2, 2.5, 2.3, 2.4} {'a', 'b', 'c'} {False, True}
```

```
In [21]: x5 = {[1,2,3,4]}
         print(x5)
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
~\AppData\Local\Temp\ipykernel_11748\2253874395.py in <module>
----> 1 x5 = {[1,2,3,4]}
      2 print(x5)

TypeError: unhashable type: 'list'
```

## Lists are mutable and hence unhashable objects in Python. Whereas, sets in Python are immutable and does not allow unhashable objects. Therefore, Python does not allow a set to store a list. You cannot add a list to a set.

## Immutable objects are a type of object that cannot be modified after they were created. Hashable objects,

```
In [22]: x5 = {(1,2,3,4)}
         print(x5)
```

```
{(1, 2, 3, 4)}
```

```
In [26]: x6 = {{"a":1, "b":2}}
         print(x6)
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
~\AppData\Local\Temp\ipykernel_11748\218602821.py in <module>
----> 1 x6 = {{"a":1, "b":2}}
      2 print(x6)

TypeError: unhashable type: 'dict'
```

## “Nested” isn't a property of a set.

```
In [28]: a ={{11,2,3},{4,5,6}}
         print(a)
         print(type(a))
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
~\AppData\Local\Temp\ipykernel_11748\81408094.py in <module>
----> 1 a ={{11,2,3},{4,5,6}}
      2 print(a)
      3 print(type(a))

TypeError: unhashable type: 'set'
```

## The set() Constructor

```
In [30]: thisset = set(("alpha","beta","gamma"))
         print(thisset)
         print(type(thisset))
```

```
{'gamma', 'beta', 'alpha'}
<class 'set'>
```

## Creating empty set

```
In [47]: a = {}
         print(a)
         print(type(a))
```

```
{}
<class 'dict'>
```

In [48]:
```
empty = set()
print(type(empty))
```
```
<class 'set'>
```

## Access Items

You cannot access items in a set by referring to an index or a key.

But you can loop through the set items using a for loop, or ask if a specified value is present in a set, by using the in keyword.

In [33]:
```
a = {"a", "b", "c", "d"}

for x in a:
    print(x)
```
```
a
b
d
c
```

## Check if "Item" is present in the set:

In [34]:
```
a = {"data", "python", "code"}

print("code" in a)
```
```
True
```

In [35]:
```
a = {"data", "python", "code"}

print("analyst" in a)
```
```
False
```

## Add Items

Once a set is created, you cannot change its items, but you can add new items.

In [36]:
```
a = {"a1", "b1", "c1", 1, 2,3}

a.add("d1")

print(a)
```
```
{'d1', 1, 2, 3, 'c1', 'b1', 'a1'}
```

In [38]:
```
x = {"data", "python", "code"}
y= {10,20,30,40,50,60,70}

x.update(y)

print(x)
print(len(x))
```

```
{70, 40, 10, 'code', 50, 'data', 20, 'python', 60, 30}
10
```

## Python - Remove Set Items

In [40]:
```
n = {70, 40, 10, 'code', 50, 'data', 20, 'python', 60, 30}
n.remove(40)
print(n)
```

```
{70, 10, 'code', 50, 'data', 20, 'python', 60, 30}
```

In [41]:
```
m = {70, 40, 10, 'code', 50, 'data', 20, 'python', 60, 30}
m.remove("python")
print(m)
```

```
{70, 40, 10, 'code', 50, 'data', 20, 60, 30}
```

## You can also use the pop() method to remove an item, but this method will remove a random item

In [44]:
```
m = {70, 40, 10, 'code', 50, 'data', 20, 'python', 60, 30}
x = m.pop()
print(x)
print(m)
```

```
70
{40, 10, 'code', 50, 'data', 20, 'python', 60, 30}
```

## The clear() method empties the set:

In [45]:
```
m = {70, 40, 10, 'code', 50, 'data', 20, 'python', 60, 30}
m.clear()
print(m)
```

```
set()
```

## The del keyword will delete the set completely

In [49]:
```
m = {70, 40, 10, 'code', 50, 'data', 20, 'python', 60, 30}
del m
print(m)
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
~\AppData\Local\Temp\ipykernel_11748\1725773102.py in <module>
      1 m = {70, 40, 10, 'code', 50, 'data', 20, 'python', 60, 30}
      2 del m
----> 3 print(m)

NameError: name 'm' is not defined
```

## Join Two Sets

The union() method returns a new set with all items from both sets:

In [50]:
```
x = {"a", "b" , "c"}
y = {1, 2, 3}

z = x.union(y)
print(z)
```

```
{'a', 'c', 1, 2, 3, 'b'}
```

## Write a Python program to find the maximum and minimum values in a set.

In [51]:
```
setn = {5, 10, 3, 15, 2, 20}
print("Original set elements:")
print(setn)
print(type(setn))
print("Maximum value of the said set:")
print(max(setn))
print("Minimum value of the said set:")
print(min(setn))
```

```
Original set elements:
{2, 3, 20, 5, 10, 15}
<class 'set'>
Maximum value of the said set:
20
Minimum value of the said set:
2
```

## Return a new set of identical items from two sets

In [54]:
```
set1 = {10, 20, 30, 40, 50}
set2 = {30, 40, 50, 60, 70}

print(set1.intersection(set2))
```

```
{40, 50, 30}
```

In [ ]:
```
```