Data Science & Artificial Intelligence

Python for Data Science

Python Collections and String Handling

Lecture No. 04

By- Kashif Sir

RECAP

List → List Comprehension

Tuple → Tuple packing
Tuple Unpacking
Methods

sort( ), sorted( ), reverse( ), reversed( )

Topics *to be covered*

- Python Collections and String Handling

SET

Initialization

```
s = { }
print(type(s))
```
<dict>

```
s = {1, 2, 3, 4, 5}
s = {"Hello", 1, 1.0, 5, 7}
```
⟶ `s = {1, 2, 3, [4, 5]}`

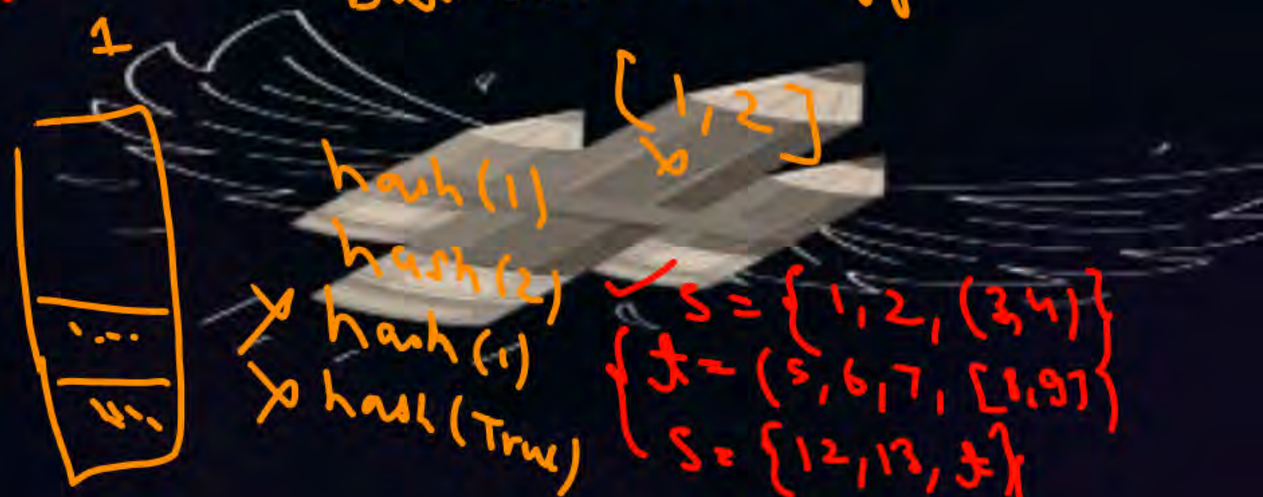
set does not contain list, set / dict

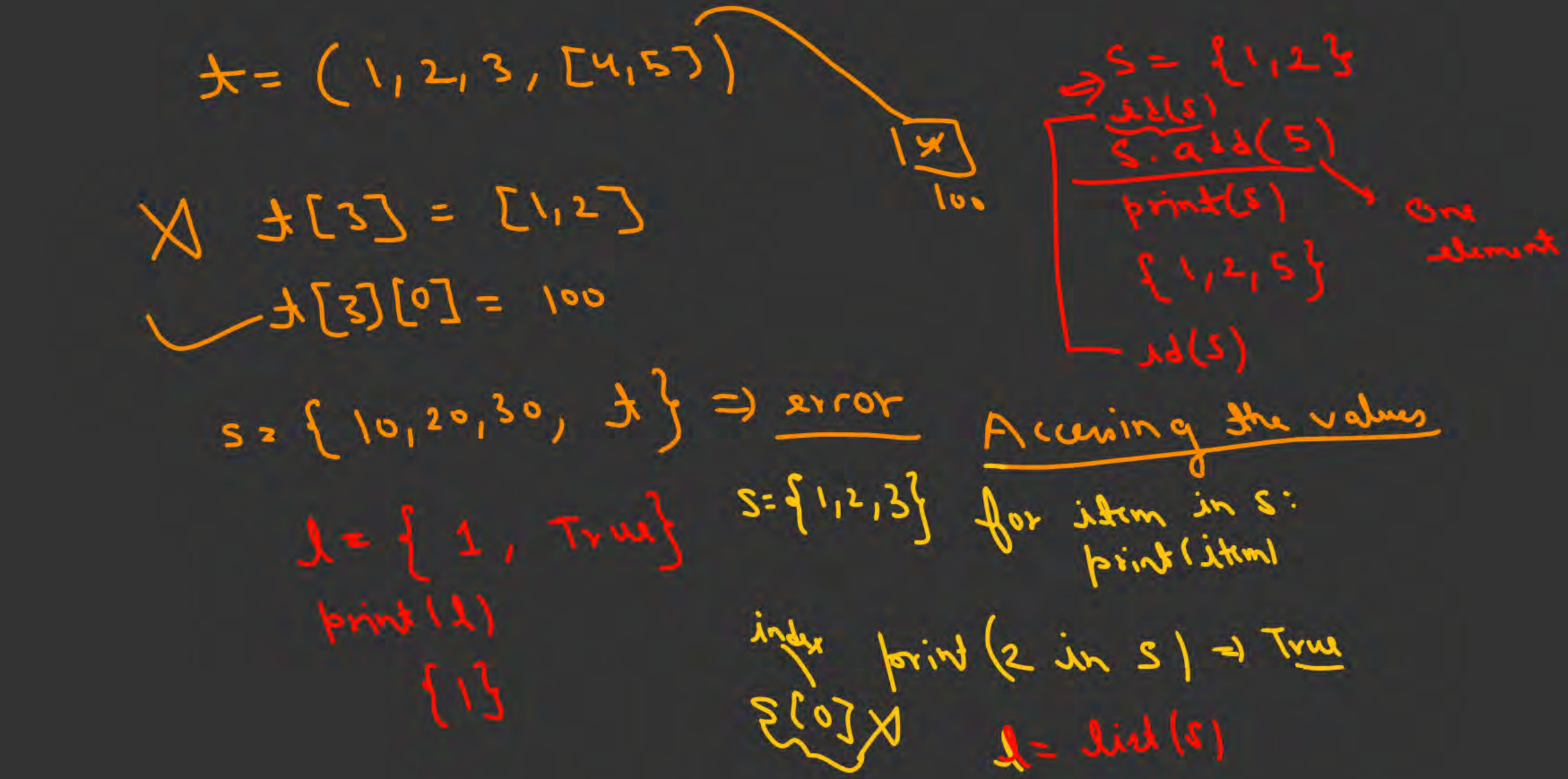
```
s = {1, 2, 3, 7}
```
<set>

{1, 2, {3, 7}} ✗

```
p = set( )
```

Unordered, mutable, duplicates not allowed, add / remove elements, but can't modify




t = (1, 2, 3, [4, 5])
x
100
t[3] = [1, 2]
t[3][0] = 100
s = {1, 2}
add(s)
s.add(5)
print(s)
{1, 2, 5}
rd(s)
one element
s = {10, 20, 30, t} => error
l = {1, True}
print(l)
{1}
Accessing the values
s={1,2,3}
for item in s:
print(item)
index
print(2 in s) => True
s[0]
l = list(s)

```
S = {1,2,3}
S.update({4,5,6})
print(S)
    {1,2,3,4,5,6}
S.update("GATE")
print(S)
    {1,2,3,"G","A","T","E"}
S.update({"GATE"})
    {1,2,3,"GATE"}
```

```
t = (9,10,11)
a = S.union(t)
print(a)
{1,2,3,9,10,11}
```

update()

S.update()

↑

It will update the set and return type will be None

union()

l = S.union(t)

returning a new set, without modifying the actual set

remove( )

a = {1, 2, 3}

a.remove(3)

print(a) ⇒ {1, 2}

a.remove(4)

↳ error

discard( )

a = {1, 2, 3}

a.discard(3)

print(a)

{1, 2}

a.discard(6)

↳ no error

S = {1, 2, 3, 2, 5, 7, 5, 9}

len(S) S = {1, 2, 3, 5, 7, 9}

pop( )

randomly deletes one element from set

a = {1, 2, 3}

a.pop( )

⇓

It will randomly delete one element from set.

S = {1, 0, 1.0, True, False, 0.0}

len(S) S = {1, 0}

2

Union

a.union(b, c, d ...)

a | b | c | d ...

b | a

a = {1, 2, 3} b = {2, 3, 5}

len(a | b)

{1, 2, 3, 5} ⇒ 4

Intersection

a = {1, 2, 3}

b = {2, 3, 4}

b & a

a.intersection(b)

len(a & b) = 2

{2, 3}

Intersection_Update

a = {1, 2, 3}

b = {2, 3, 4}

a.intersection_update(b)

print(a)

{2, 3}

a.intersection_update(b)

a = → {2, 3}

difference

a = {1,2,3}

b = {2,3,4}

c = a.difference(b)

a - b

c = {1}

b - a = {4}

difference_update

a = {4,2,3}

b = {2,3,4}

a.difference_update(b)

{1}

print(a)

= {1}

symmetric_difference

a = {1,2,3}

b = {2,3,4}

a ∪ b − a ∩ b

{1,2,3,4} − {2,3}

=> {1,4}

c = a.symmetric_difference(b)

print(c)

{1,4}

clear()

a.clear()

remove all the elements of the set

del

del a

delete the set

isdisjoint()

a = {1,2,3}

b = {4,5,6}

print(a.isdisjoint(b)) ⇒ True

copy()

a = {1,2,3}

b = a.copy()

issubset()

a = {1,2}

b = {1,2,3,4}

True = print(a.issubset(b))

False = print(b.issubset(a))

issuperset()

a = {1,2}

b = {1,2,7,7}

False = print(a.issuperset(b))

print(b.issuperset(a)) = True

FROZEN SET

- Immutable
- Unordered
- Duplicates not allowed
- Do not have add / remove methods

```
a = frozenset( )
a = frozenset( {1,2,3} )
a = {1,2,3}
```

UNION( )
INTERSECTION( )
DIFFERENCE( )
SYMMETRIC-DIFFERENCE( )
COPY( )

DICTIONARY

It is used to store data in key-value format.

Ordered, mutable, duplicates not allowed (Key)

s = {1,2,3}

l = [1,2,3]

l[0]

Initialization

d = { }, d = dict( ), d = {"a" : 1, "b" : 2, "c" : 11}

d = {1 : "a", 2.0 : [1,2,3], "d" : (1,2)}

Dictionary elements will be indexed through Keys, not with sequence.

print(d[1]) ⇒ "a"

print(d[2.0]) ⇒ [1,2,3]

```
for item in d:
    print(item)
```

1
2.0
"d"

d = { 1 : "F" , 1.0 : "AB" , 0 : "Z" , False : "AZ" , 1.0 : "X" }

d = { 1 : ~~"F"~~ ~~"AB"~~ "X" , 0 : ~~"Z"~~ "AZ" }

d = { 1 : "X" , 0 : "AZ" }

d[0] ⇒ "AZ"

d2 = { False : 'a' , 0 : 'b' }

d2 = { False : 'b' }

d1 = { "1" : 10 , 1.0 : 5 , 0 : 10 }

d1["1"] ⇒ 10

l = { 1 , 2 , 5 , 2 , 1 }

1 == 1.0 = True

{ 1 : ~~"F"~~ ~~"AB"~~ "X" , 0 : ~~"Z"~~ "AZ" }

0 == 0.0 = = False

d4 = { "a" : 12 , "b" : 12 , "c" : 12 }

d1 = dict(a=10, b=12, c=13, d=[10,20,30])

d1 = {'a':10, 'b':12, 'c':13, 'd':[10,20,30]}

d2 = {'a':(1,4), 'b':4}

d5 = dict([('a',1),('b',4),('c',3)])

d5 = {'a':1, 'b':2, 'c':3}

d = {'a':10, 'b':20, 'c':30}

d1 = d

⇒ d2 = d.copy()

⇒ d3 = dict(d)

⇒ d4 = {'a':10, 'b':20, 'c':30}

d['a'] = 100 (10 → 100)

d1['a'] ⇒ 100

1, 1.0, True

0, 0.0, False

bool(9)

True

Summary

Set

remove, discard, pop

Frozen Set

Dictionary

- Union, Intersection, Difference, Symmetric Difference
  Intersection_update, Difference_update

THANK - YOU