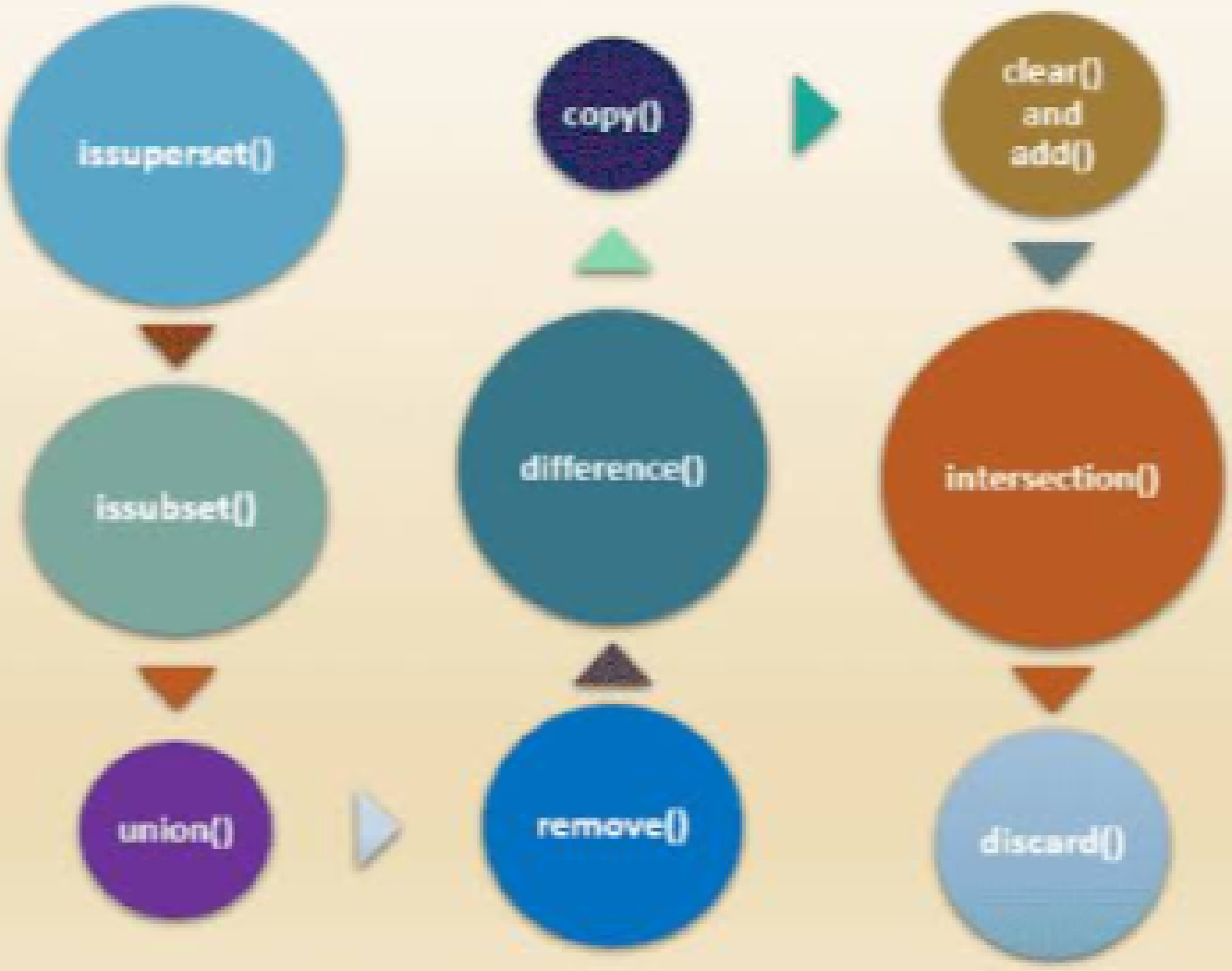
Python

Python
Sets


issuperset()
copy()
clear() and add()
issubset()
difference()
intersection()
union()
remove()
discard()

Python Set

A Python set is the collection of the unordered items. Each element in the set must be unique, immutable, and the sets remove the duplicate elements. Sets are mutable which means we can modify it after its creation.

Unlike other collections in Python, there is no index attached to the elements of the set, i.e., we cannot directly access any element of the set by the index. However, we can print them all together, or we can get the list of elements by looping through the set.

Creating a set

The set can be created by enclosing the comma-separated immutable items with the curly braces {}. Python also provides the set() method, which can be used to create the set by the passed sequence.

Example 1: Using curly braces

```
Days = {"Monday", "Tuesday", "Wednesday", "Thursday", "Fr
print(Days)
print(type(Days))
print("looping through the set elements ... ")
for i in Days:
    print(i)
```

Output:

```
{'Friday', 'Tuesday', 'Monday', 'Saturday', 'Thursday' 'Sunday', 'Wednesday'}
<class 'set'>
looping through the set elements ...
Friday
Tuesday
Monday
Saturday
Thursday
Sunday
Wednesday
```

It can contain any type of element such as integer, float, tuple etc. But mutable elements (list, dictionary, set) can't be a member of set. Consider the following example.

```
# Creating a set which have immutable elements
set1 = {1,2,3, "JavaTpoint", 20.5, 14}
print(type(set1))
#Creating a set which have mutable element
set2 = {1,2,3,["Javatpoint",4]}
print(type(set2))
```

Output:

```
<class 'set'>

Traceback (most recent call last)
<ipython-input-5-9605bb6fbc68> in <module>
      4
      5 #Creating a set which holds mutable elements
----> 6 set2 = {1,2,3,["Javatpoint",4]}
      7 print(type(set2))

TypeError: unhashable type: 'list'
```

In the above code, we have created two sets, the set **set1** have immutable elements and set2 have one mutable element as a list. While checking the type of set2, it raised an error, which means set can contain only immutable elements.

Creating an empty set is a bit different because empty curly {} braces are also used to create a dictionary as well. So Python provides the set() method used without an argument to create an empty set.

```
# Empty curly braces will create dictionary
set3 = {}
print(type(set3))

# Empty set using set() function
set4 = set()
print(type(set4))
```

Output:

```
<class 'dict'>
<class 'set'>
```

Let's see what happened if we provide the duplicate element to the set.

```
set5 = {1,2,4,4,5,8,9,9,10}
print("Return set with unique elements:",set5)
```

Output:

```
Return set with unique elements: {1, 2, 4, 5, 8, 9, 10}
```

Adding items to the set

Python provides the **add()** method and **update()** method which can be used to add some particular item to the set. The add() method is used to add a single element whereas the update() method is used to add multiple elements to the set. Consider the following example.

Example: 1 - Using add() method

```
Months = set(["January","February", "March", "April", "May",
print("\nprinting the original set ... ")
print(months)
print("\nAdding other months to the set...");
Months.add("July");
Months.add ("August");
print("\nPrinting the modified set...");
print(Months)
print("\nlooping through the set elements ... ")
for i in Months:
    print(i)
```

Output :

```
printing the original set ...
{'February', 'May', 'April', 'March', 'June', 'January'}

Adding other months to the set...

Printing the modified set...
{'February', 'July', 'May', 'April', 'March', 'August', 'June', 'January'}

looping through the set elements ...
February
July
May
April
March
August
June
January
```

To add more than one item in the set, Python provides the **update()** method. It accepts iterable as an argument.

Consider the following example.

Example - 2 Using update() function

```
Months = set(["January","February", "March", "April", "May", "
print("\nprinting the original set ... ")
print(Months)
print("\nupdating the original set ... ")
Months.update(["July","August","September","October"]);
print("\nprinting the modified set ... ")
print(Months);
```

Output:

```
printing the original set ...
{'January', 'February', 'April', 'May', 'June', 'March'}

updating the original set ...
printing the modified set ...
{'January', 'February', 'April', 'August', 'October', 'May', 'June', 'July', 'September', 'March'}
```

Removing items from the set

Python provides the **discard()** method and **remove()** method which can be used to remove the items from the set. The difference between these function, using discard() function if the item does not exist in the set then the set remain unchanged whereas remove() method will through an error.

Example-1 Using discard() method

```
months = set(["January","February", "March", "April", "May", "
print("\nprinting the original set ... ")
print(months)
print("\nRemoving some months from the set...");
months.discard("January");
months.discard("May");
print("\nPrinting the modified set...");
print(months)
print("\nlooping through the set elements ... ")
for i in months:
    print(i)
```

```
printing the original set ...
{'February', 'January', 'March', 'April', 'June', 'May'}

Removing some months from the set...

Printing the modified set...
{'February', 'March', 'April', 'June'}

looping through the set elements ...
February
March
April
June
```

Python provides also the **remove()** method to remove the item from the set. Consider the following example to remove the items using **remove()** method.

Example-2 Using remove() function

```
months = set(["January","February", "March", "April", "May", "
print("\nprinting the original set ... ")
print(months)
print("\nRemoving some months from the set...");
months.remove("January");
months.remove("May");
print("\nPrinting the modified set...");
print(months)
```

Output:

```
printing the original set ...
{'February', 'June', 'April', 'May', 'January', 'March'}

Removing some months from the set...

Printing the modified set...
{'February', 'June', 'April', 'March'}
```

We can also use the pop() method to remove the item. Generally, the pop() method will always remove the last item but the set is unordered, we can't determine which element will be popped from set.

Consider the following example to remove the item from the set using pop() method.

```
Months = set(["January","February", "March", "April", "May", "
print("\nprinting the original set ... ")
print(Months)
print("\nRemoving some months from the set...");
Months.pop();
Months.pop();
print("\nPrinting the modified set...");
print(Months)
```

```
printing the original set ...
{'June', 'January', 'May', 'April', 'February', 'March'}

Removing some months from the set...

Printing the modified set...
{'May', 'April', 'February', 'March'}
```

In the above code, the last element of the **Month** set is **March** but the pop() method removed the **June and January** because the set is unordered and the pop() method could not determine the last element of the set.

Python provides the clear() method to remove all the items from the set.

Consider the following example.

```
Months = set(["January","February", "March", "April", "May", "
print("\nprinting the original set ... ")
print(Months)
print("\nRemoving all the items from the set...");
Months.clear()
print("\nPrinting the modified set...")
print(Months)
```

Output:

```
printing the original set ...
{'January', 'May', 'June', 'April', 'March', 'February'}

Removing all the items from the set...

Printing the modified set...
set()
```

Difference between discard() and remove()

Despite the fact that **discard()** and **remove()** method both perform the same task, There is one main difference between discard() and remove().

If the key to be deleted from the set using discard() doesn't exist in the set, the Python will not give the error. The program maintains its control flow.

On the other hand, if the item to be deleted from the set using remove() doesn't exist in the set, the Python will raise an error.

Example-

```
Months = set(["January","February", "March", "April", "May", "June"])
print("\nprinting the original set ... ")
print(Months)
print("\nRemoving items through discard() method...");
Months.discard("Feb"); #will not give an error although the key feb is not available in the set
print("\nprinting the modified set...")
print(Months)
print("\nRemoving items through remove() method...");
Months.remove("Jan") #will give an error as the key jan is not available in the set.
print("\nPrinting the modified set...")
print(Months)
```

Output:

```
printing the original set ...
{'March', 'January', 'April', 'June', 'February', 'May'}

Removing items through discard() method...

printing the modified set...
{'March', 'January', 'April', 'June', 'February', 'May'}

Removing items through remove() method...
Traceback (most recent call last):
  File "set.py", line 9, in
    Months.remove("Jan")
KeyError: 'Jan'
```