**Tuple**

* a tuple is an ordered, immutable collection of elements.
* This means that once a tuple is created, you cannot modify its contents -
* you can't add, remove, or change elements.
* Tuples are defined using parentheses () and can contain elements of different data types.
* Tuples are commonly used in situations where you want to ensure the integrity of data or when you need an ordered and immutable sequence of elements.
* They are particularly useful for representing fixed collections of values.

example

T = ("sheema",25, 5.2 )

type(T)

tuple

characteristics of tuple

1. **Immutable:**

* Once a tuple is created, its elements cannot be changed.
* we have no methods while list has different methods in directory

example:

dir(tuple)

['\_\_add\_\_',

'\_\_class\_\_',

'\_\_class\_getitem\_\_',

'\_\_contains\_\_',

'\_\_delattr\_\_',

'\_\_dir\_\_',

'\_\_doc\_\_',

'\_\_eq\_\_',

'\_\_format\_\_',

'\_\_ge\_\_',

'\_\_getattribute\_\_',

'\_\_getitem\_\_',

'\_\_getnewargs\_\_',

'\_\_getstate\_\_',

'\_\_gt\_\_',

'\_\_hash\_\_',

'\_\_init\_\_',

'\_\_init\_subclass\_\_',

'\_\_iter\_\_',

'\_\_le\_\_',

'\_\_len\_\_',

'\_\_lt\_\_',

'\_\_mul\_\_',

'\_\_ne\_\_',

'\_\_new\_\_',

'\_\_reduce\_\_',

'\_\_reduce\_ex\_\_',

'\_\_repr\_\_',

'\_\_rmul\_\_',

'\_\_setattr\_\_',

'\_\_sizeof\_\_',

'\_\_str\_\_',

'\_\_subclasshook\_\_',

'count',

'index']

1. **Heterogeneous Elements:**

* Tuples can contain elements of different data types.

example

T = ('sheema', 25, 5.2)

print(type(T[0]))

=<class 'str'>

print(type(T[1]))

= <class 'int'>

print(type(T[2]))

<class 'float'>

1. **Indexing**

* we can access the individual element in the list due to its unique index (position)
* python uses zero index language (starts from 0)
* uses [] to access the single element with its position in between

|  |  |
| --- | --- |
| Positive indexing | Negative indexing |
| * left to right | * **right to left** |
| * strats from 0 , and so on (forward) | * **starts from -1 and backwards** |
| * Example   T = ('sheema', 25, 5.2)  print(T[0])  = ”sheema” | * **Example**   **T = ('sheema', 25, 5.2)**  **Print(T[-1])**  **= 5.2** |

1. **Slicing**

* continuous part of the tuple
* create a new tuple
* Use the colon : to specify a range of indices.
* The syntax is sarting\_index : end\_index
* it includes elements from start\_index to end index , excluding the elements of end\_index
* For example:

Tslice = T[0:2]

Tslice

=('sheema', 25)

type(Tslice)

=tuple

1. **Nesting:**

* Tuples can contain other tuples, allowing for nested structures
* example

my\_tuple = (("sheema","masood"), 25, 5.2)

type(my\_tuple[0])

= tuple

1. **Use as Dictionary Keys:**

* Tuples are hashable and can be used as keys in dictionaries.
* example

my\_dict = {"name":"sheema" , 'age':25, "height": 5.2}

1. **Tuple Unpacking:**

* You can unpack a tuple into individual variables.

numbers = ( 1,2,3)

a,b,c = numbers

print(a)

=1

print(b)

2

print(c)

=3

1. **Size Efficiency:**

* Tuples generally consume less memory than lists, making them more memory-efficient for certain use cases.