CLASS XI
COMPUTER SCIENCE
STUDY MATERIAL
PYTHON - TUPLE

What is Tuple?

Are sequence that are used to store a tuple of values of any type

- Tuples are immutable i.e. you cannot change the elements of a tuple in place(inplace means in original tuple).
- Python will create a fresh tuple when we make changes to an element of tuple.

Creating and accessing tuples

Tuples are created just like list except by parenthesis "()" in place of square bracket "[]"

› **Examples of tuple :**

- ()
- (1,2,3)
- (2,2.5,4,1.2)
- ('a',1,"b",2,"c",3)
- ("red","green","blue")

Creating tuples/ tuple assignment

T = () # empty tuple

T = (value1, value2, value3,….)

This construct is known as tuple display construct

1. Empty Tuple

T = () Or

T = tuple()

Single element Tuple

>>> T = (20)

>>> T

20

>>> T = 5,

```
>>> T
(5,)
>>> T = (100,)
>>> T
(100,)
```

Creating long tuples

```
roots =  (1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20)
```

Nested tuples

- Each nested tuple also has indexing inside it. So items of nested tuples are accessed like [first tuple index][nested tuple index]. A tuple can have any number of nested tuples.

```
>>> T1 = (10,20,30,(40,50,60),100)
>>> len(T1)
>>> T1[1]
>>> T1[3][1]
# 5
# 20
# 50
```

Creating tuples from existing sequence

```
T = tuple(sequence)
>>> T = tuple('python')
>>> T
('p', 'y', 't', 'h', 'o', 'n')   >>> items=[100,200,300,400]
>>> T2 = tuple(items)
>>> T2
(100, 200, 300, 400)
>>> t1 = tuple(input('enter elements:'))
enter elements:abcde
>>> t1
('a', 'b', 'c', 'd', 'e')
```

Using eval() while creating tuple -

mytuple=eval(input("enter tuple elements"))

enter tuple elements(10,'ravi',10.5)

>>> mytuple (10, 'ravi', 10.5)

Accessing Tuple elements

Just like string, every individual elements of tuples are accessed from their index position which is from 0 to length-1 in forward indexing and starts from -1 in backward indexing. This method is called as **Indexing.**

› For example

› Fruits = (“mango”,”apple”,”guaua”,”pomegranate”,”cherry”)

› In above list items from mango to cherry are 0 to 4 and from cherry to mango will be -1 to -5

0	1	2	3	4
Mango	Apple	Guaua	Pomegranate	cherry
-5	-4	-3	-2	-1

Negative(backward) and positive(forward) indexing

Accessing tuple Elements

Tuple elements are accessed just like string like str[2] means character at 2nd index **tuple1[2]** means elements at 2nd index and **tuple1[1:3]** means all items between index 1 to 2. The stop value is not included in slicing.

Tuple_name[start index : stop index : step value] ; where step value is optional.

TUPLE OPERATIONS

They are same as list operations except that tuples are immutable.

› **Length :** the function len(T) will return number of elements in tuple

› **Indexing and Slicing : T[i]** will return item at index i and **T[i:j]** means all items between index i to j-1 and **T[i:j:n]** means every nth item between index i to j-1

› **Membership operator :** both “in” and “not in” can be used to check the presence of any item in tuple

› **Concatenation and Replication :** allows the use of “+” and “*” for tuple addition and replication

print(data2)
print(data[::-1])
print(data[4:-2])

data2 = data[4:-4]
print(data[-40:4])
print(data[1:6])

Difference from Lists

Although tuples are similar to list in many ways but yet there is one major difference is “Lists are mutable” while “Tuples are immutable”

```
>>> L1 = [10,20,30]
>>> T1 = (100,200,300)
>>> L1[1]=200                #valid
>>>T1[1]= 150                #Invalid as Tuples are immutable
```

Traversing tuple

We can use “for” loop to access every element of tuple. The loops used are same as lists.

```
qualifications=("B.A.","M.A.","B.Sc","M.Sc","MCA","M.Com","B.Tech")
for q in qualifications:
    print(q)
```

Or

```
qualifications=("B.A.","M.A.","B.Sc","M.Sc","MCA","M.Com","B.Tech")
for i in range(len(qualifications)):
     print("Index :“, i, „ „, qualifications[i])
```

Tuple operations

Joining Tuple/concatenation

```
>>> t1=(10,20,30)
>>> t2=('a','b','c')
>>> t3 = t1 + t2
>>> t3
(10, 20, 30, 'a', 'b', 'c')
```

Note: you can add tuple with only another tuple and not with *int, complex number, string or list*

>>> t1 + 20 #Error

If you want to add a tuple with another tuple with one value only and if you write statement as:

>>> t1 + (20) # Error, because (20) will be treated as number

To add single value tuple just add comma(,) after the value

as:

```
>>> t1 = (10,20,30)
>>> t1 + (50,)  (10,20,30,50)
```

Replicating Tuple/Repetition:

```
>>> t1=("do","it")
>>> t1*3
('do', 'it', 'do', 'it', 'do', 'it')
```

Slicing Tuples

T [start : end]

```
data=(10,20,30,1,7,9,100,51,75,80)
data2 = data[4:-4]
print(data2)
print(data[1:6])
print(data[4:-2])
print(data[-40:4])
print(data[::-1])
print(data[::-2])
print(data[2:10:2])
```

Output

```
(7, 9)
(20, 30, 1, 7, 9)
(7, 9, 100, 51)
(10, 20, 30, 1)
(80, 75, 51, 100, 9, 7, 1, 30,
20, 10)
(80, 51, 9, 1, 20)
(30, 7, 100, 75)
```

```
>>> tp1[2:5]*3
(15, 20, 8, 15, 20, 8, 15, 20, 8)
>>> tp1[2:5] + (500,1000)
```

(15,20,8,500,1000)

Membership operator (only int datatype can be used with the tuple)

```
>>> t=(1,2,3,4,4)
>>> t*3
(1, 2, 3, 4, 4, 1, 2, 3, 4, 4, 1, 2, 3, 4, 4)
>>> t*'a'
Traceback (most recent call last):
  File "<pyshell#9>", line 1, in <module>
    t*'a'
TypeError: can't multiply sequence by non-int of type 'str'
```

Packing & Unpacking tuples

Creating a tuple from a set of values is called packing and its reverse i.e. creating individual values from tuple's elements is called unpacking.

Unpacking is done by using following syntax: var1, var2, var3, … = tuple_Object

The number of variables on the left side should be same as the number of values inside the tuple.

Example:

```
>>> t1 = (100,200,300,400)     # packing
>>> a,b,c,d = t1        #unpacking
>>> a  100
>>> b  200
>>> c
T300
```

Deleting tuples

The del statement of python is used to delete elements and objects but as you know that tuples are immutable, which also means that individual elements of tuples cannot be deleted.

For example

```
del t1[2]                # Error, coz elements of tuple cannot be deleted
```

```
>>>t1 = (10,20,30)
>>> print(t1)
>>> (10,20,30)
>>> del t1
>>> print(t1)          # Error t1 is not defined
```

Tuple functions and methods

len() : returns number of elements in the tuple

```
>>> book = (“B001",",Let Us Python",",DP",500)
>>> len(book)  4
```

max() : it returns element from tuple having maximum value

```
>>> salary=(1000,1500,800,700,1200)
>>> max(salary)
1500
>>> fruits=("mango","pine apple","apple","carrot")
>>> max(fruits)  'pine apple„
```

Note: max() function will return maximum value only if all the elements in tuple is of same type. If elements are of different type then python will raise an exception.

```
>>> t1 = (10,20,30,(40,50),)
```

min() : it returns element from tuple having minimum value

```
>>> salary=(1000,1500,800,700,1200)
>>> min(salary)  700
>>> fruits=("mango","pine apple","apple","carrot")
>>> min(fruits)  'apple„
```

Note: min() function will return minimum value only if all the elements in tuple is of same type. If elements are of different type then python will raise an exception.

```
>>> t1 = (10,20,30,(40,50),90)
>>> min(t1)   # Error
```

index() : it return index value of given element in the list, if element

not present it raises ValueError exception

```
salary.index(800)
```

salary.index(5000)

count() : It return the count of any element in the tuple i.e. how many times the given element is in the tuple. If given element not in the tuple it return 0**.**

>>> val=(10,20,30,20,10,60,80,20)

>>> val.count(20) 3

>>> val.count(80) 1

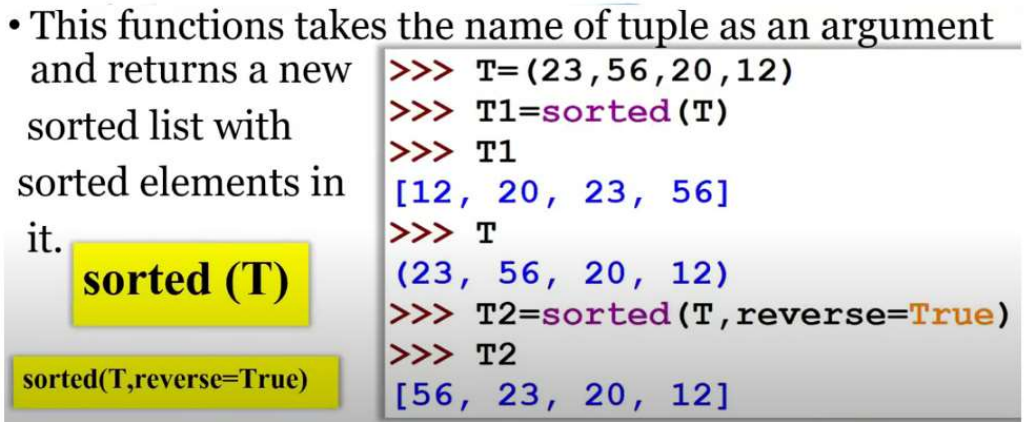
>>> val.count(100) 0

tuple(): this method is actually a constructor used to create tuples from different type of values.

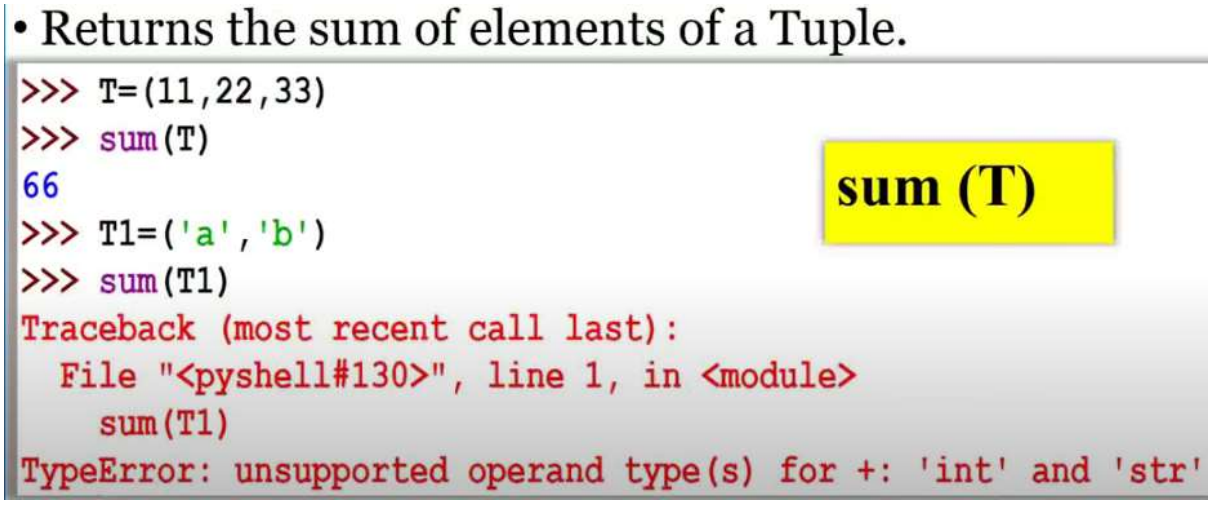
Creating empty tuple

tup = tuple()

sorted(): used to sort a tuple in an order.



sum()



Creating tuple from string

tup = tuple(“quick brown fox”)

Creating a tuple from a list

tup = tuple([1,20,40])

Creating a tuple from keys of dictionary

>>> tup = tuple({1:”One”,2:”Two”})

>>> tup # (1,2)

Note: in a tuple() we can pass only a sequence (string, list, dictionary) not a single value. If we pass value other than sequence it returns error.

>>> t = tuple(10)

Error: „int" object is not iterable

Using Tuple unpacking

>>> val = (10,20,30)

>>> a,b,c = val

>>> b=30

>>> val=(a,b,c)

>>> val (10, 30, 30)

Using constructor function of lists and tuples i.e. list() and tuple()

>>> foods=("rice","dosa","idli","mushroom","paneer")

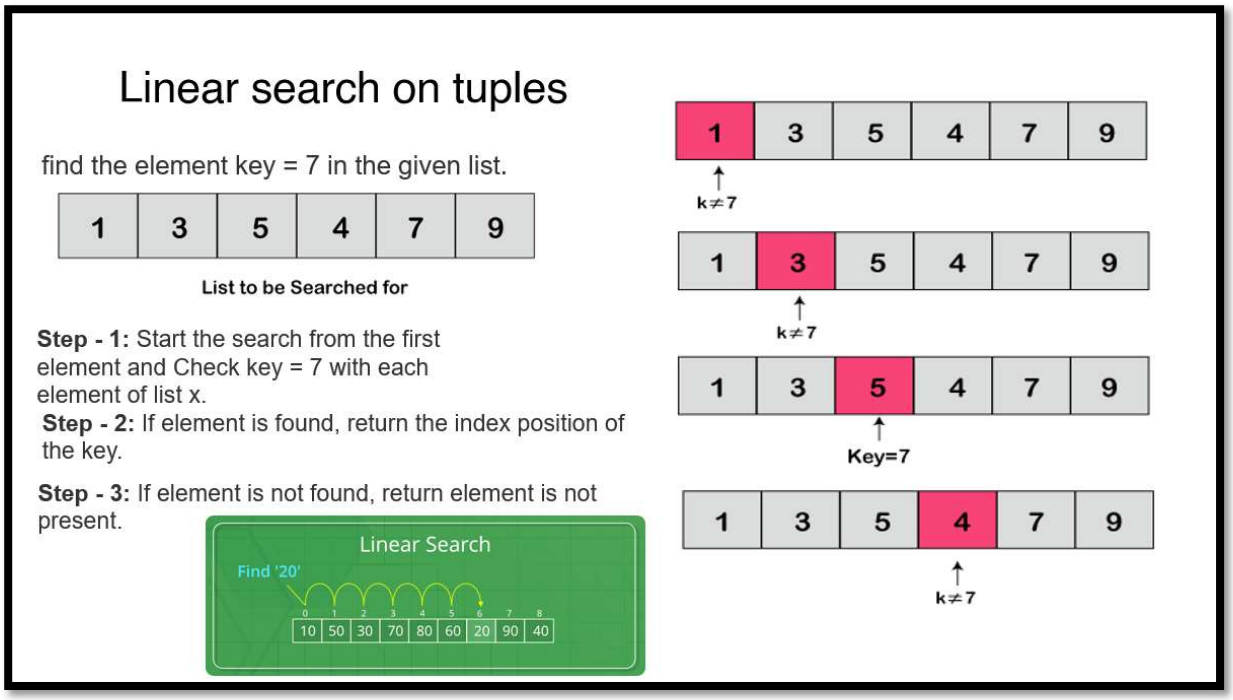
>>> myfood = list(foods)

>>> myfood[2]="biryani"

>>> foods = tuple(myfood)

>>> foods

('rice', 'dosa', 'biryani', 'mushroom', 'paneer')



How to Concatenate tuples to nested tuples

test_tup1 = (3, 4)

test_tup2 = (5, 6)

 # printing original tuples

print("The original tuple 1 : " , (test_tup1))

print("The original tuple 2 : " , (test_tup2))

 # Concatenating tuples to nested tuples

 # Using ", " operator during concatenation

res = ((test_tup1,) + (test_tup2,))

 # printing result

print("Tuples after Concatenating : ", (res))

CREATING TUPLE USING

WHILE LOOP

```
t=tuple()
n=int(input("Enter size of tuple:"))
i=0
while(i<n):
    a=int(input("enter value:"))
    t=t+(a,)
    i+=1
print(t)
```

FOR LOOP

```
t=tuple()
n=int(input("Enter size of tuple:"))
for i in range(n):
    a=int(input("enter value:"))
    t=t+(a,)

print(t)
```

MCQs

https://www.sanfoundry.com/python-mcqs-tuples/

PRACTICE QUESTIONS

https://pynative.com/python-tuple-exercise-with-solutions/

https://www.w3resource.com/python-exercises/tuple/

SUGGESTED PROGRAMS

1. Finding the minimum, maximum, mean of values stored in a tuple
2. linear search on a tuple of numbers
3. counting the frequency of elements in a tuple
4. Printing all even numbers from a tuple.
5. Print elements at even indexes in a tuple.
6. Printing all odd numbers from a tuple.
7. Print elements at odd indexes in a tuple.
8. Print all negative elements of a tuple.
9. Print those numbers from a tuple which are divisible by 5.
10. Copy only distinct elements from a tuple to a list.
11. Write a Python program to unpack a tuple in several variables

12. Write a Python program to remove an item from a tuple.
13. Calculate the product, multiplying all the numbers of a given tuple
14. Swap the values of two tuples. (hint:tuple1, tuple2 = tuple2, tuple1)

    Tuple1 = (11, 22)

    tuple2 = (99, 88)
15. Copy element 44 and 55 from the following tuple into a new tuple

    T=(11,22,33,44,55,66)