 # Tuples are used to store multiple items in a single variable.

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

 # Unchangeable

 # Tuples are unchangeable, meaning that we cannot change, add or remove items after the tuple has been created.

 # # eg.1

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

 # Tuple Length

```
# To determine how many items a tuple has, use the len() funct1ion:
thistuple = ("apple", "banana", "cherry")
print(len(thistuple))
```

 # Create Tuple With One Item

```
thistuple = "apple"
print(type(thistuple))
thistuple = ("apple",)
print(type(thistuple))
```

#NOT a tuple

```
thistuple = ("apple")
print(type(thistuple))
```

 # Tuple Items - Data Types

 # String, int and boolean data types:

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

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

tuple3 = (True, False, False)


# A tuple with strings, integers and boolean values:

tuple1 = ("abc", 34, True, 40, "male")


# type()

# From Python's perspective, tuples are defined as objects with the data type 'tuple':

mytuple = ("apple", "banana", "cherry")

print(type(mytuple))


# The tuple() Constructor

# It is also possible to use the tuple() constructor to make a tuple.


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

print(thistuple)


# Access Tuple Items

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

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

print(thistuple[1])


# Negative Indexing

# Negative indexing means start from the end.

# -1 refers to the last item, -2 refers to the second last item etc.

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

```
print(thistuple[-1])
```

 # Range of Indexes

```
thistuple = ("apple", "banana", "cherry", "orange", "kiwi", "melon", "mango")
print(thistuple[2:5])
```

Range of Negative Indexes

```
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")
```

 # Change Tuple Values

 # Once a tuple is created, you cannot change its values. Tuples are unchangeable, or immutable as it also is called.But there is a workaround. You can convert the tuple into a list, change the list, and convert the list back into a tuple.

```
x = ("apple", "banana", "cherry")
y = list(x)
y[1] = "kiwi"
x = tuple(y)
```

Add Items:

1. Convert into a list: Just like the workaround for changing a tuple, you can convert it into a list, add your item(s), and convert it back into a tuple

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

 # 2. Add tuple to a tuple. You are allowed to add tuples to tuples, so if you want to add one item, (or many), create a new tuple with the item(s), and add it to the existing tuple:

```
thistuple = ("apple", "banana", "cherry")
y = ("orange",)
thistuple += y
print(thistuple)
```

Unpacking a Tuple

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

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

 # But, in Python, we are also allowed to extract the values back into variables. This is called "unpacking":

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

 # 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])
```

 # # Join Two Tuples

```
tuple1 = ("a", "b" , "c")
tuple2 = (1, 2, 3)

tuple3 = tuple1 + tuple2
print(tuple3)
```

 # Multiply Tuples

```
fruits = ("apple", "banana", "cherry")
mytuple = fruits * 3

print(mytuple)
```