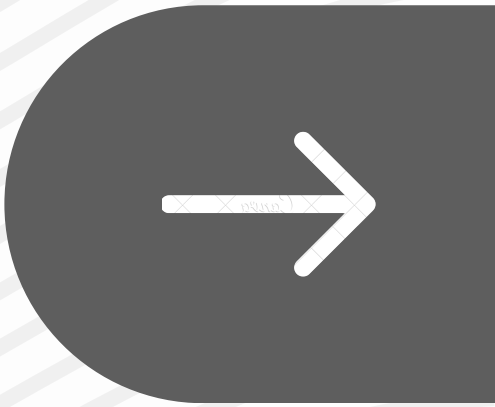
OSCAR CORTEZ



# HOW CAN YOU MAKE A "FOR" LOOP MORE ELEGANT?



@OSKRGAB

# ‘IN' keyword.

The ‘for’ loop can be simplified using the ‘in’ keyword. This approach is more readable and Pythonic.

👉This method directly accesses each element, eliminating the need for indexing.

```
def make_engineering_1(subjects):
  for subject in subjects:
    print(subject + ' Eng.')
```

# List comprehensions.

Offers a concise way to create lists. They're generally more compact and faster than normal functions and loops.

🚨 Remember, list comprehensions are best for creating new lists, not just for side effects like printing!

.

```
def make_engineering_2(subjects):
  return [print(subject + ' Eng.') for subject in subjects]
```

@OSKRGAB

# 'map()' function:

Applies a function to all items in a list. It's a powerful functional programming feature supported by Python.

✅ Ideal for applying a single operation to all list items. However, it is less readable than list comprehensions or 'for' loops.

```
def make_engineering_3(subjects):
  return list(map(lambda x: print(x + ' Eng.'), subjects))
```

# Enumerate.

Use it to add a counter to an iterable and get both the item and its index.

🧠 It's perfect for referencing array elements in loops.

```
def make_engineering_4(subjects):
  for i, subject in enumerate(subjects):
    print(f'{i}: {subject} Eng.')
```

Each of these methods has its use cases. Pythonic code is not just about being concise; it's about being readable, efficient, and expressive.

What's your go-to method for iterating in Python? Drop your thoughts below!

@OSKRGAB