

Agenda

23 December 2024 22:37

Recap

Data types

int, float, Boolean

String

List

Tuple

Dictionary

Recap

23 December 2024 23:22

- Python
- Python version release structure (alpha, beta, final)
 - dev
 - solving bugs
 - Release
- Google Colab (Gpu server free).
- Overview of datatypes
- Keyword & Identifiers

→ Data types

→ what is data type in python

→ Int, float, boolean, String

(

String:-

x = " data "

y = " dreams "

print(x + " - " + y)

{ " ".format }

x = input(" please enter your name : ")

y = input(" Please enter your age : ")

print(" my name is " + x + " and my age is " + y)

Normal Data types(int, float, Boolean)

23 December 2024 23:22



Int: It is one of the data type to store integer values (can be -ve & +ve)

Declaration syntax:

a=2 ✓
x=-34 ✓
y=56565 ✓
z=0 ✓

What operations we can perform?

a=2, b=3
a+b
a-b
a%b ✓
a/b ✓

int
(-∞ to +∞)
000
100 dig.
(IB)

Data type :-

→ { fruits }, { vegetables }, { clothes }

→ list, tuple, set, dict 99%

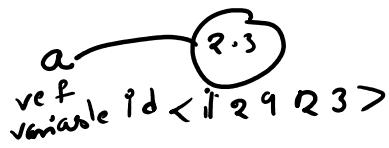
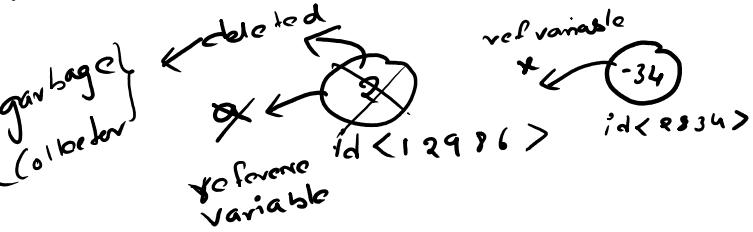
Float: It is another data type to store float values (can be -ve & +ve)

Declaration syntax:

a=2.3 ✓
x=-34.33
y=56565.45
z=0.0 ✓

What operations we can perform?

a=2.3, b=3.1
a+b
a-b
a%b
a/b



Boolean: It stores True and False

Declaration syntax:

_c = True
_f123=False

What operations we can perform?

X=True, Y=False
X and Y
X or Y
not X
not Y

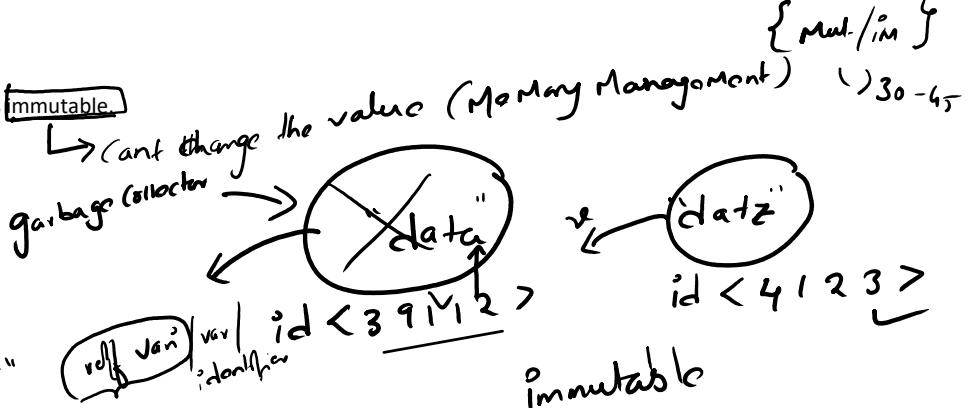
String

23 December 2024 23:12

-It is used to store sequence of characters and it is **immutable**.

Declaration syntax:

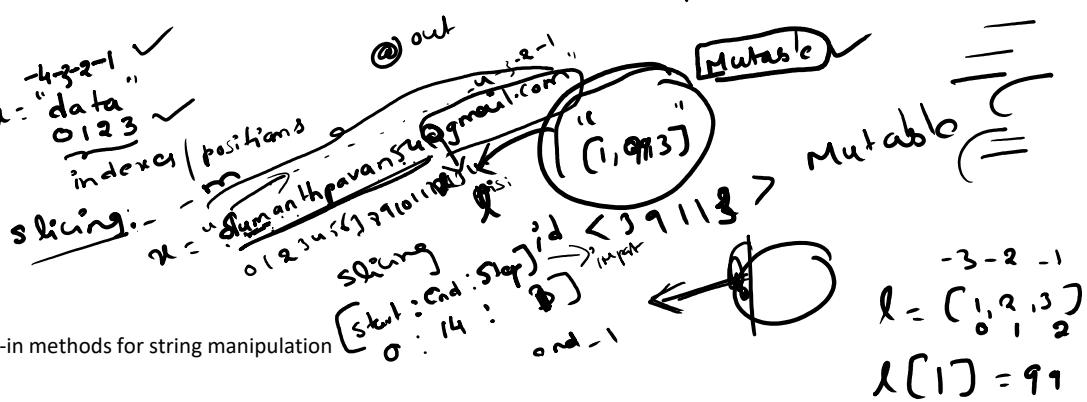
```
x='data'  
y="dreamers"  
z="I am good  
At python"  
r="""I am good  
At pyspark"""
```



What operations can we perform?

```
x='data'  
y="dreamers"
```

Concatenation
Indexing
Slicing
length



String Methods

Python provides a variety of built-in methods for string manipulation

Method	Description	Example
lower()	Converts all characters to lowercase.	"Python".lower() → "python"
upper()	Converts all characters to uppercase.	"Python".upper() → "PYTHON"
strip()	Removes leading and trailing spaces.	"Hello ".strip() → "Hello"
replace(old, new)	Replaces a substring with another.	"Python".replace("Py", "My") → "Mython"
split(delimiter)	Splits a string into a list.	"a,b,c".split(",") → ['a', 'b', 'c']
join(iterable)	Joins elements of an iterable into a string.	" ".join(['Hello', 'World']) → "Hello World"
startswith(sub)	Checks if the string starts with a substring.	"Python".startswith("Py") → True
endswith(sub)	Checks if the string ends with a substring.	"Python".endswith("on") → True

String Formatting

It helps you to display the string outputs in readable, maintainable, and adaptable to changing values.

f-strings

str.format() Method

List

23 December 2024 23:23

-It is a collection of items and it is mutable

Declaration syntax:

a=[1,2,3,4,2,3,True,"apple"]
x=[1,"apple",True,3.5,[2,3,4]]
y=[["apple","banana","apple"]]

z=list() → empty list

What operations we can perform:

Concatenation
Indexing
Slicing
length

l = [1, 2, 3, 4, 5, 6]
-6 -5 -4 -3 -2 -1
0 1 2 3 4 5
tive indexing
[start : end : step] → jump
end-1

List methods:

Method

append(item)

extend(iterable)

insert(index, item)

remove(item)

pop(index)

clear()

index(item)

count(item) ✓

Description

Adds an item to the end of the list.

Adds all the elements of an iterable (e.g., another list, tuple) to the end of the list.

Inserts an item at the specified index.

Removes the first occurrence of the specified item.

Removes and returns the item at the specified index. If no index is provided, removes the last item.

Removes all items from the list.

Returns the index of the first occurrence of the specified item.

Returns the number of times the specified item appears in the list.

sort(key=None, reverse=False) ✓ Sorts the list in ascending order by default. Accepts optional key and reverse parameters.

reverse() ✓ Reverses the order of the list in place.

copy()

Returns a shallow copy of the list.

Function:

Sorted(iterable, key=None, reverse=False) The sorted() function always returns a new list and does not modify the original

l = [1, 2, 3, 4, 5, 6, 7]
0, 1, 2, 3, 4, 5, 6
1 2 3 4 5 6 7
8 9 10 11
↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓

Sub elements inside the list

Immutable print(l)
Mutability l = [1, 2, 3, 4, 5, 6, 7]
l.append(8)
print(l)
[1, 2, 3, 4, 5, 6, 7, 8]

Syntax:-

[start : end : step] → jump
end-1

l[1:6:2]
[2, 4, 6]

l[-22:-1]

l[: :-1], l[-15:-3:1]

l[22:-1:2]

Wool and clear

fun (vs) methods

(oops)

fun (vs) methods (oops)

① functions -

inbuilt & user-defined
 defn)
 min()
 max()
 mode()
 mean()

def (function):
 =
 =

$$y = 3 \text{ years}$$

$$R = 8\% \text{ p.a.}$$

$$P = 10000$$

$$S.I = \frac{P.R}{100}$$

$$= \frac{10000 \times 3 \times 8}{100}$$

$$(or)$$

$$= 10000 \times 3 \times 0.08$$

def Validation(amount):

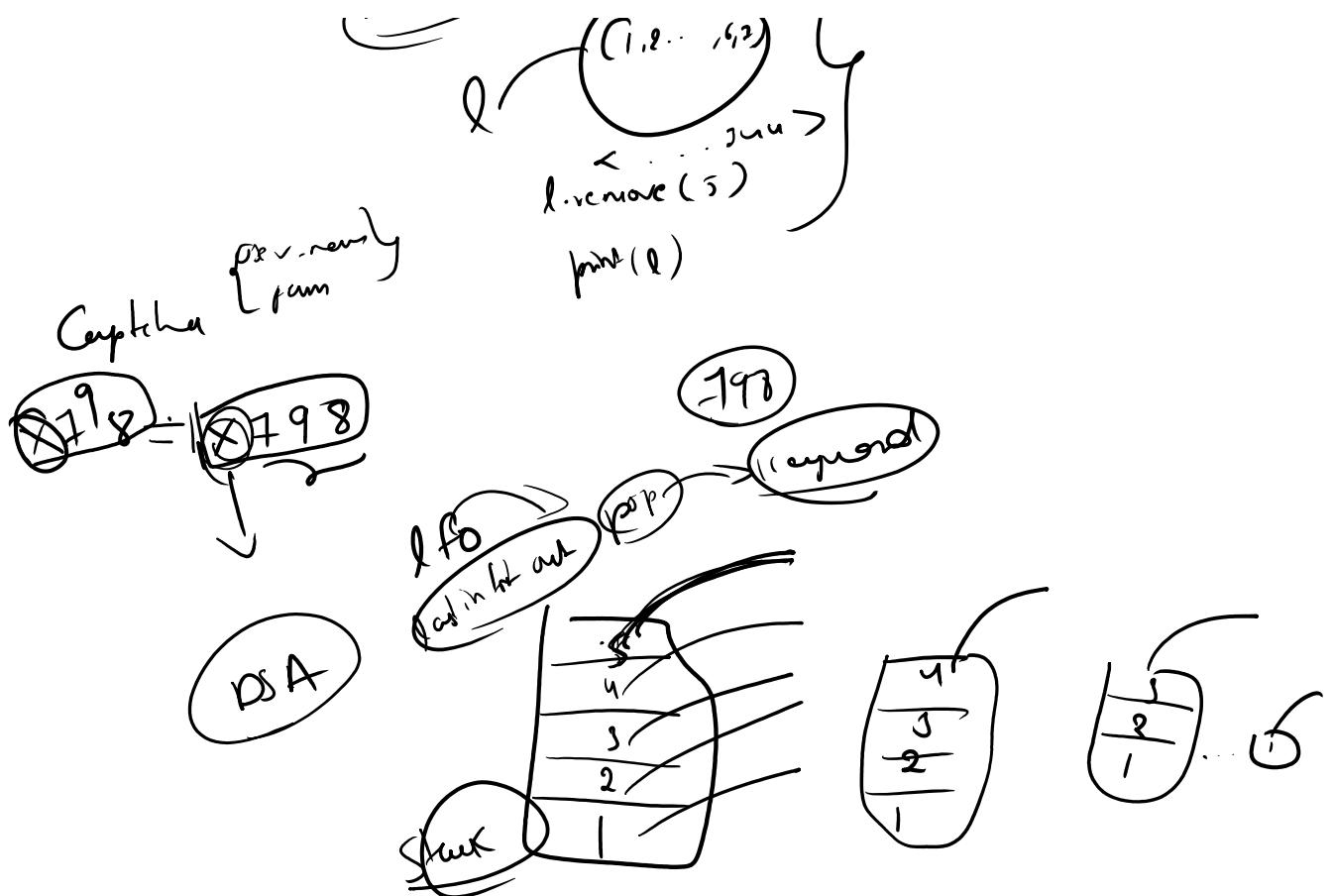
(amount)
 def SimpleInterest(P, T, R):
 $S.I = (P * R * T) / 100$
 return S.I

② method :-

class Temp:
 {
 def simple_interest(self):
 $S.I = (P * R * T) / 100$
 return S.I
 } → Method

Ex:-
 class List:
 def append():
 def extend():
 def insert():
 def remove():
 } → Methods

mutable
 (1, 2, ..., 10)



Method args/params → false
① sort(key, reverse) → Asc / Desc

३५

$l = [Smart, Savana, radio, west]$

Scammonia : -

`l.sort()` => Ascending

print(l)

(A, B, C, Washita, Studio)

Scenario 2 :- $\text{l}.\text{sort}(\text{reverse}:\text{True}) \Rightarrow \text{Descending}$

sort (reverse=True) -> reverse
[radio, weしだれ C . . O A]

Key arg/param :-

Ex:- Sort as per length of filenames }
Sort as per Capital letters }
 ||
 \>

$l.sort(\text{Key} = \text{long})$ reverse

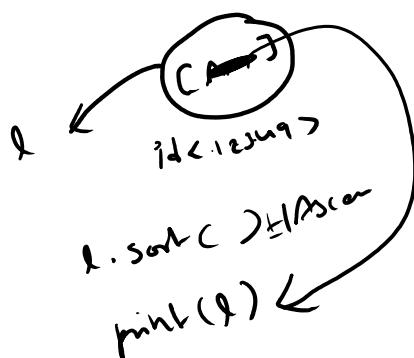
(len, Asc)

print(l)
(PA, Q, - , Pmark)

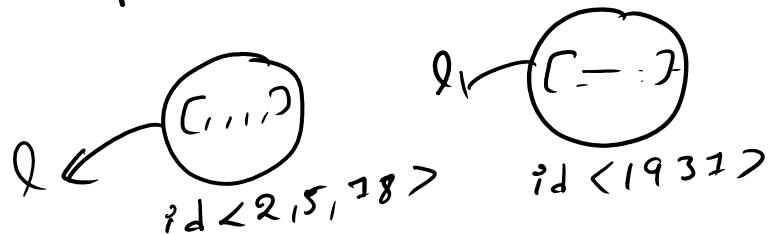
sorted (iterate) (key , reverse)

{ Any datatype
on which you
can apply looping
(can apply looping)

Diff sort, sorted
 $l = []$ ✓
 $l.sort()$



$l = [, ,]$ →
sorted(l)

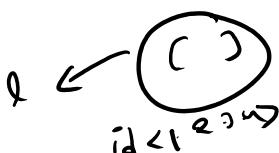


$l_1 = \text{sorted}(l)$

shallow

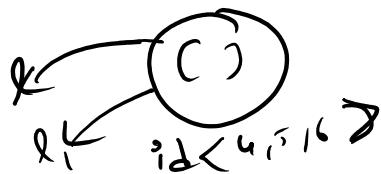
copy ✓
 $l = []$ ✓
 $l_1 = l.\text{copy}()$ ✓

Deep
(copy)



aliasing ✓

$l = []$
 $l_1 = l$



Tuple

23 December 2024 23:23

-It is a collection of items and it is **immutable**

Declaration syntax:

$x = (1, 2, 3, "apple", True)$ → heterogeneous
 $y = ("apple", "banana", "apple")$ → homogeneous
 $z = (1, 2, 3, "apple", True, (1, 2, 3, 4), [1, 2, 5, 6])$ → nested tuple
 $r = (1,)$ → single element tuple
 $t = tuple()$ → empty tuple → i) can accept duplicates

What operations we can perform:

Concatenation
Indexing
Slicing
length

Methods:

Method	Description
count(item)	Returns the number of times the specified item appears in the tuple.
index(item)	Returns the index of the first occurrence of an item. Raises ValueError if the item is not found.

Topics:

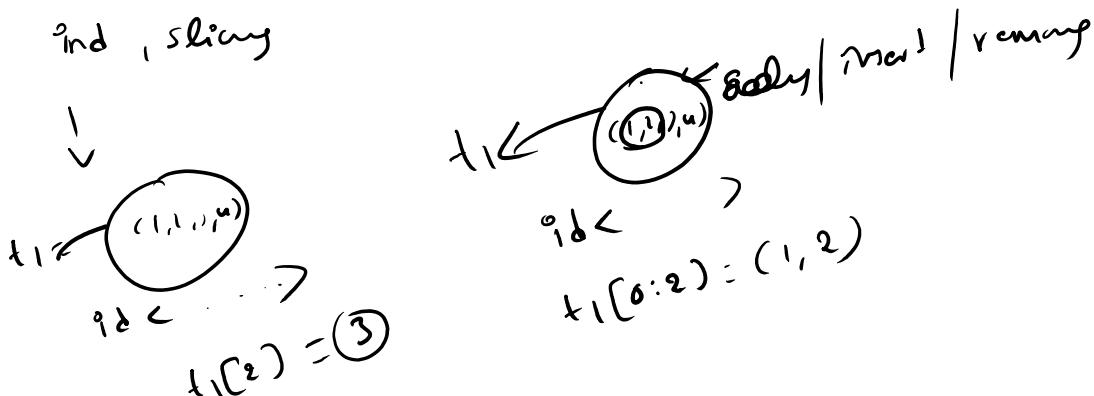
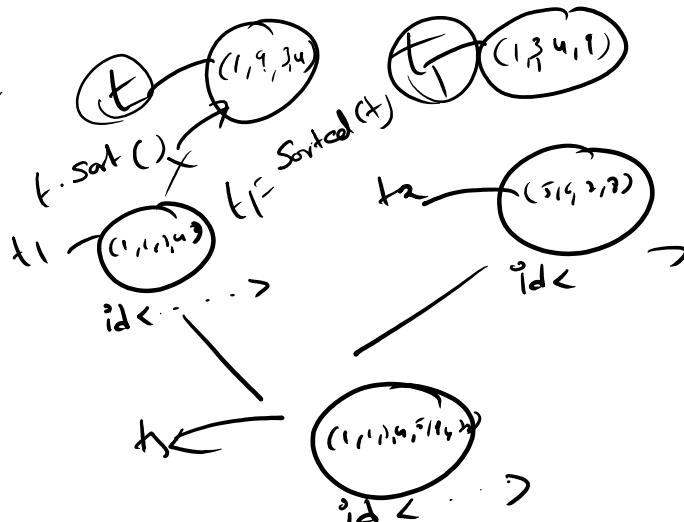
When to use tuples

Why sorted() function can be applied but not sort() ✓

Tuple unpacking

Ex:-

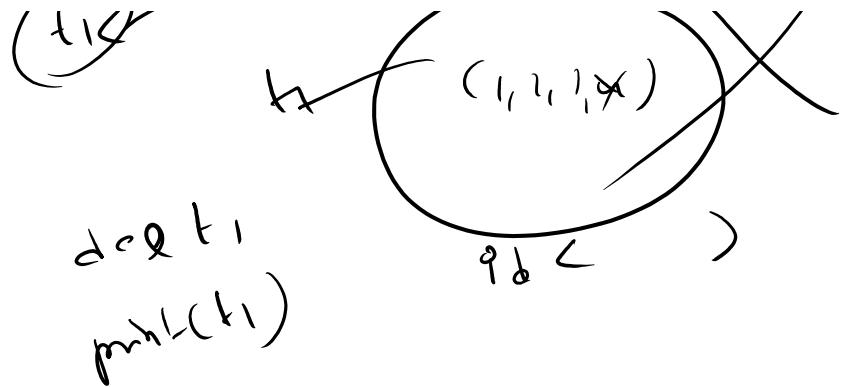
```
t1=(1,2,3,4) →  
t2=(5,6,7,8)  
t3 = t1+t2 ✓  
print(t3)  
print(id(t1))  
print(id(t2))  
print(id(t3))
```



```
t1=(1,2,3) →  
del t1  
print(t1)
```



```
t1=(1,2,3)  
del t1  
print(t1)
```



Mutability

$$l = \begin{bmatrix} -5 & -4 & -3 & -2 & -1 \\ 1, 2, 3, 4, 5 \\ 0 & 1 & 2 & 3 & 4 \end{bmatrix} \quad l \leftarrow \boxed{(1, e^{j\pi/5})}$$

$$l[2] = 300$$

print(1) %p [1,2,300,4,5]

Immutability

$$t = (1, 2, 3, 4, 5) \quad t \leftarrow (1, 2, 3, 4, 5)$$

$$t[2]=300$$

$p_{\text{int}}(t)$

'o/p:- Error

When to use tuple?

$$\text{Ex: 1) } t = (23.286, 21.796), \quad Q = [23.736, 21.796]$$

2) days-of-week = ("monday", "tuesday", ..., "sunday")

diff b/w list & tuple

- 1) list is mutable \rightarrow can change
- 2) tuple is immutable \rightarrow cannot "

$$\text{Ex: 2 } t = (1, 1, 1) \text{gu} \rightarrow t_1 = (1/2, 1/2) \text{gu}$$

$$t_1 = t \cdot \underline{\text{sort}()} \quad$$

$$L = (l_1, l_2, l_3, q_1, u)$$

$$k' = k$$

$(1, 2, 1, 9, 4)$

15 € /

$(1, e, s, 4, 9)$

$t = (1, 2, 1, 9, 4)$

$\text{L1-Sorted}(t)$



Miscellaneous Interview Questions

29 December 2024 22:47

What is Aliasing? ✓
Cloning & Copying ✓ (alternatives to aliasing, so that the existing memory remains same)

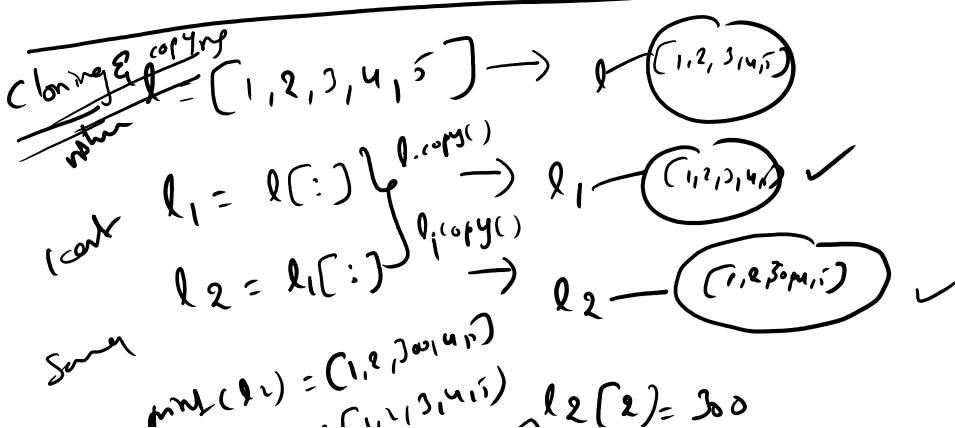
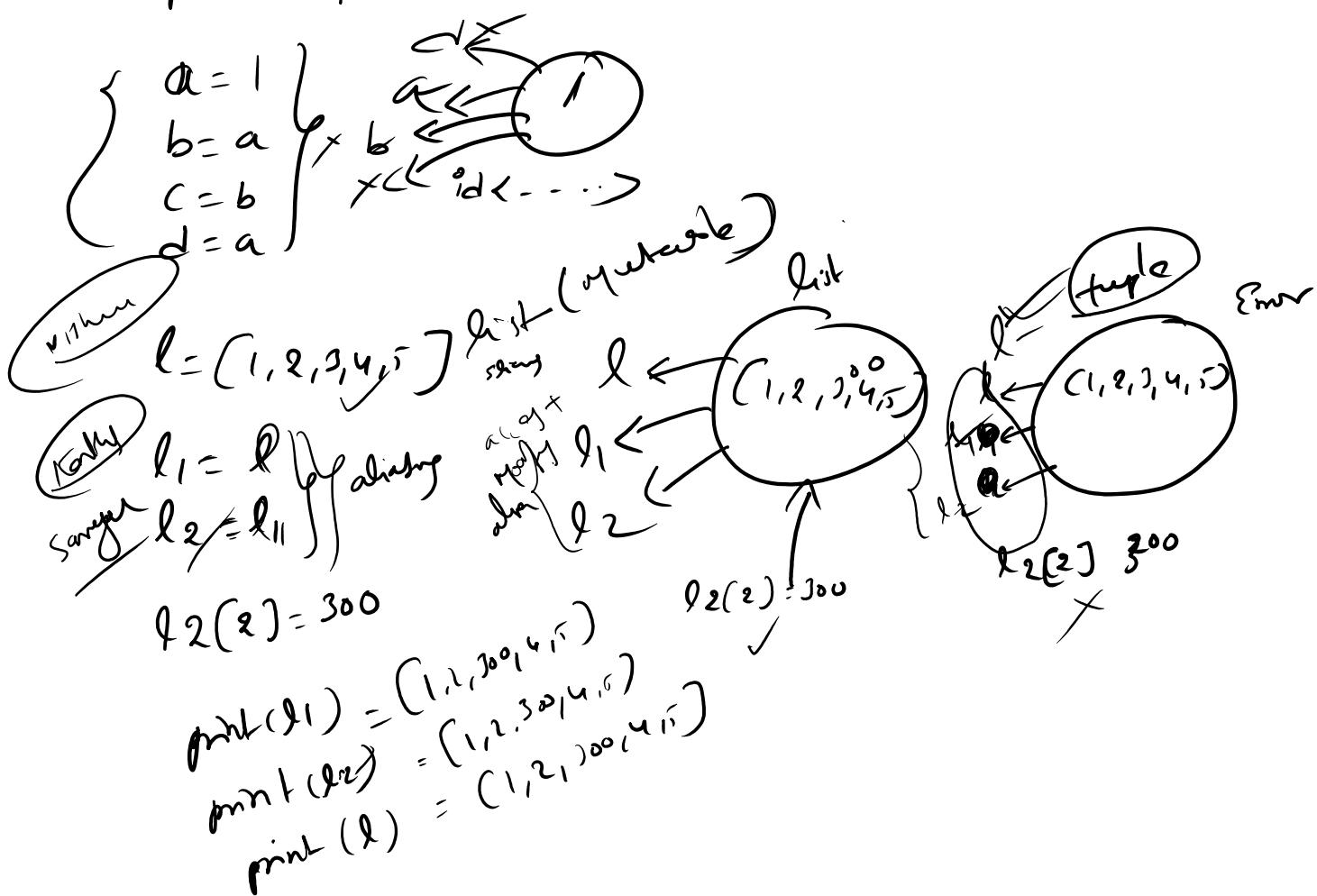
What is the role of Garbage Collector

How tuples and lists are stored? ✓
Why tuples takes less memory and time for execution? ✓

Difference between tuple and list ✓
What is mutability and how can we handle it (Cloning and Copying)
Shallow Copy vs Deep Copy

What is Aliasing?

→ pet name, original name, friend's name (same person)



$$\begin{aligned}
 \text{Sum} &= \text{prv}(t_2) = (1, e, 1)^{\omega_1 \cup \omega_2} \\
 \text{prv}(t_1) &= (1, e, 1)^{\omega_1 \cup \omega_1} \\
 \text{prv}(t) &= (1, e, 1)^{\omega_1 \cup \omega_1}
 \end{aligned}$$

shy in bo

(1,2,3)

(1,2)

(1,2,3)

(1,2)

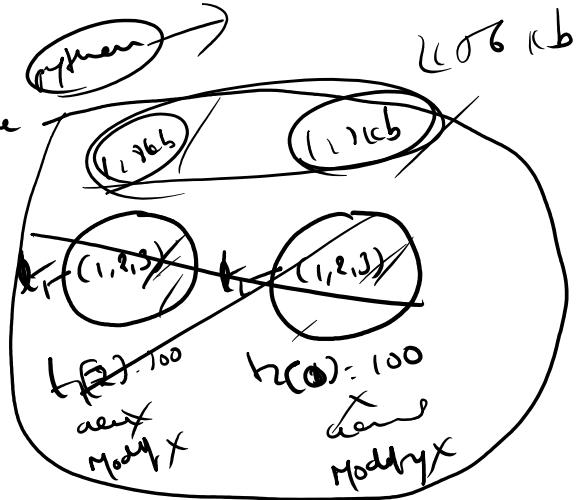
earm
natty

earm
Hoddy

(1,2,3)

(1,2)

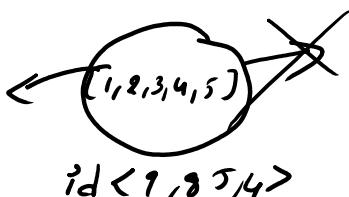
earm



What is role of garbage collector?

gew. berge Gletscher Riffen nach
einem eiszeitl. Zeitalter

$$l = [1, 2, 3, 4, 5] \longrightarrow$$



$$\ell = [6, 7, 19, 255]$$

พัฒนา

(6,7,19,255)

mom obj

Reference Counting :-

Referenten = 0

$$\begin{aligned} l &= [1, 2, 3, 4] \\ l_1 &= l \\ l_2 &= l_1 \\ l_1 &= \text{None} \end{aligned}$$

$$\ell_1 = \ell$$

8

$$12 =$$

$$g = \sin c x$$

"
I, J. Nonc

$\ell = \text{none}$

$\ell \leftarrow \text{id} < (0/230)$

garbage collector

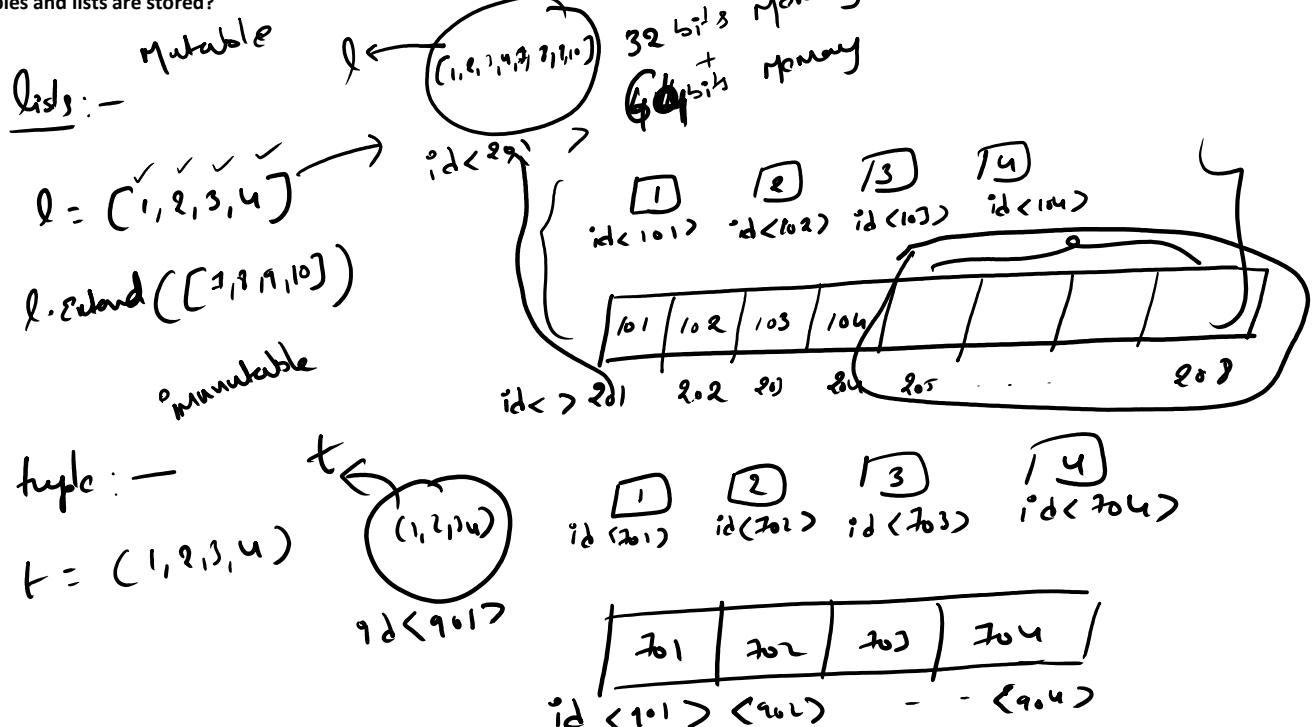
{ } { }

$(1, 2, 3, 4, 5)$ X

A diagram illustrating a sequence of three nodes, l_2 , l_1 , and l_3 . Each node is represented by a circle containing the word "None". Arrows indicate a flow or connection from l_2 to l_1 and from l_1 to l_3 .

Note :- Python is a dynamic in allocating & deallocating the Memory

How tuples and lists are stored?



Why tuples takes less memory and time for execution?

- list will take more memory than tuple ↳ (dynamic free space)
- list will take more execution time than tuple ↳ (cuz of mutability)

Difference between tuple and list

- 1) list
 mutable
- 2) More memory space
- 3) More execution time

- tuple
 immutable
- Low memory space
- Low execution time
 ... low is diff ()

3) More execution time

a) Syntax is different []

Yours -

Syntax is diff ()

What is mutability and how can we handle it (Cloning and Copying)

Ans copy

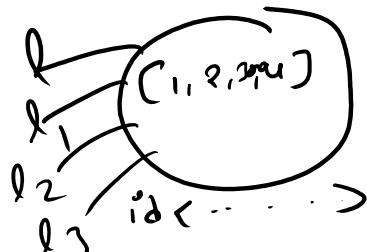
aliasing:-

def l = [1, 2, 3, 4]

l1 = l
l2 = l1

l3 = l

l2[2] = 300



l1[0:3:1]
for i in range(3):
 l1[i] = 100

l2 = l1[:]
l3 = l1[:]

l1[0:3:1]
for i in range(3):
 l1[i] = 200

l2[2] = 300

l1[0:3:1]
for i in range(3):
 l1[i] = 300

l2 = l1[:]

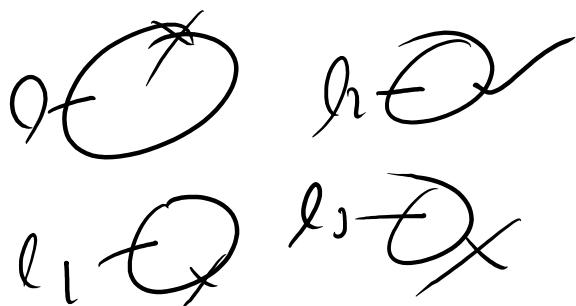
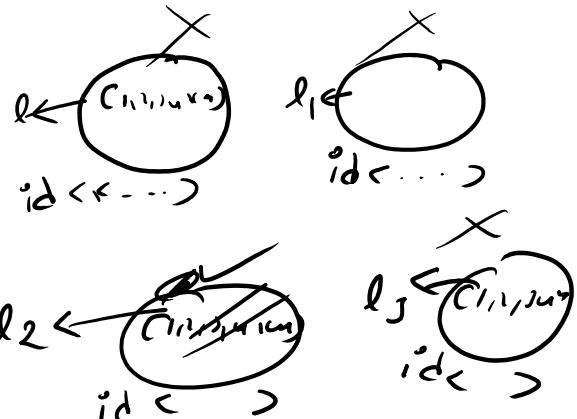
l1[0:3:1]
for i in range(3):
 l1[i] = 400

l3 = l1[:]

l2[2] = 200

l2[2] = 300

l2[2] = 400



Shallow Copy vs Deep Copy

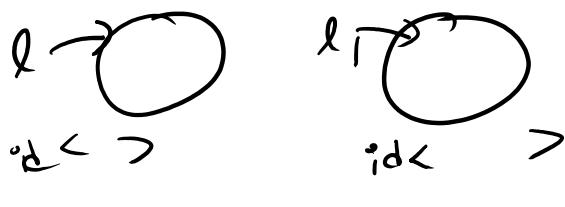
l = [1, 2, 3, 4]

l → O

l1 → O

Shallow Copy vs Deep Copy

$l = [1, 2, 3, 4]$



$$\lambda = [1, 2, 3, [4, 5, 6, 7], 8]$$

1
2
3
(u, s, b, ?)

8
l₁ = l.copy()

```
l1 = copy.deepcopy(l)
```

Configuration

id

host

$$91 \leftarrow (1, 1, 2, 1, \{[4, 5, 16, 2]\})_3$$

deep 40% ()
(1) (2) (3) (4)
861 862 863 864

$$g \in \text{loop} \\ g([x]) = x$$

The diagram illustrates a binary search tree and its corresponding array representation. The tree has the following structure:

- Root node: 101
- Left child of Root: 102
- Right child of Root: 103
- Left child of 103: 107
- Right child of 103: 108
- Right child of 108: 1010

A circled '4' is placed above the root node, and a circled '8' is placed above the rightmost leaf node (1010). A curved arrow points from the circled '4' to the circled '8'. The array representation below shows the nodes in level-order traversal:

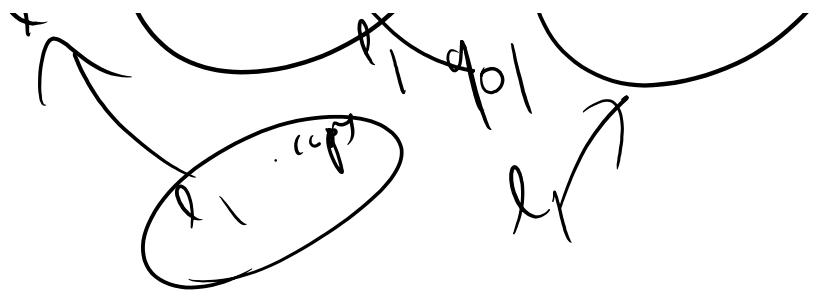
101	102	103	107	1010				
201	202	-	-	-	.	.	.	1

$$f(1) = (1, 1, 1, 1) \quad \text{and} \quad f(0) = (0, 0, 0, 0)$$

deepcore →

A hand-drawn diagram of a cell with various organelles labeled:

- Nucleus
- Chloroplast
- Endoplasmic reticulum
- Golgi apparatus
- Microvilli
- Cell wall
- Cell membrane
- Chlorophyll
- Protoplasm



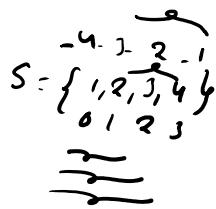
Set

25 December 2024 19:53

It is also collection of items ✓

Remember 4 rules:

1. Set doesn't allow duplicates ✓
2. Set is by default mutable ✓
3. Set doesn't allow indexing and slicing ✓
4. Set allows immutable datatypes inside it. ✓



String — immutable ✓
List — mutable
tuple — immutable ✓
int — immutable
float — 1 —
boolean — " " —
complex — " " —

Creation of Sets:

=====

1. s={1,2,3,4}
2. s=set()
3. s={"ugiug","uhieg","yftyut"}
4. s={1,2,3,"juhg","True"}
5. s=set("god")

Operations:

=====

Concatenation

Indexing

Slicing

Length

in

Method	Description
add(element)	Adds a single element to the set. If the element is already present, it has no effect.
remove(element)	Removes a specified element from the set. Raises a KeyError if the element is not found.
discard(element)	Removes a specified element from the set if it is present. Does nothing if the element is absent.
pop()	Removes and returns an arbitrary element from the set. Raises a KeyError if the set is empty.
clear()	Removes all elements from the set, making it empty.
union(*others)	Returns a new set containing all unique elements from the set and other sets (or iterables).
intersection(*others)	Returns a new set with elements common to the set and other sets (or iterables).
difference(*others)	Returns a new set with elements in the set but not in the other sets (or iterables).
symmetric_difference(other)	Returns a new set with elements in either the set or the other set, but not both.
update(*others)	Updates the set by adding elements from other sets (or iterables).
intersection_update(*others)	Updates the set with the intersection of itself and other sets (or iterables).
difference_update(*others)	Updates the set by removing elements found in other sets (or iterables).
symmetric_difference_update(other)	Updates the set with the symmetric difference of itself and another set.
isdisjoint(other)	Returns True if the set has no elements in common with another set.
issubset(other)	Returns True if the set is a subset of another set.
issuperset(other)	Returns True if the set is a superset of another set.
copy()	Returns a shallow copy of the set.

1) add

$$s = \{1, 2, 3\} \rightarrow$$

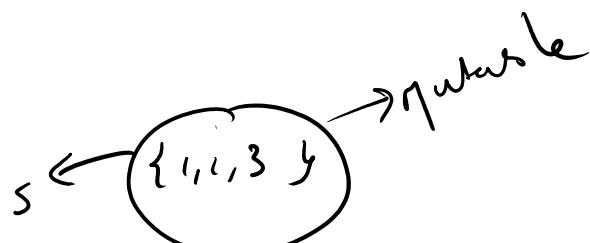
$$s.add(4) \checkmark$$



print(s)

2) remove

$$s = \{1, 2, 3, 4\} \rightarrow$$



2) remove

$s = \{1, 2, 3, 4\}$ → $s \leftarrow \{1, 2, 3\}$

$s.remove(4)$ ✓

$\text{print}(s)$ $\{1, 2, 3\}$

Markable

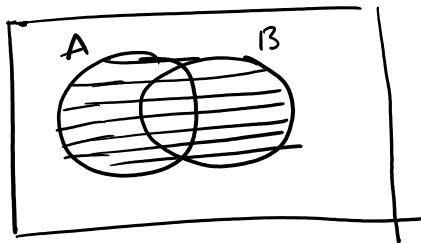
4) pop

$s = \{1, 2, 3, 4\}$

$\text{print}(s.pop())$

$\text{print}(s)$

Venn diagrams



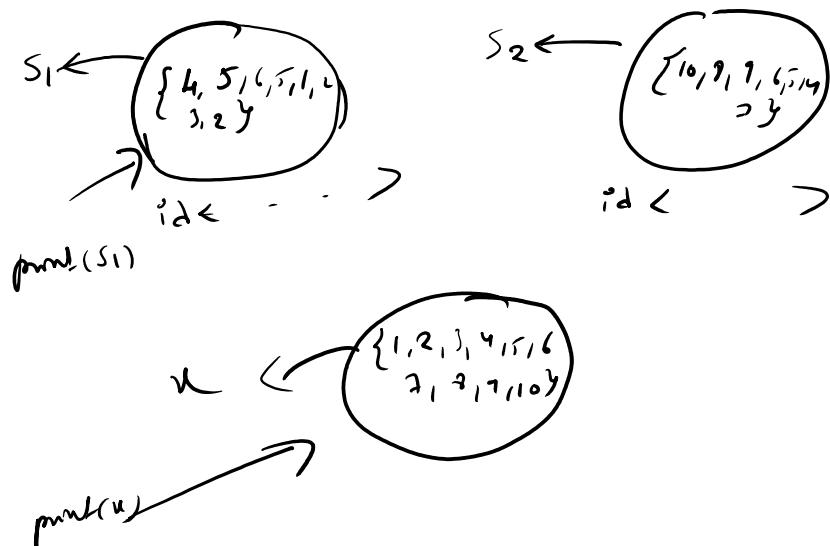
$A \cup B$ *union*

$A \cap B$

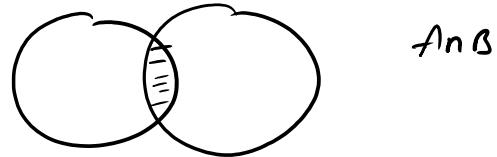
$A \setminus B$

union(*others)

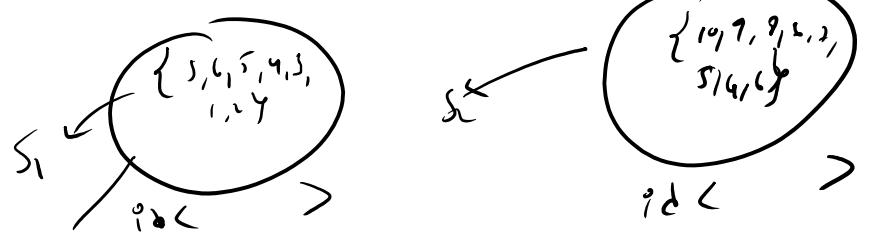
$s1 = \{1, 2, 3, 4, 5, 6, 5\}$
 $s2 = \{4, 5, 6, 7, 8, 9, 10\}$
 $\text{print}(s1.union(s2))$ ✓



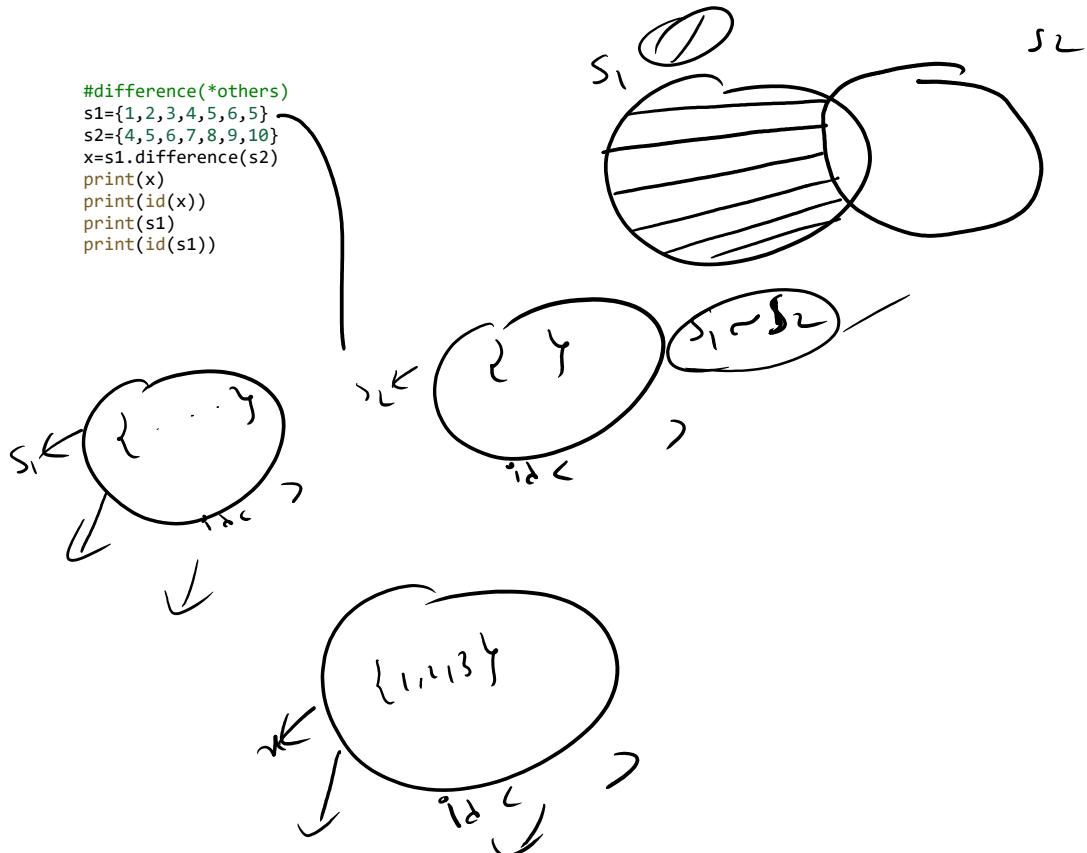
intersection(*others)



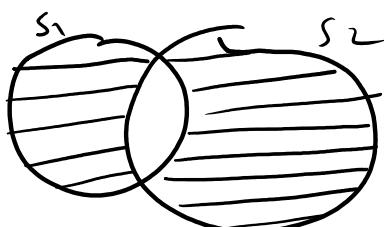
```
#intersection(*others)
s1={1,2,3,4,5,6,5}
s2={4,5,6,7,8,9,10}
x=s1.intersection(s2)
print(x)
print(id(x))
print(s1)
print(id(s1))
```



```
#difference(*others)
s1={1,2,3,4,5,6,5}
s2={4,5,6,7,8,9,10}
x=s1.difference(s2)
print(x)
print(id(x))
print(s1)
print(id(s1))
```

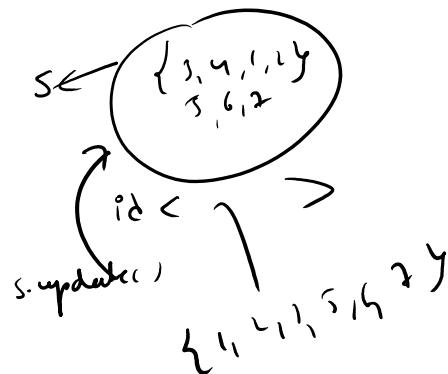


```
symmetric_difference(other)
```

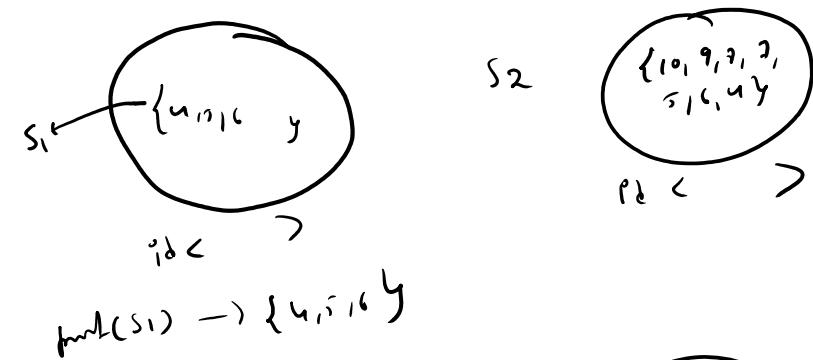


```
#symmetric_difference(*others)
s1={1,2,3,4,5,6,5}
s2={4,5,6,7,8,9,10}
x=s1.symmetric_difference(s2)
print(x)
print(id(x))
print(s1)
print(id(s1))
```

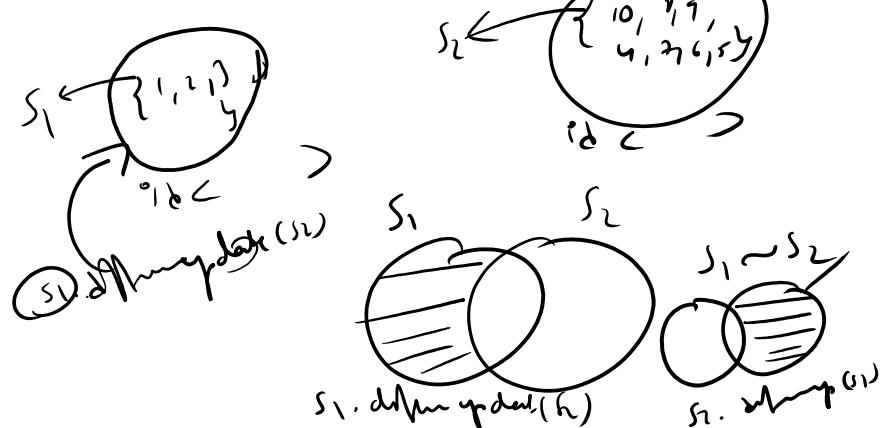
```
#update(*others)
s={1,2,3,4}
print(id(s))
s.update({5,6,6,7})
print(s)
print(id(s))
```



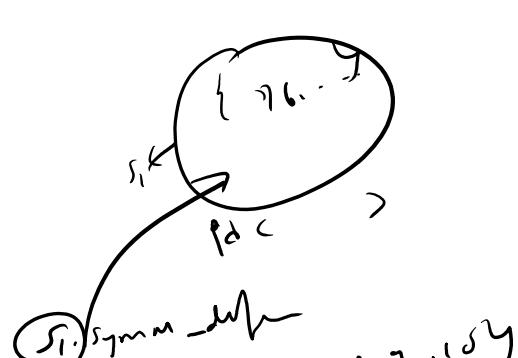
```
#intersection_update(*others)
s1={1,2,3,4,5,6,5}
s2={4,5,6,7,8,9,10}
s1.intersection_update(s2)
print(s1)
```



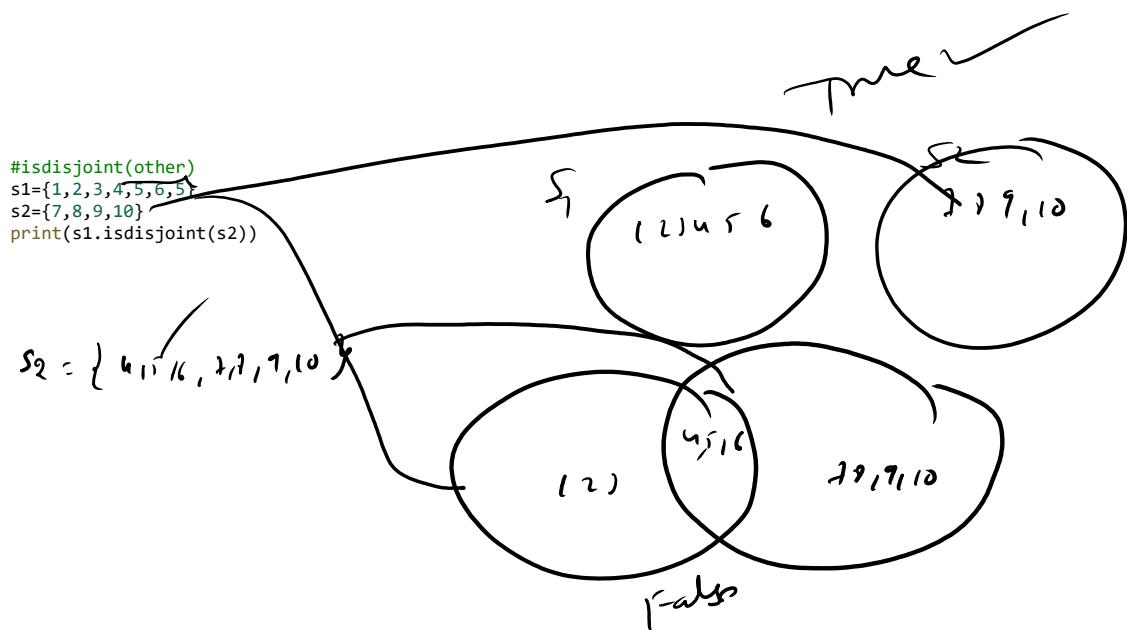
```
#difference_update(*others)
s1={1,2,3,4,5,6,5}
s2={4,5,6,7,8,9,10}
s1.difference_update(s2)
print(s1)
```



```
#symmetric_difference_update(*others)
s1={1,2,3,4,5,6,5}
s2={4,5,6,7,8,9,10}
s1.symmetric_difference_update(s2)
print(s1)
```



s_1 .symmetric
 $\{1, 2, 3, 1, 2, 3, 1, 2, 1, 3\}$



issubset(other)

$s_1 = \{1, 2, 3, 4\}$
 $s_2 = \{1, 2, 3, 4, 7, 8, 9, 10, 11, 12\}$
print(s_1 .issubset(s_2))

True

copy
 $s = \{1, 2, 3, 4\}$
 $s_1 = s.copy()$ ✓
print(s)
print(s_1)



Dictionary

23 December 2024 23:23

It is a collection of key-value pairs

Remember 4 rules:

-
1. Dictionary is mutable
 2. Dictionary keys should be immutable but values can be mutable or immutable
 3. Dictionary doesn't support indexing and slicing like set
 4. Dictionary keys should be unique

Creation of Dictionary:

```
_ = {"Name": "data", "age": 25}
_x_ = {}
__123 = dict()
c_gpa = {"Name": "data", "age": 25, "marks": ["maths": 96, "english": 87]}
```

Operations:

Concatenation

Indexing

Slicing

Length

in

Method	Description
dict.clear()	Removes all items from the dictionary.
dict.copy()	Returns a shallow copy of the dictionary.
dict.fromkeys(iterable, value=None)	Creates a new dictionary with keys from the iterable and values set to value.
dict.get(key, default=None)	Returns the value associated with the key if it exists; otherwise, returns the default value.
dict.items()	Returns a view object displaying a list of dictionary's key-value tuple pairs.
dict.keys()	Returns a view object displaying a list of all the keys in the dictionary.
dict.values()	Returns a view object displaying a list of all the values in the dictionary.
dict.pop(key, default=None)	Removes the specified key and returns its value. If the key is not found, returns the default value.
dict.popitem()	Removes and returns the last key-value pair in the dictionary as a tuple.
dict.setdefault(key, default=None)	Returns the value of the key. If the key does not exist, inserts the key with the specified default value.
dict.update([other])	Updates the dictionary with the key-value pairs from another dictionary or an iterable of key-value pairs.

