SETs:

1. What is set?
2. Creation
3. Unordered
4. Mutable
5. What is Set?

L1=[1,2,3,4,5,6] 🡪 List

T1=(1,2,3,4,5,6) -> Tuple

S1={1,2,3,4,5,6} 🡪 Sets

* Unordered collection of Distinct Elements
* It’s Heterogenous
* Its Mutable

S2={1,2,2,3,3,4,5,6}

#Collection of distinct elements

print(S2)

#Disnct elements

#Output: {1, 2, 3, 4, 5, 6}

S3={1,2.5,"Jack",True,3+4j}

#heterogeneous elements

# Indexing and Slicing are not allowed in Sets

# S3[2]

#TypeError: 'set' object is not subscriptable

#s3[1:4]

#TypeError: 'set' object is not subscriptable

#Unordered

#Creation of sets

S1={1,2,3,4,5,6}

print(type(S1))

#<class 'set'>

print(S1)

#Output: {1, 2, 3, 4, 5, 6}

#S2=set(iterable)

S2=set([1,2,3,4,5,6])

print(type(S2))

# {<class 'set'> }

print(S2)

#Output: {1, 2, 3, 4, 5, 6}

S3=set('Python')

print(type(S3))

# <class 'set'>

print(S3)

#Output: {'h', 'P', 'n', 'o', 't', 'y'}

S4={5}

print(type(S4))

#<class 'set'>

print(S4)

#Output: {5}

S5=set()

print(type(S5))

#<class 'set'>

print(bool(S5))

#Output: False

print(S5)

#Output: set()

S6={}

print(type(S6))

#<class 'dict'>

########### Unordered

S1={10,20,30,40,50}

print(S1)

#Output: {40, 10, 50, 20, 30}

print("\n")

for i in S1:

    print(i,end=" ")

print("\n")

#40 10 50 20 30

#Mutable

S1={10,20,30,40,50}

S1.add(60)

print(S1)

#Output: {40, 10, 50, 20, 60, 30}

S1.add("Hi")

print(S1)

#Output: {40, 10, 50, 20, 'Hi', 60, 30}

S1.add((1,2,3))

print(S1)

#{40, 'Hi', 10, (1, 2, 3), 50, 20, 60, 30}

# S1.add([1,2,3])

# print(S1)

#list is mutable

# #TypeError: unhashable type: 'list'

#### REmove

S1={10,20,30,40,50}

S1.remove(30)

print(S1)

#Output: {40, 10, 50, 20}

print(S1.pop())

#Output: 40

# Dont know which element will be removed

#Mutable are not hashable

INTERNAL Working of sets:

Set in Mathematics:

1. Set
2. Subset
3. Superset
4. Proper subset
5. Proper Superset
6. Disjoint sets

S={1,2,3,4,5,6,8,9,10};

A={1,2,3,5,7}

B=={5,7,9,10}

C={1,2,3,4,5,6,7,8,9,10}

* No dups in sets

A subset of S

B is a subset of S

C is equal and subset of S

S is a superset of A

S is a superset of B

S is a superset of C

A is proper subset of S

B is proper subset of S

Only few values

C is not a proper subset of S

🡪C is a proper superset of S

A screenshot of a computer

AI-generated content may be incorrect.

D={1,2,3,4,5}

E={6,7,8,9,10}

No common elements are Disjoint sets

Disjoint subsets

Set Operations in Maths:

A={1,2,3,5,7}

B={5,7,9,10,11}

Venn Diagrams

A screenshot of a computer

AI-generated content may be incorrect.

Set Operations:

1. Union (iterable)
2. Intersection(iterable)
3. Intersection update(iterable)
4. Difference (iterable)
5. Difference\_update
6. Symmetric\_difference(iterable)
7. Symmetric\_difference\_update(iterable)

#Union

#Union

s1={1,2,3,5,7}

s2={5,7,9,10,11}

s3=s1.union(s2)

print(s3)

#Output: {1, 2, 3, 5, 7, 9, 10, 11}

#Intersection

s3=s1.intersection(s2)

print(s3)

#Output: {5, 7}

#intersection update

s1.intersection\_update(s2)

print(s1)

#Output: {5, 7}

#calling on S1 will be updated

#Difference

s1={1,2,3,5,7}

s2={5,7,9,10,11}

s3=s1.difference(s2)

print(s3)

#Output: {1, 2, 3}

#s1 will not be updated

#difference update

s1.difference\_update(s2)

print(s1)

#Output: {1, 2, 3}

#symmetric difference

s1={1,2,3,5,7}

s2={5,7,9,10,11}

s3=s1.symmetric\_difference(s2)

print(s3)

#Output: {1, 2, 3, 9, 10, 11}

#s1 will not be updated

#symmetric difference update

s1.symmetric\_difference\_update(s2)

print(s1)

#Output: {1, 2, 3, 9, 10, 11}

#update

s1={1,2,3,5,7}

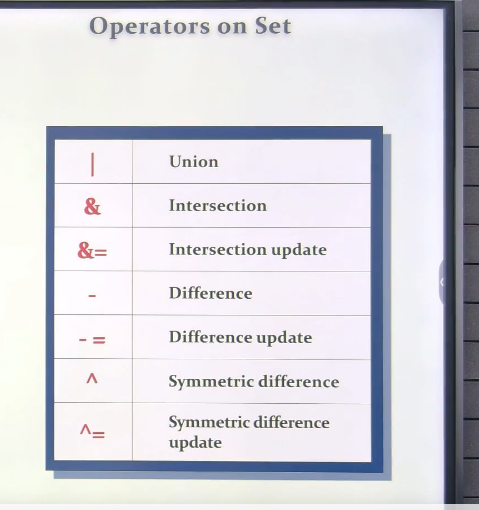
s2={5,7,9,10,11}

s1.update(s2)

print(s1)

#Output: {1, 2, 3, 5, 7, 9, 10, 11}

Operators on set:



#  |               --> Union

#  |=             --> Update

#  &               --> Intersection

#  &=              --> Intersection update

#  -              --> Difference

#  -=             --> Difference update

#  ^              --> Symmetric difference

#  ^=             --> Symmetric difference update

#union

s1={1,2,3,5,7}

s2={5,7,9,10,11}

s3=s1 | s2

print(s3)

#Output: {1, 2, 3, 5, 7, 9, 10, 11}

#Update

s1 |= s2

print(s1)

#Output: {1, 2, 3, 5, 7, 9, 10, 11}

#Intersection

s1={1,2,3,5,7}

s2={5,7,9,10,11}

s3=s1 & s2

print(s3)

#Output: {5, 7}

#Intersection update

s1 &= s2

print(s1)

#Output: {5, 7}

#difference

s1={1,2,3,5,7}

s2={5,7,9,10,11}

s3=s1 - s2

print(s3)

# Output: {1, 2, 3}

#difference update

s1 -= s2

print(s1)

#Output: {1, 2, 3}

#symmetric difference

s1={1,2,3,5,7}

s2={5,7,9,10,11}

s3=s1 ^ s2

print(s3)

#Output: {1, 2, 3, 9, 10, 11}

#symmetric difference update

s1 ^= s2

print(s1)

#Output: {1, 2, 3, 9, 10, 11}

Set methods:

# #add(element)

# #update(iterable)

# #copy()

# #pop() --> Revmove top most element

# #remove(element)

# #discard(element)

# #rclear()

#Add

s1={10,20,30,40,50}

print(len(s1))

#5

s1.add(60)

print(s1)

#{40, 10, 50, 20, 60, 30}

# s1.add(70,80)

# #TypeError: set.add() takes exactly one argument (2 given)

s1.add((70,80))

print(s1)

#{(70, 80), 40, 10, 50, 20, 60, 30}

#as tuple single element

#update(iterable)

s1={10,20,30,40,50}

s1.update((60,70))

print(s1)

#  {40, 10, 70, 50, 20, 60, 30}

s1={10,20,30,40,50}

s1.update('ruby')

print(s1)

#{'b', 40, 10, 'y', 50, 20, 'r', 30, 'u'}

#copy

s1={10,20,30,40,50}

s2=s1.copy()

print(id(s1),id(s2))

# 1967206105160 1967206063240

#pop

s1={10,20,30,40,50}

print(s1.pop())

#40

print(s1)

#{10, 50, 20, 30}

print(s1.pop())

#10

print(s1)

#{50, 20, 30}

#discard

#

#

s1={10,20,30,40,50}

print(s1.discard(30))

#None

print(s1)

#{40, 10, 50, 20}

print(s1.discard(100))

#None

#  #no Error

print(s1)

#{40, 10, 50, 20}

#remove

s1={10,20,30,40,50}

print(s1.remove(30))

#None

print(s1)

#{40, 10, 50, 20}

# print(s1.remove(100))

# #KeyError: 100

# s1.remove(10,20)

# #TypeError: set.remove() takes exactly one argument (2 given)

s1={10,(20,30),40,50}

s1.remove((20,30))

print(s1)

#{40, 10, 50}

#clear

s1={10,20,30,40,50}

s1.clear()

print(s1)

#set()

#del

s1={10,20,30,40,50}

del s1

print(s1)

#NameError: name 's1' is not defined

SET Comprehensions:

#s={iterable}

#s={exp for item in iterable if condition}

S1={x for x in range(1,5)}

print(S1)

#{1, 2, 3, 4}

print(type(S1))

#<class 'set'>

S2={x\*x for x in {-2,-1,0,1,2}}

print(S2)

#   {0, 1, 4}

print(type(S2))

#<class 'set'>

S3={x.lower() for x in "PHILIPins"}

print(S3)

#  {'h', 'i', 'p', 'l', 'a', 's', 'n'}

print(type(S3))

#<class 'set'>

#Set comprehension with no dups

#{}--> flower braces

#[]--> List

#[]-->Allow dups

Scrambled Words:

word\_set={"plea","medical","listen","leap","decimal","silent","pale","enlist"}

result=set()

print("scrambed word pairs are:")

for word1 in word\_set:

    for word2 in word\_set:

        if word1!=word2 and sorted(word1)==sorted(word2):

            pair=tuple(sorted((word1,word2)))

            result.add(pair)

for pair in result:

    print(pair)

Output:

scrambed word pairs are:

('decimal', 'medical')

('pale', 'plea')

('enlist', 'listen')

('enlist', 'silent')

('listen', 'silent')

('leap', 'pale')

('leap', 'plea')

<https://github.com/mohammedabdulbari/Python/blob/main/SetChallenges/Jaccard.py>

JACCARD:

'''

'''

import re

str1 = ('Time is the most valuable thing we have,'

' and once lost, it never returns.')

str2 = ("We never get time back once it's "

"gone—it's truly the most valuable resource.")

words1 = re.findall(r'\w+', str1.lower())

print(words1)

words2 = re.findall(r'\w+', str2.lower())

print(words2)

wset1 = set(words1)

# ['time', 'is', 'the', 'most', 'valuable', 'thing', 'we', 'have', 'and', 'once', 'lost', 'it', 'never', 'returns']

wset2 = set(words2)

# ['we', 'never', 'get', 'time', 'back', 'once', 'it', 's', 'gone', 'it', 's', 'truly', 'the', 'most', 'valuable', 'resource']

# |               --> Union

 # &               --> Intersection

common = wset1 & wset2

# {'most', 'valuable', 'have', 'thing', 'lost', 'never', 'is', 'once', 'returns', 'we', 'and', 'time', 'the', 'it'}

unique = wset1 | wset2

# {'most', 'valuable', 'have', 'thing', 'lost', 'never', 'is', 'once', 'returns', 'we', 'and', 'time', 'the', 'it'}

ratio = len(common) / len(unique)

print(f"Jaccard Similarity:{ratio:.2f}")

# Jaccard Similarity:0.40