Python Tuples

Python provides another type that is an ordered collection of objects, called a tuple.

Tuples are identical to lists in all respects, except for the following properties:

1. Tuples are defined by enclosing the elements in parentheses (()) instead of square brackets ([]).
2. Tuples are immutable

Immutable – unchangeable (technically: Fixed length object or fixed object values)

“““A tuple is a sequence of immutable Python objects (fixed length object). Tuples are sequences, just like lists. The differences between tuples and lists are, the tuples cannot be changed unlike lists. tuples use parentheses (), whereas lists use square bracket [].”””

Creating tuple

A tuple is created by placing all the items (objects) inside parentheses () and separated by commas (,). Sometimes parentheses are optional.

A tuple can have any number of items and they may be of different types (integer, float, list, string, etc.).

```
# initialization Empty tuple
t = ()
t = tuple()  # constructor
print(t, type(t))  # () <class 'tuple'>

# init tuple with single object
t = (10,)
t = 10,
print(t, type(t))  # (10,) <class 'tuple'>

# tuple operation
# accessing the elements
#     1. indexing : element a position
t = (2, 43, 'hello', [2, 43])
d = t[1]
print(d, type(d))  # 43 <class 'int'>

#     2. slicing : elements at given range
t = (2, 43, 'hello', [2, 43])
d = t[1:3]
print(d, type(d))  # (43, 'hello') <class 'tuple'>
```

```python
# item assignment
t = (2, 43, 'hello', [2, 43])
t[0] = 'hello'  # TypeError: 'tuple' object does not support item
assignment
print(t)
# output error

# item deletion
t = (2, 43, 'hello', [2, 43])
del t[0]  # TypeError: 'tuple' object doesn't support item deletion
print(t)
# output error


# list in tuple
t = (2, 43, 'hello', [2, 43])
t[-1].append('Hi')  # list can add or remove the elements without
changing there id in namespace
t[-1].clear()
print(t)  # (2, 43, 'hello', [])

# tuple creation with multiple elements number
# using tuple constructor or typecast
a1 = tuple(range(1, 10))
print(a1, type(a1))  # (1, 2, 3, 4, 5, 6, 7, 8, 9) <class 'tuple'>

# using concat
a2 = ()
for i in range(1, 10):
    a2 += (i,)  # two tuples can concat or append like string but
change the id every time
print(a2, type(a2))  # (1, 2, 3, 4, 5, 6, 7, 8, 9) <class 'tuple'>


# tuple generator (iterator)
a3 = (i for i in range(1, 10))
print(a3, type(a3))  # <generator object <genexpr> at 0x0E324F30>
<class 'generator'>
# A generator is a special kind of iterator, which stores the
instructions for how to generate each
# of its members, in order, along with its current state of
iterations.
# It generates each member, one at a time, only as it is requested via
iteration.
```

```python
tp = tuple(a3)  # gen to tuple type
print(tp, type(tp))  # (1, 2, 3, 4, 5, 6, 7, 8, 9) <class 'tuple'>


# tuple other operation
# membership
t = (2, 4, 6, 89, 1)
item = 1
out = item in t
print(out)  # True
item = 100
out = item in t
print(out)  # False

# multiplication with number
t = (2,5)
out = t*2
print(out, type(out)) # (2, 5, 2, 5) <class 'tuple'>

out = t*0
print(out, type(out))  # () <class 'tuple'>

# Addition (concat)
t1 = (2, 4)
t2 = (4, 8)
t = t1 + t2
print(t, type(t))  # (2, 4, 4, 8) <class 'tuple'>

# iteration with tuple (traverse)
t = (2, 54, 'hello')
for i in t:
    print(i)
# 2
# 54
# hello

# methods in tuple
'''
Only two methods in tuple
1. index
2. count

both methods returns integer value
'''
```

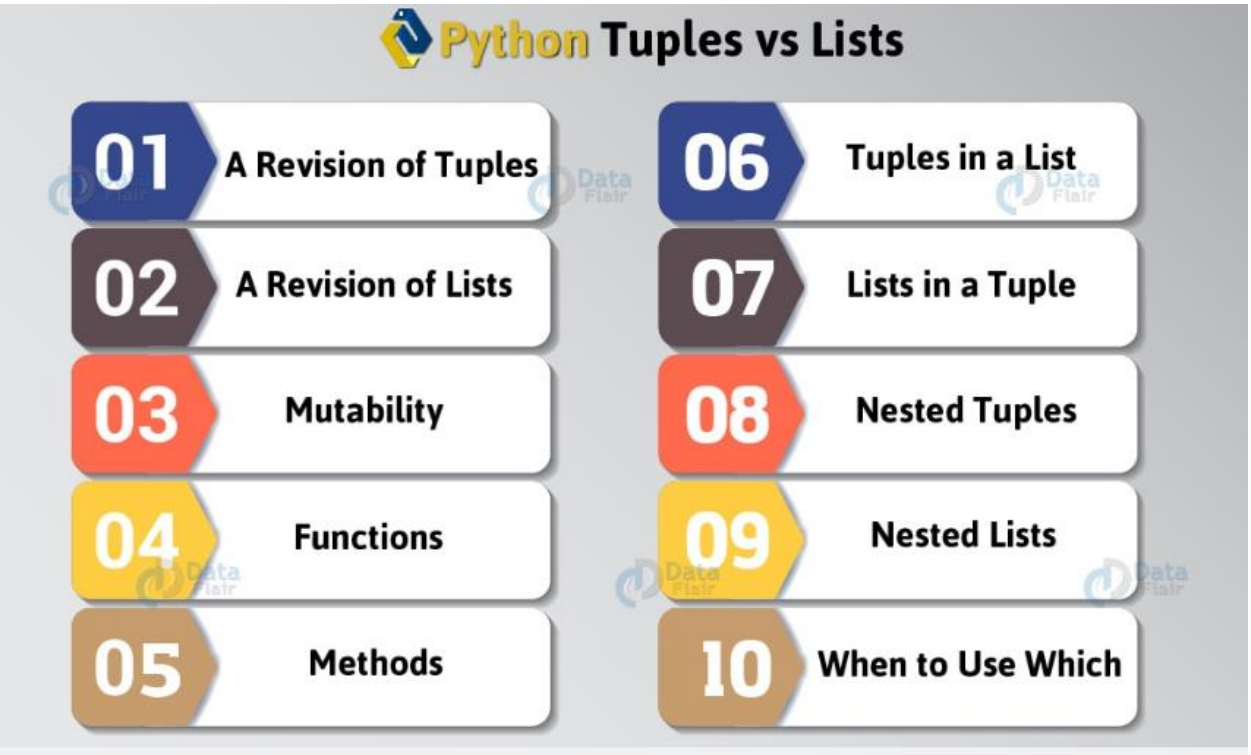
```python
# index : return match index of the element
t = ('a', 'h', 'Hi', 2, 65)
out = t.index(2)
print(out)  # 3

out = t.index('Hello')  # ValueError: tuple.index(x): x not in tuple
print(out) # error

# count : count the frequency of the elements within the tuple
t = ('a', 'h', 'Hi', 2, 65)
out = t.count(2)
print(out)  # 1

out = t.count('Hello')
print(out)  # 0
```

Assignment: **List vs Tuple** (note down the points for comparison)



Programs for practice

1. Write a Python program to create a tuple.

2. Write a Python program to create a tuple with different data types.

3. Write a Python program to create a tuple with numbers and print one item.

4. Write a Python program to unpack a tuple in several variables.

5. Write a Python program to add an item in a tuple.

6. Write a Python program to convert a tuple to a string.

7. Write a Python program to get the 4th element and 4th element from last of a tuple.

8. Write a Python program to create the colon of a tuple.

9. Write a Python program to find the repeated items of a tuple.

10. Write a Python program to check whether an element exists within a tuple.

11. Write a Python program to convert a list to a tuple.

12. Write a Python program to remove an item from a tuple.

13. Write a Python program to slice a tuple.

14. Write a Python program to find the index of an item of a tuple.

15. Write a Python program to find the length of a tuple.

16. Write a Python program to convert a tuple to a dictionary.

17. Write a Python program to unzip a list of tuples into individual lists.

18. Write a Python program to reverse a tuple.

19. Write a Python program to convert a list of tuples into a dictionary.

20. Write a Python program to print a tuple with string formatting.
Sample tuple : (100, 200, 300)
Output : This is a tuple (100, 200, 300)

21. Write a Python program to replace last value of tuples in a list.
Sample list: [(10, 20, 40), (40, 50, 60), (70, 80, 90)]
Expected Output: [(10, 20, 100), (40, 50, 100), (70, 80, 100)]

22. Write a Python program to remove an empty tuple(s) from a list of tuples.
Sample data: [(), (), ('',), ('a', 'b'), ('a', 'b', 'c'), ('d')]
Expected output: [('',), ('a', 'b'), ('a', 'b', 'c'), 'd']

23. Write a Python program to sort a tuple by its float element.
Sample data: [('item1', '12.20'), ('item2', '15.10'), ('item3', '24.5')]
Expected Output: [('item3', '24.5'), ('item2', '15.10'), ('item1', '12.20')]

24. Write a Python program to count the elements in a list until an element is a tuple