In [ ]:

```
Python Sets

Set

Sets are used to store multiple items in a single variable.

Set is one of 4 built-in data types in Python used to store collections of data, the other

A set is a collection which is both unordered and unindexed.

Sets are written with curly brackets.
```

In [ ]:

```
Set Items

Set items are unordered, unchangeable, and do not allow duplicate values.

Unordered

Unordered means that the items in a set do not have a defined order.

Set items can appear in a different order every time you use them, and cannot be referred t

Unchangeable

Sets are unchangeable, meaning that we cannot change the items after the set has been creat

Duplicates Not Allowed

Sets cannot have two items with the same value.
```

In [ ]:

```
Set Methods

Python has a set of built-in methods that you can use on sets.

Method  Description
add()   Adds an element to the set
clear() Removes all the elements from the set
copy()  Returns a copy of the set
difference()    Returns a set containing the difference between two or more sets
difference_update() Removes the items in this set that are also included in another, specif
discard()   Remove the specified item
intersection()  Returns a set, that is the intersection of two other sets
intersection_update()   Removes the items in this set that are not present in other, specif
isdisjoint()    Returns whether two sets have a intersection or not
issubset()  Returns whether another set contains this set or not
issuperset()    Returns whether this set contains another set or not
pop()   Removes an element from the set
remove()    Removes the specified element
symmetric_difference()  Returns a set with the symmetric differences of two sets
symmetric_difference_update()   inserts the symmetric differences from this set and another
union() Return a set containing the union of sets
update()    Update the set with the union of this set and others
```

In [1]:

```
thisset = {"apple", "banana", "cherry"}
print(thisset)

# Note: the set list is unordered, meaning: the items will appear in a random order.

# Refresh this page to see the change in the result.
```

```
{'apple', 'banana', 'cherry'}
```

In [2]:

```
thisset = {"apple", "banana", "cherry", "apple"}

print(thisset)
```

```
{'apple', 'banana', 'cherry'}
```

In [3]:

```
thisset = {"apple", "banana", "cherry"}

print(len(thisset))
```

```
3
```

In [4]:

```
set1 = {"apple", "banana", "cherry"}
set2 = {1, 5, 7, 9, 3}
set3 = {True, False, False}

print(set1)
print(set2)
print(set3)
```

```
{'apple', 'banana', 'cherry'}
{1, 3, 5, 7, 9}
{False, True}
```

In [1]:

```
set1 = {"abc", 34, True, 40, "male"}

print(set1)
```

```
{True, 34, 'male', 40, 'abc'}
```

In [2]:

```
myset = {"apple", "banana", "cherry"}

print(type(myset))
```

```
<class 'set'>
```

In [7]:

```
thisset = set(("apple", "banana", "cherry"))
print(thisset)
# Note: the set list is unordered, so the result will display the items in a random order.
```

```
{'apple', 'banana', 'cherry'}
```

In [8]:

```
thisset = {"apple", "banana", "cherry"}

for x in thisset:
  print(x)
```

```
apple
banana
cherry
```

In [9]:

```
thisset = {"apple", "banana", "cherry"}

print("banana" in thisset)
```

```
True
```

In [2]:

```
thisset = {"apple", "banana", "cherry"}

#Set.add("Value")
thisset.add("orange")

print(thisset)
```

```
{'banana', 'apple', 'cherry', 'orange'}
```

In [11]:

```
thisset = {"apple", "banana", "cherry"}
tropical = {"pineapple", "mango", "papaya"}

#Set1.update(Set2)
thisset.update(tropical)

print(thisset)
```

```
{'papaya', 'apple', 'mango', 'cherry', 'pineapple', 'banana'}
```

In [12]:

```
thisset = {"apple", "banana", "cherry"}
mylist = ["kiwi", "orange"]

thisset.update(mylist)

print(thisset)
```

```
{'apple', 'kiwi', 'cherry', 'orange', 'banana'}
```

In [13]:

```
thisset = {"apple", "banana", "cherry"}

thisset.remove("banana")

print(thisset)
```

```
{'apple', 'cherry'}
```

In [14]:

```
thisset = {"apple", "banana", "cherry"}

thisset.discard("banana")

print(thisset)
```

```
{'apple', 'cherry'}
```

In [15]:

```
thisset = {"apple", "banana", "cherry"}

x = thisset.pop()

print(x) #removed item

print(thisset) #the set after removal
```

```
apple
{'banana', 'cherry'}
```

In [4]:

```
thisset = {"apple", "banana", "cherry"}

thisset.clear()

print(thisset)
```

```
set()
```

In [17]:

```
thisset = {"apple", "banana", "cherry"}

del thisset

print(thisset) #this will raise an error because the set no longer exists
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-17-a14d992200ab> in <module>
      3 del thisset
      4
----> 5 print(thisset) #this will raise an error because the set no longer e
xists

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

In [18]:

```
thisset = {"apple", "banana", "cherry"}

for x in thisset:
  print(x)
```

```
apple
banana
cherry
```

In [19]:

```
set1 = {"a", "b" , "c"}
set2 = {1, 2, 3}

set3 = set1.union(set2)
print(set3)
```

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

In [20]:

```
set1 = {"a", "b" , "c"}
set2 = {1, 2, 3}

set1.update(set2)
print(set1)
```

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

In [5]:

```
x = {"apple", "banana", "cherry"}
y = {"google", "microsoft","banana", "apple"}

x.intersection_update(y)

print(x)
```

```
{'apple', 'banana'}
```

In [22]:

```
x = {"apple", "banana", "cherry"}
y = {"google", "microsoft", "apple"}

z = x.intersection(y)

print(z)
```

```
{'apple'}
```

In [23]:

```
x = {"apple", "banana", "cherry"}
y = {"google", "microsoft", "apple"}

x.symmetric_difference_update(y)

print(x)
```

```
{'microsoft', 'google', 'cherry', 'banana'}
```

In [24]:

```
x = {"apple", "banana", "cherry"}
y = {"google", "microsoft", "apple"}

z = x.symmetric_difference(y)

print(z)
```

```
{'microsoft', 'google', 'cherry', 'banana'}
```