**Tuple**

1. Tuple a collection of objects separated by commas.
2. Tuple is one of the 4 built in data types in python used to store collection of data.
3. Tuple contains a group of elements which can be same or different data types
4. Tuples are immutable(unchangeable), meaning that we cannot change, add or remove items after the tuple has been created.
5. tuple is similar to a list in terms of indexing, nested objects, and repetition but a tuple is immutable, unlike lists which are mutable.

Example: tuple1 = (0, 1, 2, 3)

tuple1[0] = 4

print(tuple1)

1. It occupies less memory compare to list.
2. Represented using ()
3. Tuples are ordered, when we say that tuples are ordered, it means that the items have a defined order, and that order will not change.
4. Allows duplicates, since tuples are indexed, they can have items with the same value.
5. Tuple items are indexed, the first item has index [0], the second item has index [1] etc.

**CREATING TUPLE:**

We can create tuple by writing elements separated by commas inside parentheses.

1. With one element

~~Tup=(24)~~ = it ll become integer

Tup= (24,) = this is the correct way to write single element in tuple

1. With multiple elements

Tup=(10,20,30,40,50)

Tup=(33,”priyanka”,12.5,-122,”deo”)

Tup=(“priyanka”,100,[1,2,3],”deo”,”priyanka”) #allows duplicate values too

Tup=10,20,-50,”priyanka” #without parenthesis

1. Creating empty tuple

Tup=()

1. Creating tuple with tuple() constructor

Tup= tuple((10,20,30,40,50))

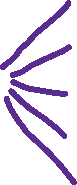
**INDEXING**

Tuple items are indexed, the first item has index [0], the second item has index [1] etc.

Tup= (10,20,30,40,50)

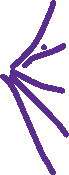
|  |  |
| --- | --- |
| **[0]** | **10** |
| **[1]** | **20** |
| **[2]** | **30** |
| **[3]** | **40** |
| **[4]** | **50** |

**Tup**



Negative indexing

Tup



|  |  |
| --- | --- |
| [-5] | 10 |
| [-4] | 20 |
| [-3] | 30 |
| [-2] | 40 |
| [-1] | 50 |

**ACESSING TUPLE ELEMENTS**

1. **Index number**

You can access tuple items by referring to the index number, inside square brackets

1. Positive indexing

*Tup=(10,20,30,40,50)*

*Print(Tup[0])*

*Print(Tup[1])*

1. Negative indexing

*Tup=(10,20,30,40,50)*

*Print(Tup[-1])*

*Print(Tup[-2])*

1. **Range of indexes (slicing)**

You can specify a range of indexes by specifying where to start and where to end the range.

When specifying a range, the return value will be a new tuple with the specified items.

1. **Slicing**

Syntax :

*new\_tuple\_name = tuple\_name[start:stop:stepsize]*

Return the third, fourth, and fifth item

The search will start at index 2 (included) and end at index 5 (not included).

tup=(10,20,30,40,"priyanka",12.5)

print(tup[2:5])

1. **Slice from the start**

By leaving out the start index, the range will start at the first item:

*tup=(10,20,30,40,"priyanka",12.5)*

*print(tup[ :4])*

1. **Slice to the end**

*tup=(10,20,30,40,"priyanka",12.5)*

*print(tup[1: ])*

1. ***Negative slicing***

*tup=(10,20,30,40,"priyanka",12.5)*

*print(tup[-5:-3])*

*print(tup[-5:])*

*print(tup[:-3])*

1. **Using For loop**

*a = (10, 20, 30,40,50)*

Without index

*for i in a:*

*print(i)*

With index

*n = len(a)*

*for i in range(n):*

*print(a[i])*

1. **Using While loop**

*a=(10,20,30,40,50)*

*i=0*

*n=len(a)*

*while i<n:*

*print(a[i])*

*i+=1*

**CONCATINATION OF TUPLES**

we use + operator to concatenate 2 tuples

*t1=(12,33,41,"priyanka")*

*t2=("deo",25,76,1)*

*t3=(t1+t2)*

*print(t3)*

**MULTILY TUPLES**

*t1=(12,33,41,"priyanka")*

*t2=t1\*3*

*print(t2)*

**NESTING OF TUPLES**

*t1=(12,33,41,"priyanka")*

*t2=("deo",25,76,1)*

*t3=(t1,t2)*

*print(t3)*

**Accessing nested tuple**

*t1=((12,33,41,"priyanka"),*

*("deo",25,76,1),*

*(21,65,80)*

*)*

*print(t1)*

*for i in t1: #outer for loop represent the rows*

*for j in i: #inner for loop represt the columns in each row*

*print(j ,end="")*

*print()*

**REPEATION TUPLES**

*t1=("priyanka",)\*6*

*print(t1)*

**DELETING A TUPLE**

*tuple3 = ( 0, 1)*

*del tuple3*

*print(tuple3)*

**ALIASING TUPLE**

Aliasing means giving another name to the existing object. It doesn’t mean copying.

*a = (10, 20, 30, 40, 50)*

*b = a*

*print(b)*

**COPYING TUPLE**

We can copy elements of tuple into another tuple using slice.

*a = (10, 20, 30, 40, 50)*

*b = a*

*b = a[0:5]*

*print(b)*

*b = a[1:4]*

*print(b)*

**CONVERTING LIST TO TUPLE**

*list1 = [0, 1, 2]*

*print(tuple(list1))*

*print(tuple('python'))*

**UPDATE TUPLE**

Once a tuple is created, you cannot change its values. Tuples are **unchangeable**, or **immutable** as it also is called.

There is one way, Convert the tuple into a list update it and convert the list back to tuple.

**change tuple values**

*t=('priyanka',24,12.5,456,75)*

*x=list(t)*

*x[3]='deo'*

*y=tuple(x)*

*print(y)*

**add items**

*t=('priyanka',24,12.5,456,75)*

*x=list(t)*

*x.append('Pune')*

*y=tuple(x)*

*print(y)*

**Add Tuple to the tuple**

*t=('priyanka',24,12.5,456,75)*

*x=('deo',)*

*t+=x*

*print(t)*

**Remove items**

*t=('priyanka',24,12.5,456,75)*

*x=list(t)*

*x.remove(12.5)*

*y=tuple(x)*

*print(y)*

**Delete tuple completely**

*t=('priyanka',24,12.5,456,75)*

*del t*

*print(t) #this will raise an error*

**PASSING TUPLE TO THE FUNCTION**

We can pass a tuple to a function while calling function.

*def show(t):*

*print(t)*

*print(type(t))*

*for i in t:*

*print(i)*

*t = (10, 20, 30, 'priyanka')*

*show(t)*

**UNPACK TUPLES**

When we create a tuple, we normally assign values to it. This is called "packing" a tuple

But, in Python, we are also allowed to extract the values back into variables. This is called "unpacking":

*user\_details=("priyanka",30,"MSC","Pune")*

*(name,age,education,address)=user\_details*

*print(name)*

*print(age)*

*print(education)*

*print(address)*

**Using Asterisk**\*

If the number of variables is less than the number of values, you can add an \* to the variable name and the values will be assigned to the variable as a list:

*user\_details=("priyanka",30,"MSC","Ambegoan","Pune","Maharashtra")*

*(name,age,education,\*address)=user\_details*

*print(name)*

*print(age)*

*print(education)*

*print(address)*

If the asterisk is added to another variable name than the last, Python will assign values to the variable until the number of values left matches the number of variables left.

*user\_details=("priyanka","Ambegoan","Pune","Maharashtra",30,"MSC",)*

*(name,\*address,age,education)=user\_details*

*print(name)*

*print(address)*

*print(age)*

*print(education)*

**TUPLE METHODS**

1. COUNT() : The count() method returns the number of times a specified value appears in the tuple

Syntax

tuple.count(value)

*t=('priyanka',24,12.5,24,456,75,24)*

*t.count(24)*

1. INDEX() : Searches the tuple for a specified value and returns the position of where it was found

Syntax

tuple.index(value)

*t=('priyanka',24,12.5,24,456,75,24)*

*t.index(24)*

**Python programs from assignments**

*1.* *Remove empty tuples from a list*

*lst=[1,(),2,3,(),4,(1,2,3)]*

*print("original list:",lst)*

*emptylst=[]*

*for i in lst:*

*if i==():*

*lst.remove(i)*

*print("resulted list:",lst)*

***2.Python | Convert a list of Tuples into Dictionary***

*def Convert(tup, di):*

*di = dict(tup)*

*return di*

*tups = [("akash", 10), ("gaurav", 12), ("anand", 14),*

*("suraj", 20), ("akhil", 25), ("ashish", 30)]*

*di = {}*

*print (Convert(tups, di))*

# ***3. program to find Tuples with positive elements in List of tuples***

*test\_list = [(4, 5, 9), (-3, 2, 3), (-3, 5, 6), (4, 6)]*

*print("The original list is : " + str(test\_list))*

*res = list(filter(lambda sub: all(ele >= 0 for ele in sub), test\_list))*

*print("Positive elements Tuples : " + str(res))*