Collective Datatypes in Pthon(list, set, Dictionary and Tuple)

```
# List. Ordered and Changable(mutable) and [] are used and allow Duplicates.
thislist = ["apple", "banana", "cherry"]
print(thislist)
```

```
['apple', 'banana', 'cherry']
```

```
# examples
thislist = ["apple", "banana", "cherry", "apple", "cherry"]
print(thislist)
```

```
['apple', 'banana', 'cherry', 'apple', 'cherry']
```

```
# length of a list
thislist = ["apple", "banana", "cherry"]
print(len(thislist))
```

```
3
```

```
# list datatypes.(list can be of any datatypes)
list1 = ["apple", "banana", "cherry"]
list2 = [1, 5, 7, 9, 3]
list3 = [True, False, False]
```

```
# list type
print(type(list1))
```

```
<class 'list'>
```

```
# List datatypes
print(type(1))
```

```
<class 'int'>
```

```
print(type('True'))
print(type(True))
```

```
<class 'str'>
<class 'bool'>
```

```
# The list() Constructor.
# It is also possible to use the list() constructor when creating a new list.
thislist = list(("apple", "banana", "cherry")) # note the double round-brackets
print(thislist)
```

```
['apple', 'banana', 'cherry']
```

```
# Access Items,
# List items are indexed and you can access them by referring to the index number.
# This is positive indexing
thislist = ["apple", "banana", "cherry"]
print(thislist[1])
```

```
banana
```

```
# Negative Indexing.
# Negative indexing means start from the end.
```

```
# -1 refers to the last item, -2 refers to the second last item etc.
list = ["apple", "banana", "cherry"]
print(list[-1])
```

```
cherry
```

```
# Range of Indexes
# first index is 0.
thislist = ["apple", "banana", "cherry", "orange", "kiwi", "melon", "mango"]
print(thislist[2:5])

thislist = ["apple", "banana", "cherry", "orange", "kiwi", "melon", "mango"]
print(thislist[:4])

thislist = ["apple", "banana", "cherry", "orange", "kiwi", "melon", "mango"]
print(thislist[2:])

# Range of Negative indexex.
thislist = ["apple", "banana", "cherry", "orange", "kiwi", "melon", "mango"]
print(thislist[-4:-1])
```

```
['cherry', 'orange', 'kiwi']
['apple', 'banana', 'cherry', 'orange']
['cherry', 'orange', 'kiwi', 'melon', 'mango']
['orange', 'kiwi', 'melon']
```

```
# Change Item Value
thislist = ["apple", "banana", "cherry"]
thislist[1] = "blackcurrant"
print(thislist)

# Insert Items
thislist = ["apple", "banana", "cherry"]
thislist.insert(2, "watermelon")
print(thislist)

thislist = ["apple", "banana", "cherry"]
thislist.insert(1, "orange")
print(thislist)

# Append Items.
thislist = ["apple", "banana", "cherry"]
thislist.append("orange")
print(thislist)

# Extend list
thislist = ["apple", "banana", "cherry"]
tropical = ["mango", "pineapple", "papaya"]
thislist.extend(tropical)
print(thislist)
```

```
['apple', 'blackcurrant', 'cherry']
['apple', 'banana', 'watermelon', 'cherry']
['apple', 'orange', 'banana', 'cherry']
['apple', 'banana', 'cherry', 'orange']
['apple', 'banana', 'cherry', 'mango', 'pineapple', 'papaya']
```

```
# Remove List Items
thislist = ["apple", "banana", "cherry"]
thislist.remove("banana")
print(thislist)
```

```
thislist = ["apple", "banana", "cherry"]
thislist.pop(1)
print(thislist)

# pop
thislist = ["apple", "banana", "cherry"]
thislist.pop() #last ele pop.
print(thislist)

# clear
thislist = ["apple", "banana", "cherry"]
thislist.clear()
print(thislist)
```

```
['apple', 'cherry']
['apple', 'cherry']
['apple', 'banana']
[]
```

```
# Loop Through a List
thislist = ["apple", "banana", "cherry"]
for x in thislist:
  print(x)

  # Loop Through the Index Numbers
  # Use the range() and len() functions to create a suitable iterable.
  thislist = ["apple", "banana", "cherry"]
for i in range(len(thislist)):
  print(thislist[i])
```

```
apple
banana
cherry
apple
banana
cherry
```

```
# Tuples: ordered and unmutable(not changable) and () are used, and allow duolicates.
thistuple = ("apple", "banana", "cherry")
print(thistuple)

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

# len.
thistuple = ("apple", "banana", "cherry", "apple", "cherry")
print(thistuple)

# One item tuple, remember the comma:********
thistuple = ("apple",)
print(type(thistuple))

#NOT a tuple
thistuple = ("apple")
print(type(thistuple))

mytuple = ("apple", "banana", "cherry")
print(type(mytuple))
```

```
('apple', 'banana', 'cherry')
('apple', 'banana', 'cherry', 'apple', 'cherry')
('apple', 'banana', 'cherry', 'apple', 'cherry')
<class 'tuple'>
```

```
<class 'str'>
<class 'tuple'>
```

```
# The tuple() Constructor
thistuple = tuple(("apple", "banana", "cherry")) # note the double round-brackets
print(thistuple)

# Access Tuple Items
thistuple = ("apple", "banana", "cherry")
print(thistuple[1])

thistuple = ("apple", "banana", "cherry")
print(thistuple[-1])

# Range of Indexes
thistuple = ("apple", "banana", "cherry", "orange", "kiwi", "melon", "mango")
print(thistuple[2:5])
thistuple = ("apple", "banana", "cherry", "orange", "kiwi", "melon", "mango")
print(thistuple[:4])
thistuple = ("apple", "banana", "cherry", "orange", "kiwi", "melon", "mango")
print(thistuple[2:])
thistuple = ("apple", "banana", "cherry", "orange", "kiwi", "melon", "mango")
print(thistuple[-4:-1])

# Check if Item Exists
thistuple = ("apple", "banana", "cherry")
if "apple" in thistuple:
  print("Yes, 'apple' is in the fruits tuple")
```

```
('apple', 'banana', 'cherry')
banana
cherry
('cherry', 'orange', 'kiwi')
('apple', 'banana', 'cherry', 'orange')
('cherry', 'orange', 'kiwi', 'melon', 'mango')
('orange', 'kiwi', 'melon')
Yes, 'apple' is in the fruits tuple
```

```
# Add tuple to a tuple
thistuple = ("apple", "banana", "cherry")
y = ("orange",)
thistuple += y

print(thistuple)
```

```
('apple', 'banana', 'cherry', 'orange')
```

```
# Remove Items
thistuple = ("apple", "banana", "cherry")
del thistuple
print(thistuple) #this will raise an error because the tuple no longer exists
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-19-3257b0e525ad> in <module>
      2 thistuple = ("apple", "banana", "cherry")
      3 del thistuple
----> 4 print(thistuple) #this will raise an error because the tuple no longer
```

```
# Add Items
thistuple = ("apple", "banana", "cherry")
y = list(thistuple)
y.append("orange")
thistuple = tuple(y)
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-20-af4f9919d616> in <module>
      1 # Add Items
      2 thistuple = ("apple", "banana", "cherry")
----> 3 y = list(thistuple)
      4 y.append("orange")
      5 thistuple = tuple(y)

TypeError: 'list' object is not callable
```

SEARCH STACK OVERFLOW

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

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-21-ac6161d440f4> in <module>
      1 thistuple = ("apple", "banana", "cherry")
----> 2 y = list(thistuple)
      3 y.remove("apple")
      4 thistuple = tuple(y)

TypeError: 'list' object is not callable
```

SEARCH STACK OVERFLOW

```
# Loop Through a Tuple
thistuple = ("apple", "banana", "cherry")
for x in thistuple:
  print(x)

  # Loop Through the Index Numbers
thistuple = ("apple", "banana", "cherry")
for i in range(len(thistuple)):
  print(thistuple[i])
```

```
apple
banana
cherry
apple
banana
cherry
```

```
# Set:unordered, unchangeable*, and unindexed.{} are used. Duplicates Not Allowed
thisset = {"apple", "banana", "cherry"}
print(thisset)

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

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

# length
thisset = {"apple", "banana", "cherry"}
print(len(thisset))

# Set Items - Data Types
set1 = {"apple", "banana", "cherry"}
set2 = {1, 5, 7, 9, 3}
set3 = {True, False, False}
print(type(set1))

set1 = {"abc", 34, True, 40, "male"}
print(type(set1))
```

```
{'apple', 'cherry', 'banana'}
{True, 'apple', 2, 'cherry', 'banana'}
{'apple', 'cherry', 'banana'}
3
<class 'set'>
<class 'set'>
```

```
# Access Items
# Loop through the set, and print the values:
thisset = {"apple", "banana", "cherry"}
for x in thisset:
  print(x)
thisset = {"apple", "banana", "cherry"}
print("orange" in thisset)
```

```
apple
cherry
banana
False
```

```
# Add Items
thisset = {"apple", "banana", "cherry"}
thisset.add("orange")
print(thisset)

# Add sets
tropical = {"pineapple", "mango", "papaya"}

thisset.update(tropical)

print(thisset)

# add any datatype
thisset = {"apple", "banana", "cherry"}
mylist = ["kiwi", "orange"]

thisset.update(mylist)

print(thisset)
```

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

```
# Remove Item
thisset = {"apple", "banana", "cherry"}

thisset.remove("banana")
```

```
print(thisset)
```

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

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

thisset.discard("banana")

print(thisset)

# pop()
thisset = {"apple", "banana", "cherry"}

x = thisset.pop()

print(x)

print(thisset)

# clear
thisset = {"apple", "banana", "cherry"}

thisset.clear()

print(thisset)
```

```
{'apple', 'cherry'}
apple
{'cherry', 'banana'}
set()
```

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

del thisset

print(thisset)
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-32-e36ab6394dd4> in <module>
      4 del thisset
      5
----> 6 print(thisset)

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

SEARCH STACK OVERFLOW

```
# loop through sets
thisset = {"apple", "banana", "cherry"}
for x in thisset:
  print(x)

  # join two sets [union]
  set1 = {"a", "b" , "c"}
set2 = {1, 2, 3}
set3 = set1.union(set2)
print(set3)
```

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

set1.update(set2)
print(set1)

# Keep ONLY the Duplicates
x = {"apple", "banana", "cherry"}
y = {"google", "microsoft", "apple"}

x.intersection_update(y)

print(x)

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

z = x.intersection(y)

print(z)

# Keep All, But NOT the Duplicates
x = {"apple", "banana", "cherry"}
y = {"google", "microsoft", "apple"}

x.symmetric_difference_update(y)

print(x)
```

```
apple
cherry
banana
{'a', 1, 2, 3, 'b', 'c'}
{'a', 1, 2, 3, 'b', 'c'}
{'apple'}
{'apple'}
{'banana', 'google', 'cherry', 'microsoft'}
```

```
# Dictionaries: changable and duplicates are not allowed.
thisdict = {
  "brand": "Ford",
  "model": "Mustang",
  "year": 1964
}
print(thisdict)

# Dictionary Items
thisdict = {
  "brand": "Ford",
  "model": "Mustang",
  "year": 1964
}
print(thisdict["brand"])

# overwrite the duplicates
thisdict = {
  "brand": "Ford",
  "model": "Mustang",
  "year": 1964,
  "year": 2020
}
print(thisdict)
```

```
     {'brand': 'Ford', 'model': 'Mustang', 'year': 1964}
     Ford
     {'brand': 'Ford', 'model': 'Mustang', 'year': 2020}
```

```
# Accessing Items
thisdict = {
  "brand": "Ford",
  "model": "Mustang",
  "year": 1964
}
x = thisdict["model"]

# get method
x = thisdict.get("model")

# get keys
x = thisdict.keys()

car = {
"brand": "Ford",
"model": "Mustang",
"year": 1964
}
x = car.keys()
print(x) #before the change
car["color"] = "white"
print(x) #after the change
```

```
     dict_keys(['brand', 'model', 'year'])
     dict_keys(['brand', 'model', 'year', 'color'])
```

```
# Get Values
x = thisdict.values()
print(x)

# make a change
car = {
"brand": "Ford",
"model": "Mustang",
"year": 1964
}
x = car.values()
print(x) #before the change
car["year"] = 2020
print(x) #after the change

# add new
car = {
"brand": "Ford",
"model": "Mustang",
"year": 1964
}
x = car.values()
print(x) #before the change
car["color"] = "red"
print(x) #after the change
```

```
     dict_values(['Ford', 'Mustang', 1964])
     dict_values(['Ford', 'Mustang', 1964])
     dict_values(['Ford', 'Mustang', 2020])
     dict_values(['Ford', 'Mustang', 1964])
     dict_values(['Ford', 'Mustang', 1964, 'red'])
```

```
x = thisdict.items()
```

```
print(x)
```

```
     dict_items([('brand', 'Ford'), ('model', 'Mustang'), ('year', 1964)])
```

```
# Check if Key Exists
thisdict = {
  "brand": "Ford",
  "model": "Mustang",
  "year": 1964
}
if "model" in thisdict:
  print("Yes, 'model' is one of the keys in the thisdict dictionary")

# Change Values
thisdict = {
  "brand": "Ford",
  "model": "Mustang",
  "year": 1964
}
thisdict["year"] = 2018

# Update Dictionary
thisdict = {
  "brand": "Ford",
  "model": "Mustang",
  "year": 1964
}
thisdict.update({"year": 2020})
print(thisdict)
```

```
     Yes, 'model' is one of the keys in the thisdict dictionary
     {'brand': 'Ford', 'model': 'Mustang', 'year': 2020}
```

```
# Add Dictionary Items
thisdict = {
  "brand": "Ford",
  "model": "Mustang",
  "year": 1964
}
thisdict["color"] = "red"
print(thisdict)
```

```
     {'brand': 'Ford', 'model': 'Mustang', 'year': 1964, 'color': 'red'}
```

```
# Update Dictionary
thisdict = {
  "brand": "Ford",
  "model": "Mustang",
  "year": 1964
}
thisdict.update({"color": "red"})
```

```
# Remove Dictionary
thisdict = {
  "brand": "Ford",
  "model": "Mustang",
  "year": 1964
}
```

```
thisdict.pop("model")
print(thisdict)
```

```
{'brand': 'Ford', 'year': 1964}
```

```
thisdict = {
  "brand": "Ford",
  "model": "Mustang",
  "year": 1964
}
thisdict.popitem()
print(thisdict)
```

```
{'brand': 'Ford', 'model': 'Mustang'}
```

```
# del
thisdict = {
  "brand": "Ford",
  "model": "Mustang",
  "year": 1964
}
del thisdict["model"]
print(thisdict)
```

```
{'brand': 'Ford', 'year': 1964}
```

```
# clear
thisdict = {
  "brand": "Ford",
  "model": "Mustang",
  "year": 1964
}
thisdict.clear()
print(thisdict)
```

```
{}
```

```
#  Loop Dictionaries
for x in thisdict:
  print(x)

for x in thisdict:
  print(thisdict[x])

# loop through values
for x in thisdict.values():
  print(x)

# loop through keys
for x in thisdict.keys():
  print(x)

# for both keys and values
for x, y in thisdict.items():
  print(x, y)



# Copy Dictionaries
thisdict = {
  "brand": "Ford",
  "model": "Mustang",
  "year": 1964
}
```

```
mydict = thisdict.copy()
print(mydict)
```

```
{'brand': 'Ford', 'model': 'Mustang', 'year': 1964}
```

```
# Nested Dictionaries
myfamily = {
  "child1" : {
    "name" : "Emil",
    "year" : 2004
  },
  "child2" : {
    "name" : "Tobias",
    "year" : 2007
  },
  "child3" : {
    "name" : "Linus",
    "year" : 2011
  }
}
print(myfamily)
```

```
{'child1': {'name': 'Emil', 'year': 2004}, 'child2': {'name': 'Tobias', 'year': 2007}, 'child3': {'name': 'Linus',
```