|  |  |
| --- | --- |
| *#alike DICTIONARY pass all elements WITHOUT A KEY...................* py\_set={**"shit"**, **"bra"**, **"panti"**, **"bikini"**} print(py\_set)  print() *#using SET function, and passing elements like LIST ["E1","E2","E3", ...... "En"]* py\_set2=set([**"shit"**, **"bra"**, **"panti"**, **"bikini"**]) print(py\_set2)  **for** i **in** py\_set2:  print(i) | {'shit', 'bra', 'panti', 'bikini'}  {'shit', 'bra', 'panti', 'bikini'}  shit  bra  panti  bikini |
|  |  |
| py\_set2.add(**"DILDO"**) py\_set.add(**"DILDO"**) *#this will never gives op IN SAME ORDER* print(py\_set) print(py\_set2) | {'DILDO', 'panti', 'shit', 'bikini', 'bra'}  {'DILDO', 'panti', 'shit', 'bikini', 'bra'} |
|  |  |
| *#An empty set can not be formed useing (), instade it creates a DICTIONARY* empty\_set=set() print(empty\_set) | set() |
|  |  |
| *#though an empty set {} creates a DICTIONARY instade of set, #so .add method wont work here* empty\_set.add(**"a"**) print(empty\_set)  empty\_set2={} *# empty\_set2.add("a") # print(empty\_set2) #error* | {'a'} |
|  |  |
| *#TUPLE to SET* tup=(1,2,5,8,9,) set\_tup=set(tup) print(set\_tup) | {0, 2, 4, 6, 8, 10, 12, 14, 16, 18} |
|  |  |
| *# union and intersection operation* even\_no=set(range(0,30,2)) print(even\_no) print(len(even\_no)) print() tup2=(1,5,8,10,16,9) nos=set(tup2) print(nos) print(len(nos))  print(**"inter"**) *# intersection* inter=even\_no.intersection(tup2) print(inter) print(len(inter))  print(**"union"**) union=even\_no.union(tup2) print(union) print(len(union)) | {0, 2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28}  15  {1, 5, 8, 9, 10, 16}  6  inter  {8, 16, 10}  3  union  {0, 1, 2, 4, 5, 6, 8, 9, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28}  18 |
|  |  |
| print(even\_no & nos) *# intersection* | {8, 16, 10} |
|  |  |
| *#sorting a set coz it never gurenties the same value in same order # using the set values as before [even\_no] & [nos] # print(even\_no.sort()) -- wrong* print(sorted(even\_no)) print(sorted(nos)) *#becomes stable in case of order* | [0, 2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28]  [1, 5, 8, 9, 10, 16] |
|  |  |
| *#subtracting sets # even\_no - nos* print(sorted(even\_no.difference(nos))) print(sorted(even\_no-nos)) | [0, 2, 4, 6, 12, 14, 18, 20, 22, 24, 26, 28]  [0, 2, 4, 6, 12, 14, 18, 20, 22, 24, 26, 28] |
|  |  |
| *# nos-even\_no* print(sorted(nos.difference(even\_no))) print(sorted(nos-even\_no)) | [1, 5, 9]  [1, 5, 9] |
|  |  |
| *#difference\_update : deletes all the same things from even\_no with #respect to nos* print(sorted(even\_no)) print(sorted(nos)) *# returns None bcz it doesnt have a return type* print(even\_no.difference\_update(nos)) print(nos.difference\_update(even\_no)) even\_no.difference\_update(nos) print(even\_no) nos.difference\_update((even\_no)) print(nos) | [0, 2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28]  [1, 5, 8, 9, 10, 16]  None  None  {0, 2, 4, 6, 12, 14, 18, 20, 22, 24, 26, 28}  {1, 5, 8, 9, 10, 16} |
|  |  |
| *# symmetric\_difference() -> deletes element from list that are identical # #AND NON IDENTIALS ARE INSERTED in one shot* print(sorted(even\_no)) print(sorted(nos)) print(sorted(even\_no.symmetric\_difference(nos))) print(sorted(nos.symmetric\_difference(even\_no))) | [0, 2, 4, 6, 12, 14, 18, 20, 22, 24, 26, 28]  [1, 5, 8, 9, 10, 16]  [0, 1, 2, 4, 5, 6, 8, 9, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28]  [0, 1, 2, 4, 5, 6, 8, 9, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28] |
|  |  |
| *#remove and discard #remove throws an error if the element doesnt exist #discard doesnt give any error if the element doesnt exist* print(nos) nos.remove(1) nos.discard(5) print(nos) print() nos.discard(20) *#no error # nos.remove(20) #error* print(nos) | {1, 5, 8, 9, 10, 16}  {8, 9, 10, 16}  {8, 9, 10, 16} |
|  |  |
| **if** a.issubset(b):  print(**"a is a subset of b"**) **else**:  print(**"b is a subset of a"**)  **if** b.issuperset(a):  print(**"b is a superset of a"**) **else**:  print(**"a is a superset of b"**) | b is a subset of a  a is a superset of b |
|  |  |
| *#Frozen Set : immutable set : a set that cnt be changed # frozen key can be usedas a dictionary key #frozen set can also be added as a member of a set # no ADD REMOVE DISCARD is available in frozen set # UNION INTERSECTION SUBTRACTION ALL ARE ALLOWED IN FROZEN SET, LIKE OTHER*  P=frozenset(range(0,20,2)) print(P) | frozenset({0, 2, 4, 6, 8, 10, 12, 14, 16, 18}) |