```
# A tuple is an immutable, ordered collection of elements in Python.
# It can contain duplicates.
# Like Lists, tuples are ordered and we can access their elements using their index values
# We cannot update items to a tuple once it is created.
# Tuples cannot be appended or extended.
# We cannot remove items from a tuple once it is created.
# It allows storing multiple values in a single variable. Tuples are defined using parentheses ().
```

```
# Empty Tuple
empty_tuple = ()

# Tuple with values
sample_tuple = (1, 2, 3, "Hello", 4.5)

# Single element tuple (comma is necessary)
single_element_tuple = (10,)

print(type(single_element_tuple))  # Output: <class 'tuple'>
```

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

```
# Python tuples have only two built-in methods because they are immutable.
t = (1, 2, 3, 2, 5, 2)

# Count occurrences of 2
print(t.count(2))  # Output: 3

# Find the first index of 5
print(t.index(5))  # Output: 4
```

```
3
4
```

```
# Elements in a tuple can be accessed using indexing and slicing.
t = (10, 20, 30, 40)
print(t[0])  # Output: 10
print(t[-1]) # Output: 40 (Negative indexing)

t = (1, 2, 3, 4, 5, 6)
print(t[1:4])   # Output: (2, 3, 4)
print(t[:3])    # Output: (1, 2, 3)
print(t[::2])   # Output: (1, 3, 5)  # Step slicing
```

```
10
40
(2, 3, 4)
(1, 2, 3)
(1, 3, 5)
```

```
# To Concatenation of Python Tuples, we will use plus operators(+).
t = ("Govind","Sahil")
t1 = (1,2,3,1)
t2 = (2,"Govind",1)
res = t+t1+t2
print(res)
```

```
('Govind', 'Sahil', 1, 2, 3, 1, 2, 'Govind', 1)
```

```
# iterating using for
t = ("Kite","Flying",12,31)
for i in t:
  print(i, end = " ")
```

```
Kite Flying 12 31
```

```
# Code for creating nested tuples
t1 = (0, 1, 2, 3)
t2 = ('python', 'geek')

t3 = (t1, t2)
print(t3)
```

```
t4 = ((11,22,33),(44,55,66))
print(t4)
```

```
((0, 1, 2, 3), ('python', 'geek'))
((11, 22, 33), (44, 55, 66))
```

```
# Code to create a tuple with repetition
t = ('python',)*3
print(t)
t = (1, 2)
print(t * 3)  # Output: (1, 2, 1, 2, 1, 2)
```

```
('python', 'python', 'python')
(1, 2, 1, 2, 1, 2)
```

```
#basic reversing
print(t[::-1])
```

```
(2, 1)
```

```
#deleting a tuple
del t
# print(t)
```

```
# len of tuple
print(len(t1))
```

```
4
```

```
# tuple with different datatypes
t = ("immutable", True, 23)
print(t)
```

```
('immutable', True, 23)
```

```
# Code for converting a list  into a tuple
a = [0, 1, 2]
t = tuple(a)

print(t)
```

```
(0, 1, 2)
```

```
# Creating a tuple without brackets
t = 4, 5, 6
print(t)  # Output: (4, 5, 6)
```

```
(4, 5, 6)
```

```
# Tuple packing
a, b, c = 11, 12, 13
t = (a, b, c)
print(t)  # Output: (11, 12, 13)
```

```
(11, 12, 13)
```

```
# Tuple Unpacking
t = (10, 20, 30)
a, b, c = t

print(a)  # Output: 10
print(b)  # Output: 20
print(c)  # Output: 30
```

```
10
20
30
```

```
# Tuple operations are faster than list operations.

# Tuples consume less memory than lists.
```

```
# Tuple Comprehension (Using Generators)
# Unlike lists, tuples do not support comprehension. However, we can use a generator expression to create a tuple dynamically.

# Generator Expression inside tuple()
t = tuple(x**2 for x in range(5))
print(t)  # Output: (0, 1, 4, 9, 16)
# Since tuples are immutable, Python does not allow modifying elements directly. Instead, we use tuple reconstruction.
```

```
(0, 1, 4, 9, 16)
```

```
nested_tuple = ((1, 2, 3), (4, 5, 6), (7, 8, 9))
print(nested_tuple[1])    # Output: (4, 5, 6)
print(nested_tuple[1][2]) # Output: 6
```

```
(4, 5, 6)
6
```

```
# Unpacking a Tuple with * (Variable-Length Argument)
a, *b, c = (10, 20, 30, 40, 50)
print(a)  # Output: 10
print(b)  # Output: [20, 30, 40] (List)
print(c)  # Output: 50
```

```
10
[20, 30, 40]
50
```

```
tup1 = (1,2,[3,4,5,6])
print(tup1)
tup1[2][0]=33
print(tup1)
```

```
(1, 2, [3, 4, 5, 6])
(1, 2, [33, 4, 5, 6])
```