Tuples In Python

A tuple in Python is an immutable, ordered collection of elements. It is similar to a list in that it can store a variety of data types, including numbers, strings, and other objects, but unlike lists, tuples cannot be modified once they are created.

This immutability means that you can't add, remove, or change elements within a tuple after it's been defined.

Characterstics

- Ordered
- Unchangeble
- Allows duplicate

Creating Tuples

In [ ]:
```
# How to create empty tuple
t1 =()
print(t1)

# creating tuple with single item
t2 = ('hello',)
print(t2)
print(type(t2))

# homo
t3 = (1,2,3,4,5)
print(t3)

# hetero
t4 = (1,2.5,True,[1,2,3])
print(t4)

# tuple
t5 = (1,2,3,(4,5))
print(t5)

# using type conversion
t6 = tuple('hello')
print(t6)
```

```
()
('hello',)
<class 'tuple'>
(1, 2, 3, 4, 5)
(1, 2.5, True, [1, 2, 3])
(1, 2, 3, (4, 5))
('h', 'e', 'l', 'l', 'o')
```

Accessing items in Tuples

- Indexing
- Slicing

In [ ]:
```
print(t3)
```

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

In [ ]:
```
# extract 4 from the tuple
print(t3[3])

# extract 3,4,5
print(t3[2:])
```

```
4
(3, 4, 5)
```

In [ ]:
```
print(t5)
t5[-1][0]
```

```
(1, 2, 3, (4, 5))
```
Out[ ]:
```
4
```

Editing items in tuple/Deleting in tuple

- Tuple are immutable like string it cannot change or Delete

In [ ]:
```
print(t5)
del t5[-1]
```

```
(1, 2, 3, (4, 5))
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
c:\Users\disha\Downloads\Pythoon 100 Days\100_Days_OF_Python\Day11 - Tuples In Python.ipynb Cell 10 line 2
      <a href='vscode-notebook-cell:/c%3A/Users/disha/Downloads/Pythoon%20100%20Days/100_Days_OF_Python/Day11%20-%20Tuples%20In%20Python.ipynb#X22sZmlsZQ%3D%3D?line=0'>1</a> print(t5)
----> <a href='vscode-notebook-cell:/c%3A/Users/disha/Downloads/Pythoon%20100%20Days/100_Days_OF_Python/Day11%20-%20Tuples%20In%20Python.ipynb#X22sZmlsZQ%3D%3D?line=1'>2</a> del t5[-1]

TypeError: 'tuple' object doesn't support item deletion
```

Operation on Tuples

In [ ]:
```
# + and *
t1 = (1,2,3,4)
t2 = (5,6,7,8)

print(t1 + t2)

# membership
print(1 in t1)

# iteration/loops
for i in t3:
    print(i)
```

```
(1, 2, 3, 4, 5, 6, 7, 8)
True
1
2
3
4
5
```

Tuple Functions

In [ ]:
```
# len/sum/min/max/sorted
t = (1,2,3,4,5)

print(len(t))

print(sum(t))

print(min(t))

print(max(t))

print(sorted(t))
```

```
5
15
1
5
[1, 2, 3, 4, 5]
```

In [ ]:
```
# count
t = (1,2,3,4,5)

t.count(3)
```

Out[ ]:
```
1
```

In [ ]:
```
# index
t = (4,64567,3454,11,33,55)

t.index(64567)
```

Out[ ]:
```
1
```

Special Syntax

In [ ]:
```
# tuple unpacking
a,b,c = (1,2,3)
print(a,b,c)
```

```
1 2 3
```

In [ ]:
```
a = 1
b = 2
a,b = b,a

print(a,b)
```

```
2 1
```

In [ ]:
```
a,b,*others = (1,2,3,4)
print(a,b)
```

```
print(others)
```

```
1 2
[3, 4]
```

In [ ]:
```
# zipping tuples
a = (1,2,3,4)
b = (5,6,7,8)

tuple(zip(a,b))
```

Out[ ]:
```
((1, 5), (2, 6), (3, 7), (4, 8))
```

Difference between Lists and Tuples

Lists and tuples are both data structures in Python, but they have several differences in terms of syntax, mutability, speed, memory usage, built-in functionality, error-proneness, and usability. Let's compare them in these aspects:

1. Syntax:
   - Lists are defined using square brackets `[ ]`, e.g., `my_list = [1, 2, 3]`.
   - Tuples are defined using parentheses `( )`, e.g., `my_tuple = (1, 2, 3)`.
2. Mutability:
   - Lists are mutable, which means you can change their elements after creation using methods like `append()`, `insert()`, or direct assignment.
   - Tuples are immutable, which means once created, you cannot change their elements. You would need to create a new tuple if you want to modify its contents.
3. Speed:
   - Lists are slightly slower than tuples in terms of performance because they are mutable. Modifying a list may involve resizing or copying elements, which can introduce overhead.
   - Tuples, being immutable, are generally faster for accessing elements because they are more memory-efficient and do not require resizing or copying.
4. Memory:
   - Lists consume more memory than tuples due to their mutability. Lists require extra memory to accommodate potential resizing and other internal bookkeeping.
   - Tuples are more memory-efficient since they don't have the overhead associated with mutable data structures.
5. Built-in Functionality:
   - Both lists and tuples have common operations like indexing, slicing, and iteration.
   - Lists offer more built-in methods for manipulation, such as `append()`, `insert()`, `remove()`, and `extend()`. Lists are better suited for situations where you need to add or remove elements frequently.
   - Tuples have a more limited set of operations due to their immutability but offer security against accidental modifications.
6. Error-Prone:

- Lists are more error-prone when it comes to accidental modification, especially in large codebases, as they can be changed throughout the code.
- Tuples are less error-prone since they cannot be modified once created, making them more predictable.

7. Usability:

- Lists are typically used when you need a collection of items that can change over time. They are suitable for situations where you want to add, remove, or modify elements.
- Tuples are used when you want to create a collection of items that should not change during the program's execution. They provide safety and immutability.

In summary, the choice between lists and tuples depends on your specific needs. Use lists when you require mutability and dynamic resizing, and use tuples when you want to ensure immutability and need a more memory-efficient, error-resistant data structure.