Python Practice:

**Python hash() function**is a built-in function and returns the hash value of an object if it has one. The hash value is an integer which is used to quickly compare dictionary keys while looking at a dictionary.

# initializing objects

int\_val **=** 4

str\_val **=** 'GeeksforGeeks'

flt\_val **=** 24.56

# Printing the hash values.

# Notice Integer value doesn't change

# You'l have answer later in article.

**print**("The integer hash value is : " **+** str(hash(int\_val)))

print("The string hash value is : " **+** str(hash(str\_val)))

print("The float hash value is : " **+** str(hash(flt\_val)))

The integer hash value is: 4

The string hash value is: -5570917502994512005

The float hash value is: 1291272085159665688

The hash values are integers an used to compare dictionary keys during a dictionary lookup. We can hash only these types:

Hashable types: \* bool \* int \* long **\* float \* string** \* Unicode \* **tuple** \* code object

We cannot hash of these types:

Non-hashable types: \* bytearray **\* list \* set \* dictionary** \* memoryview

**strip()** is an inbuilt function in Python programming language that returns a copy of the string with both leading and trailing characters removed (based on the string argument passed).

string **=** """    geeks for geeks    """

# prints the string without stripping

print(string)

# prints the string by removing leading and trailing whitespaces

**print**(string.strip())

geeks for geeks

geeks for geeks

# Python3 program for union() function

set1 **=** {2, 4, 5, 6}

set2 **=** {4, 6, 7, 8}

set3 **=** {7, 8, 9, 10}

# union of two sets

print("set1 U set2 : ", set1.union(set2))

# union of three sets

print("set1 U set2 U set3 :", set1.union(set2, set3))

set1 U set2 : {2, 4, 5, 6, 7, 8}

set1 U set2 U set3 : {2, 4, 5, 6, 7, 8, 9, 10}

**power [ pow(a,b,m) = a^b mod m ]**

pow(3,4,5)

= (3\*\*4)%5

= 1

Input : x = 8, y = 3

Output :(2, 2)

**OrderedDict:**

An OrderedDict is a dictionary that remembers the order of the keys that were inserted first. If a new entry overwrites an existing entry, the original insertion position is left unchanged.

**Divmod**

The divmod() method in python takes two numbers and returns a pair of numbers consisting of their quotient and remainder.

**from** collections **import** OrderedDict

**print**("\nThis is an Ordered Dict:\n")

od **=** OrderedDict()

od['a'] **=** 1

od['b'] **=** 2

od['c'] **=** 3

od['d'] **=** 4

**for** key, value **in** od.items():

    print(key, value)

An **OrderedDict** is a dictionary subclass that remembers the order that keys were first inserted. The only difference between [dict()](https://www.geeksforgeeks.org/python-set-4-dictionary-keywords-python/) and OrderedDict() is that:

OrderedDict **preserves the order** in which the keys are inserted. A regular dict doesn’t track the insertion order and iterating it gives the values in an arbitrary order. By contrast, the order the items are inserted is remembered by OrderedDict.

**pop() in set**

Remove a random item from the set:

fruits = {"apple", "banana", "cherry"}  
  
fruits.pop()  
  
print(fruits)

**.remove(x)**

This operation removes element  from the set.  
If element  does not exist, it raises a KeyError.

**.discard(x)**

This operation also removes element  from the set.  
If element  does not exist, it **does not** raise a KeyError.

Operator Overloading

Operator Overloading means giving extended meaning beyond their predefined operational meaning. For example operator + is used to add two integers as well as join two strings and merge two lists. It is achievable because ‘+’ operator is overloaded by int class and str class.

Lexicographic Order

What is lexicographic order example?

Lexicographical order is nothing but the dictionary order or preferably the order in which words appear in the dictonary. For example, **let's take three strings, "short", "shorthand" and "small"**. In the dictionary, "short" comes before "shorthand" and "shorthand" comes before "small". This is lexicographical order.

**Sorted Function**

Sorted() sorts any sequence (list, tuple) and always returns a list with the elements in a sorted manner, without modifying the original sequence.

x **=** [2, 8, 1, 4, 6, 3, 7]

**print**("Sorted List returned :"),

print(sorted(x))

print("\nReverse sort :"),

print(sorted(x, reverse**=**True))

print("\nOriginal list not modified :"),

print(x)

**Output:**

Sorted List returned : [1, 2, 3, 4, 6, 7, 8]

Reverse sort : [8, 7, 6, 4, 3, 2, 1]

Original list not modified : [2, 8, 1, 4, 6, 3, 7]

**Join**

string*.join(*iterable*)*

myTuple = ("John", "Peter", "Vicky")  
  
x = "#".join(myTuple)  
  
print(x)

>>> H = set("Hacker")

>>> R = set("Rank")

>>> H.update(R)

>>> print H

set(['a', 'c', 'e', 'H', 'k', 'n', 'r', 'R'])

>>> H = set("Hacker")

>>> R = set("Rank")

>>> H.intersection\_update(R)

>>> print H

set(['a', 'k'])

>>> H = set("Hacker")

>>> R = set("Rank")

>>> H.difference\_update(R)

>>> print H

set(['c', 'e', 'H', 'r'])

John#Peter#Vicky

**Break Continue**

The break statement is used to terminate the loop or statement in which it is present. After that, the control will pass to the statements that are present after the break statement, if available.

Continue is also a loop control statement just like the break statement. continue statement is opposite to that of break statement, instead of terminating the loop, it forces to execute the next iteration of the loop.

# Use of break statement inside the loop

for val in "string":

if val == "i":

break

print(val)

print("The end")

**Output**

s

t

r

The end

# Program to show the use of continue statement inside loops

for val in "string":

if val == "i":

continue

print(val)

print("The end")

**Output**

s

t

r

n

g

The end# import counter class from collections module

**from** collections **import** Counter

# Creation of a Counter Class object using

# string as an iterable data container

x **=** Counter("geeksforgeeks")

# printing the elements of counter object

**for** i **in** x.elements():

    print ( i, end **=** " ")