- **created by**: Sudip Ghimire
- **URL**: https://www.sudipghimire.com.np
- **GitHub**: https://github.com/ghimiresdp

go to course contents

Chapter 3.2. Tuple

Table of Contents

Introduction

Tuples are immutable sequences, typically used to store collections of heterogeneous data (such as the 2-tuples produced by the enumerate() built-in). Tuples are also used for cases where an immutable sequence of homogeneous data is needed (such as allowing storage in a set or dict instance). Reference: https://docs.python.org/3.9/library/stdtypes.html?highlight=tuple#tuples

Some of the features of tuple are as follows:

- Tuples are ordered
- Tuples index starts with 0
- Tuples can have more than one data types
- Tuples can have non-unique items
- Tuples can not be changed or modified once initialized
- Tuples are represented by small brackets `()` instead of `[]`
- Tuples are faster than lists

Creating/Constructing a tuple

Tuples can be constructed with various methods which are as follows:

- Using a pair of parentheses to denote an empty tuple. Eg: `()`
- Using a trailing comma for a singleton tuple. Eg: `(a, )`
- Using multiple values separated with commas within parenthesis. Eg: `(1, 2, 3)`
- Using the `tuple()` method. Eg: `tuple([1,2,3])`

Some examples:

```
tuple_1 = ()  # creates an empty tuple

tuple_2 = (1,)    # creates a tuple with only one element

tuple_3 = (1, 2, 3) # creates a tuple containing 1, 2, 3

tuple_4 = tuple([1, 2, 3])  # creates a tuple from a list [1, 2, 3]

# a tuple with items organized in multiple lines
tuple_5 = (
    'cat',
    'dog',
    'cow'
)

# A tuple containing random data types
tuple_6 = (1, 'John', 'Moon', True, 45.62)


t1: tuple = (1, 2, 3)   # type hinting for tuple
```

Note: *Remember, if we do not add comma to `tuple_2`, then it initializes an integer with value `1` rather than initializing the tuple containing single element.*

accessing an item of a tuple

Accessing an item of a tuple is similar to that of the list. The index of tuple also starts at zero. We use index inside large brackets `[]` to access an item of the tuple at that index

example:

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

print(numbers[2])   # 3
```

getting the slice of a tuple

Getting slice of a tuple is also similar to that of a list. For more, you can review the list chapter.

Example:

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

print(numbers[:3])  # (1, 2, 3)
```

Hack: Changing the value of the tuple

As tuples are immutable, we cannot change the value of tuple directly.

Example:

```
odd = (1, 3, 5)

odd[0] = -1
# TypeError: 'tuple' object does not support item assignment
```

So If we want to change the value of tuple, then we can add a new value to the existing tuple and re-initialize it.

Example:

```
tuple_1 = (1, 2, 3)
tuple_1 = tuple_1 + (4, 5, 6)
```

or we can convert tuple into list by typecasting and then perform assignment operations and then convert it back to tuple again.

Example:

```
even = (2, 4, 6, 8, 9)

even = list(even)   # [2, 4, 6, 8, 9]
even[-1] = 10       # [2, 4, 6, 8, 10]
even = tuple(even)  # (2, 4, 6, 8, 10)
```

We can also use unpacking method to achieve similar results

```
even = (2, 4, 6, 8)
other_even = (10, 12, 14)
even = (*even, *other_even, 16) # (2, 4, 6, 8, 10, 12, 14, 16)
```

Note: *we will learn more about unpacking once we complete advanced data types.*

finding out the length of tuple

```
tup = (1, 45, 23, 56, 99, -42)
print(len(tup)) # 6
```

some useful functions of tuple that are similar to list

- `tuple.count()`
- `tuple.index()`