TUPLE

Tuple in Python

Tuple is a collection of Python objects much like a list.

- Ordered
- Immutable
- Allow Duplicates

Tuple Creation : A tuple is created by placing all the items (elements) inside parentheses (), separated by commas. A tuple can have any number of items and they may be of different types (integer, float, list, string, etc.).

```
# tuple with mixed datatypes
tup1 = (1, "Hello", 3.4)

# nested tuple
tup2 = ("hello", [8, 4, 6], (1, 2, 3))
```

Accessing and Updating Tuple

Access:

- Indexing and Negative Indexing:

```
fruit_tup = ("apple", "banana", "cherry")
print(fruit_tup[1])
print(fruit_tup[-1])
```

- Slicing

```
print(fruit_tup[1:3])
```

Update and Delete Tuple:

- Tuple is immutable so can’t update or delete.
- However we can delete entire tuple.

E.g: del fruit_tup

Unpack Tuple

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

```
fruit_tup = ("apple", "banana", "cherry") #Packing a Tuple
```

Unpacking a tuple:

```
fruit_tup = ("apple", "banana", "cherry")
(green, yellow, red) = fruit_tup
```

Use of *:

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

```
fruits = ("apple", "banana", "cherry", "strawberry", "raspberry")
(green, yellow, *red) = fruits
print(red)
```

Tuple Methods

```
len() : len(fruit_tup)
my_tuple = ('a', 'p', 'p', 'l', 'e',)
print(my_tuple.count('p'))  # Output: 2
print(my_tuple.index('l'))  # Output: 3
```

Advantages of Tuple over List:

- Since tuples are immutable, iterating through a tuple is faster than with list. So there is a slight performance boost.
- Tuples that contain immutable elements can be used as a key for a dictionary. With lists, this is not possible.
- If you have data that doesn't change, implementing it as tuple will guarantee that it remains write-protected.

SET

Set in Python

In Python, Set is an **unordered** collection of data type that is **iterable**, **mutable** and has **no duplicate** elements.

Set Creation: A set is created by placing all the items (elements) inside curly braces {}, separated by comma,

```
num_set = {1, 2, 3}
```

Accessing Sets:

Loop:

```
for x in num_set:
  print(x)
```

- We can’t access or change an element of a set using indexing or slicing

Modifying Set Items

add():

```
num_set.add(5)
{1, 2, 3, 5}
```

update():

```
num_set.update([5,6,7])
{1, 2, 3, 5, 6, 7}
```

discard():

```
num_set.discard(7)
{1, 2, 3, 5, 6}
```

remove():

```
num_set.remove(6)
num_set
{1, 2, 3, 5}
num_set.remove(6)
*Error
```

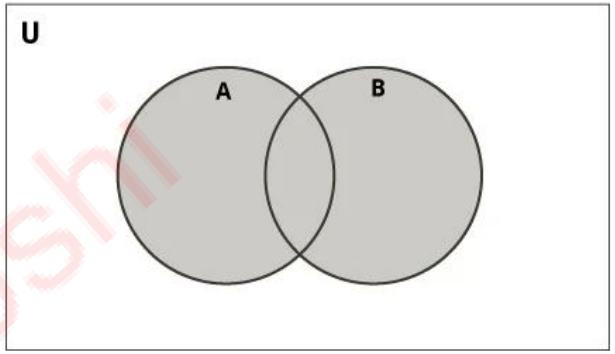
The only difference between the two is that the discard() function leaves a set unchanged if the element is not present in the set. On the other hand, the remove() function will raise an error in such a when if element is not present in the set.

Set Operations

Union (|):

All elements from both the sets.

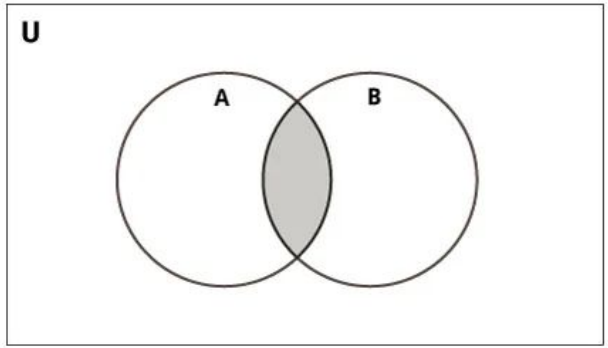
```
a={1,2,3,4,5}
b={4,5,6,7,8}
a|b
{1, 2, 3, 4, 5, 6, 7, 8}
a.union(b)
{1, 2, 3, 4, 5, 6, 7, 8}
b.union(a)
{1, 2, 3, 4, 5, 6, 7, 8}
```



Intersection (&):

Common elements from both the sets.

```
a&b
{4, 5}
b&a
{4, 5}
a.intersection(b)
{4, 5}
```

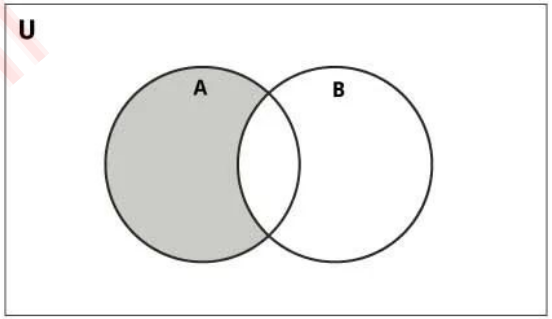


Set Operations (Cont…)

Set Difference (-):

Difference of the set B from set A(A - B) is a set of elements that are only in A but not in B. Similarly, B - A is a set of elements in B but not in A.

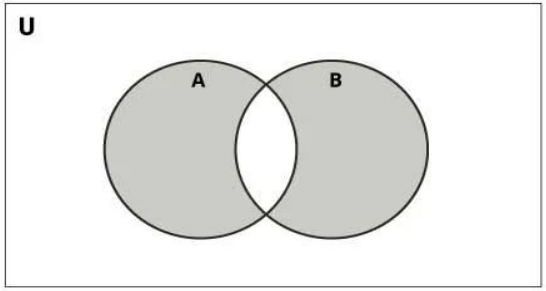
```
a-b
{1, 2, 3}
b-a
{8, 6, 7}
a.difference(b)
{1, 2, 3}
b.difference(a)
{8, 6, 7}
```



Symmetric Difference (^):

Symmetric Difference of A and B is a set of elements in A and B but not in both (excluding the intersection).

```
a^b
{1, 2, 3, 6, 7, 8}
a.symmetric_difference(b)
{1, 2, 3, 6, 7, 8}
```



DICTIONARY

Dictionary in Python

Python dictionary is an unordered collection of items. Each item of a dictionary has a key/value pair.

- Unordered
- Mutable
- Don’t Allow Duplicates

Dictionary Creation: Creating a dictionary is as simple as placing items inside curly braces {} separated by commas.
An item has a key and a corresponding value that is expressed as a pair (key: value).
E.g:

```
std={'id':101,'name':'Arjun','subs':['Python','Java','BigData']}
```

Accessing Dictionary

- Value can be accessed by using **dict_name[key]** or by using **get()**

```
std['id']                   #Op: 101
std.get('subs')             #Op: ['python', 'java', 'BigData']
print(std['subs'][2])       #Op: BigData
```

Getting Keys:

```
std.keys()
dict_keys(['id', 'name', 'subs']) #Return List of all keys
```

Getting Values:

```
std.values()
dict_values([101, 'Arjun', ['python', 'java', 'BigData']]) #Return List off all values
```

Getting Items:

```
std.items()    #Return all (key,value) as tuple in a List
dict_items([('id', 101), ('name', 'Arjun'), ('subs', ['python', 'java', 'BigData'])])
```

Looping Dictionary

Get Keys:

```
for k in std:
  print(k)

for k in std.keys():
    print(k)
```

Get Values:

```
for k in std:
    print(std[k])

for val in std.values():
    print(val)
```

Get Key,Values

```
for k,v in std.items():
    print(k,v)
```

Modify Dictionary Items

If the key is already present, then the existing value gets updated. In case the key is not present, a new (key: value) pair is added to the dictionary.

```
std_dict={'id':101,'name':'Arjun','age':25,'address':'Bangalore'}
std_dict['age']=26
print(std_dict)         #Op: {'id': 101, 'name': 'Arjun', 'age': 26, 'address': 'Bangalore'}
std_dict['tech']='python'
print(std_dict)   #op: {'id': 101, 'name': 'Arjun', 'age': 26, 'address': 'Bangalore', 'tech': 'python'}

std_dict.update({'birth_year':'1990'})
#op: {'id': 101, 'name': 'Arjun', 'age': 26, 'address': 'Bangalore', 'tech': 'python', 'birth_year': '1990'}
```

pop() : It removes an item with the provided key and returns the value.

```
std_dict.pop('tech')
print(std_dict)   #{'id': 101, 'name': 'Arjun', 'age': 26, 'address': 'Bangalore', 'birth_year': '1990'}
```

popitem(): It remove and return an arbitrary (key, value) item pair

```
std_dict.popitem()
```

Hands-On 3

1. Develop a menu driven calculator which will take input from user and perform basic calculations(+,-,*,/,%) using dictionaries.
2. WAP to print sum of all dictionary items.
3. WAP to calculate total number of key,value pair present in a dictionary.
4. WAP to take input from user and swap those numbers.
5. WAP which can register new user and can login existing user using dictionary.
6. WAP to find total number of integer type key in a dictionary.
7. Create two sets and check if the 2nd set is subset of 1st set and if it is subset find the difference of both set.
8. Create a tuple and if item of tuple are repeated make the item as key and count and value and create a dictionary with that kay, value.
   Sample Tuple: (88,1,1,1,5,5)
   Dictionary Op: {1: 3, 5: 2}