**PYTHON NOTES**

**About Python:**

-Python supports cross platform development .

-It was designed by guido van Rossum in 1991

-It was used for web development , data base system , software development .

**History of python:**

-It was invented in Netherlands early 90,s by van Rossum

-First version of python is released in February 1991 at cwi(centrum wiskunde and information)

-Python was derived from ABC Programming language

**Why Python:**

-easy to learn

-easy to read

-easy to maintain

-a broad standard library

-portable

**Disadvantages:**

-speed

-Not to use in mobile development

-memory consumption

-runtime error

**Python versions:**

-version 0 renamed as 0.9.0 in 1991

-version 1.0 jan 1994

-version 2.0 sep 2006

-version 3.0 dec 2008

-latest 3.7

**Comments:**

-Single line comments #-------

-Multi line comments #---------------

-----------------------#

**Indentation:**

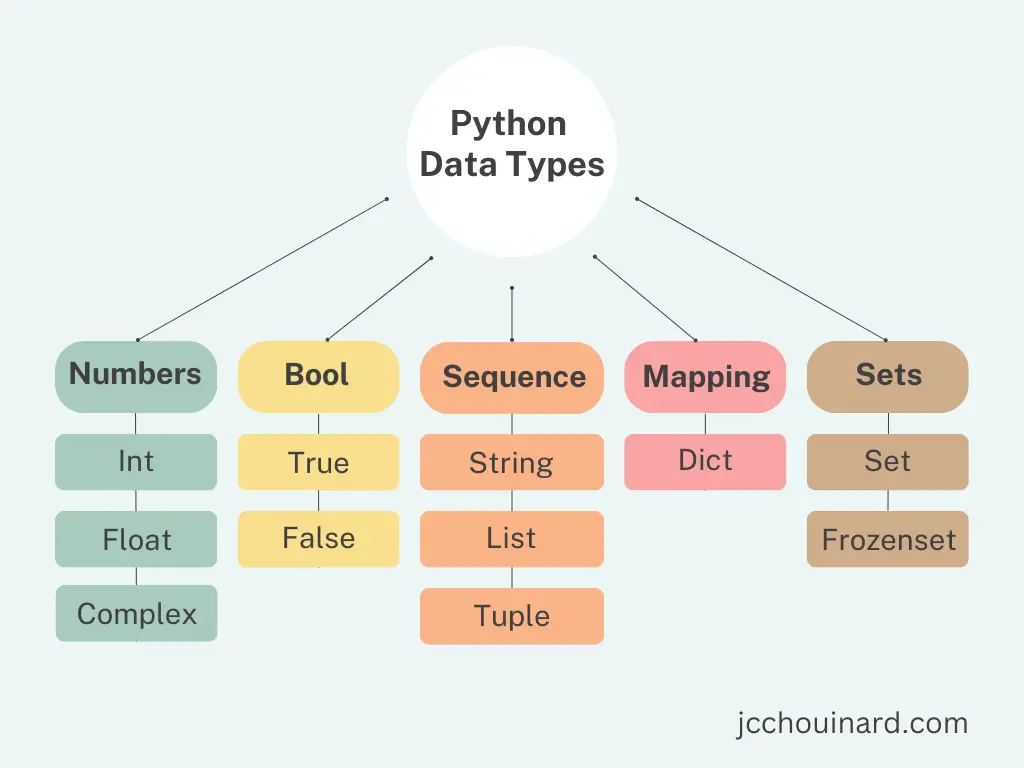
-Represent a block of statement with specific symbol (:)

**Identifiers:**

-Is a name given to a variable.

**Constants:**

-These are the fixed values which cannot be changed

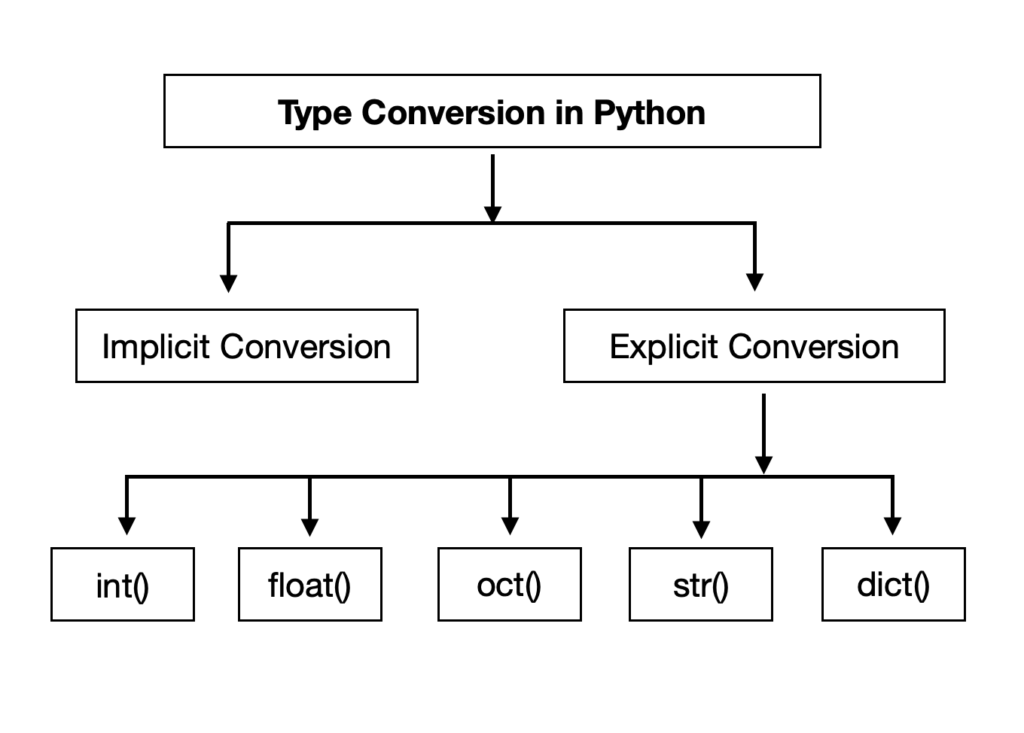
**Data Types:**

**Primitive:** integer , float , String , Bool , complex

**Derived :** List , Set , Dictionary , tuple

Note: Frozen set : Once we given data we cannot modify the data it will freeze the data.

**Type conversions:**



**Python Operators**:

-Arithmetic Operators(+,\*,-,/,//,%,\*\*)

/->Return quotient with float value.

//->Retun The Quotient in integer value.

\*\*->exponentiation

-Relational operator (<,>,<=,>=,==,!=)

-Assignment operator(=,+=,-=.\*=,/=,%=,//=,\*\*=)

-Bitwise operator (&,|,^,<<,>>)

-Membership operator (in , not in)

**Note:**

-1 bite = 8 bits

-1kb =1024 bytes

-1 mb = 1024kb

-1 gb = 1024mb

**Conditional statements**:

-if statement

-if – else statement

-elif statement

-Nested-if Statement

-Switch case in python.

**Control Statement**:

-for loop: for i in range(1,10):

-while loop: while condition:

**Loop Control Statements:**

-Break Statement

-Continue Statement

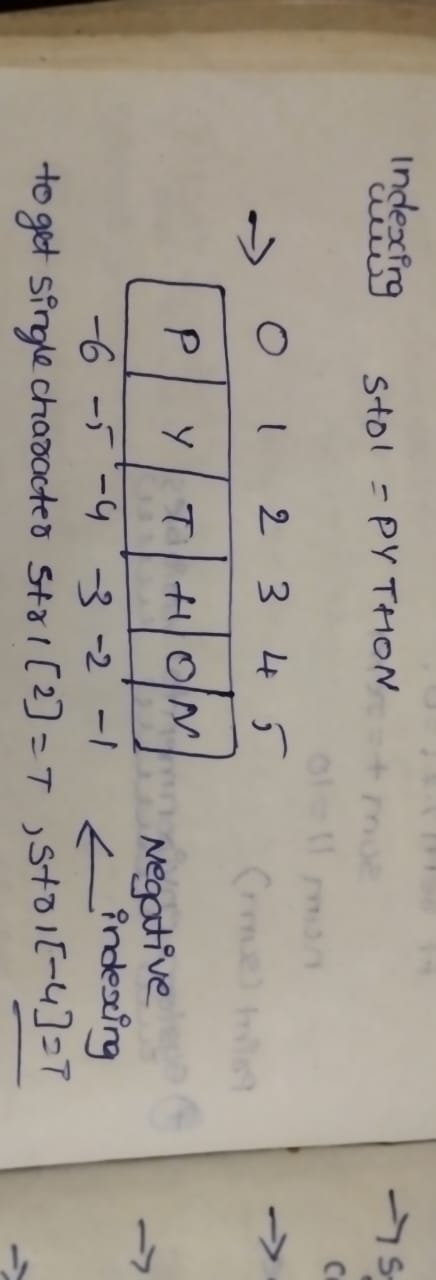
-Pass Statement

**String Manipulation:**

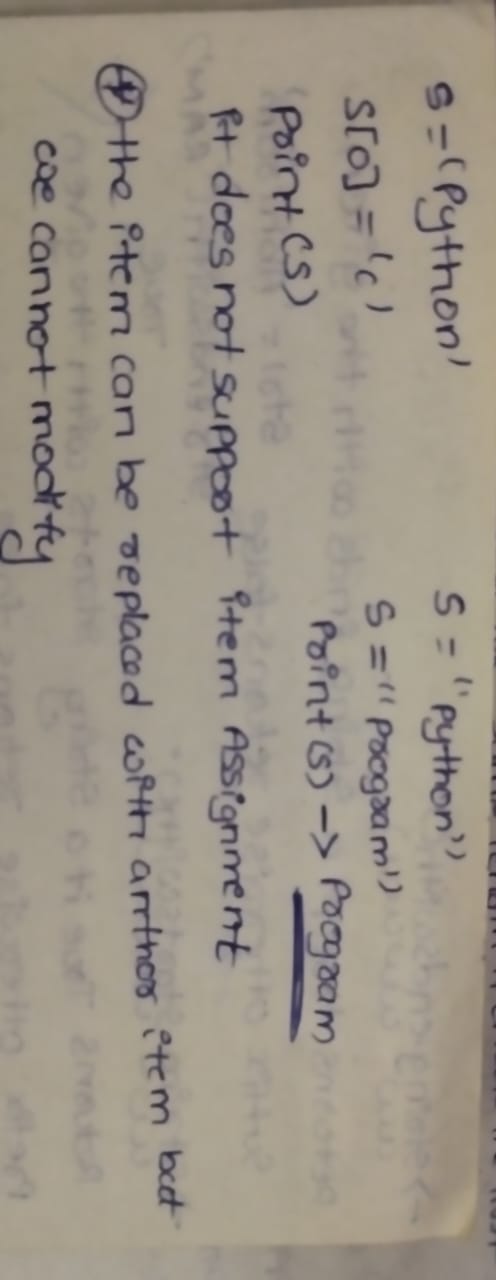
-it is a collection of characters

-the data is completely immutable

-A String literal are surrounded by either single quotation or double quotes

**Indexing in String:**

-String have both positive and negative indexing



**String Modify:**

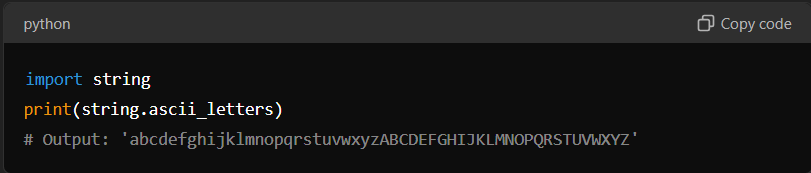
-Strings are immutable hence elements of a string cannot be changed once it has been assigned

-only new string can be re assigned to the same one

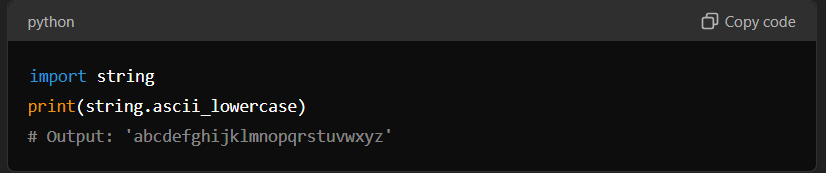
String Methods and Functions:

**String Constants:**

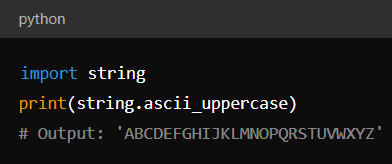
**-string.ascii\_letters**: Returns a string containing all ASCII letters (both lowercase and uppercase).



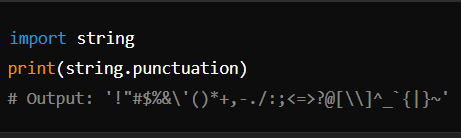
**-string.ascii\_lowercase**: Returns a string containing all ASCII lowercase letters.



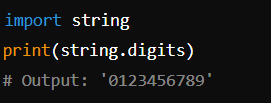
-**string.ascii\_uppercase**: Returns a string containing all ASCII uppercase letters.



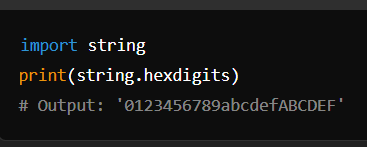
**-string.punctuation:**Returns a string containing all punctuation characters.



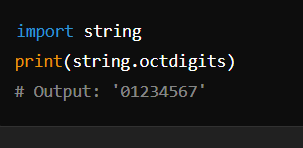
**-string.digits**: Returns a string containing all digit characters.



**-string.hexdigits** :Returns a string containing all hexadecimal digits (0-9 and a-f).



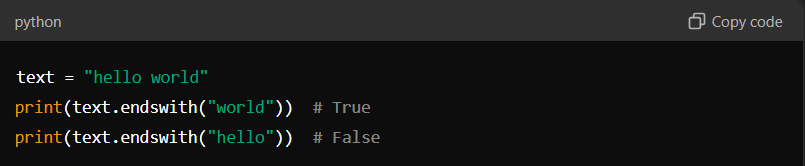
**-string.octdigits**: Returns a string containing all octal digits (0-7).



**String Methods:**

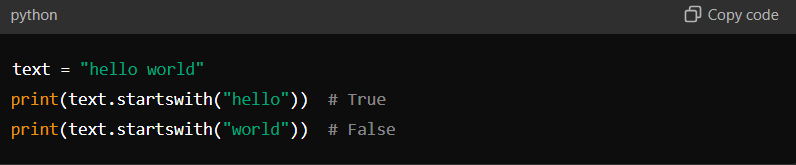
-**str.endswith(suffix[, start[, end]])**

* Returns True if the string ends with the specified suffix; otherwise, False.



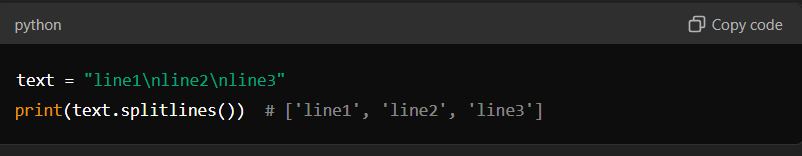
-**str.startswith(prefix[, start[, end]])**

* Returns True if the string starts with the specified prefix; otherwise, False.



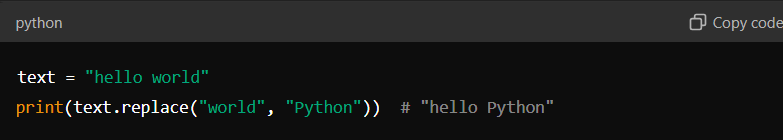
-**str.splitlines([keepends])**

* Returns a list of lines in the string, breaking at line boundaries.



-**str.replace(old, new[, count])**

* Returns a copy of the string with all occurrences of the substring old replaced by new.



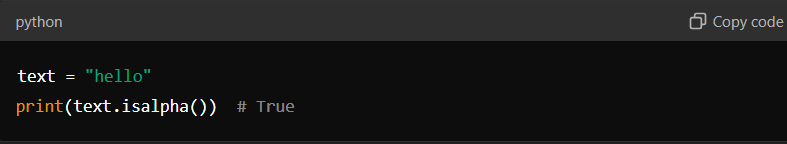
-**str.isdigit()**

* Returns True if all characters in the string are digits.



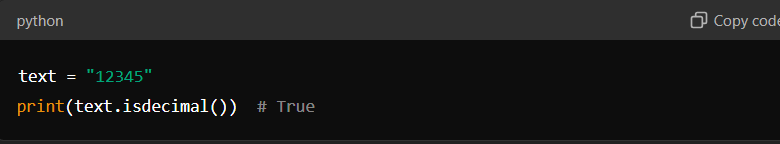
-**str.isalpha()**

* Returns True if all characters in the string are alphabetic.



-**str.isdecimal()**

* Returns True if all characters in the string are decimal characters.



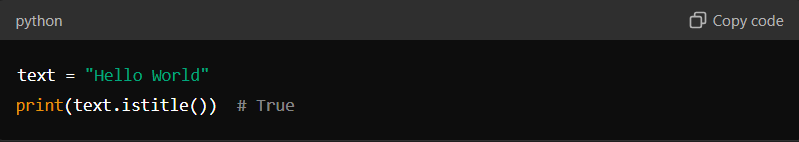
-**str.isalnum()**

* Returns True if all characters in the string are alphanumeric (letters and numbers).



-**str.istitle()**

* Returns True if the string is a titlecased string (each word starts with an uppercase letter and the rest are lowercase).



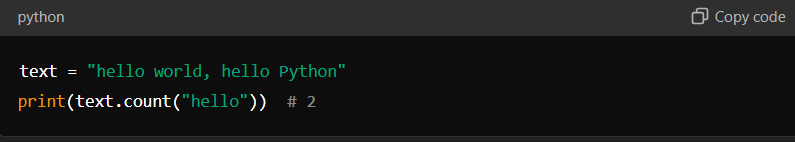
-**str.upper()**

* Returns a copy of the string with all characters in uppercase.



-**str.count(sub[, start[, end]])**

* Returns the number of non-overlapping occurrences of substring sub in the string.



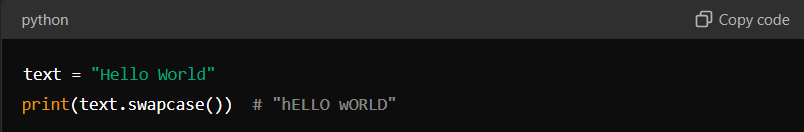
-**str.lower()**

* Returns a copy of the string with all characters in lowercase.



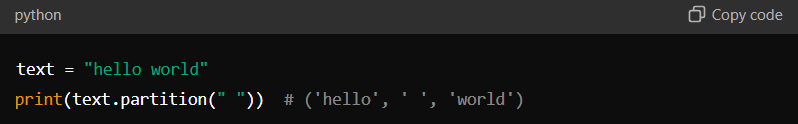
-**str.swapcase()**

* Returns a copy of the string with uppercase characters converted to lowercase and vice versa.



**str.partition(separator)**

* Splits the string into a 3-tuple (head, separator, tail) where separator is the first occurrence of the separator.



-**str.index(sub[, start[, end]])**

* Returns the index of the first occurrence of the substring sub. Raises ValueError if not found.



-**str.find(sub[, start[, end]])**

* Returns the index of the first occurrence of the substring sub. Returns -1 if not found.



-**str.rfind(sub[, start[, end]])**

* Returns the index of the last occurrence of the substring sub. Returns -1 if not found.

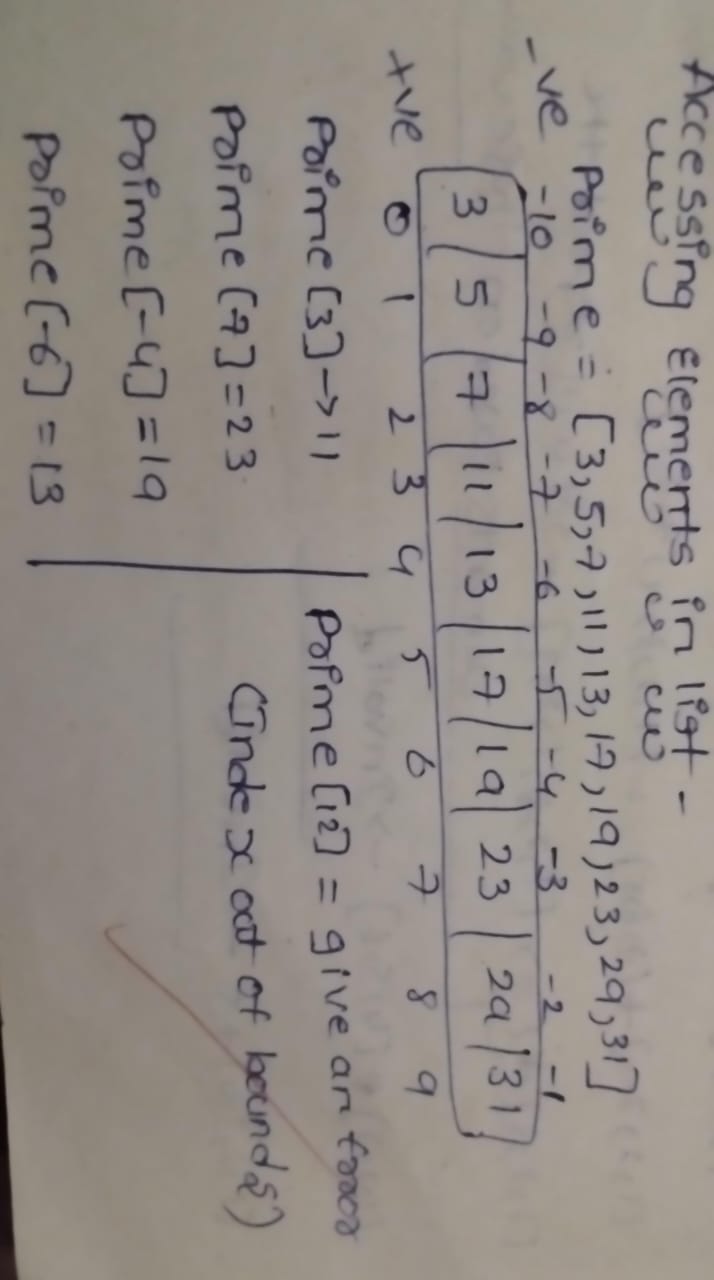


**List:**

-List is a ordered collection of similar (or) different data elements

-it supports both +ve(Start from 0) & -ve(Start from -1) indexing

-List is mutable in nature

-Where each element is separated by ‘,’.

Empty List:

Mylist =[]

Lst = List()

**Syntax:**

Mylist = [101,’ABC’,’IT’,True]

-List allows duplicate elements

**Accessing elements in a List**:

Slicing in List:

Prime[start:Stop:Steps]

Start – inclusive(Start from here)

Stop – Exclusive (We get output before stop value)

-All 3 are optional

**Default values:**

Start – 0

Stop:Length of list

Stepsize:1

Ex:

Prime[2:6] =>[7,11,13,17]

Prime[:5] => [3,5,7,11]

Prime[2:8:2]=>[7,13,19]

Prime[2:8:-1]=>[23,19,17,13,11,17]

**Build in functions in List:**

**-type() :** return type of an object

Ex:type(prime)

-len()-returns the length of an object in integer format

Ex: len(prime)

-max() – it returns maximum element from list

Ex: max(prime)

-min() – it returns minimum element from the list

Ex: min(prime)

-sum() – return sum of all numericals

Ex: sum(prime)

-sorted() – returns a list in sorted order without effecting the original list.

Ex:sorted(prime)

**List Methods:**

-.append() - it takes one argument as element and append at the end of the list.

Ex:prime.append(10)

-.count() - returns no.of occurrence of given element in the list

Ex: prime.count(3)

-.index() – it returns the index of give element at first occurrence

-.extend() – it extends the original list with another list

-.remove() – removes an element from the list

Note: if the element is not in list it raise error

-.pop() – it removes an element from the list and return it

Ex: prime.pop(index)

-.sort() – it performs sorting on given list in ascending order

Ex: prime.sort()

Prime.sort(rev = True) //descending order

-.insert() – insert element at specified index

Ex: prime.index(1,4) 1-index 4-value

-.copy() – it returns a copy of list

-.reverse() – it returns the given list

Ex: prime.reverse()

-.clear() – It clear all the elements from the list

Ex: prime.clear()

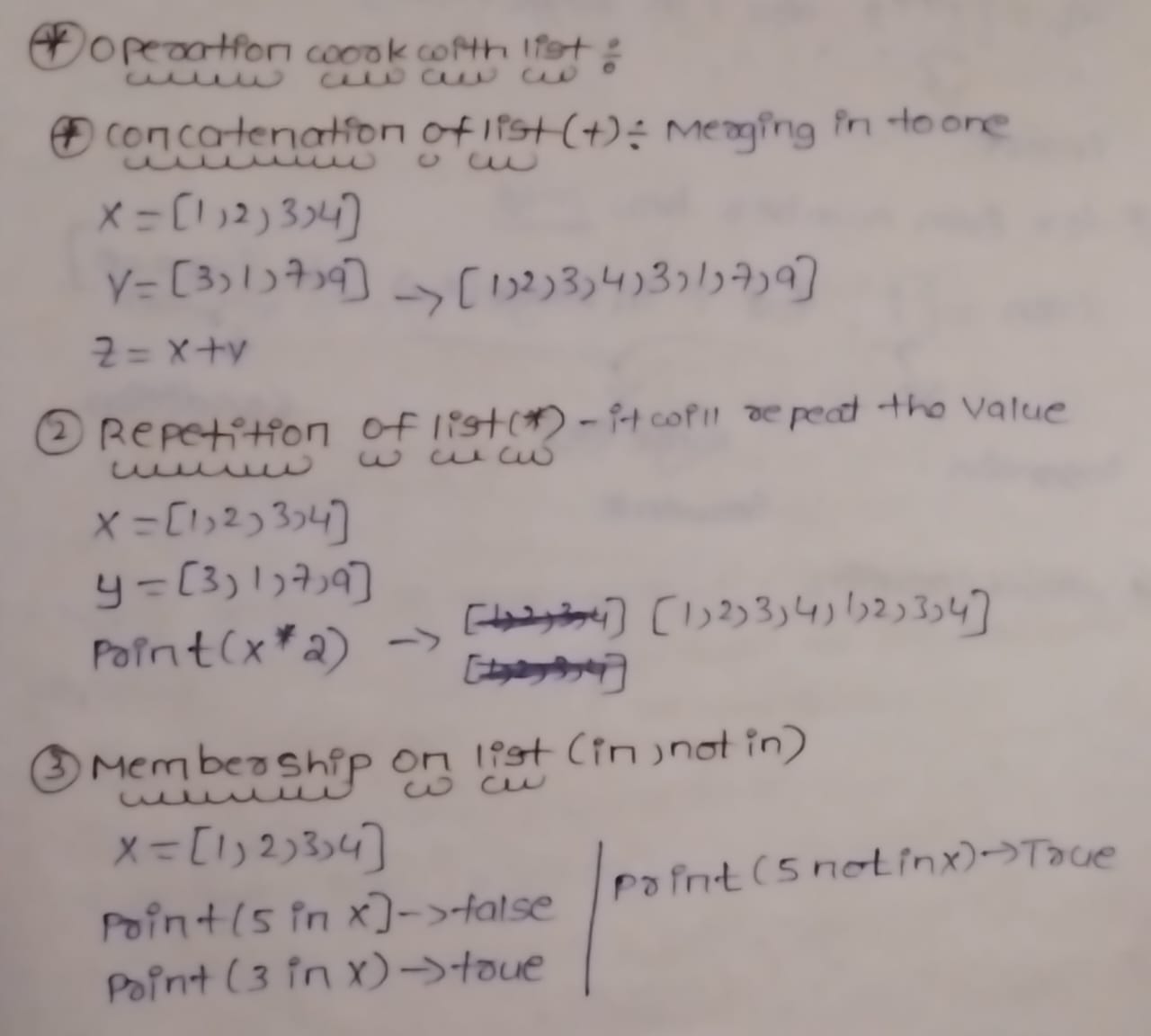
**Creating list from String:**

Mystr = “Hello”

Print(list(mystr))

**List Comprehension:**

Even = [ i for i in range(2,10) if i%2 == 0]



**Tuple:**

-it is a ordered collection of similar (or) different data elements

-support both +ve & -ve

-can be created using () , where each element is separated by ‘,’

-it is immutable

Empty Tuple:

T =()

T =tuple()

**Tuple Methods:**

-count() - returns no.of occurrence of given element in the list

-index()- it returns the index of give element at first occurrence

**Note:**

If tuple has only one value we need to keep , .

**Functions:**

-sum()

-max()

-min()

-all() – Returns true if all elements are true or if tuple is empty

-any() – Return true if any element of the tuple is true , else return false

-len()

(()) ->empty tuple

((),)->A tuple contains empty tuple

**Enumeration:**

For I,n in enumerate(prime):

Print(i,n)

Taking Input Dynamically:

Tup =int(input(“enter”))

Tuple Unpacking:

Mytuple = (2,4,6,8)

T1,t2,t3,t4 = mytuple

**Sets:**

-A Set is an unordered collection of elements

-No sequential order

-sets are mutable

-set doesn’t support indexing

-set doesn’t allow duplicates to store in it

-set can be created using curley braces {}

**Empty Set:**

S =set()

S={}-> it become a dictionary

-Elements are stored based on hashing value

-we use sets to eliminate duplicate value.

-in set there is no nested set

-a tuple can have list inside it A List have Tuple inside it

{[]} -not supported

{()} – Accepted

{“”}-Accepted

{{}} – not supported

-While storing an element in a set its hash value is computed using a hashing technique to determine where it should be stored in the set.

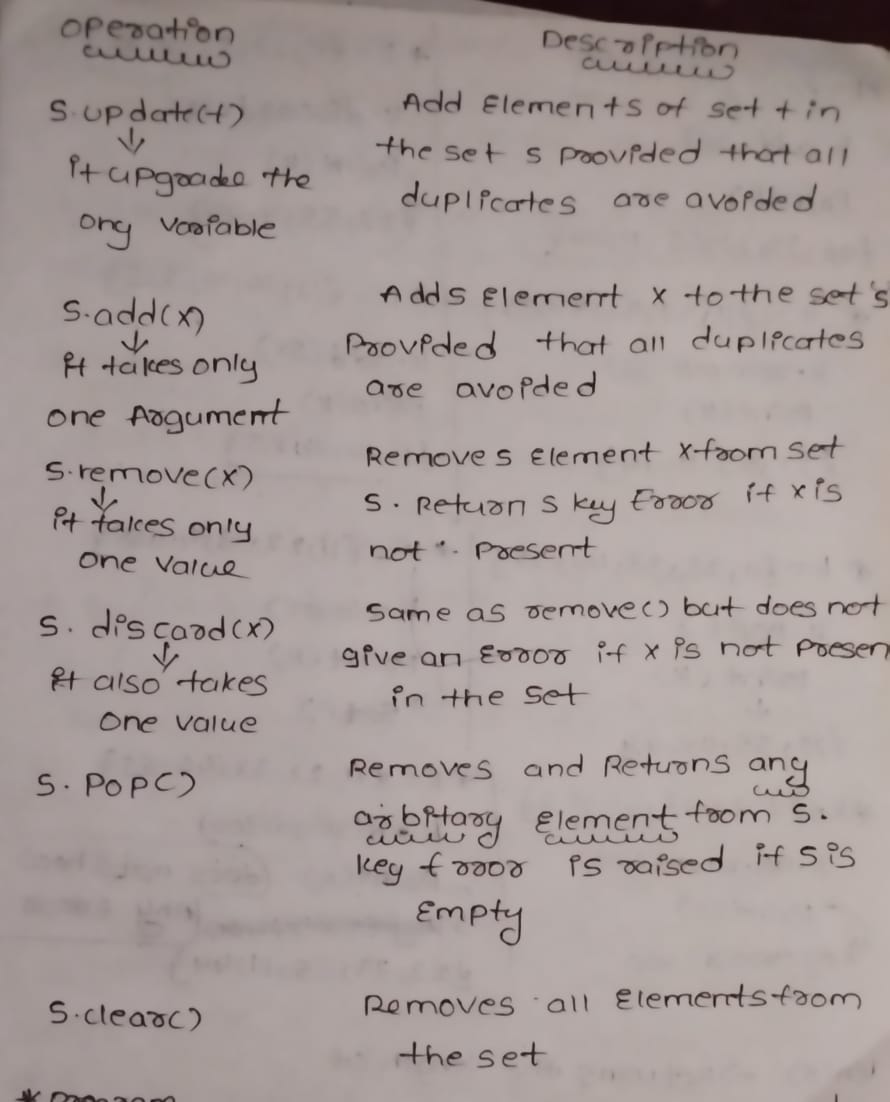
**How to Access Set Elements:**

-By using name of the set

-set cannot be sliced using []

**Looping in set:**

* We can iterate using a for loop
* We can also use enumerate() to loop



-.add() – add an element

-s.update(s1) – update a existing set with new set

-.remove(element) – remove an element from the set

-.pop() – delete the last element

-.discard(element) – remove the element from the set

-.clear() – empty the set

-x in s -> returns true if x is present in set s else false

-x not in s -> returns true if x is not present in set s else false

-s.subset(t) -> returns true if every element in set s is present in set t else false

-s.issuperset(t) -> return true if every element in t is present in set s else false

-s.union(t) -> Combine the two set s & t in to one

-s,intersection(t) - >Returns the comman term in both

-s.intersection\_update(t) – returns a set that has elements which are common to both the set s and t and update to s .

-s.difference(t) – returns a new set that has element in set s but not in t

-s.difference\_update(t) -> update the elements in to s.

**Dictionary:**

-Collection of key value pairs

-each value is associated with key

-it is mutable

-it doesn’t support indexing

-values can be accessible with keys

-keys are unique but values are not

**Create a empty dictionary:**

D1=dict()

D2={}

-Different keys may have same values

-if keys are same but values are different latest key value pair gets stored

**Accessing Values:**

-We can print entire dictionary using name

-Elements are not accessed using the position

-Elements can be Accessed by using Keys

-Dictionaries can not be sliced using []

**Loop in Dictionaries:**

For k,v in d.items():

Print(k,v)

**Adding Keys and values:**

-you can add a new key value pair to a dictionary by using the

Dictionaryname[key] = value

**Deleting Keys :**

-to delete entire dictionary we use

Del dic\_name

-to delete a keys based

Del dic[key]

Ex:

d = {‘ajay’:2,’kumar’:3}

del(d[‘ajay’])

d.clear() -> it delete all elements and print empty dictionary{}

del(d)-> it delete all elements and shoe d is not defined

**Sorting :**

-Sorted(d.keys())

-use list to store a collection of data that doesnot need random access

-use list if the data has to be modifies frequently

-use a set if you want to ensure that every element in the data structure is unique

-use tuples when you want your data to not modify

-use dictionary when we want to store data in key value pair form.

**DATA ANALYSIS WITH PYTHON**

**Numpy:**

**-**Numerical Python

-it is not a built in package

-it is a third party package

How to Install:

-go to cmd

>pip install <package\_name>

Ex: pip install numpy

>pip show numpy

-import numpy as np

-numpy Arrays mainly used in scientific application because it is efficient to perform mathematical calculations

-numpy arrays improves time complexity, it will take less time when compared to normal list

Refer notes.