Lists, Tuples and Sets

June 13, 2024

Expected Output?

```
def remove_dups(L1, L2):
    for e in L1:
        if e in L2:
            L1.remove(e)
L1 = [1, 2, 3, 4]
L2 = [1, 2, 5, 6]
remove_dups(L1, L2)
print(L1)
```

Removing Duplicates

```
def remove_dups(L1, L2):
    L1COPY = L1
    for e in L1COPY:
      if e in L2:
          L1.remove(e)
L1 = [1, 2, 3, 4]
L2 = [1, 2, 5, 6]
remove_dups(L1, L2)
print(L1)
```

Removing Duplicates

```
def remove_dups(L1, L2):
    L1COPY = L1[:]
    for e in L1COPY:
      if e in L2:
          L1.remove(e)
L1 = [1, 2, 3, 4]
L2 = [1, 2, 5, 6]
remove_dups(L1, L2)
print(L1)
```

Define and Print a tuple

```
mytuple = ("Python", "Pascal", "CPlus-Plus")

print(mytuple)
```

Define and Print a tuple

```
mytuple = ("Python", "Pascal", "CPlus-Plus")

print(mytuple)
```

Output: ('Python','Pascal','CPlusPlus')

Ordered, Unchangeable and allows Duplication

- Tuple items are ordered, unchangeable, and allow duplicate values.

Ordered, Unchangeable and allows Duplication

- Tuple items are ordered, unchangeable, and allow duplicate values.

- Tuple items are indexed, the first item has index [0], the second item has index [1] etc.

Tuple with Duplicates

thistuple = ("CPlusPlus", "C", "PASCAL",
"CPlusPlus")

Tuple with Duplicates

```
thistuple = ("CPlusPlus", "C", "PASCAL",
"CPlusPlus")

print(thistuple)
```

Tuple with Duplicates

```
thistuple = ("CPlusPlus", "C", "PASCAL",
"CPlusPlus")

print(thistuple)
```

Output: ('CPlusPlus','C','PASCAL','CPlusPlus)

Tuple Length

thistuple = ("C", "CC", "CCc")

Tuple Length

```
thistuple = ("C", "CC", "CCc")

print(len(thistuple))
```

Tuple Length

```
thistuple = ("C", "CC", "CCc")

print(len(thistuple))
```

Output: 3

One Item Tuple

```
thistuple = ("apple",)

print(type(thistuple))
```

One Item Tuple

```
thistuple = ("apple",)

print(type(thistuple))
```

Output: <class 'tuple'>

Tuples of same data types

```
tuple1 = ("apple", "banana", "cherry")

tuple2 = (1, 5, 7, 9, 3)

tuple3 = (True, False, False)
```

Tuples of Different data types

Information = ("abc", 34, True, 40, "male")

Accessing Elements by Indexing

```
thistuple = ("ABC", "DEF", "GHI")

print(thistuple[1])
```

Accessing Elements by Indexing

```
thistuple = ("ABC", "DEF", "GHI")

print(thistuple[1])
```

Output: DEF

Negative Indexing

```
thistuple = ("ABC", "DEF", "GHI")

print(thistuple[-2])
```

Negative Indexing

```
thistuple = ("ABC", "DEF", "GHI")

print(thistuple[-2])
```

Output: DEF

Range of Indices

```
thistuple = ("A", "AB", "ABC", "ABCD",
"ABCDE")

print(thistuple[2:4])
```

Range of Indices

```
thistuple = ("A", "AB", "ABC", "ABCD",
"ABCDE")

print(thistuple[2:4])
```

Output: ('ABC', 'ABCD')

Range of Indices

```
thistuple = ("A", "AB", "ABC", "ABCD",
"ABCDE")

print(thistuple[:4])
```

Range of Indices

```
thistuple = ("A", "AB", "ABC", "ABCD",
"ABCDE")

print(thistuple[:4])
```

Output: ('A','AB','ABC','ABCD')

Element Existence

```
thistuple = ("apple", "banana", "cherry")

if 'apple' in thistuple:

    print("Yes, apple is in the fruits tuple")
```

Element Existence

```
thistuple = ("apple", "banana", "cherry")

if 'apple' in thistuple:

    print("Yes, apple is in the fruits tuple")
```

Output: Yes, 'apple' is in the fruits tuple

Changing Tuple Values

```
x = ("Pascal", "Python", "C")

y = list(x)

y[1] = "cplusplus"

x = tuple(y)

print(x)
```

Changing Tuple Values

```
x = ("Pascal", "Python", "C")

y = list(x)

y[1] = "cplusplus"

x = tuple(y)

print(x)
```

Output: ('Pascal', 'cplusplus', 'C')

Add Items

```
thistuple = ("apple", "banana", "cherry")

y = list(thistuple)

y.append("orange")

thistuple = tuple(y)

print(thistuple)
```

Add Items

```
thistuple = ("apple", "banana", "cherry")

y = list(thistuple)

y.append("orange")

thistuple = tuple(y)

print(thistuple)
```

Output: ('apple','banana','cherry','orange')

Add tuple to a tuple

```
thistuple = ("apple", "banana", "cherry")

y = ("orange",)

thistuple += y

print(thistuple)
```

Add tuple to a tuple

```
thistuple = ("apple", "banana", "cherry")

y = ("orange",)

thistuple += y

print(thistuple)
```

Output: ('apple', 'banana', 'cherry', 'orange')

Remove an element from a tuple

```
thistuple = ("apple", "banana", "cherry")
y = list(thistuple)
y.remove("apple")
thistuple = tuple(y)
print(thistuple)
```

Remove an element from a tuple

```
thistuple = ("apple", "banana", "cherry")
y = list(thistuple)
y.remove("apple")
thistuple = tuple(y)
print(thistuple)
```

Output: ('banana', 'cherry')

Delete Tuple

```
thistuple = ("apple", "banana", "cherry")

del thistuple

print(thistuple) #this will raise an error because the tuple no longer exists
```

Unpacking a tuple

```
fruits = ("apple", "banana", "cherry")

(green, yellow, red) = fruits

print(green)

print(yellow)

print(red)
```

Unpacking a tuple

```
fruits = ("apple", "banana", "cherry")

(green, yellow, red) = fruits

print(green)

print(yellow)

print(red)
```

Output:
apple
banana
cherry

Using Asterisk

```
fruits = ("apple", "banana", "cherry", "strawberry", "raspberry")
(green, yellow, *red) = fruits
print(green)
print(yellow)
print(red)
```

Using Asterisk

```
fruits = ("apple", "banana", "cherry", "strawberry", "raspberry")
(green, yellow, *red) = fruits
print(green)
print(yellow)
print(red)
```

Output:
apple
banana
['cherry', 'strawberry', 'raspberry']

```
fruits = ("apple", "mango", "papaya", "pineapple", "cherry")

(green, *tropic, red) = fruits

print(green)

print(tropic)

print(red)
```

```
fruits = ("apple", "mango", "papaya", "pineapple", "cherry")

(green, *tropic, red) = fruits

print(green)

print(tropic)

print(red)
```

Output:
apple
['mango', 'papaya', 'pineapple']
cherry

Loop Tuples - For loop

```
thistuple = ("apple", "banana", "cherry")

for i in range(len(thistuple)):

        print(thistuple[i])
```

Loop Tuples - For loop

```
thistuple = ("apple", "banana", "cherry")

for i in range(len(thistuple)):

        print(thistuple[i])
```

Output:
apple
banana
cherry

Loop Tuples - While loop

```
thistuple = ("apple", "banana", "cherry")

i = 0

while i < len(thistuple):

        print(thistuple[i])

        i = i + 1
```

Loop Tuples - While loop

```
thistuple = ("apple", "banana", "cherry")

i = 0

while i < len(thistuple):

        print(thistuple[i])

        i = i + 1
```

Join Tuples

```
tuple1 = ("a", "b" , "c")

tuple2 = (1, 2, 3)

tuple3 = tuple1 + tuple2

print(tuple3)
```

Join Tuples

```
tuple1 = ("a", "b" , "c")

tuple2 = (1, 2, 3)

tuple3 = tuple1 + tuple2

print(tuple3)
```

Output: ('a', 'b', 'c', 1, 2, 3)

Multiply Tuples

```
fruits = ("apple", "banana", "cherry")

mytuple = fruits * 2

print(mytuple)
```

Multiply Tuples

```
fruits = ("apple", "banana", "cherry")

mytuple = fruits * 2

print(mytuple)
```

Output: ('apple', 'banana', 'cherry', 'apple', 'banana', 'cherry')

Python program to swap the contents of two variables using tuple.

Python program to swap the contents of two variables using tuple.

```
a = 10

b = 20

a, b = (b, a)

print("a =", a)

print("b =", b)
```

Python program to swap the contents of two variables using tuple.

```
a = 10

b = 20

a, b = (b, a)

print("a =", a)

print("b =", b)
```

Output: a = 20 b=10

Write a Python program to remove all tuple of length k from a list of tuples.

Write a Python program to remove all tuple of length k from a list of tuples.

```
tupleList = [(1, 4), (9, 4, 2), (4,5,6,8), (2, 6, 8), (3, 0, 1), (4, 4, 1)]

K = 2

print("Initial List : " + str(tupleList))

filteredList = [tup for tup in tupleList if len(tup) != K]

print("List of tuples after removing tuple of length k : " + str(filteredList))
```

Python program to sort a list of tuples by second item.

Python program to sort a list of tuples by second item.

```
tupleList = [(2, 5), (9, 1), (4, 6), (2, 8), (1, 7)]
print("Unordered list : ", str(tupleList))
listLen = len(tupleList)
for i in range(0, listLen):
    for j in range(0, (listLen - i - 1)):
        if(tupleList[j][1] > tupleList[j+1][1]):
            temp = tupleList[j]
            tupleList[j] = tupleList[j+1]
            tupleList[j+1] = temp

print("Sorted List : ", str(tupleList))
```

Sets

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

- A set is a collection which is unordered, unchangeable*, and unindexed.

Print Set

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

print(thisset)
```

Print Set

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

print(thisset)
```

Output: {'apple', 'banana', 'cherry'}

Set with Duplicates

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

print(thisset)
```

Set with Duplicates

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

print(thisset)
```

Output: {'apple', 'cherry', 'banana'}

True and 1 are same

```
thisset = {"apple", "banana", "cherry", True,
1, 2}

print(thisset)
```

True and 1 are same

thisset = {"apple", "banana", "cherry", True, 1, 2}

print(thisset)

Output: {True, 2, 'apple', 'cherry', 'banana'}

False and 0 are same

```
thisset = {"apple", "banana", "cherry", False,
True, 0}

print(thisset)
```

False and 0 are same

```
thisset = {"apple", "banana", "cherry", False,
True, 0}

print(thisset)
```

Output: {False, True, 'apple', 'banana', 'cherry'}

Cardinality of a set

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

print(len(thisset))
```

Cardinality of a set

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

print(len(thisset))
```

Output: 3
Duplicates are not counted

Examples of Set with same data type

set1 = {"apple", "banana", "cherry"}

set2 = {1, 5, 7, 9, 3}

set3 = {True, False, False}

Example with different data types

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

Accessing Items in a set - No Indexing

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

for x in thisset:

    print(x)
```

Accessing Items in a set - No Indexing

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

for x in thisset:

    print(x)
```

Output:
banana
cherry
apple

Membership

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

print("banana" in thisset)
```

Membership

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

print("banana" in thisset)
```

Output: True

Membership

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

print("banana" not in thisset)
```

Membership

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

print("banana" not in thisset)
```

Output: False

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

print("banana" not in thisset)
```

Output: False

Once Set is created you cannot change items, but you can add and remove items.

Adding an item to a set

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

thisset.add("orange")

print(thisset)
```

Adding an item to a set

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

thisset.add("orange")

print(thisset)
```

Output: {'orange', 'apple', 'cherry', 'banana'}

Add Sets

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

tropical = {"pineapple", "mango", "papaya"}

thisset.update(tropical)

print(thisset)
```

Add Sets

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

tropical = {"pineapple", "mango", "papaya"}

thisset.update(tropical)

print(thisset)
```

Output: {'mango', 'pineapple', 'papaya', 'cherry', 'banana', 'apple'}

Add list to a set

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

mylist = ["kiwi", "orange"]

thisset.update(mylist)

print(thisset)
```

Add list to a set

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

mylist = ["kiwi", "orange"]

thisset.update(mylist)

print(thisset)
```

Output: {'kiwi', 'orange', 'banana', 'cherry', 'apple'}

Remove an item from a set

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

thisset.remove("banana")

print(thisset)
```

Remove an item from a set

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

thisset.remove("banana")

print(thisset)
```

Output: {'cherry', 'apple'}

Remove an item from a set

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

thisset.discard("banana")

print(thisset)
```

Remove an item from a set

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

thisset.discard("banana")

print(thisset)
```

Output: {'apple', 'cherry'}

Remove using Pop - Random

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

Remove using Pop - Random

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

Output:

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

Clear Method

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

thisset.clear()

print(thisset)
```

Clear Method

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

thisset.clear()

print(thisset)
```

Output: set()

Deleting the set

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

del thisset

print(thisset)
```

Deleting the set

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

del thisset

print(thisset)
```

Output: ERROR

Loop through the set

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

for x in thisset:

    print(x)
```

Loop through the set

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

for x in thisset:

    print(x)
```

Output:
cherry
banana
apple

Union of two sets

```
set1 = {"a", "b", "c"}

set2 = {1, 2, 3}

set3 = set1.union(set2)

print(set3)
```

Union of two sets

```
set1 = {"a", "b", "c"}

set2 = {1, 2, 3}

set3 = set1 | set2

print(set3)
```

Union of two sets

```
set1 = {"a", "b", "c"}

set2 = {1, 2, 3}

set3 = set1 | set2

print(set3)
```

Union of Multiple Sets

```
set1 = {"a", "b", "c"}
set2 = {1, 2, 3}
set3 = {"John", "Elena"}
set4 = {"apple", "bananas", "cherry"}
myset = set1.union(set2, set3, set4)
print(myset)
```

Union of Multiple Sets

```
set1 = {"a", "b", "c"}
set2 = {1, 2, 3}
set3 = {"John", "Elena"}
set4 = {"apple", "bananas", "cherry"}
myset = set1.union(set2, set3, set4)
print(myset)
```

Union of Multiple Sets

```
set1 = {"a", "b", "c"}
set2 = {1, 2, 3}
set3 = {"John", "Elena"}
set4 = {"apple", "bananas", "cherry"}
myset = set1 | set2 | set3 |set4
print(myset)
```

Union of Multiple Sets

```
set1 = {"a", "b", "c"}
set2 = {1, 2, 3}
set3 = {"John", "Elena"}
set4 = {"apple", "bananas", "cherry"}
myset = set1 | set2 | set3 |set4
print(myset)
```

Union of a tuple with a set

```
x = {"a", "b", "c"}
y = (1, 2, 3)
z = x.union(y)
print(z)
```

Union of a tuple with a set

```
x = {"a", "b", "c"}
y = (1, 2, 3)
z = x.union(y)
print(z)
```

Union of a tuple with a set

```
x = {"a", "b", "c"}
y = (1, 2, 3)
z = x.union(y)
print(z)
```

The | operator only allows you to join sets with sets, and not with other data types like you can with the union() method.

The update method

```
set1 = {"a", "b", "c"}
set2 = {1, 2, 3}
set1.update(set2)
print(set1)
```

The update method

```
set1 = {"a", "b", "c"}
set2 = {1, 2, 3}
set1.update(set2)
print(set1)
Output: {1, 'c', 2, 'b', 3, 'a'}
```

The update method

```
set1 = {"a", "b", "c"}
set2 = {1, 2, 3}
set1.update(set2)
print(set1)
Output: {1, 'c', 2, 'b', 3, 'a'}
```

Note: Both union() and update() will exclude any duplicate items.

Intersection

```
set1 = {"apple", "banana", "cherry"}

set2 = {"google", "microsoft", "apple"}

set3 = set1.intersection(set2)

print(set3)
```

Intersection

```
set1 = {"apple", "banana", "cherry"}

set2 = {"google", "microsoft", "apple"}

set3 = set1.intersection(set2)

print(set3)
```

Intersection using &

```
set1 = {"apple", "banana", "cherry"}

set2 = {"google", "microsoft", "apple"}

set3 = set1 & set2

print(set3)
```

Intersection using &

```
set1 = {"apple", "banana", "cherry"}

set2 = {"google", "microsoft", "apple"}

set3 = set1 & set2

print(set3)
```

Intersection using update

```
set1 = {"apple", "banana", "cherry"}

set2 = {"google", "microsoft", "apple"}

set1.intersection_update(set2)

print(set1)
```

Intersection using update

```
set1 = {"apple", "banana", "cherry"}

set2 = {"google", "microsoft", "apple"}

set1.intersection_update(set2)

print(set1)
```

Intersection

```
set1 = {"apple", 1, "banana", 0, "cherry"}

set2 = {False, "google", 1, "apple", 2, True}

set3 = set1.intersection(set2)

print(set3)
```

Intersection

```
set1 = {"apple", 1, "banana", 0, "cherry"}

set2 = {False, "google", 1, "apple", 2, True}

set3 = set1.intersection(set2)

print(set3)
```

Difference

```
set1 = {"apple", "banana", "cherry"}

set2 = {"google", "microsoft", "apple"}

set3 = set1.difference(set2)

print(set3)
```

Difference

```
set1 = {"apple", "banana", "cherry"}

set2 = {"google", "microsoft", "apple"}

set3 = set1.difference(set2)

print(set3)
```

Difference using -

```
set1 = {"apple", "banana", "cherry"}
set2 = {"google", "microsoft", "apple"}
set3 = set1 - set2
print(set3)
```

Difference using -

```
set1 = {"apple", "banana", "cherry"}
set2 = {"google", "microsoft", "apple"}
set3 = set1 - set2
print(set3)
```

Difference using -

```
set1 = {"apple", "banana", "cherry"}
set2 = {"google", "microsoft", "apple"}
set3 = set1 - set2
print(set3)
```

The - operator only allows you to join sets with sets, and not with other data types like you can with the difference() method.

Difference Update

```
set1 = {"apple", "banana", "cherry"}

set2 = {"google", "microsoft", "apple"}

set1.difference_update(set2)

print(set1)
```

Difference Update

```
set1 = {"apple", "banana", "cherry"}

set2 = {"google", "microsoft", "apple"}

set1.difference_update(set2)

print(set1)
```

Symmetric Difference

```
set1 = {"apple", "banana", "cherry"}

set2 = {"google", "microsoft", "apple"}

set3 = set1.symmetric_difference(set2)

print(set3)
```

Symmetric Difference

```
set1 = {"apple", "banana", "cherry"}

set2 = {"google", "microsoft", "apple"}

set3 = set1.symmetric_difference(set2)

print(set3)
```

Symmetric Difference using ^ operator

```
set1 = {"apple", "banana", "cherry"}
set2 = {"google", "microsoft", "apple"}
set3 = set1 ^ set2
print(set3)
```

Symmetric Difference using ^ operator

```
set1 = {"apple", "banana", "cherry"}
set2 = {"google", "microsoft", "apple"}
set3 = set1 ^ set2
print(set3)
```

The ^ operator only allows you to join sets with sets, and not with other data types like you can with the symmetric_difference() method

Symmetric Difference Update

```
set1 = {"apple", "banana", "cherry"}

set2 = {"google", "microsoft", "apple"}

set1.symmetric_difference_update(set2)

print(set1)
```

Symmetric Difference Update

```
set1 = {"apple", "banana", "cherry"}

set2 = {"google", "microsoft", "apple"}

set1.symmetric_difference_update(set2)

print(set1)
```

Write a Python Program that includes a function that returns a tuple.

Write a Python Program that includes a function that returns a tuple.

```
def quotient_and_remainder(x, y):

    q = x // y

    r=x%y

    return (q, r)

print(quotient_and_remainder(7,2))
```

NumPy

- NumPy is a Python library.
- NumPy is used for working with arrays.
- NumPy is short for "Numerical Python".