Date / /
Page

Unit - 3

Lists, Dictionaries and Strings.

→ Check the presence of element in list:

```
l1 = [3,4,5]
if 4 in l1:
        print('Yes')
else:
    print('no')
```

* Basic List Operations / List operators

- Repetation:
```
l1 = [3,2,4]
print(l1*2)   # [3,2,4,3,2,4]
```

- Concatenation:
```
l1 = [3,2,4]
l2 = [0,1,5]
print(l1+l2)   # [3,2,4,0,1,5]
```

- Length:
```
l1 = [3,2,4]
print(len(l1))   # 3
```

- Iteration:
```
l1 = [3,2,4]      # 3
for i in l1:      # 2
    print(i)      # 4
```

- Membership:
```
l1 = [3,2,4]
print(3 in l1)   # True
```

[This give Output in True & False]

Teacher Signature ..... Parents Signature .....

Date / /
Page:

* Methods of list :

- **Append** : We can add one element at a time. Add elements at the end of the list.

eg
```
l1 = [ ]
l1.append (5)        # [5]
print (l1)
```

Append tuple in list :
```
l1 = [ ]
l1.append ((5,4))        # [(5,4)]
print (l1)
```

Append list in list :
```
l1 = [ ]                 # [[5,4]]
l1.append ([5,4])
print (l1)
```

- extend : Add multiple value at the end. (Concatenate l1 and l2.)
```
l1 = [ ]
l1.extend ([5,6,'Hlo'])      # [5,6,'Hlo']
print (l1)
```

Teacher Signature ..... Parents Signature .....

We can add list, set also in list.

- Insert :

```
l1 = [1, 2, 3]
l1.insert(1, 'Hlo')      # [1, 'Hlo', 2, 3]
print(l1)
```

eg :
```
String = '1234'
String.insert(0, '5')    # attribute error
print(String)
```

Note : Want to insert any element in the beginning we give index = 0

:: To add the element at the end of list using insert keyword.

```
l1 = [1, 2, 3]
l1.insert(len(l1), 4)    # [1, 2, 3, 4].
print(l1)
```

:: Remove elements from list.

eg.
```
l1 = [2, 3, 4]
l1.remove(4)             # [2, 3]
print(l1)
```

↓ Remove only 1 element.

↓ Remove specific item from list.

Teacher Signature ..... Parents Signature .....

Date / /
Page

- delete more elements

```
l1 = [2,3,4,5,6,7,8]
del l1[0:2]                    # [4,5,6,7,8]
print(l1)
```

- To remove last element

```
l1 = [2,3,4,5]
l1.pop()                       # [2,3,4]
print(l1)
```

- To find the minimum value

```
l1 = [1,2,3,4]                 # 1
print(min(l1))
```

- To find the maximum value

```
l1 = [1,2,3,4]                 # 4
print(max(l1))
```

- Reverse the values

```
l1 = [1,2,3,4]
l1.reverse()                   # [4,3,2,1
print(l1)
```

Teacher Signature .....

Parents Signature .....

Date / /
Page

- Count

```
l1 = [2, 3, 4, 7, 7, 7, 8]
a = l1.count(7)          # 3
print(a)
```

- Repeat

```
l1 = [2, 3]
print(l1 * 3)            # [2, 3, 2, 3, 2, 3]
```

- Sort (Ascending order)

```
l1 = [5, 4, 2, 1, 3]
l1.sort()                # [1, 2, 3, 4, 5]
print(l1)
```

- Clear

```
l1 = [1, 2, 3, 4]
l1.clear()               # []
print(l1)
```

- Replacing element in the list.

```
l1 = ['Hello', 1, 2, 3]
l1[2] = 'xyz'            # ['Hello', 1, 'xyz', 3]
print(l1)
```

Teacher Signature .....

Parents Signature .....

Date / /
Page

- Replacing using for loop:

```
l1 = ['hlo', 1, 2, 3]
for i in range(len(l1)):                    # ['hlo', 1, 'xyz', 3]
        if l1[i] == 2:
            l1[i] = 'xyz'
print(l1)
```

- Replace using lambda

In this method, we use lambda & map fxn to replace the value in the list. Map() is a built in fxn in python to iterate over a list without using any loop statement.

Syntax `l = list(map(lambda x: x.replace('old value', 'new value'), l))`

eg
```
l = ['Ram', 'Sham', 'Rohit', 'Rahul', 'Pant']
l = list(map(lambda x: x.replace('Pant', 'Ishan'), l))
print(l)
                # ['Ram', 'Sham', 'Rohit', 'Rahul', 'Ishan']
```

- Find character in string:

```
S1 = 'This is class'
print(S1.find('c'))    #8
                       #Return the index
```

Teacher Signature .....

Parents Signature .....

Date / /
Page:

- Replace part of String

```
S1 = 'coding'
print(S1.replace('ing', 'ers'))          # coders
```

old value → 'ing', new value → 'ers'

- Count letters in string [duplicate / or any]

```
S1 = 'codingg'
print(S1.count('g'))          # 2
```

- Split the string

```
S1 = 'This is class'
print(S1.split(" "))          # ['This', 'is', 'class']
```

- Stratswith fxn()

```
S1 = 'This is class'
print(S1.startswith('T'))     # True
```

- Endswith fxn()

```
S1 = 'This is class'
print(S1.endswith('s'))       # True
```

- Upper()

```
S1 = 'This is class'          # THIS IS CLASS
print(S1.upper())
```

Teacher Signature .....

Parents Signature .....

Date / /
Page:

- lower ()

```
s1 = 'THIS IS Class'
print (s1. lower ())          # this is class
```

- Capitalize ()

```
s1 = 'this is class'
print (s1. Capitalize ())     # This is class
```

- String. Swapcase ()

```
s1 = 'abcd'
print (s1. swapcase())        #ABCD
```

- title ()

```
s1 = 'This is book'           # This Is Book
print (s1. title ())
```

- Strip ( ) — lstrip() / rstrip()

```
s1 = "**** abc ****"
print (s1. Strip ('*'))       #  abc
print (s1. lstrip ('*'))      #  abc ****
print (s1. rstrip ('*'))      #  **** abc
```

⇨ Isalnum method return 'True' if all the characters in the string are alphanumeric (either alphabets or numbers)

Teacher Signature ..... Parents Signature .....

Date / /
Page:

* word = 'Hello1'
print (word.isalnum()) # True
print (word.isalpha()) # False
print (word.isdigit()) # False

Dictionary literals.

3.7 Version and above dictionary → ordered
3.6 and less are unordered.

* Create a dict
d1 = {1: 'xyz', 2: 'abc'}
print (d1) # {1: 'xyz', 2: 'abc'}

* Access dict
d1 = {1: '3', 2: 'xyz'}
print (d1[1]) # '3'

* dicts are changable:
d1 = {1: 'abc', 2: 'kkb', 2: 'xyz'}
print (d1) # {1: 'abc', 2: 'xyz'}

Teacher Signature ..... Parents Signature .....

Date / /
Page:

* Adding keys in dict

```
d1 = {1: 'abc', 2: 'xyz'}
d1['2'] = 'zz'
print(d1)          # {1: 'abc', 2: 'xyz', '2': 'zz'}
```

* Replace Variable

```
d1 = {1: 'abc', 2: 'xyz'}
d1[2] = 'zz'
print(d1)          # {1: 'abc', 2: 'zz'}
```

* Remove keys

```
d1 = {1: 'abc', 2: 'xyz'}
del d1[2]
print(d1)          # {1: 'abc'}
```

Pop ()

```
d1 = {1: 'abc', 2: 'xyz'}
d1.pop(2)
print(d1)          # {1: 'abc'}
```

* delete whole dict

```
d1 = {1: 'abc', 2: 'xyz'}
print(d1.clear())  # None
```

Teacher Signature .....

Parents Signature .....

Date / /
Page:

- Note: Dictionarie keys vare case senstive.

* Nested dictionaries:

```
D1 = {1: 'abc', 2: 'xyz', 3: {4: 'hlo'}}
print(D1)          # {1: 'abc', 2: 'xyz', 3: {4: 'hlo'}}
print(D1[3][4])    # Hlo
```

* Get method to access elements/values.

```
d1 = {1: 'abc', 2: 'xyz', 3: {4: 'hlo'}}
print(d1.get(2))   # xyz
```

* Add key value pair in dict

```
d1 = {1:'abc', 2: 'xyz', 3: 'hlo'}
d1.__setitem__(4, 'hi')
print(d1)   # {1: 'abc', 2: 'xyz', 3: 'hlo', 4: 'hi'}
```

OR

```
d1 = {1: 'abc', 2: 'xyz', 3: 'hlo'}
d2 = {**d1, **{4: 'hi'}}
print(d2)   # {1: 'abc', 2: 'xyz', 3: 'hlo', 4: 'hi'}
```

* Adding a new key without value:

```
d1 = {1: 'abc', 2: 'xyz'}
d1[3] = 'none'
print(d1)   # {1: 'abc', 2: 'xyz', 3: 'none'}
```

Teacher Signature ..... Parents Signature .....

Date / /
Page

* Adding multiple key-value pairs using update method.

```
d1 = {1: 'orange'}
d1.update ({2: 'Red', 3: 'green'})
print (d1)
```

* Replace dict value from other dict:

```
test_dict = {'Gfg': 5, 'is': 8, 'Best': 10, 'for': 8,
             'Geeks': 9}
print (test_dict) # {'Gfg': 5, 'is': 8, 'Best': 10, 'for': 8,
                    'Geeks': 9}
Updict = {'Gfg': 10, 'Best': 17}
for key in test_dict:
    if key in updict:
        test_dict[key] = updict[key]
print (test_dict) # {'Gfg': 10, 'is': 8, 'Best': 17,
                    'for': 8, 'Geeks': 9}
```

⇨ Traversing Dictionaries:

- Traversing over key :-

```
d1 = {1: 'hlo', 2: 'hi'}
for key in d1:
    print (key)    # 1
                   2
```

Teacher Signature ..... Parents Signature .....

key,

- For 1 value :

```
d1 = {1:'Hlo', 2:'Hi'}
for key, values in d1.items():
        print(key, values)  # 1 Hlo
                            2 Hi
```

Note: We cannot make mutable items as a key. It will give unhashable error.

⇨ Built-In functions / Methods of Dict. in Python.

* len()

```
D1 = {1:'Hi', 2:'Hlo'}
print(len(D1))          # 2
```

* Clear()

```
D1 = {1:'Hi', 2:'Hlo'}
print(D1.clear())       # None
```

* Sorted() [Sort keys]

```
D1 = {2:'Hlo', 5:'Hi', 1:'Bye'}
print(Sorted(D1))       # [1, 2, 5]
```

Teacher Signature ..... Parents Signature .....

Date / /
Page:

* copy()

```
D1 = {9: 'a', 3: 'b'}
d = D1.copy()
print(d)          # {9: 'a', 3: 'b'}
```

* Pop()

```
D1 = {9: 'a', 3: 'b'}
D1.Pop(9)
print(D1)         # {3: 'b'}
```

* Popitem() [del. the recent key]

```
D1 = {9: 'a', 3: 'b'}
D1.popitem()
print(D1)         # {9: 'a'}
```

* keys()

```
D1 = {9: 'a', 3: 'b'}
D2 = D1.keys()
print(D2)         # dict_keys([9, 3])
```

* Values()

```
D1 = {9: 'a', 3: 'b'}
D2 = D1.values()
print(D2)         # dict_values(['a', 'b'])
```

Teacher Signature .....

Parents Signature .....

Note! Dict's are mutuable.

Date / /
Page:

* items ()

```
d1 = {9: 'a', 3: 'b'}
d = d1.items()
print (d)          # dict_items ([(9, 'a'), (3, 'b')])
```

* get ()

```
d1 = {9: 'a', 3: 'b'}
d = d1.get(9)
print (d)          # b
```

* Update ()

```
d1 = {9: 'a', 3: 'b'}
d1.update ({3: 'c'})
print (d1)         # {9: 'a', 3: 'c'}
```

⇨ Dict. Membership Test.

Note! It only work on keys.

```
d1 = {9: 'a', 3: 'b'}
print (9 in d1)    # True.
```

Teacher Signature .....

Parents Signature .....