

List and Dict Comprehensions, and Generator Expressions

- List comprehensions
- Dict comprehensions
- Generator expressions
- Nested comprehensions

List Comprehensions

- The difference between statements and expressions
- An expressive alternative to for statements – list comprehensions

Statements

Instructions

A 'for' loop is a statement

Statements don't evaluate
to something

Not used in (pure)
functional programming

Expression- Things that evaluate

Anything that evaluates to something

For example – $10 + 10$

Used heavily in functional programming

List Comprehensions
Are an Expressive
Alternative to 'for'
Statements

Dict Comprehensions

Generator Expressions

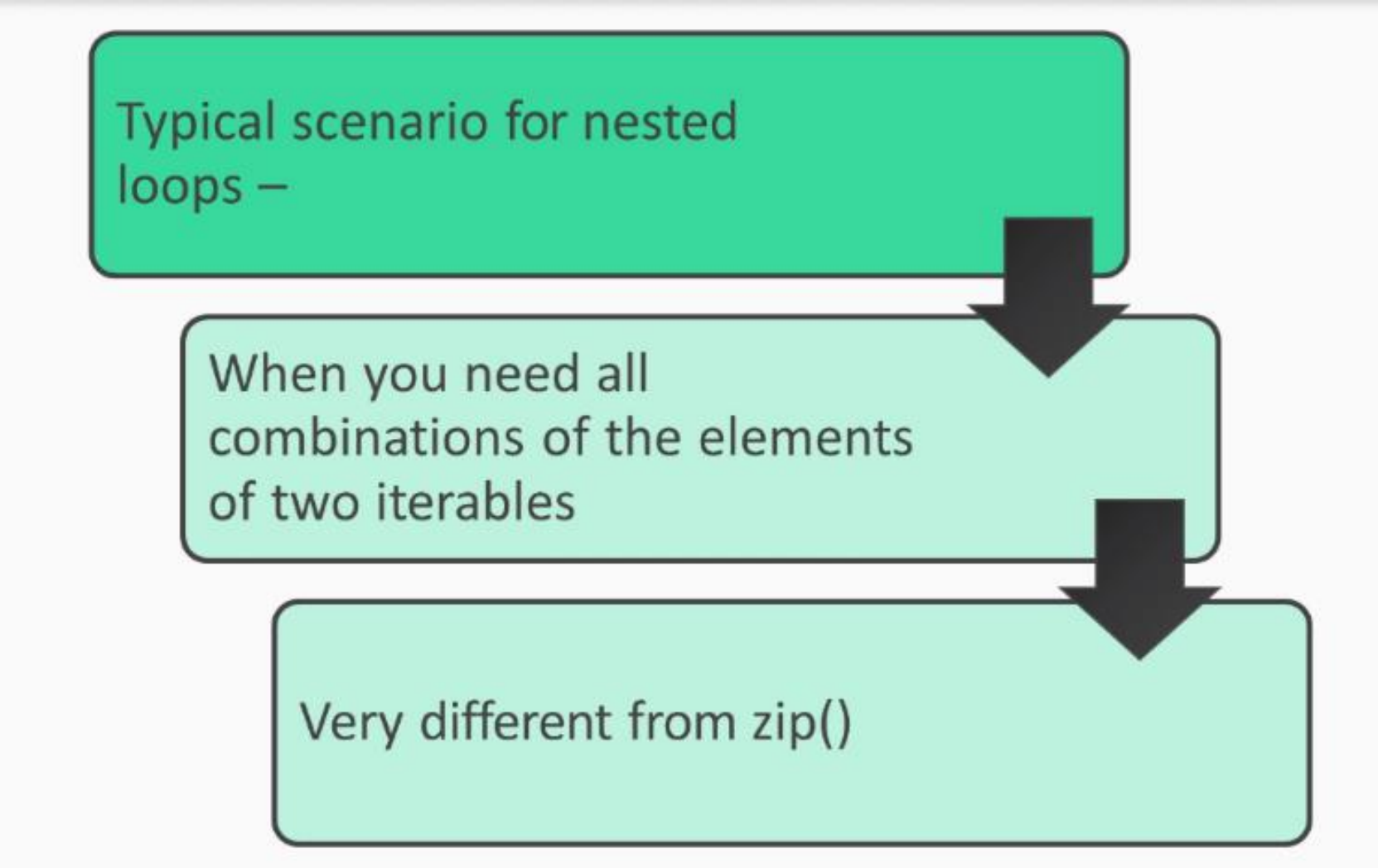
- What a generator expression is
- How it implements lazy evaluation
- How to implement break-like behavior

Nested Comprehensions

- What a nested loop is
- How you can implement this in a list comprehension

Nested loops- Combining Iterables

Typical scenario for nested loops –



```
graph TD; A[Typical scenario for nested loops –] --> B[When you need all combinations of the elements of two iterables]; B --> C[Very different from zip()];
```

When you need all combinations of the elements of two iterables

Very different from `zip()`

Summary

- Explained that statements are instructions to the Python interpreter and expressions are things that evaluate to something
- Explored that functional programming relies heavily on expressions
- Studied that list and dict comprehensions are expressive ways to implement loops
- Learned that comprehensions support filtering and nesting but not breaking
- Understood that generator expressions are lazy list comprehensions