Lesson 2: Tuples

1. **Definition**: A tuple is an ordered, immutable collection of items in Python, defined using `()` (optional) or the `tuple()` constructor. Example: `(1, 2, "hello")`.
2. **Creating Tuples**:
   - Empty: `()`
   - Single-item: `(5,)` (trailing comma required)
   - Mixed types: `(1, "text", 3.14)`
   - No parentheses: `1, 2, 3`
3. **Key Properties**:
   - Ordered (maintains insertion order)
   - Immutable (cannot modify after creation)
   - Indexed (starts at 0)
   - Allows duplicates
4. **Accessing Elements**:
   - Positive indexing: `t[0]`
   - Negative indexing: `t[-1]`
   - Slicing: `t[1:3]`
5. **Immutability**: Cannot change, add, or remove elements, but mutable objects inside (e.g., lists) can be modified. Example: `t = (1, [2, 3], 4)` → `t[1][0] = 5`.
6. **Methods**:
   - `len(t)`: Number of items
   - `count(item)`: Counts occurrences
   - `index(item)`: First index of item
7. **Operations**:
   - Concatenation: `(1, 2) + (3, 4)` → `(1, 2, 3, 4)`
   - Repetition: `(1,) * 3` → `(1, 1, 1)`
   - Membership: `2 in (1, 2, 3)` → `True`
8. **Tuple Unpacking**:
   - Basic: `x, y = (1, 2)`
   - With `*`: `a, b, *rest = (1, 2, 3, 4)` → `rest = [3, 4]`
9. **Nested Tuples**: Example: `((1, 2), (3, 4))`, access with `t[0][1]`.
10. **Tuples vs Lists**:
    - Tuples: Immutable, faster, fewer methods
    - Lists: Mutable, more flexible

---

11. **Practical Example**:

    1. Create a tuple `student` with `"Alice"`, `20`, `"A"`.
    2. Unpack it into `name`, `age`, `grade` and print: `"Name: Alice, Age: 20, Grade: A"`.
    3. Create `extra_info` with `"Math"`, `95`.
    4. Combine into `full_record` and print: "`Full Record: ('Alice', 20, 'A', 'Math', 95)`".

---

13. **Exercises**:
    1. Create a tuple containing the first 5 prime numbers (2, 3, 5, 7, 11). Then, print the second and fourth elements using indexing.