LIST COMPREHENSIONS IN PYTHON

What are they?

- List comprehensions are a super-efficient way to make new lists in Python.
- They pack complex list creation into one short line of readable code.

Basic Structure

Python
```python
new_list = [expression for item in iterable if condition]
```

- **expression:** The thing you want in your new list (can be calculations, method calls, etc.)
- **item:** A variable representing each thing in your existing list.
- **iterable:** The list (or other iterable object) you're working with.
- **condition:** Optional. A filter to include only certain items.

Why use them?

- **Readability:** They're clear and concise, making your code easier to understand.
- **Speed:** Often faster than traditional `for` loops, especially for smaller lists.
- **Pythonic:** It's the cool, Python-native way of doing things!

Examples

1. **Squares:**

   Python
   ```python
   numbers = [1, 2, 3, 4]
   squares = [x**2 for x in numbers]
   print(squares)  # Output: [1, 4, 9, 16]
   ```

2. **Filtering even numbers:**

   Python
   ```python
   numbers = [1, 2, 3, 4, 5, 6]
   even_numbers = [x for x in numbers if x % 2 == 0]
   print(even_numbers)  # Output: [2, 4, 6]
   ```

3. **Uppercase strings:**

Python
```python
fruits = ["apple", "banana", "cherry"]
uppercase_fruits = [fruit.upper() for fruit in fruits]
print(uppercase_fruits)  # Output: ["APPLE", "BANANA", "CHERRY"]
```

Things to Note

- **Nested list comprehensions:** You can create multidimensional lists (like a matrix).
- **Order matters:** The `for` loop part comes before the `if` condition.
- **Be careful with complexity:** Too much stuff crammed into one line can be hard to read.