Tuple

* Tuple is defined as an object with the data type “tuple”
* When creating a tuple by assigning values to it called packing a tuple
* Tuples are used to store multiple items in a single variable.
* Tuple is one of 4 built-in data types in Python used to store collections of data, the other 3 are [List](https://www.w3schools.com/python/python_lists.asp), [Set](https://www.w3schools.com/python/python_sets.asp), and [Dictionary](https://www.w3schools.com/python/python_dictionaries.asp), all with different qualities and usage.
* A tuple is a collection which is ordered , unchangeable, and allow duplicate values.
* Tuple item can be of any data type
* Tuples are written with round/open brackets. Ex: tuple=()
* Tuple items are indexed, first item has index[0] and second item has index [1]
* Tuple() constructor can be used to make tuple
  + Ex: Using a tuple() create a tuple
    - t2=tuple((1,2,3,4,5))  
      print(t2)*#(1, 2, 3, 4, 5)*
    - t1=[1,2,3]  
      t2=tuple(t1)  
      print(t2)*#(1, 2, 3)*
* Don’t forget to include comma after an item even though there is one item.
  + Ex: One item tuple with comma
    - T1=("one",)  
      print(type(T1)) # <class 'tuple'>
  + Ex: One item tuple without comma considers as string
    - T1=("one")  
      print(type(T1)) # <class str>

Access Tuple Items

* By referring to the index number, able to access the tuple items
* Using negative indexing also can access the tuple items
* Using index range, can access the tuple items
* Using loop, can access the tuple items

tuple1=(1,25,6,85,1.2)  
print(tuple1[2]) *#6*print(tuple1[-2])*#85*print(tuple1[1:4])*#25,6,85*if 25 in tuple1:  
 print("Yes, 25 is in tuple1")*#Yes, 25 is in tuple1*else:  
 print("Cannot find")

Update the tuple items

Once the tuple is created , cannot change its value. But there is workaround. Convert the tuple into list and do the modification and then converted back to tuple.

t1=(1,2,3,5)  
l1=list(t1) *# convert tuple to a list*l1[1]=50 *# add item by referring an index*l1.append(100) *# add item at the end of the list by appending*t1=tuple(l1) *# convert list to a tuple*print(t1) *#(1,* ***50****, 3, 5,* ***100****)*

Also allowed to add tuple to a tuple

t1=(1,2,3,5)  
t2=(6,) *# create tuple with new items*t1+=t2 *# add new tuple to current tuple*print(t1)*#(1, 2, 3, 5, 6)*

Remove Items in Tuple is not allowed, but there is a workaround – converting to a list

t1=(1,2,3,5)  
l1=list(t1) *# convert tuple to a list*l1.remove(5) *# remove value 5*t1=tuple(l1)*# convert list back to a tuple*print( t1)*# (1, 2, 3)*

Completely remove tuple

t1=(1,2,3,5)  
t2=(6,) *# create tuple with new items*print(t1, t2)*#(1, 2, 3, 5) (6,)*del t2 # delete t2 tuple  
print( t2)*# name 't2' is not defined.*

Unpacking the items

When extracting the tuple values back into variables called unpacking. However, **note:** The number of variables must match the number of values in the tuple, if not, you must use an asterisk to collect the remaining values as a list.

fruits = ("apple", "banana", "cherry")  
(v1,v2,v3)=fruits *# unpacking by assigning tuple values to variables*print(v1, "is assigned to v1") *# apple is assigned to v1*print(v2, "is assigned to v2") *# banana is assigned to v2*print(v3, "is assigned to v2") *# cherry is assigned to v3*

Using asterisk (\*)

If the number of variables is less than the number of values, you can add an \* to the variable name and the values will be assigned to the variable as a list:

fruits = ("apple", "banana", "cherry", "strawberry", "raspberry")  
(v\_1,v\_2,\*v\_3) = fruits *## unpacking by assigning tuple values to variables*print(v\_1, "is assigned to v\_1") *# apple is assigned to v\_1*print(v\_2, "is assigned to v\_2") *# banana is assigned to v\_2*print(v\_3, "are assigned to v\_3 as a list") *# ['cherry', 'strawberry', 'raspberry'] are assigned to v\_3 as a list*

If the asterisk is added to in a middle variable name than the last, Python will assign values to the variable until the number of values left matches the number of variables left.

fruits = ("apple", "banana", "cherry", "strawberry", "raspberry")  
(\_v1,\*\_v2,\_v3) = fruits *# unpacking by assigning tuple values to variables*print(\_v1, "is assigned to \_v1") *# apple is assigned to \_v1*print(\_v2, "are assigned to \_v2 as a list") *# ['banana', 'cherry', 'strawberry'] are assigned to \_v2 as a list*print(\_v3, "is assigned to \_v3 ") *# raspberry is assigned to \_v3*

Join tuples

Using + operator can join two or more tuples

f1 = ("apple", "banana", "cherry")  
c1 = ("red", "yellow", "purple")  
f1+=c1 *# join new tuple to an existing tuple*print(f1) *#('apple', 'banana', 'cherry', 'red', 'yellow', 'purple')*

Using \* operator, can replicate the content of a tuple a given number of times

f1 = ("apple", "banana", "cherry")  
f2 = f1\*2 *# replicate the content of a f1 twice*print(f2) *#('apple', 'banana', 'cherry', 'apple', 'banana', 'cherry')*

count()— Returns the number of times a specified value occurs in a tuple

f1 = ('apple', 'banana', 'cherry', 'apple', 'banana', 'banana')  
print(f1.count('apple'),f1.count('banana'),f1.count('cherry')) *#2 3 1*

index()—Searches the tuple for a specified value and returns the position of where it was found

f1 = ('apple', 'banana', 'cherry', 'apple', 'banana', 'banana')  
print(f1.index('apple'),f1.index('banana'),f1.index('cherry')) *#0 1 2*