2(a) How are dictionaries different from lists in Python?

• Lists store ordered collections of elements accessed by index.

• Dictionaries store key–value pairs accessed by keys (unordered before Python 3.7, insertion ordered from 3.7+).

Example:

List: [10, 20, 30] → list[1] = 20

Dictionary: {'a': 10, 'b': 20} → dict['b'] = 20

2(b) What is a lambda function? How is it different from a regular function?

A lambda is an anonymous, single■expression function defined using the lambda keyword.

Differences:

• lambda: single expression, no name, inline

• def function: can have multiple statements, name, docstring

Example:

lambda x: x*x (square of x)

2(c) What is string immutability in Python?

Strings cannot be modified after creation. Any operation creates a new string.

Example:

s = "hello"; s[0] = 'H' → Error

s = "Hello" (new string created)

2(d) Slice 'PythonProgramming' to get 'thonPro'.

text = 'PythonProgramming'

text[2:8] → 'thonPr' (to include 'o', use)

text[2:9] → 'thonPro'

2(e) What does *args do in a function?

*args allows passing variable number of positional arguments into a function.

Example:

def add(*args): return sum(args)

add(1,2,3) → 6

22(a) Using sorted() and sort() with lists:

• sort() sorts a list in place and returns None.

• sorted() returns a new sorted list, original unchanged.

Example:

a = [3,1,2]; sorted(a) → [1,2,3]; a.sort() → a becomes [1,2,3]

22(b) How to get a random number in Python?

Use random module:
random.randint(1,10) → integer 1–10
random.random() → float 0–1

22(c) What are map and reduce functions?
map(): applies a function to each iterable element and returns an iterator.
reduce(): applies function cumulatively to elements to reduce to single value (in functools).
Example:
map(str.upper, ['a','b']) → ['A','B']

22(d) Difference between reshape() and resize():
reshape(): returns new reshaped array without changing original (if possible).
resize(): changes array in place and may change its size.

22(e) Difference between pivot table and crosstab (pandas):
Pivot table:
• Summarizes data with aggregation using index/columns/values
Crosstab:
• Computes frequency between two or more factors (contingency table)
Example:
pd.pivot_table(df, values='sales', index='city', columns='year')
pd.crosstab(df['city'], df['year'])