Class Notes

Prepared by **VIJITH P**

Class: **XI - B Subjects**: **COMPUTER SCIENCE**

Chapter/Unit: **Unit 8: TUPLES IN PYTHON**

Page/s From **1** to **13**

SILVER HILLS PUBLIC SCHOOL

Affiliated to CBSE New Delhi, No. 930433, Code No. 75439

Paroppady, Marikunnu PO, Kozhikode - 673012 Kerala, India

Phone: 0495 2370075, 2375144 Fax: 0495 2375155

E-mail: shpsoffice@gmail.com Web: www.silverhillspublicschool.org

a cmi institution • nurturing a youth sublime

TUPLES

Tuple is an ordered sequence of data enclosed within opening and closing braces ( ) and the elements are separated by commas.(,).

Characteristics of Tuples

Ordered – Tuple items are stored in an order and this order cannot be changed.

Immutable – Tuples are unchangeable. You cannot change, add, or remove items after the tuple has been created.

Allows duplicate values – Tuples can contain duplicate values.

Allows different data types values – The tuples can contain items of different data types.

Creating Tuples

A tuple can be created by enclosing elements inside parenthesis ( ) and all the elements can be separated by commas.

Syntax: variable=(value1,value2,value3,……)

Eg:

t=(1,2,3,4)

t1=('a','air',23,4.5) – tuple with different data types.

An empty tuple can be created as follows:

>>>tup=( ) – An empty tuple named 'tup' has been created.

>>>print(tup)

( ) – Output as empty tuple.

More examples

- ✓ tup=('hello',1,2,3.5,'python') – Heterogenous tuple.
- ✓ tup=10,20,30 – Parenthesis are not must.
- ✓ tup=((1,2,3),(4,5,6)) – Nested tuple.
- ✓ tup=((01,2,3),['hello','world']) – Tuple and List within a tuple.
- ✓ tup=([1,2,3],[4,5,6])- Lists within a tuple.
- ✓ tuple=('a','b','c',[10],[20],[30]) – A tuple containing strings and list.

Singleton tuple

A tuple having a single element is known as singleton tuple. If a tuple comprises a single element, the element should be followed by a comma to distinguish a tuple from a parenthesized expression.

Eg: tup=(70,)

Advantages using tuple:

- As tuple is immutable, iterating through a tuple is faster as compared to a list.
- If we have data that is not to be changed, then storing this data in a tuple will ensure that it is not changed accidently.

The use of tuple( ) function

An empty tuple can be created using tuple () function.

Syntax: variable=tuple()

Eg:1 >>>t=tuple()

>>>print(t)

>>>( )

Eg:2

>>> >>> T=tuple("string")

>>> print(T)

o/p: ('s', 't', 'r', 'i', 'n', 'g')

Eg:3

>>>t=tuple([1,2,3])

>>> print(t)

o/p: (1, 2, 3)

creating tuple by accepting user input using while loop

```
t=tuple()
n=int(input("Enter number of elements:"))
i=1
while(i<=n):
    a=input("Enter the number:")
    t=t+(a,)
    i=i+1
print("Tuple created as:")
print(t)
```

o/p:

```
Enter number of elements:5
Enter the number:1
Enter the number:4
Enter the number:8
Enter the number:9
Enter the number:7
Tuple created as:
('1', '4', '8', '9', '7')
```

Modify the above program using for loop

```
t=tuple()
n=int(input("Enter number of elements:"))
for i in range(n):
    a=input("Enter Number:")
    t=t+(a,)
print("output is:")
print(t)
```

o/p:

Enter number of elements:5

Enter Number:1

Enter Number:5

Enter Number:6

Enter Number:7

Enter Number:9

output is:

('1', '5', '6', '7', '9')

NESTING OF TUPLES

Tuples can be placed inside another tuples. When you add one or more tuples inside another tuple, the items in the nested tuples are combined together to form a new tuple.

Eg:

```
>>> tuple1=(1,2,3,4)
>>> tuple2=('python','book')
>>> tuple3=(tuple1,tuple2)
>>> print(tuple3)
```

o/p: ((1, 2, 3, 4), ('python', 'book'))

Implementation of tuple in real time situations:

Create a program to store roll number, name, and marks of students.

```
st=((200,"SANJAY",88),(201,"DEEPIKA",98),(202,"RADHIKA",97),(203,"SACHIN",99))
print("s_No",'\t',"Roll_No","Name",'\t\t',"Mark")
for i in range(0,len(st)):
    print((i+1),'\t',st[i][0],'\t',st[i][1],'\t',st[i][2])
```

o/p:

s_No	Roll_No	Name	Mark
1	200	SANJAY	88
2	201	DEEPIKA	98
3	202	RADHIKA	97
4	203	SACHIN	99

ACCESSING AND TRAVERSING A TUPLE

A tuple is a sequence of values which can be of any type and they are indexed by integer. Like lists and string there could be positive indexing like 0,1,2,3,4….. and negative indexing like -1.-2,-3,-4…..

Tuple Slicing

The items in a tuple can be accessed by using the slicing operator ':'. Tuple indices start at **zero**.

Syntax:

S1=t(start:stop:step)

Eg:

```
>>> tup=('Monday','Tuesday',2.5,32,'w')
>>> print(tup[2])
2.5
>>> print(tup[-2])
32
>>> print(tup[1:4])
('Tuesday', 2.5, 32)
>>> print(tup[-3:-1])
(2.5, 32)
>>> print(tup[-3::])
(2.5, 32, 'w')
```

```
>>> print(tup[-5:1])
('Monday',)
>>> tup[-5:2]==tup[0:3]
False
>>> tup[-5:2]==tup[0:2]
True
>>>
```

TRAVERSING A TUPLE

Traversing a tuple means accessing each element of a tuple one after the other at the same time. This can be done by *while* and *for* loop.

1. **Using in operator using for loop**

   'in' operator used with for loop to iterate each element of a tuple in sequence.

   Eg:

```
tup=('p','y','t','h','o','n')        o/p: p
for i in tup:                                y
   print(i)                                  t
                                             h
                                             o
                                             n
```

2. **Using range() function**

   Same as that of list operation.

```
Eg:                                  o/p
tup=('p','y','t','h','o','n')             p
n=len(tup)                                 y
for i in range(n):                         t
   print(tup[i])                           h
                                           o
                                           n
```

3. Using while loop

It needs length of tuple to keep the bounds and indexing to access the items of tuple.

Eg:

```
tup=('p','y','t','h','o','n')
i=0
while i<len(tup):
   print(tup[i])
   i=i+1
```

COMMON TUPLE OPERATIONS

1. Tuple multiplication/Repetition
   The '*' operator is used to repeat the elements of atuple by a specified number of times.

Operations on Tuple

- Concatenation
- Repetition
- Membership Testing
- Indexing
- Slicing
- Comparing Tuples
- Deleting a tuple

Concatenation

Concatenation means join two values together. You can concatenate two tuples in python using '+' operator.

Eg:

```
>>> x=(1,2,3,4)
>>> y=(5,6,7,8)
>>> z=x+y
>>> print(z)
o/p: (1, 2, 3, 4, 5, 6, 7, 8)
```

we can use '+' operator with tuple slice also.

Eg:

```
>>> tup=(1,2,3,4)
>>> tup[1:3]+(5,6)
o/p: (2, 3, 5, 6)
```

Note: we can only add tuple to a tuple.

```
>>>tup[1:4]+tup[0]
  Traceback (most recent call last):
   File "<pyshell#10>", line 1, in <module>
   tup[1:4]+tup[0]
   TypeError: can only concatenate tuple (not "int") to tuple
```

It can be written as follows:

```
>>> tup[1:4]+(tup[0],)
      o/p: (2, 3, 4, 1)
```

Repetition

Sometimes, while working with data, we might have a problem in which we need to replicate, i.e construct duplicates of tuples. The multiplication operator can be used to construct the duplicates of a container. This also can be extended to tuples even though tuples are immutable.

Example 1:

```
>>> t=(1,2)
>>> t1=t*2
 >>> t1
     (1, 2, 1, 2)
```

Example2:

```
>>> res=((t,)*2)
>>> print(res)
        ((1, 2), (1, 2))
```

Membership Testing

We can test if an item exists in a tuple or not, using the keyword 'in'.

Example:

```
>>> T=(1,2,3,'a','b','c')
>>> 1 in T
        True
>>> 'd' in T
        False
```

Indexing

We can use the index operator [] to access an item in a tuple, where the index starts from 0.

So, a tuple having 6 elements will have indices from 0 to 5. Trying to access an index outside of the tuple index range(6,7,... in this example) will raise an index error.

Example 1:

```
>>> T=(1,2,3,'a','b','c')
>>> T[2]
3
>>> T[4]
'b'
```

Example 2:

```
n_tuple = ("mouse", [8, 4, 6], (1, 2, 3))
# nested index
print(n_tuple[0][3]) # 's'
print(n_tuple[1][1]) # 4 6
```

Python allows negative indexing for its sequences.

The index of -1 refers to the last item, -2 to the second last item and so on.

Example:

```
>>> T=("mouse", [8, 4, 6], (1, 2, 3))
>>> T[-1]
(1, 2, 3)
```

Slicing

We can access a range of items in a tuple by using the slicing operator colon ':'

Example:

```
>>> T=(1,2,3,4,5,6,7,8,9,10)
>>> T[3:7]
(4, 5, 6, 7)
>>> T[-1:-5:-1]
(10, 9, 8, 7)
>>> T[::]
(1, 2, 3, 4, 5, 6, 7, 8, 9, 10)
```

Comparing Tuples

Example:

```
>>> T=(1,2,3)
>>> T1=(1.0,2.0,3.0)
>>> T==T1
True
>>> T<T1
False
```

Deleting a tuple

We cannot change the elements in a tuple. It means that we cannot delete or remove items from a tuple.

Deleting a tuple entirely, however, is possible using the keyword del.

7

```
>>> T1=(1.0,2.0,3.0)
>>> del T
>>> T
NameError: name 'T' is not defined
```

TUPLE FUNCTIONS:

Being immutable in nature unlike lists, tuple do not support methods such as append(), extend(), remove(), insert() and pop(). Since all these operations require modifications to be made on a tuple which is not permitted.

However, tuples work well with several built-in methods which are as follows:

1. len() – This function returns the length of a tuple, i.e., it counts total number of elements present in a tuple and returns the same.

```
>>> a=(5,'book',4,4,'new')
>>>len(a)
    5
```

2. count() – This function is used to count the occurrence of an item in the tuple.

```
>>> a=(5,'book',4,4,'new')
>>> a.count(4)
     2
```

3. any() – This function returns True if a tuple is having at least one item. If the tuple is empty, it will return False.

```
>>>  a=(1,)
>>>  any(a)
       True
>>>  b=()
>>>  any(b)
       False
```

4. min() and max()

   min() function shall return the element with minimum value from the tuple. max() function shall return the element with maximum value from the tuple.

   Note: To use the min() and max() function, all values in the tuple must be of same type.

```
>>>t=(10,'a',30)
       max(t)
  Traceback (most recent call last):
  TypeError: '>' not supported between instances of 'str' and   'int'
>>> a=(1,2,3,4,5)
>>> max(a)
      5
>>> a=(1,2,3,4,5)
      min(a)
      1
>>> str1=('Apple')
      max(str1)
      'p'
>>>   tup1=("ALI","RAHUL","VIKAS")
       max(tup1)
       'VIKAS'
```

5. sum(): This function returns sum of the elements of the tuple. It must be remembered that sum() method works on numeric values only.

```
>>> tuple1=(10,22,55,18,8,77,30)
>>> sum(tuple1)
      220
```

6. sorted() : It is used to sort the elements of a tuple. it returns a **list** after sorting and does not modify the original tuple. The sorting by default is done in ascending order.

```
>>> a=(6,7,4,2,1,5,3)
>>> sorted(a)
    [1, 2, 3, 4, 5, 6, 7]
```

7. index()

index() function finds the first index of a specified item and returns the index. It returns the first occurrence of a given element in a tuple, if found, else returns an error.

Syntax:

```
tuple.index(value,start,end)
```

Here, start and end are two optional parameters.

Eg:

```
>>> vote=('yes','no','yes','yes','no')
>>> vote.index("no")
    1
>>> vote.index('yes')
    0
>>>vote.index("yes",1)
    2
>>>vote.index("no",2)
    4
>>>vote.index("yes",1,4)
    2
>>> vote.index("maybe")
    Traceback (most recent call last):
    File "<pyshell#31>", line 1, in <module>
      vote.index("maybe")
    ValueError: tuple.index(x): x not in tuple
```

Link for reference:
https://python.org
https://geeksforgeeks.org

****