LECTURE 6: TUPLES AND SETS

INTRODUCTION TO TUPLES

- **Tuples consists of a number of values separated by comma and enclosed within parentheses.**

- **Tuple is similar to list, values in a list can be changed but not in a tuple.**

ADVANTAGES OF TUPLES OVER LIST

1. The elements of a **list** are changeable (**mutable**) whereas the elements of a **tuple** are unchangeable (**immutable**), this is the key difference between tuples and list.

2. The elements of a **list** are enclosed within **square brackets**. But, the elements of a **tuple** are enclosed by **parenthesis**.

3. Iterating tuples is faster than list.

CREATING TUPLES

- In tuples, elements may be enclosed by parenthesis.
- The elements of a tuple can be even defined without parenthesis.
- Whether the elements defined within parenthesis or without parenthesis, there is no different in it's function.

SYNTAX

- # Empty tuple
- **Tuple_Name = ( )**
- # Tuple with n number elements
- **Tuple_Name = (E1, E2, E2 ……. En)**
- # Elements of a tuple without parenthesis
- **Tuple_Name = E1, E2, E3 ….. En**

Example

```
tuple1 = (1, 2, 4, 10, 'A', 'Ali')
tuple2 = 1, 2, 4, 10, 'A', 'Ali'
print(tuple1)
print(tuple2)
```

Output

```
(1, 2, 4, 10, 'A', 'Ali')
(1, 2, 4, 10, 'A', 'Ali')
```

CREATING TUPLES USING TUPLE( ) FUNCTION

- The tuple( ) function is used to create Tuples from a list.
- When you create a tuple, from a list, the elements should be enclosed within square brackets.
- Syntax:

```
Tuple_Name = tuple( [list elements] )
```

Example

```
tuple_of_list= tuple([1, 2, 4, 10, 'A'])
print(tuple_of_list)
```

Output

```
(1, 2, 4, 10, 'A')
```

CREATING SINGLE ELEMENT TUPLE

- **While creating a tuple with a single element, add a comma at the end of the element.**

- **In the absence of a comma, Python will consider the element as an ordinary data type; not a tuple. Creating a Tuple with one element is called "Singleton" tuple.**

Example

```
single_tuple=(10)
print(type(single_tuple))
```

Output

```
<class 'int'>
```

If , was not used with single element then the data type won't be tuple

```
single_tuple=(10,)
print(type(single_tuple))
```

Output

```
<class 'tuple'>
```

If , was used with single element then the data type will be tuple

ACCESSING VALUES IN A TUPLE

- **Like list, each element of tuple has an index number starting from zero.**
- **The elements of a tuple can be easily accessed by using index number.**

Example

```
fruites =('apple', 'banana', 'orange')
print(fruites[0])
print(fruites[-1])
print(fruites[1:3])
```

Output

```
apple
orange
('banana', 'orange')
```

Example

```
tup=(10,90,105,'A','bus',200)
print(tup)
print(tup[0])
print(tup[-5])
print(tup[1:4])
print(tup[0:5:2])
```

Negative indexes	-6	-5	-4	-3	-2	-1
Positive indexes	0	1	2	3	4	5
Tuple elements	10	90	105	A	bus	200

Output

```
(10, 90, 105, 'A', 'bus', 200)
10
90
(90, 105, 'A')
(10, 105, 'bus')
```

UPDATE AND DELETE TUPLE

- A tuple is **immutable**, the elements in a tuple **cannot be changed**.
- Instead of altering values in a tuple, **joining** two tuples or deleting the entire tuple is **possible**.
- For deleting a tuple use del keyword.

Syntax:

del tuple_name

Example

```
fruites =('apple', 'banana', 'orange')
weights=(10, 20, 30)
tup=fruites+weights
print(tup)
```

Output

```
('apple', 'banana', 'orange', 10, 20, 30)
```

Example

```
fruites =('apple', 'banana', 'orange')
weights=(10, 20, 30)
tup=fruites+weights
print(tup)
del tup
print(tup)
```

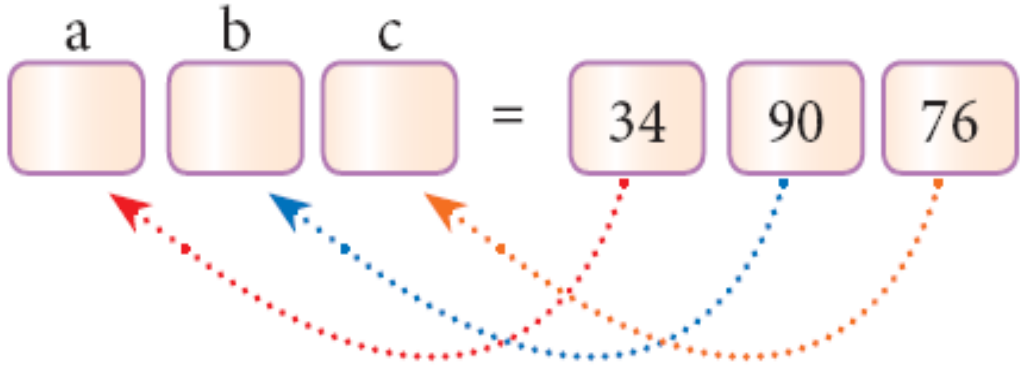
Output

```
('apple', 'banana', 'orange', 10, 20, 30)
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
/tmp/ipython-input-2847572916.py in <cell line: 0>()
      4 print(tup)
      5 del(tup)
----> 6 print(tup)

NameError: name 'tup' is not defined
```

TUPLE ASSIGNMENT

- Tuple assignment is a powerful feature in Python.
- It allows a tuple variable on the left of the assignment operator to be assigned to the values on the right side of the assignment
- operator.
- Each value is assigned to its respective variable.



Example

```
(a,b,c,d) = (10,20,30,40)
print("a=",a)
print("b=",b)
print("c=",c)
print("d=",d)
e=a+b+c+d
print("the value of adding a+b+c+d=",e)
```

Output

```
a= 10
b= 20
c= 30
d= 40
the value of adding a+b+c+d= 100
```

RETURNING MULTIPLE VALUES IN TUPLES

- **A function can return only one value at a time, but Python returns more than one value from a function.**
- **Python groups multiple values and returns them together.**

Example

```
def Min_Max(n):
  a = max(n)
  b = min(n)
  return(a, b)

Num = (12, 65, 84, 1, 18, 85, 99)
(Max_Num, Min_Num) = Min_Max(Num)
print("Maximum value = ", Max_Num)
print("Minimum value = ", Min_Num)
```

Output

```
Maximum value =  99
Minimum value =  1
```

NESTED TUPLES

- In Python, a tuple can be defined inside another tuple; called Nested tuple.
- In a nested tuple, each tuple is considered as an element.
- The for loop will be useful to access all the elements in a nested tuple.

Example

```
Mylist = ((3.4,.76,65,786),(4,6,7),(65,65,43,4))
a=len(Mylist)
print(Mylist)
print("the length of the previous tuple is:", a)

Mylist = ((3.4,.76,65,786),(4,6,7),(65,65,43,4))
item=0
for i in Mylist:
  item+=1
  print("element", item, "of the tuple is:", i)
```

Output

```
((3.4, 0.76, 65, 786), (4, 6, 7), (65, 65, 43, 4))
the length of the previous tuple is: 3
element 1 of the tuple is: (3.4, 0.76, 65, 786)
element 2 of the tuple is: (4, 6, 7)
element 3 of the tuple is: (65, 65, 43, 4)
```

INTRODUCTION TO SET

- **A set is another type of collection data type.**
- **A Set is a mutable and an unordered collection of elements without duplicates.**
- **That means the elements within a set cannot be repeated.**
- **This feature used to include membership testing and eliminating duplicate elements.**

CREATING A SET

- **A set is created by placing all the elements separated by comma within a pair of curly brackets{ }.**

- **The set( ) function can also used to create sets in Python.**

syntax

```
Set_Variable = {E1, E2, E3 ..... En}
```

Example

```
s1={1,2,2,3,4,5,5,6}
print(s1)
s2=set([1, 'Ali', 2, 'Ali'])
print(s2)
```

Output

```
{1, 2, 3, 4, 5, 6}
{1, 2, 'Ali'}
```

CREATING SET USING LIST OR TUPLE

- **A list or Tuple can be converted as set by using**
- **set( ) function.**

- **First you have to create a list or Tuple then, substitute its variable within set( ) function as argument.**

Example

```
L=[1,2,3]
print("type of L is:",type(L))
T=(1,2,3)
print("type of T is:",type(T))
s1=set(L)
print("type of s1 is:",type(s1))
s2=set(T)
print("type of s2 is:",type(s2))
```

Output

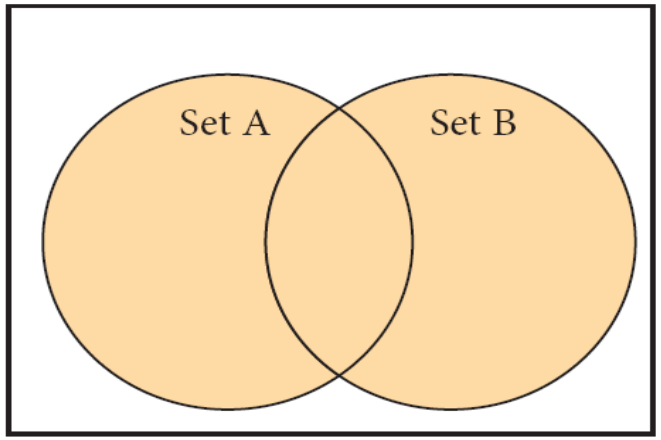
```
type of L is: <class 'list'>
type of T is: <class 'tuple'>
type of s1 is: <class 'set'>
type of s2 is: <class 'set'>
```

SET OPERATIONS

- The python is also supports the set operations such as
  - | union
  - & intersection
  - – difference
  - ^ symmetric difference

union

- **It includes all elements from two or more sets**
- **In python, the operator | is used to union of two sets.**
- **The function union( ) is also used to join two sets in python.**
-



Example

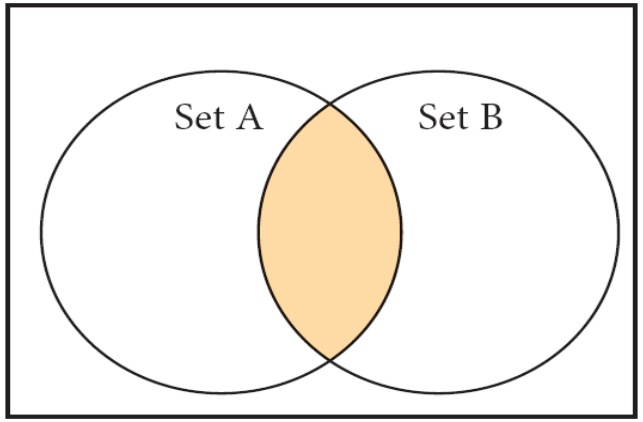
```
A = {0, 2, 4, 6, 8};
B = {1, 2, 3, 4, 5};
print("Union using the symbol A | B:", A | B)
print("Union using the method A.union(B):", A.union(B))
```

Output

```
Union using the symbol A | B: {0, 1, 2, 3, 4, 5, 6, 8}
Union using the method A.union(B): {0, 1, 2, 3, 4, 5, 6, 8}
```

INTERSECTION

- It includes the common elements in two sets
- The operator **&** is used to intersect two sets in python
- The function **intersection()** is also used to intersect two sets in python



Example

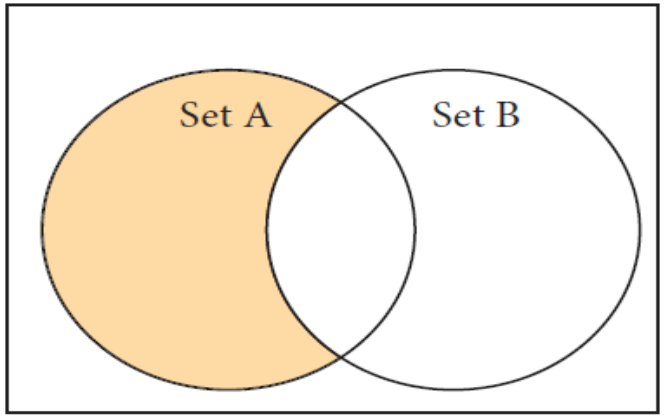
```
A = {0, 2, 4, 6, 8};
B = {1, 2, 3, 4, 5};
inter_1=A & B
inter_2=A.intersection(B)
print("Intersection using the symbol A & B:",inter_1)
print("Intersection using the method A.intersection(B):", inter_2)
```

Output

```
Intersection using the symbol A & B: {2, 4}
Intersection using the method A.intersection(B): {2, 4}
```

Difference

- **It includes all elements that are in first set (say set A) but not in the second set (say set B)**
- **The minus (-) operator is used to difference set operation in python.**
- **The function difference() is also used to intersect two sets in python**



Example

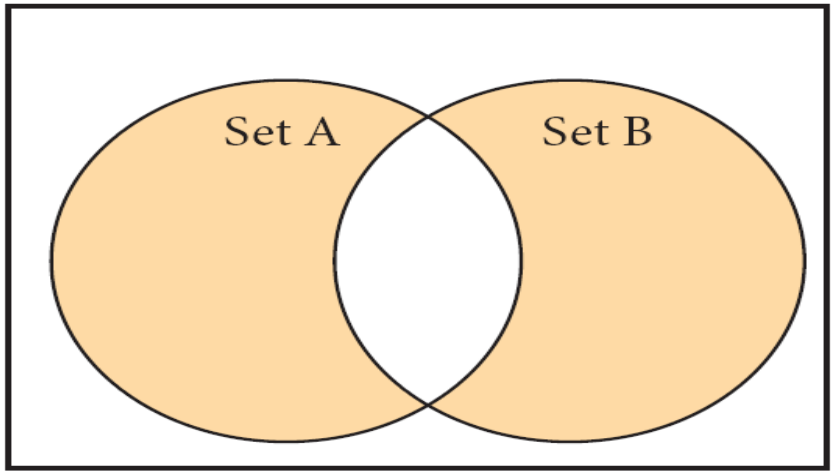
```
A = {0, 2, 4, 6, 8};
B = {1, 2, 3, 4, 5};
diff_1=A - B
diff_2=A.difference(B)
print("The difference using the symbol A - B:",diff_1)
print("The difference using the method A.difference(B):", diff_2)
```

Output

```
The difference using the symbol A - B: {0, 8, 6}
The difference using the method A.difference(B): {0, 8, 6}
```

Symmetric difference

- **It includes all the elements that are in two sets (say sets A and B) but not the one that are common to two sets.**
- **The caret (^) operator is used to symmetric difference set operation in python. The function symmetric_difference() is also used to do the same operation.**



Example

```
A = {0, 2, 4, 6, 8};
B = {1, 2, 3, 4, 5};
print("The symmetric difference using the symbol A - B:",A ^ B)
print("The symmetric difference using the method A.difference(B):", A.symmetric_difference(B))
```

Output

```
The symmetric difference using the symbol A - B: {0, 1, 3, 5, 6, 8}
The symmetric difference using the method A.difference(B): {0, 1, 3, 5, 6, 8}
```