Python Comprehensions vs Generators vs For-Loops (20 Examples)

Each example shows: list comprehension, generator expression, and the equivalent for-loop.

1) Squares of integers

List Comprehension:

```
squares = [n**2 for n in [1,2,3,4,5]]  # [1,4,9,16,25]
```

Generator Expression:

```
squares_gen = (n**2 for n in [1,2,3,4,5])  # generator (iterate to consume)
```

For-loop Equivalent:

```
nums=[1,2,3,4,5]
squares=[]
for n in nums:
    squares.append(n**2)
```

2) Filter even numbers

List Comprehension:

```
evens = [n for n in [1,2,3,4,5,6] if n % 2 == 0]  # [2,4,6]
```

Generator Expression:

```
evens_gen = (n for n in [1,2,3,4,5,6] if n % 2 == 0)
```

For-loop Equivalent:

```
nums=[1,2,3,4,5,6]
evens=[]
for n in nums:
    if n % 2 == 0:
        evens.append(n)
```

3) Uppercase all words

List Comprehension:

```
upper_words = [w.upper() for w in ['python','rocks','hard']]
```

Generator Expression:

```
upper_words_gen = (w.upper() for w in ['python','rocks','hard'])
```

For-loop Equivalent:

```
words=['python','rocks','hard']
upper_words=[]
for w in words:
    upper_words.append(w.upper())
```

4) First letters of words

List Comprehension:

```
first_letters = [w[0] for w in ['data','science','machine','learning']]
```

Generator Expression:

```
first_letters_gen = (w[0] for w in ['data','science','machine','learning'])
```

For-loop Equivalent:

```
words=['data','science','machine','learning']
first_letters=[]
for w in words:
    first_letters.append(w[0])
```

Python Comprehensions vs Generators vs For-Loops (20 Examples)

Each example shows: list comprehension, generator expression, and the equivalent for-loop.

5) Conditional transform (square evens, cube odds)

List Comprehension:

```
result = [n**2 if n % 2 == 0 else n**3 for n in [1,2,3,4,5]]
```

Generator Expression:

```
result_gen = (n**2 if n % 2 == 0 else n**3 for n in [1,2,3,4,5])
```

For-loop Equivalent:

```
nums=[1,2,3,4,5]
result=[]
for n in nums:
    result.append(n**2 if n % 2 == 0 else n**3)
```

6) Flatten a nested list

List Comprehension:

```
flat = [x for row in [[1,2,3],[4,5],[6]] for x in row]
```

Generator Expression:

```
flat_gen = (x for row in [[1,2,3],[4,5],[6]] for x in row)
```

For-loop Equivalent:

```
matrix=[[1,2,3],[4,5],[6]]
flat=[]
for row in matrix:
    for x in row:
        flat.append(x)
```

7) Remove vowels from a string

List Comprehension:

```
no_vowels = ''.join([ch for ch in 'comprehension' if ch not in 'aeiou'])
```

Generator Expression:

```
no_vowels_gen = (ch for ch in 'comprehension' if ch not in 'aeiou')  # use ''.join(...) to
materialize
```

For-loop Equivalent:

```
text='comprehension'
no_vowels_chars=[]
for ch in text:
    if ch not in 'aeiou':
        no_vowels_chars.append(ch)
no_vowels=''.join(no_vowels_chars)
```

8) Dict comprehension: squares

List Comprehension:

```
squares = {n: n**2 for n in [1,2,3,4]}
```

Generator Expression:

```
pairs_gen = ((n, n**2) for n in [1,2,3,4])  # dict(pairs_gen) to build dict lazily
```

For-loop Equivalent:

```
nums=[1,2,3,4]
squares={}
for n in nums:
    squares[n]=n**2
```

Python Comprehensions vs Generators vs For-Loops (20 Examples)

Each example shows: list comprehension, generator expression, and the equivalent for-loop.

9) Reverse each word in a sentence

List Comprehension:

```
reversed_words = [w[::-1] for w in 'python is powerful'.split()]
```

Generator Expression:

```
reversed_words_gen = (w[::-1] for w in 'python is powerful'.split())
```

For-loop Equivalent:

```
sentence='python is powerful'
reversed_words=[]
for w in sentence.split():
    reversed_words.append(w[::-1])
```

10) Multiple filters (n > 2 and even)

List Comprehension:

```
filtered = [n for n in range(10) if n > 2 if n % 2 == 0]
```

Generator Expression:

```
filtered_gen = (n for n in range(10) if n > 2 if n % 2 == 0)
```

For-loop Equivalent:

```
filtered=[]
for n in range(10):
    if n > 2 and n % 2 == 0:
        filtered.append(n)
```

11) Cartesian product (pairs)

List Comprehension:

```
pairs = [(x,y) for x in [1,2,3] for y in ['x','y']]
```

Generator Expression:

```
pairs_gen = ((x,y) for x in [1,2,3] for y in ['x','y'])
```

For-loop Equivalent:

```
pairs=[]
for x in [1,2,3]:
    for y in ['x','y']:
        pairs.append((x,y))
```

12) Word lengths (dict)

List Comprehension:

```
lengths = {w: len(w) for w in ['analytics','is','fun']}
```

Generator Expression:

```
lengths_pairs_gen = ((w, len(w)) for w in ['analytics','is','fun'])  # dict(...) to build
```

For-loop Equivalent:

```
words=['analytics','is','fun']
lengths={}
for w in words:
    lengths[w]=len(w)
```

Python Comprehensions vs Generators vs For-Loops (20 Examples)

Each example shows: list comprehension, generator expression, and the equivalent for-loop.

13) Divisible by both 3 and 5

List Comprehension:

```
div_3_5 = [n for n in range(1,51) if n % 3 == 0 and n % 5 == 0]
```

Generator Expression:

```
div_3_5_gen = (n for n in range(1,51) if n % 3 == 0 and n % 5 == 0)
```

For-loop Equivalent:

```
div_3_5=[]
for n in range(1,51):
    if n % 3 == 0 and n % 5 == 0:
        div_3_5.append(n)
```

14) Multiplication table (1..3 x 1..3)

List Comprehension:

```
table = [f"{i}x{j}={i*j}" for i in range(1,4) for j in range(1,4)]
```

Generator Expression:

```
table_gen = (f"{i}x{j}={i*j}" for i in range(1,4) for j in range(1,4))
```

For-loop Equivalent:

```
table=[]
for i in range(1,4):
    for j in range(1,4):
        table.append(f"{i}x{j}={i*j}")
```

15) Extract digits from string

List Comprehension:

```
digits = [int(ch) for ch in 'abc123xyz456' if ch.isdigit()]
```

Generator Expression:

```
digits_gen = (int(ch) for ch in 'abc123xyz456' if ch.isdigit())
```

For-loop Equivalent:

```
text='abc123xyz456'
digits=[]
for ch in text:
    if ch.isdigit():
        digits.append(int(ch))
```

16) Set comprehension: unique letters

List Comprehension:

```
unique_letters = {ch for ch in 'programming'}
```

Generator Expression:

```
unique_letters_gen = (ch for ch in 'programming')  # wrap with set(...) to consume
```

For-loop Equivalent:

```
text='programming'
unique_letters=set()
for ch in text:
    unique_letters.add(ch)
```

Python Comprehensions vs Generators vs For-Loops (20 Examples)

Each example shows: list comprehension, generator expression, and the equivalent for-loop.

17) Filter palindromes

List Comprehension:

```
palindromes = [w for w in ['level','python','madam','racecar'] if w == w[::-1]]
```

Generator Expression:

```
palindromes_gen = (w for w in ['level','python','madam','racecar'] if w == w[::-1])
```

For-loop Equivalent:

```
words=['level','python','madam','racecar']
palindromes=[]
for w in words:
    if w == w[::-1]:
        palindromes.append(w)
```

18) Double each character in a string

List Comprehension:

```
doubled = ''.join([ch*2 for ch in 'data'])
```

Generator Expression:

```
doubled_gen = (ch*2 for ch in 'data')  # ''.join(doubled_gen) to realize
```

For-loop Equivalent:

```
text='data'
chars=[]
for ch in text:
    chars.append(ch*2)
doubled=''.join(chars)
```

19) Even squares as dict

List Comprehension:

```
even_squares = {n: n**2 for n in range(1,10) if n % 2 == 0}
```

Generator Expression:

```
even_squares_pairs_gen = ((n, n**2) for n in range(1,10) if n % 2 == 0)
```

For-loop Equivalent:

```
even_squares={}
for n in range(1,10):
    if n % 2 == 0:
        even_squares[n]=n**2
```

20) Transpose a 2x3 matrix

List Comprehension:

```
transpose = [[row[i] for row in [[1,2,3],[4,5,6]]] for i in range(3)]
```

Generator Expression:

```
transpose_rows_gen = ([row[i] for row in [[1,2,3],[4,5,6]]] for i in range(3))  # list(...) to
realize
```

For-loop Equivalent:

```
matrix=[[1,2,3],[4,5,6]]
transpose=[]
for i in range(3):
    col=[]
    for row in matrix:
        col.append(row[i])
```

Python Comprehensions vs Generators vs For-Loops (20 Examples)

Each example shows: list comprehension, generator expression, and the equivalent for-loop.

Notes & Interview Tips

- List comprehensions MATERIALIZE a list immediately; generator expressions are LAZY (produce items on demand).
- Use generators for large data or streaming pipelines; wrap with list()/set()/dict() to realize.
- Dict/set comprehensions are not generators; but you can feed generators into dict()/set().
- Readability first: prefer multi-line loops when logic gets too nested.
- Time complexity is the same as equivalent for-loops; comprehensions are often slightly faster due to C-level implementation.