Python:

1) features: High level programming
   Interpreter
   Dynamic typing
   Extensive library & framework
   Cross platform compatibility – Python Virtual Machine (os Independent)

2. Standard libraries:
   Interact with OS: os, sys, platform
   os-specific modules

3. Virtual Envir.:
   Py -m venv .venv

4. GUI & web Develop.
   ↓ → Django & Flask
   Tkinter, PyQt, Kivy

5. Support object oriented programming (composition & Inheritance)

6. Function = first class object
   assign to variable, return for other method,
   Pass them as arguments

7. List: Sequence of item – Mutable, More Memory Slower iteration
   Tuple: Immutable
   Dict: assoviate pairs of two item

List	Tuple
More memore consume	Less Memory consume
Slow iteration	fast iteration
Better – insert & delete	
Various Built in Method	Limited built in Method

Mutable datatype

Modified After creation

List, Dict, sets

Elements added/removed/changed

Immutable Datatype

cannot Modified After creation

Numeric (int, float)
String, tuple

CODING CHALLENGE

→ List Comprehension:

one liner syntax

```
Lst = [i for i in range(1,10)]
Dict = {i:i**2 for i in range(1,10)}
Tup = (i for i in range(5))  — generator Not tuple
    = tuple(i for i in range(5))
```

Generator

type: <class 'generator'>

Lazy Evaluation

Generate values at one time

Mutable

faster for large dataset

produces values on demand

Tuple

<class 'tuple'>

Store all element in memory

→ Negative Index : List[-1], List[-2]

→ How to modify string :?

```
Str_val = "Hellow"
Str_val[0] = "K"  → Error
```

soln. convert to list & modify

→ "".join(s) → Join 's' with "No separator"

Ex: s = ["H", "E", "L", "L", "O"]

"".join(s) = "Hello"

Separator.join(iterable)

Eg2: dt = ["2024", "2", "1"]

"-".join(dt) o/p => 2024-2-1

→ Swap 2 variable : a, b = b, a
(without temp variable)

→ Print ("Even" if num % 2 == 0 else "odd")

→ Input ("Enter value")
int (Input ("enter value"))

→ Largest Number in list : max (list)

→ Reverse string : s[::-1]

→ Anagram :

s1 = "listen"
s2 = "silent"

sorted (s1) == sorted (s2)

→ Recursive function : Factorial

Return 1 if n==0 else n * fun(n-1)

→ count frequency

from collections import counter

dict (counter (str))

→ 2nd Largest : list.remove (max (list))
Print => max (list)

→ Merge 2 dictionary :

Dict = {**dict1, **dict2} — unpack key-value pair
— merge two dict

dict1.update(dict2) — duplicate key : dict2 will overwrite dict1

dict1 | dict2

→ fibonacci :

list(func(n))

```
a, b = 0, 1
for _ in range(n)
    yield a
    a, b = b, a+b
```

turns function into generator

Pass execution & returns value

don't store value in memory

```
gen = fibonacci(5)
print(next(gen))   o/p = 0
"      "     "     o/p = 1
"      "     "     o/p = 1
"      "     "     o/p = 2
```

⋮

→ All prime nos. in a range

```
[num for num in range(2,n) if all(num%i!=0
    for i in range(2,int(num**0.5)+1))]
```

→ Important Built in function :

a) map → apply func. to each element of iterable

```
list(map(lambda x: x**2, [1,2,3]))
```

b) filter : filter on iterable based on func. condition

```
list(filter(lambda x: x%2==0, [1,2,3,4]))
```

→ reduce() :

from functools import reduce

Applies function cummutatively to iterable

numbers = [1, 2, 3, 4, 5]

reduce (lambda x, y : x * y , numbers)

→ Zip() → combine 2 list

list (zip(list1, list2))

list1 = ("Kunan", "Ram")

list 2 = (20, 30)

list(Zip (list1, list2))

⇓ o/p

[("Kunan", 20), ("Ram", 30)]

→ enumerate

add an Index to iterable

for Index, Element in (list, start = 1) :

print (Index, Element)

→ sorted (list)

sorted (list, Reverse = True)

→ all() → all element are true

any() → any of the element are true

→ Min() , max() , sum()

→ list (reversed (list))

→ type