EXPERIMENT NO : 2B

**NAME : AKASH RAMKRIT YADAV**

**ID.NO: VU4F2122016**

**BATCH : A**

**BRANCH : IT**

**DIV : A**

**Python Programs To Implement Tuple Operations using Built-in functions.**

# Aim :- Python Programs To Implement Tuple Operations

**- using Built-in functions.**

***THEORY:***

#### OUTPUT:

Python 3.11.0a4 (main, Jan 17 2022, 12:57:32) [MSC v.1929 32 bit

(Intel)] on win32

Type "help", "copyright", "credits" or "license()" for more information.

#AKASH RAMKRIT YADAV #ID NO:VU4F2122016 DATE:25/01/2023

**#TUPLES**

Tuples are used to store multiple items in a single variable. Tuple is one of 4 built-in data types in Python used to store collections of data, the other 3 are List, Set, and Dictionary, all with different qualities and usage.A tuple is a collection which is ordered and unchangeable.Tuples are written with round brackets.

## #Create a Tuple:

*>> TUPLE=("AKASH","RAMKRIT","YADAV")*

*print(TUPLE)*

*('AKASH', 'RAMKRIT', 'YADAV'***)**

### #Tuples allow duplicate values:

*>> tuple=("akash","akash","yadav","viram","suraj","viram") print(tuple)*

*('akash', 'akash', 'yadav', 'viram', 'suraj', 'viram')*

### #Tuple Length

To determine how many items a tuple has, use the len() function: Print the number of items in the tuple:

*>> TUPLE=("AKASH","RAMKRIT","YADAV")*

*print(len(TUPLE)) 3*

### #Create Tuple With One Item

To create a tuple with only one item, you have to add a comma after the item, otherwise Python will not recognize it as a tuple.

### One item tuple, remember the comma:

*>> TUPLE=("AKASH","RAMKRIT","YADAV")*

*print(TUPLE)*

*('AKASH', 'RAMKRIT', 'YADAV')*

### #NOT TUPLE:

*>>TUPLE=("AKASH")*

*print(TUPLE) AKASH*

## # DATA type

*>>TUPLE=("AKASH","RAMKRIT","YADAV")*

*print(type(TUPLE))*

*<class 'tuple'>*

#NOT TUPLE

*>>TUPLE=("AKASH")*

*print(type(TUPLE))*

*<class 'str'>*

### #Tuple Items - Data Types

Tuple items can be of any data type:

String, int and boolean data types:

***#CREATING TUPLE***

*tuple1=("akash","ramkrit","yadav") tuple2=(1,2,3,4,5,6,7,8)*

*tuple3=(True,False,True)*

*>>print(tuple1)*

*('akash', 'ramkrit', 'yadav')*

*>>print( tuple2 )*

*(1, 2, 3, 4, 5, 6, 7, 8)*

*>>print(tuple3) (True, False, True)*

# #Access Tuple Items

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

i.e) Print the second item in the tuple:

***#CREATING TUPLE***

*>>tuple1=("akash","ramkrit","yadav") print(tuple[1])*

*akash*

### #Negative Indexing

Negative indexing means start from the end.

-1 refers to the last item, -2 refers to the second last item etc**.**

i.e) Print the last item of the tuple:

***#CREATING TUPLE***

*>>tuple1=("akash","ramkrit","yadav") print(tuple1[-1])*

*yadav*

### #Range of Indexes

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.

i.e) Return the third, fourth, and fifth item:

***#CREATING TUPLE***

*>>tuple=("akash","akash","yadav","viram","suraj","viram") print(tuple[2:5])*

*('yadav', 'viram', 'suraj')*

*('yadav', 'viram', 'suraj')*

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

This example returns the items from the beginning to, but NOT included, "akash":

***#CREATING TUPLE***

*>>tuple=("akash","akash","yadav","viram","suraj","viram") print(tuple[:1])*

*('akash',)*

### #By leaving out the end value, the range will go on to the end of the list:

This example returns the items from "yadav" and to the end:

***#CREATING TUPLE***

*>>tuple=("akash","akash","yadav","viram","suraj","viram") print(tuple[2:])*

*('yadav', 'viram', 'suraj', 'viram')*

### #Range of Negative Indexes

Specify negative indexes if you want to start the search from the end of the tuple:

i.e) This example returns the items from index -4 (included) to index -1 (excluded)

***#CREATING TUPLE***

*>>tuple=("akash","akash","yadav","viram","suraj","viram") print(tuple[-4:-1])*

*('yadav', 'viram', 'suraj')*

**#Check if Item Exists**

To determine if a specified item is present in a tuple use the **in** keyword:

### “ if in tuple\_name ”

i.e)Check if "akash" is present in the tuple:

***#CREATING TUPLE***

**>>***tuple=("akash","akash","yadav","viram","suraj","viram") if "akash" in tuple:*

*print("Yes, 'akash' is in the tuple")*

*Yes, 'akash' is in the tuple*

# #Update Tuples

Tuples are unchangeable, meaning that you cannot change, add, or remove items once the tuple is created.

### #Change Tuple Values

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

But there is a workaround. You can convert the tuple into a list, change the list, and convert the list back into a tuple.

i.e)Convert the tuple into a list to be able to change it:

**#CREATING TUPLE**

*>> a=("akash","ramkrit","yadav") b=list(a)*

*b[1]="RAM"*

*a=tuple(b) print(a)*

*('akash', 'RAM', 'yadav')*

### #Add Items

Since tuples are immutable, they do not have a build-in append() method, but there are other ways to add items to a tuple.

#### #1. Convert into a list: Just like the workaround for changing a tuple, you can convert it into a list, add your item(s), and convert it back into a tuple.

i.e)Convert the tuple into a list, add "ravi", and convert it back into a tuple

*>>tuple1=("akash","ramkrit","yadav") y=list(tuple1)*

*y.append("ravi") tuple1=tuple(y) print(tuple1)*

*('akash', 'ramkrit', 'yadav', 'ravi')*

### #2. Add tuple to a tuple. You are allowed to add tuples to tuples, so if you want to add one item, (or many), create a new tuple with the item(s), and add it to the existing tuple:

**i.e)Create a new tuple with the value "raju", and add that tuple:**

>> tuple1=("akash","ramkrit","yadav") y = ("raju",)

tuple1 += y print(tuple1)

('akash', 'ramkrit', 'yadav', 'raju')

### #Remove Items

Note: You cannot remove items in a tuple.

Tuples are unchangeable, so you cannot remove items from it, but you can use the same workaround as we used for changing and adding tuple items:

i.e)Convert the tuple into a list, remove "apple", and convert it back into a

**#CREATING TUPLE**

*>>tuple1=("akash","ramkrit","yadav") y = list(tuple1)*

*y.remove("akash") tuple1=tuple(y) print(tuple1) ('ramkrit', 'yadav')*

### #you can delete the tuple completely:

The del keyword can delete the tuple completely:

*>>tuple1=("akash","ramkrit","yadav") del tuple1*

*print(tuple1)*

*Traceback (most recent call last):*

*File "<string>", line 3, in <module>*

*NameError: name 'tuple1' is not defined. Did you mean: 'tuple'?*

# # Unpack Tuples

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

### Packing a tuple:

**#CREATING TUPLE**

*>>name=("AKASH","RAM","RAJU","VIRAM")*

*print(tuple)*

*<class 'tuple'> print(name)*

*'AKASH', 'RAM', 'RAJU', 'VIRAM')*

### Unpack Tuples

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

*>> name=("AKASH","suraj","VIRAM")*

*(yadav1,yadav2,yadav3)= name print(yadav1)*

*print(yadav2) print(yadav3)*

*AKASH*

*suraj VIRAM*

# #Join Tuples

Join Two Tuples

To join two or more tuples you can use the + operator:

**#CREATING TUPLE**

*l1=(1,2,3,4,5,6)*

*l2=("akash","yadav") l3=l1+l2*

*print(l3)*

*(1, 2, 3, 4, 5, 6, 'akash', 'yadav')*

# #Multiply Tuples

If you want to multiply the content of a tuple a given number of times, you can use the \* operator:

i.e) Multiply the name tuple by 2:

**#CREATING TUPLE**

**>>***name=("akash","viram","suraj") l1= name \* 2*

*print(l1)*

*('akash', 'viram', 'suraj', 'akash', 'viram', 'suraj')*

# # Tuples loop

Loop Through a Tuple

You can loop through the tuple items by using a for loop. Iterate through the items and print the values:

>>name=("akash","viram","suraj") for x in name:

print(name)

('akash', 'viram', 'suraj')

('akash', 'viram', 'suraj')

('akash', 'viram', 'suraj')