

**Tuple Data Structure**

1. **Tuple is exactly same as List except that it is immutable. i.e once we creates Tuple object,we cannot perform any changes in that object.**

Hence Tuple is Read Only version of List.

1. **If our data is fixed and never changes then we should go for Tuple.**
2. **Insertion Order is preserved**
3. **Duplicates are allowed**
4. **Heterogeneous objects are allowed.**
5. **We can preserve insertion order and we can differentiate duplicate objects by using index. Hence index will play very important role in Tuple also.**

Tuple support both +ve and -ve index. +ve index means forward direction(from left to right) and -ve index means backward direction(from right to left)

1. **We can represent Tuple elements within Parenthesis and with comma seperator. Parenethesis are optional but recommended to use.**

Eg:

|  |
| --- |
| **1. t=10,20,30,40** |
| **2. print(t)** |
| **3. print(type(t))** |
| **4.** |
| **5. Output** |
| **6. (10, 20, 30, 40)** |
| **7. <class 'tuple'>** |
| **8.** |
| **9. t=()** |
| **10. print(type(t)) # tuple** |

Note: We have to take special care about single valued tuple.compulsary the value should ends with comma,otherwise it is not treated as tuple.

Eg:

|  |  |
| --- | --- |
| **1.** | **t=(10)** |
| **2.** | **print(t)** |
| **3.** | **print(type(t))** |
| **4.** | |
| **5.** | **Output** |
| **6.** | **10** |
| **7.** | **<class 'int'>** |



Eg:

|  |  |
| --- | --- |
| **1.** | **t=(10,)** |
| **2.** | **print(t)** |
| **3.** | **print(type(t))** |
| **4.** | |
| **5.** | **Output** |
| **6.** | **(10,)** |
| **7.** | **<class 'tuple'>** |

## Q. Which of the following are valid tuples?

1. t=()

2. t=10,20,30,40

3. t=10

4. t=10,

5. t=(10)

6. t=(10,)

7.t=(10,20,30,40)

**Tuple creation:**

## 1. t=()

creation of empty tuple

## 2. t=(10,)

t=10,

creation of single valued tuple ,parenthesis are optional,should ends with comma

## 3. t=10,20,30

t=(10,20,30)

creation of multi values tuples & parenthesis are optional

## 4. By using tuple() function:

|  |  |
| --- | --- |
| **1.** | **list=[10,20,30]** |
| **2.** | **t=tuple(list)** |
| **3.** | **print(t)** |
| **4.** | |
| **5.** | **t=tuple(range(10,20,2))** |
| **6.** | **print(t)** |



# Accessing elements of tuple:

We can access either by index or by slice operator

## By using index:

|  |  |
| --- | --- |
| **1.** | **t=(10,20,30,40,50,60)** |
| **2.** | **print(t[0]) #10** |
| **3.** | **print(t[-1]) #60** |
| **4.** | **print(t[100]) IndexError: tuple index out of range** |

1. **By using slice operator:**

|  |  |
| --- | --- |
| **1.** | **t=(10,20,30,40,50,60)** |
| **2.** | **print(t[2:5])** |
| **3.** | **print(t[2:100])** |
| **4.** | **print(t[::2])** |
| **5.** | |
| **6.** | **Output** |
| **7.** | **(30, 40, 50)** |
| **8.** | **(30, 40, 50, 60)** |
| **9.** | **(10, 30, 50)** |

**Tuple vs immutability:**

Once we creates tuple,we cannot change its content. Hence tuple objects are immutable.

Eg: t=(10,20,30,40)

t[1]=70 TypeError: 'tuple' object does not support item assignment

# Mathematical operators for tuple:

We can apply + and \* operators for tuple

## Concatenation Operator(+):

|  |  |
| --- | --- |
| **1.** | **t1=(10,20,30)** |
| **2.** | **t2=(40,50,60)** |
| **3.** | **t3=t1+t2** |
| **4.** | **print(t3) # (10,20,30,40,50,60)** |



## Multiplication operator or repetition operator(\*)

|  |  |
| --- | --- |
| **1.** | **t1=(10,20,30)** |
| **2.** | **t2=t1\*3** |
| **3.** | **print(t2) #(10,20,30,10,20,30,10,20,30)** |

**Important functions of Tuple:**

1. **len()**

To return number of elements present in the tuple

Eg: t=(10,20,30,40)

print(len(t)) #4

## count()

To return number of occurrences of given element in the tuple

Eg: t=(10,20,10,10,20)

print(t.count(10)) #3

## index()

returns index of first occurrence of the given element.

If the specified element is not available then we will get ValueError.

Eg:

t=(10,20,10,10,20)

print(t.index(10)) #0

print(t.index(30)) ValueError: tuple.index(x): x not in tuple

## sorted()

To sort elements based on default natural sorting order

|  |  |
| --- | --- |
| **1.** | **t=(40,10,30,20)** |
| **2.** | **t1=sorted(t)** |
| **3.** | **print(t1)** |
| **4.** | **print(t)** |
| **5.** | |
| **6.** | **Output** |
| **7.** | **[10, 20, 30, 40]** |
| **8.** | **(40, 10, 30, 20)** |

We can sort according to reverse of default natural sorting order as follows



t1=sorted(t,reverse=True) print(t1) [40, 30, 20, 10]

## min() and max() functions:

These functions return min and max values according to default natural sorting order. Eg:

|  |  |
| --- | --- |
| **1.** | **t=(40,10,30,20)** |
| **2.** | **print(min(t)) #10** |
| **3.** | **print(max(t)) #40** |

## cmp():

It compares the elements of both tuples. If both tuples are equal then returns 0

If the first tuple is less than second tuple then it returns -1

If the first tuple is greater than second tuple then it returns +1 Eg:

|  |  |
| --- | --- |
| **1.** | **t1=(10,20,30)** |
| **2.** | **t2=(40,50,60)** |
| **3.** | **t3=(10,20,30)** |
| **4.** | **print(cmp(t1,t2)) # -1** |
| **5.** | **print(cmp(t1,t3)) # 0** |
| **6.** | **print(cmp(t2,t3)) # +1** |

Note: cmp() function is available only in Python2 but not in Python 3

# Tuple Packing and Unpacking:

We can create a tuple by packing a group of variables. Eg:

a=10 b=20 c=30 d=40

t=a,b,c,d

print(t) #(10, 20, 30, 40)

Here a,b,c,d are packed into a tuple t. This is nothing but tuple packing.



Tuple unpacking is the reverse process of tuple packing

We can unpack a tuple and assign its values to different variables Eg:

|  |
| --- |
| **1. t=(10,20,30,40)** |
| **2. a,b,c,d=t** |
| **3. print("a=",a,"b=",b,"c=",c,"d=",d)** |
| **4.** |
| **5. Output** |
| **6. a= 10 b= 20 c= 30 d= 40** |

Note: At the time of tuple unpacking the number of variables and number of values should be same. ,otherwise we will get ValueError.

Eg: t=(10,20,30,40)

a,b,c=t #ValueError: too many values to unpack (expected 3)

# Tuple Comprehension:

Tuple Comprehension is not supported by Python. t= ( x\*\*2 for x in range(1,6))

Here we are not getting tuple object and we are getting generator object.

|  |
| --- |
| **1. t= ( x\*\*2 for x in range(1,6))** |
| **2. print(type(t))** |
| **3. for x in t:** |
| **4. print(x)** |
| **5.** |
| **6. Output** |
| **7. D:\Python\_classes>py test.py** |
| **8. <class 'generator'>** |
| **9. 1** |
| **10. 4** |
| **11. 9** |
| **12. 16** |
| **13. 25** |



Q. Write a program to take a tuple of numbers from the keyboard and print its sum and average?

|  |
| --- |
| **1. t=eval(input("Enter Tuple of Numbers:"))** |
| **2. l=len(t)** |
| **3. sum=0** |
| **4. for x in t:** |
| **5. sum=sum+x** |
| **6. print("The Sum=",sum)** |
| **7. print("The Average=",sum/l)** |
| **8.** |
| **9. D:\Python\_classes>py test.py** |
| **10. Enter Tuple of Numbers:(10,20,30,40)** |
| **11. The Sum= 100** |
| **12. The Average= 25.0** |
| **13.** |
| **14. D:\Python\_classes>py test.py** |
| **15. Enter Tuple of Numbers:(100,200,300)** |
| **16. The Sum= 600** |
| **17. The Average= 200.0** |

# Differences between List and Tuple:

List and Tuple are exactly same except small difference: List objects are mutable where as Tuple objects are immutable.

In both cases insertion order is preserved, duplicate objects are allowed, heterogenous objects are allowed, index and slicing are supported.

|  |  |
| --- | --- |
| **List** | **Tuple** |
| **1) List is a Group of Comma separeated Values within Square Brackets and Square Brackets are mandatory.**  **Eg: i = [10, 20, 30, 40]** | **1) Tuple is a Group of Comma separeated Values within Parenthesis and Parenthesis are optional.**  **Eg: t = (10, 20, 30, 40)**  **t = 10, 20, 30, 40** |
| **2) List Objects are Mutable i.e. once we creates List Object we can perform any changes in that Object.**  **Eg: i[1] = 70** | **2) Tuple Objeccts are Immutable i.e. once we creates Tuple Object we cannot change its content.**  **t[1] = 70  ValueError: tuple object does**  **not support item assignment.** |
| **3) If the Content is not fixed and keep on**  **changing then we should go for List.** | **3) If the content is fixed and never changes**  **then we should go for Tuple.** |
| **4) List Objects can not used as Keys for Dictionries because Keys should be**  **Hashable and Immutable.** | **4) Tuple Objects can be used as Keys for Dictionries because Keys should be**  **Hashable and Immutable.** |