Week 1 - Lesson 2

Functions + Comprehensions (Python)

Date: 24 Dec 2025

Lesson goals

• Write small functions with clear inputs/outputs (less copy-paste, easier debugging).

• Understand what changes (mutable like lists) vs what doesn't (immutable like ints/strings).

• Use comprehensions to build lists/dicts cleanly, without messy loops.

1) Functions: the basics

• Define with def, give parameters, return a value.

• Prefer returning values instead of printing inside functions.

```
def add(a, b):
    return a + b

result = add(2, 3)
print(result)  # 5
```

Common patterns

• **Pure function**: returns a result, no side effects (best for testing).

• **Side effect**: changes something outside (like appending to a list, writing a file).

• Use **default values** to make parameters optional.

```
def greet(name, ending="!"):
    return f"Hi {name}{ending}"

print(greet("Ali"))
print(greet("Ali", "!!"))
```

2) Mutability: why one example changed the list

Python uses **pass-by-assignment**: variables point to objects.

• Immutable objects (int, float, str, bool, tuple) cannot be changed in place.

• Mutable objects (list, dict, set) can be changed in place.

```
def f(x):
    x = x + 10   # makes a NEW int
    return x

a = 5
b = f(a)
print(a, b)  # 5 15
```

```
def add_one(lst):
    lst.append(1)  # changes the SAME list object

x = [5]
```

```
add_one(x)
print(x)  # [5, 1]
```

Rule of thumb: if you call .append/.pop/.update on a list/dict, you are mutating it.

3) Comprehensions: shorter, cleaner building

A list comprehension is a compact way to build a new list.

```
nums = [1, 2, 3, 4, 5]

# squares of even numbers
ev_sq = [n*n for n in nums if n % 2 == 0]
print(ev_sq)  # [4, 16]
```

Dict comprehension (great for quick mappings)

```
words = ["python", "java", "python", "go"]

# initialize keys with a default value
unique = {w: 0 for w in set(words)}
print(unique)  # {'python': 0, 'java': 0, 'go': 0}
```

Tip: If a comprehension becomes hard to read, use a normal loop.

4) Tiny cheatsheet

Task	Best tool
Build a result list	[expr for x in items if condition]
Need index + value	for i, x in enumerate(items): ...
Search first match	loop + if + break (or next(..., None))
Keep asking user	while True: validate -> break

Practice (recommended)

• Write a function clean_task(s) that returns None for empty/space-only strings, else returns the cleaned string.

• Write find_first(items, target) returning the index or None (use enumerate).

• Convert a filter loop into a list comprehension (example: keep only even numbers).

• In your CLI app, keep functions small: add_item(), list_items(), remove_item(), prompt_choice().

Next lesson: Files/JSON + Exceptions (so your CLI app can save/load safely).