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

```
t1=(1,2,3,4)
print(t1)
print(type(t1))
```

```
(1, 2, 3, 4)
<class 'tuple'>
```

```
#using constructor
t2= tuple((1,2,3,4))
print(t2)
```

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

Tuple

- **Ordered -** items have a defined order, and that order will not change
- **Indexed** - items are indexed, the first item has index [0]
- **Unchangeable** - we cannot change, add or remove items after the tuple has been created
- **Duplicates allowed** - Since tuples are indexed, they can have items with the same value

```
t2=(1,2,2,3,4,4,4,5)
print(t2)
```

```
(1, 2, 2, 3, 4, 4, 4, 5)
```

Tuple Length

```
t2=(1,2,2,3,4,4,4,5)
print(len(t2))
```

```
8
```

- To determine how many items a tuple has, use the len() function

Create Tuple With One Item

- To create a tuple with only one item, you have to add a comma after the item, otherwise Python will not recognize it as a tuple

```
# not a tuple
t3=(2)
print(t3)
print(type(t3))
```

```
2
<class 'int'>
```

```
# tuple with one item
t3=(2,)
print(t3)
print(type(t3))
```

```
(2,)
<class 'tuple'>
```

Access Tuple Items

You can access tuple items by referring to the index number, inside square brackets

```
# Access tuple item
t2=(1,2,2,3,4,4,4,5)
print(t2[3])
```

```
3
```

```
# Access tuple item using negative indexing
t2=(1,2,2,3,4,4,4,5)
print(t2[-1])
```

```
5
```

Check if Item Exists

- To determine if a specified item is present in a tuple use the in keyword

```
t4 = ("a", "b", "c")
if "a" in t4:
    print("Yes, 'a' is in the tuple t4")
```

```
Yes, 'a' is in the tuple t4
```

Change Tuple Values

- Once a tuple is created, you **cannot** change its values. Tuples are **unchangeable**, or **immutable.**

```
t4 = ("a", "b", "c")
t4[2]="d"
print(t4)
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
/tmp/ipykernel_33/203000124.py in <module>
      1 t4 = ("a", "b", "c")
----> 2 t4[2]="d"
      3 print(t4)

TypeError: 'tuple' object does not support item assignment
```

Delete the tuple

```
t4 = ("a", "b", "c")
del t4
print(t4)
```

```
---------------------------------------------------------
NameError                                 Traceback (mo
/tmp/ipykernel_33/3942123789.py in <module>
      1 t4 = ("a", "b", "c")
      2 del t4
----> 3 print(t4)

NameError: name 't4' is not defined
```

- The del keyword can delete the tuple completely

Unpack Tuples

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.

When we create a tuple, we normally assign values to it. This is called "packing" a tuple

In Python, we are also allowed to extract the values back into variables. This is called "unpacking“

```
# upacking a tuple

numbers = (1,2,3)

(green, yellow, red) = numbers

print(green)
print(yellow)
print(red)
```

```
1
2
3
```

```
# Assign the rest of the values as a list called "red"
numbers = (1,2,3,4,5,6,7,8)

(green, yellow, *red) = numbers

print(green)
print(yellow)
print(red)
```

```
1
2
[3, 4, 5, 6, 7, 8]
```

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

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

```
numbers = (1,2,3,4,5,6,7,8)

(green, *yellow, red) = numbers

print(green)
print(yellow)
print(red)
```

```
1
[2, 3, 4, 5, 6, 7]
8
```

Loop Through a Tuple

```
# looping through tuple
t=(1,2,3,4,5,6,7,8)
for x in t:
    print(x)
```

```
1
2
3
4
5
6
7
8
```

Loop Through the Index Numbers

```
# loop through index number
# Print all items by referring to their index number
t=(10,20,30,40,50,60,70,80)
for i in range(len(t)):
  print(t[i])
```

```
10
20
30
40
50
60
70
80
```