INDEX

STRING

① Any sequence of characters within either single quotes or double quotes

Access string ⇒ ① Index ② Slice operator

mathematical operator ⇒ ① + ⇒ concatenatn ② * ⇒ repetetn

checking membership ⇒ ① in ② not in

comparision of strings ⇒ ① comparision operators (<, <=, >, >=)
② equality operators (==, !=)

Removing space from string ⇒ ① rstrip() ② lstrip() ③ strip()

finding substrings ⇒

① for forward direction ⇒ ① find() ② index()

② for backward direction ⇒ ① rfind() ② rindex()

s.find(substring) ⇒ returns index of first occurence of the given substring if it is not available then we will get -1.

s.index(subs) ⇒ same as find() method except that if the specified substring is not available then we will get ValueError.

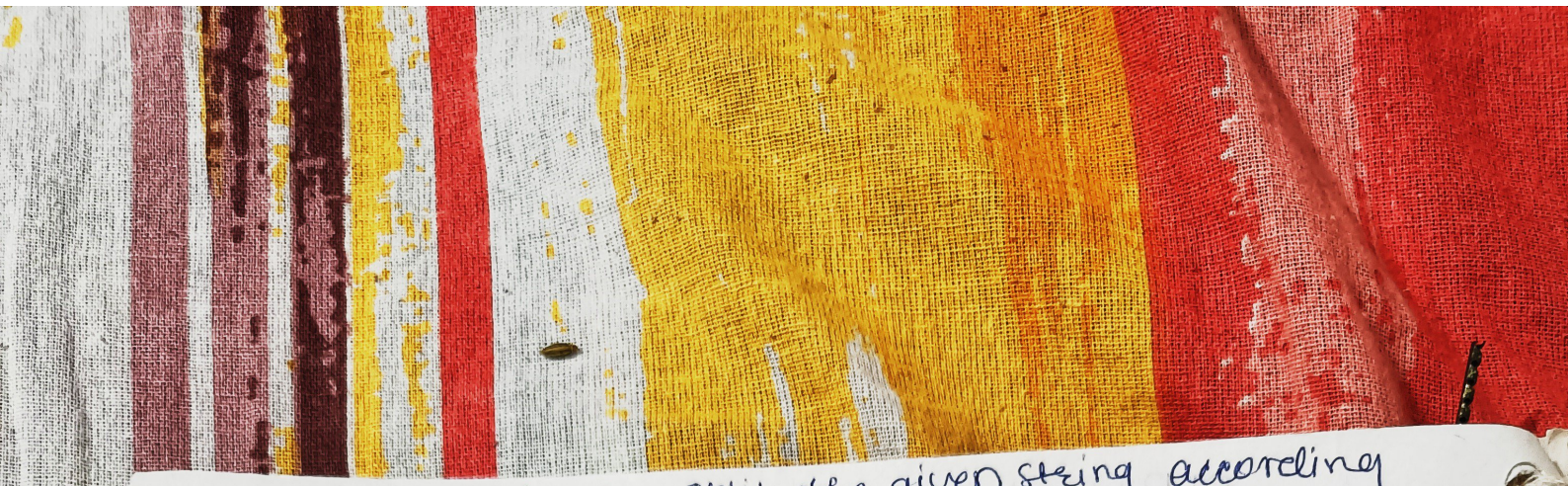
Counting substring in the given string :-

find no. of occurrences of substring present in the given string by using count() method.

s.count(substring)

Replacing a string with another string :-

s.replace(oldstring, newstring)



Splitting of Strings:- split the given string according to specified seperator by using split() method

`n = s.split (seperator)`

Joining of strings:- we can join a group of strings (list or tuple) w.r.t the given seperator.

`s = seperator.join(group of strings)`

Changing Case of a String :- ① upper() ② lower() ③ swapcase() ④ title() ⑤ capitalize()

Checking Starting & Ending part of the String:-

① s.startswith(substring) ② s.endswith(substring)

To check type of characters present in a string:-

① isalnum() ② isalpha() ③ isdigit() ④ islower() ⑤ isupper() ⑥ istitle() ⑦ isspace()

return ⇒ True, False

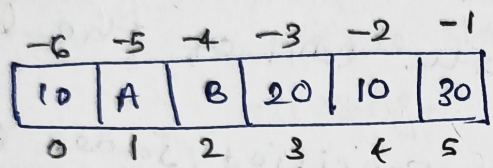
LIST

→ If we want to represent a group of individual objects as a single entity where insertion order preserved & duplicates are allowed, then we should go for list.

→ insertion order preserved.

→ duplicate objects are allowed.

→ list is dynamic.



→ list objects are mutable. # can change content.

Creation of List Objects :-

① can create empty list ⇒ list = [ ]

② with Dynamic Input ⇒ list = eval (input ("Enter list:"))

③ with list() function ⇒ I = list (range (0, 10, 2))

④ If we know element already then ⇒ list = [10, 20, 30]

⑤ with split() functn ⇒ I = s.split ( )

Accessing Elements of List :-

① By using Index ② by using slice operator(:)

Traversing the Elements of List :-

↳ sequential access of each element in the list

① By using while loop

② By using for loop

FUNCTIONS OF LIST ⇒

To get Information about list :-

① len() ⇒ no. of ele present in list

② count() ⇒ returns the no. of occurrences of specified item in the list.

③ index() ⇒ returns the index of first occurrence of item. if ele. not present then get ValueError.

Manipulating Elements of list :-

① append() :- add item at the end of the list.

② insert() :- insert item at specified index posit^n.

③ extend() :- To add all items of one list to another list.

④ remove() :- we can remove specified item from list. [1st preference] if not present then ValueError

⑤ pop() :- removes & returns the last element of the list
If list is empty the pop() raises IndexError
n.pop(index) ⇒ To remove & return element present at specified index.

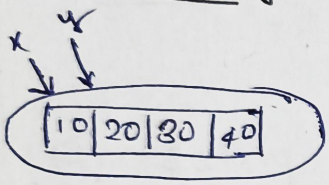
remove()	pop()
① we can use to remove special element from the list.	① we can use to remove last element from the list.
② It can't return any value.	② It returned removed element.
③ If special element not available then we get value error	③ If list is empty then we get error.
append(), insert(), extend() → for increasing the size / growable nature	
remove(), pop() → for decreasing the size / shrinking nature	

Ordering Elements of list:-

① **reverse():-**
We can use to reverse() order of elements of list.

② **sort():-** In list by default insertion order is preserved.
To use sort(), compulsory list should contain only homogeneous elements.

Aliasing ⇒ The process of giving another reference variable to the existing list is called aliasing.



x = [10, 20, 30]
y = x

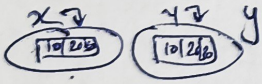
If we are changing content, then those changes will be reflected to the other reference variable.

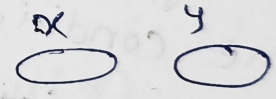
To overcome this problem we should go for **cloning**

cloning by using slice operator or by using copy()

① x[10, 20, 30]
y = x[:]

② x = [10, 20, 30]
y = x.copy()





Using Mathematical Operators for List Objects:-

① concatenate operator (+) ⇒ both argument should be list objects otherwise ⇒ TYPEERROR

② Repetition operator (*) ⇒
use to repeat elements of list specified no. of times.

Whenever we are using relational operators (<, <=, >, >=) betn List Objects, only 1st Element comparision will be performed.

Membership Operators

We can check whether element is a member of the list or not by using membership operators.

1. in operator
2. Not in operator

① clear() function :-

We can use clear() function to remove all elements of list.

② Nested Lists ⇒

We can take one list inside another list.

List Comprehensions :-

It is very easy and compact way of creating list objects from any iterable objects, based on some condition. (list, tuple, dict, Range etc)

Syntax ⇒ list = [expression for item in list if condⁿ]

```
s = [x * x for x in range (1, 11)]
```

[or]

```
m = [x for x in s if x%2 == 0]
print(m)
```

TUPLE

① It is **immutable**.
② Tuple is Read only version of List.
③ Insertion order is preserved.
④ Duplicates are allowed.
⑤ Heterogeneous objects are allowed.
⑥ We can preserve insertion order & we can different - ate duplicate objects by using index.

Syntax ⇒
```
t=10,20,30
t=()
t=(10,)
```

Accessing Elements of Tuple :-

① By using index
② By using Slice Operator.

Mathematical Operators for Tuple :-

① Concatenation operator (+)
② Multiplicatn or Repetition operator(*).

Imp function of Tuple :- t=(10,20,30,40)

① len()
② count() ⇒ t.count(10) $\xrightarrow{o/p}$ 1
③ index() ⇒ i) return index of first occurrence of given element.
ii) t.index(10) $\xrightarrow{o/p}$ 0 if not present then ValueError

④ sorted() ⇒ sorted(t)
⑤ min() & max() functions
⑥ cmp()

Tuple Packing and Unpacking

```
t = a,b,c,d # a,b,c,d are packed into a tuple t.
              called as tuple packing.
```

Tuple unpacking is the reverse process of tuple packing. we can unpack a tuple & assign its values to diff variables.

```
t = (10,20,30,40)
a,b,c,d = t
```

Tuple Comprehension

```
t = ( x**2 for x in range(1,6))
```

SET

① If we want to represent a group of unique values as a single entity then we should go for set.

② Duplicates are not allowed.

③ Insertion order is not preserved. But we can sort the elements.

④ Indexing and slicing are not allowed for the set.

⑤ Heterogeneous elements are allowed.

⑥ Set objects are mutable.

⑦ We can apply mathematical operations like union, intersection, difference etc. on set objects.

Creation of set objects

i) S = {10, 20, 30, 40}

ii) L = [10, 20, 30]
S = set(L)

iii) s = set(range(5))

Important functions of Set

① add(x)

② update(X, Y, Z)

③ copy() ⇒ s1 = s.copy()

④ pop() ⇒ s.pop() ⇒ It removes & returns some random element from the set.

⑤ remove(x) ⇒ give error

⑥ discard(x) ⇒ No error

⑦ clear() ⇒ s.clear() ⇒ remove all elements from set.

Mathematical operations on the set =>

① union() => x.union(y) or x|y

② intersection() => x.intersection(y) or x&y

③ difference() => x.difference(y) or x-y

Returns the elements present in x but not in y.

④ symmetric_difference() => x.symmetric_difference(y) or x^y

Membership Operators :- (in, not in)

DICTIONARY

→ If we want to represent a group of objects as key-value pairs then we should go for Dictionary.

⇒ Duplicate keys are not allowed but values can be duplicated.

⇒ Heterogeneous objects are allowed for both key & values.

⇒ Insertion order is not preserved.

⇒ Dictionaries are mutable.

⇒ Dictionaries are dynamic.

⇒ Indexing and slicing concepts are not applicable.

Creation of Dictionary

① d = {} or d = dict()

Accessing Data from Dictionary

① Access data by using keys.
If not available then we will get KeyError.

Delete Element from Dictionary

① del d[key]

② d.clear() → To remove all entries from the dictionary.

③ del d

Functions of Dictionary

1. dict()
2. len()
3. clear()
4. get() ⇒ d.get(key) ⇒ No error
5. d.get(key, defaultvalue)
6. pop(): ⇒ d.pop(key)
7. popitem() ⇒ It removes an arbitrary item (key-value) d.popitem() from the dictionary and returns it.
8. keys() ⇒ d.keys()
9. value() ⇒ d.values()
10. items() ⇒ returns list of tuples representing key-value pairs.
11. copy() ⇒ d.copy()
12. setdefault(): ⇒
13. update() ⇒ d.update(x)

FUNCTION

→ If a group of statements is repeatedly required then it is not recommended to write these statements everything seperately. we have to define these statements as a single unit and we can call that unit any no. of times based on our requirement without rewriting. This unit is nothing but function.

adv ⇒ code reusability.

→ Python supports 2 types of functions
  i) Built in functions.
  ii) User Defined function.

parameters

→ it is i/p to the function.

Return statement ⇒

function can take i/p values as parameters & executes business logic, & returns o/p to the caller with return statements.

Types of Arguments

① positional Arguments
② keyword Arguments
④ Default Arguments
⑤ Variable Length Arguments.

⇒ first we have to take positional arguments & then keyword arguments.

⇒ After default arguments we should not take non default arguments.

global keyword ⇒ i) To declare global variable inside function.

ii) To make global variable to the function so that we can perform required modification.

1) Global Variables

The variable which are declared outside of function are called global variables.

2) Local Variables :-

The variables which are declared inside a fcn are called local variables.

Anonymous functions

→ We can declare a function without any name, such type of nameless functions are called anonymous funcn or lambda funct^n.

→ it's main purpose is just for instant use (for one time usage)

```
lambda  argument_list : expression
```

① filter() ⇒

use to filter values from the given sequence based on some condn.

```
filter (function, sequence)
```

② map() ⇒

→ for every element present in the given sequence, apply some functionality and generate new element with the required modificatn. for this we should go for map() function.

```
map( function , sequence)
```

③ reduce() ⇒

→ It reduces sequence of elements into a single element by applying the specified function.

⇒ reduce (function, sequence)

MODULES

⇒ A group of functions, variables and classes saved to a file, which is nothing but module.

⇒ every python file acts as a module.

⇒ whenever we are using a module in our program, for that module compiled file will be generated and stored in the hard disk permanently.

⇒ dir() ⇒ To list out all members of current module.

The Special Variable __name__ :-

by using this __name__ variable we can identify whether the program executed directly or as a module.

PACKAGES

→ It is an encapsulation mechanism to group related modules into a single unit.

→ package is nothing but folder or directory which represents collection of python modules

→ Any folder or directory contains __init__.py file is considered as a python package. this file can be empty.