Assignment – Unit 02

Collections and Functions

(Lists, Tuples, Sets, Dictionaries, Functions)

Instructor: Tofik Ali **Submission Deadline:** March 31, 2026

Student Name: ___________________ **Roll No.:** _________

Instructions

- Answer **all** questions.
- There is **no time limit**. Submit on or before **March 31, 2026**.
- Write **clean code**: meaningful variable names, functions, and comments where needed.
- Handle basic edge cases (empty input, invalid input types, etc.) and write assumptions.
- Unless a question says otherwise, do **not** use external libraries.

Easy (10 Questions)

E1. **List basics:** Given the list `a = [10, 20, 30, 40, 50, 60, 70]` write Python expressions for:

(a) the last 3 elements

(b) all elements except the first and last

(c) every 2nd element starting from index 0

(d) the reversed list (using slicing)

E2. **List methods:** Using a list `nums = [5, 1, 5, 2, 9, 2]`, write code to:

(a) append 7

(b) remove the first occurrence of 5

(c) sort in ascending order

(d) print how many times 2 occurs

E3. **Tuple unpacking:** Let `t = (ÜPES; ČSE; 2026)`. Unpack it into three variables. Then print: `CSE - UPES (2026)`.

E4. **Set basics:** Given two sets `s1 = {1,2,3,4}` and `s2 = {3,4,5,6}` write code to print: union, intersection, `s1 - s2`, and `s2 - s1`.

E5. **Functions:** Write a function `area_rectangle(length, width=1)` that returns the area. Demonstrate with at least two calls: one using default width, one using a keyword argument.

E6. **List comprehension (strings):** Given:

```
names = ["Aman", "Bina", "Chetan", "Divya"]
```

create:

(a) a list of uppercased names

(b) a list of name lengths

   (c) a list of (name, length) tuples

E7. **Tuple slicing + membership:** Given `t = (11, 22, 33, 44, 55)` write expressions to:

   (a) get the middle three elements

   (b) check whether `33` exists in the tuple

   (c) create a new tuple with `99` added at the end

E8. **Dictionary safe access:** Given `marks = {"Python":  95, "Math":  88}`, write code to:

   (a) print Python marks using indexing

   (b) print Physics marks using `get` with default value 0

   (c) update Math marks to 90

E9. **Unique words using a set:** Read a sentence from the user, split into words, and:

   (a) print the number of unique words

   (b) print the unique words in sorted order (as a list)

E10. **Return multiple values (tuple):** Write a function `min_max(nums)` that returns a tuple `(min_value, max_value)` for a list of numbers. Demonstrate with a sample list.

Medium (10 Questions)

M1. **List comprehension (if–else):** Given a list of integers, create a new list such that:

- if a number is even, store its square
- else, store its cube

Do it using a **single** list comprehension and show a sample run.

M2. **Dictionary frequency:** Write a function `word_freq(sentence)` that returns a dictionary of word frequencies (case-insensitive). Ignore leading/trailing spaces and treat multiple spaces as one separator. Example input: `P̈ython is fun and Python is powerful¨`.

M3. **Tuple + list processing:** You are given a list of (name, marks) tuples: `students = [(Äman,̈ 78), (B̈ina,̈ 92), (C̈hetan,̈ 58), (D̈ivya,̈ 92)]`. Write code to:

   (a) find the highest marks

   (b) print the names of all toppers (if tie)

   (c) compute the average marks

M4. **Functions with variable arguments:** Create a function `stats(*nums)` that returns a tuple `(min, max, average)`. If no numbers are passed, return `None`.

M5. **Sorting dictionaries:** Given a dictionary of subjects to marks, print the subjects in decreasing order of marks. Example: `{M̈ath:̈  88, P̈ython:̈  95, P̈hysics:̈  72}`.

M6. **Comprehension with multiple results:** Given `nums = [1, 2, 3, 4, 5]`, create a list of tuples `(n, n*n, n*n*n)` using a single list comprehension.

M7. **Create a dictionary using zip:** Given `subjects = ["Python", "Math", "English"]` and `scores = [85, 78, 69]`, create a dictionary and compute total and average.

M8. **Remove duplicates (preserve order):** Write a function named `unique_preserve_order(items)` that returns a new list with duplicates removed **without** changing the first occurrence order. Example: `[1,2,1,3,2]` → `[1,2,3]`.

M9. **Avoid mutation:** Write a function `add_bonus(marks, bonus)` that returns a **new** list with each value increased by bonus. Show that the original list does not change.

M10. **Recursion:** Write a recursive function `sum_to_n(n)` that returns `1+2+...+n`. Include a correct base case and test with `n=5`.

Hard (10 Questions)

H1. **Mini gradebook (lists + dicts + functions):** Maintain student marks for multiple subjects. Use the data structure:

```
records = [
  {"name": "Aman", "marks": {"Python": 85, "Math": 78, "English": 69}},
  {"name": "Bina", "marks": {"Python": 92, "Math": 88, "English": 74}},
  {"name": "Chetan", "marks": {"Python": 58, "Math": 61, "English": 55}},
]
```

Write functions to:

(a) compute each student's total and average

(b) find the top student by total

(c) list students who are failing in any one subject (threshold 60)

H2. **Merge dictionaries (numeric values):** Write a function `merge_sum(d1, d2)` that returns a new dictionary:

- keys present in either dictionary should appear once
- if a key exists in both, sum the values
- do not modify the original dictionaries

H3. **2D list (matrix) tasks:** For a matrix represented as a nested list, write code to compute:

(a) the transpose

(b) sum of each row

(c) sum of each column

Use a **nested list comprehension** for at least one part.

H4. **Recursion (with explanation):** Write a recursive function `flatten(lst)` that converts a nested list (e.g., `[1, [2, 3], [4, [5]]]`) into a flat list `[1,2,3,4,5]`. Write 4–6 lines explaining the base case and recursive case.

H5. **Functional patterns:** Given a list of integers `nums`, do the following without writing an explicit `for` loop:

(a) filter only positive numbers

(b) map them to their squares

(c) compute the sum of squares

Use `filter`, `map`, and `sum`. (Optional: show an equivalent list-comprehension solution.)

H6. **Top-K frequent words:** Write a function `top_k_frequent(words, k)` that returns the top-k most frequent words from a list of words. Use a dictionary for counting and sort by frequency. Demonstrate with a sample sentence you choose.

H7. **Deep copy of a 2D list:** Write a function `deep_copy_2d(matrix)` that returns a new 2D list such that modifying the copy does not affect the original. Demonstrate with an example.

H8. **Function composition:** Write a function `compose(f, g)` that returns a new function `h(x)=f(g(x))`. Demonstrate using two simple functions (e.g., `double` and `add_3`).

H9. **Balanced parentheses (stack):** Write a function `is_balanced(s)` that returns `True` if parentheses are balanced for strings containing only `(` and `)`. Test your function with at least 3 cases.

H10. **Group anagrams:** Given a list of words, group them into anagrams. Example input (order can vary): `["eat", "tea", "tan", "ate", "nat", "bat"]`
One possible grouped output: `["eat", "tea", "ate"]`, `["tan", "nat"]`, `["bat"]`.
Implement using a dictionary where the key is a canonical form (e.g., sorted letters).

End of Assignment