Python

General python:

For binary: try to calculate with 2’s power then sees the sum of it, how much you are getting to that decides the number of bits, and likewise you can then decide the 1 and 0’s.

Bin of (111) = 1 \* 2^2 + 1 \* 2^1 + 1 \* 2\*0

Hashable: objects which can be hashed, int, str, tuple (with unique values)

What is hash= unique and fixed (no change), immutable objects are hashable, whereas mutable are not hashable.

Mutable in python: List, dict, sets etc.

Immutable: int, float, str, tuple, frozen set etc.

Comprehensions: Runs the code from left to right, i.e. the left one is outer one and right one is inner one and likewise the hierarchy in the loops. Left to right, same is the case for the conditions, left to right condition after the loop is condition for that loop. To simply the evaluation just tries the logic with left loop as outer loop and first immediate condition as outer condition right loop as in inner one and right condition after that is the inner loops condition.

What is hexadecimal representation?

* It is the representation with 0-9 and then A-F, use int(‘FF’,16) for int conversion like you do for the int(binary,2)

Lexicographical sorting: sorting based of the ascii values.

Sorted ([‘Harshal’, ‘Nayan’]), sorting start based off comparing from left, if unmatched found then it stops there itself. Same for the numbers.

Same for the comparison, i.e. “Harshal” < “Nayan”, same lexicographical comparison, start form the left then stops as soon as the inequality is matched.

**Module, package and library?**

A **module** is a single file containing functions, classes, and variables; a **package** is a directory with multiple related modules and an \_\_init\_\_.py file to organize them; and a **library** is a collection of packages and modules bundled together to provide broader reusable functionality across projects. Modules form the basic building blocks, packages group these modules, and libraries encompass multiple packages for larger-scale use.

**Modules:** math, random, os, sys, re, time, Json, shutil, queue, pprint

**Packages:**  
flask, django, numpy, pandas, scipy, matplotlib, sklearn, urllib, xml, tkinter

**Libraries:**  
NumPy, Pandas, Matplotlib, Scikit-learn, TensorFlow, Requests, Flask, Django, BeautifulSoup, Pygame, SciPy, Seaborn

e.g. from itertools import combinations

from itertools import groupby

or itertools.combinations() or itertools.groupby()

both ways works, meaning you can access the classes and function in the module just like the methods of any object instance.

**OOPS:**

Special methods (Dunder methods):

Methods which have special ability, which gets externally called based off the conditions specified mostly externally, the logic is written for the special condition in terms of new method, and as the condition is mate the same method is called.

Example, add, write the special method inside the class with \_\_add\_\_ where it takes one set of input from the current (object) from the future, and the other input from the other object where the other object must be passed to the special method like the self. The special condition here would be the use of + in the object1 and object2. Likewise other methods are defined many are specific to single object whereas the other are interactive. E.g. \_\_len\_\_, \_\_str\_\_ etc. here \_\_str\_\_ get called on the condition of the using str or print on the object.

Whenever you create attributes from outside those just get sorted as it is, without any self and not in the constructor or anything.

NumPy:

Np.round (a,2) where second parameter describes up to what digit to round off to.

Masking and indexing, both are used for accessing the specific elements of the array, indexing is based off the index values whereas the masking is based off the Boolean conditions (True/False) of the similar shape.

Np.tile – repeats the array in the specific directions.

Np.repeats- repeats the individual values multiple times.