List Comprehensions in Python / List Comprehensions στην Python

What are List Comprehensions / Τι είναι τα List Comprehensions

List comprehensions are a concise and expressive way to create lists in Python. They are a language idiom that allows the creation of new lists based on existing lists or other iterable data structures.

Τα list comprehensions είναι ένας συνοπτικός και εκφραστικός τρόπος δημιουργίας λιστών στην Python. Αποτελούν έναν ιδιωματισμό της γλώσσας, που επιτρέπει τη δημιουργία νέων λιστών βασισμένων σε υπάρχουσες λίστες ή άλλες επαναλήψιμες δομές δεδομένων.

Why Use List Comprehensions / Γιατί να χρησιμοποιήσουμε List Comprehensions

1. **Conciseness / Συνοπτικότητα**:
   - EN: They allow you to condense multiple lines of code into a single line
   - GR: Επιτρέπουν τη συμπύκνωση πολλαπλών γραμμών κώδικα σε μία μόνο γραμμή
2. **Readability / Αναγνωσιμότητα**:
   - EN: When used properly, they can make code more readable
   - GR: Όταν χρησιμοποιούνται σωστά, μπορούν να κάνουν τον κώδικα πιο ευανάγνωστο
3. **Efficiency / Αποδοτικότητα**:
   - EN: They are often faster than equivalent for loops
   - GR: Συχνά είναι πιο γρήγορα από τους αντίστοιχους βρόχους for

Basic Syntax / Βασική Σύνταξη

The basic syntax of a list comprehension is / Η βασική σύνταξη ενός list comprehension είναι:

```
[expression for item in iterable if condition]
```

Components / Συστατικά μέρη:

- `expression`:
  - EN: The expression to be applied to each element
  - GR: Η έκφραση που θα εφαρμοστεί σε κάθε στοιχείο
- `item`:
  - EN: The variable representing each element in the iteration
  - GR: Η μεταβλητή που αντιπροσωπεύει κάθε στοιχείο στην επανάληψη
- `iterable`:
  - EN: The list, tuple, or any other iterable data structure
  - GR: Η λίστα, το tuple, ή οποιαδήποτε άλλη επαναλήψιμη δομή δεδομένων
- `if condition`:
  - EN: (Optional) A condition for filtering elements
  - GR: (Προαιρετικό) Μια συνθήκη για το φιλτράρισμα των στοιχείων

Examples / Παραδείγματα

1. Creating a List of Squares / Δημιουργία λίστας με τετράγωνα αριθμών

```
# Without list comprehension / Χωρίς list comprehension
squares = []
for i in range(10):
    squares.append(i**2)

# With list comprehension / Με list comprehension
squares = [i**2 for i in range(10)]

print(squares)  # [0, 1, 4, 9, 16, 25, 36, 49, 64, 81]
```

2. Filtering Even Numbers / Φιλτράρισμα ζυγών αριθμών

```
# Without list comprehension / Χωρίς list comprehension
even_numbers = []
for i in range(20):
    if i % 2 == 0:
        even_numbers.append(i)

# With list comprehension / Με list comprehension
even_numbers = [i for i in range(20) if i % 2 == 0]

print(even_numbers)  # [0, 2, 4, 6, 8, 10, 12, 14, 16, 18]
```

3. Converting Words to Uppercase / Μετατροπή λέξεων σε κεφαλαία

```
words = ['python', 'is', 'awesome']

# Without list comprehension / Χωρίς list comprehension
upper_words = []
for word in words:
    upper_words.append(word.upper())

# With list comprehension / Με list comprehension
upper_words = [word.upper() for word in words]

print(upper_words)  # ['PYTHON', 'IS', 'AWESOME']
```

4. Creating List of Pairs / Δημιουργία λίστας ζευγών (x, y)

```
# With list comprehension / Με list comprehension
pairs = [(x, y) for x in range(3) for y in range(2)]

print(pairs)  # [(0, 0), (0, 1), (1, 0), (1, 1), (2, 0), (2, 1)]
```

In [ ]:

String Methods in Python: split() and join() / Μέθοδοι Συμβολοσειρών στην Python: split() και join()

The split() Method / Η Μέθοδος split()

Basic Description / Βασική Περιγραφή

English: The `split()` method divides a string into a list of substrings based on a specified delimiter. If no delimiter is specified, whitespace is used by default.

Ελληνικά: Η μέθοδος `split()` διαιρεί μια συμβολοσειρά σε μια λίστα υποσυμβολοσειρών με βάση έναν καθορισμένο διαχωριστή. Εάν δεν καθοριστεί διαχωριστής, χρησιμοποιείται ως προεπιλογή το κενό διάστημα.

Basic Syntax / Βασική Σύνταξη

```
string.split(separator, maxsplit)
```

- `separator` (EN): The delimiter to use for splitting (optional)
- `separator` (GR): Ο διαχωριστής που θα χρησιμοποιηθεί για τον διαχωρισμό (προαιρετικό)
- `maxsplit` (EN): Maximum number of splits to perform (optional)
- `maxsplit` (GR): Μέγιστος αριθμός διαχωρισμών που θα εκτελεστούν (προαιρετικό)

Examples / Παραδείγματα

```python
# Basic split with whitespace / Βασικός διαχωρισμός με κενό
text = "Python is awesome"
words = text.split()
print(words)  # ['Python', 'is', 'awesome']

# Split with specific delimiter / Διαχωρισμός με συγκεκριμένο διαχωριστή
data = "apple,banana,orange"
fruits = data.split(',')
print(fruits)  # ['apple', 'banana', 'orange']

# Split with limit / Διαχωρισμός με όριο
text = "one:two:three:four"
parts = text.split(':', 2)
print(parts)  # ['one', 'two', 'three:four']
```

The join() Method / Η Μέθοδος join()

Basic Description / Βασική Περιγραφή

English: The `join()` method concatenates a list of strings using a specified separator. It's called on the separator string and takes an iterable of strings as its argument.

Ελληνικά: Η μέθοδος `join()` συνενώνει μια λίστα συμβολοσειρών χρησιμοποιώντας έναν καθορισμένο διαχωριστή. Καλείται στη συμβολοσειρά του διαχωριστή και δέχεται ως όρισμα μια επαναλήψιμη δομή συμβολοσειρών.

Basic Syntax / Βασική Σύνταξη

```
separator.join(iterable)
```

Examples / Παραδείγματα

```python
# Basic join with space / Βασική συνένωση με κενό
words = ['Python', 'is', 'awesome']
sentence = ' '.join(words)
print(sentence)  # 'Python is awesome'
```

```python
# Join with comma / Συνένωση με κόμμα
fruits = ['apple', 'banana', 'orange']
fruit_list = ', '.join(fruits)
print(fruit_list)  # 'apple, banana, orange'

# Join with empty string / Συνένωση με κενή συμβολοσειρά
characters = ['H', 'e', 'l', 'l', 'o']
word = ''.join(characters)
print(word)  # 'Hello'
```

Common Use Cases / Συνήθεις Περιπτώσεις Χρήσης

1. Processing CSV Data / Επεξεργασία Δεδομένων CSV

```python
# Split CSV line / Διαχωρισμός γραμμής CSV
csv_line = "John,Doe,30,New York"
data = csv_line.split(',')
print(data)  # ['John', 'Doe', '30', 'New York']

# Create CSV line / Δημιουργία γραμμής CSV
data = ['Jane', 'Smith', '25', 'London']
csv_line = ','.join(data)
print(csv_line)  # 'Jane,Smith,25,London'
```

2. URL Processing / Επεξεργασία URL

```python
# Split URL parts / Διαχωρισμός μερών URL
url = "https://www.example.com/path/to/page"
parts = url.split('/')
print(parts)  # ['https:', '', 'www.example.com', 'path', 'to', 'page']

# Join URL parts / Συνένωση μερών URL
path_parts = ['users', 'profile', '123']
path = '/'.join(path_parts)
print(path)  # 'users/profile/123'
```

3. Text Processing / Επεξεργασία Κειμένου

```python
# Clean extra whitespace / Καθαρισμός επιπλέον κενών
text = "too    many    spaces    here"
clean_text = ' '.join(text.split())
print(clean_text)  # 'too many spaces here'

# Create sentence from words / Δημιουργία πρότασης από λέξεις
words = ['The', 'quick', 'brown', 'fox']
sentence = ' '.join(words) + '.'
print(sentence)  # 'The quick brown fox.'
```

Important Notes / Σημαντικές Σημειώσεις

English:

- `split()` always returns a list of strings
- `join()` can only be used with an iterable containing strings
- Both methods are case-sensitive
- Neither method modifies the original string (strings are immutable in Python)

Ελληνικά:

- Η `split()` επιστρέφει πάντα μια λίστα συμβολοσειρών
- Η `join()` μπορεί να χρησιμοποιηθεί μόνο με επαναλήψιμη δομή που περιέχει συμβολοσειρές
- Και οι δύο μέθοδοι κάνουν διάκριση πεζών-κεφαλαίων
- Καμία μέθοδος δεν τροποποιεί την αρχική συμβολοσειρά (οι συμβολοσειρές είναι αμετάβλητες στην Python)

In [ ]:

Python Logical Functions and Falsy Values / Λογικές Συναρτήσεις και Ψευδείς Τιμές στην Python

all() and any() Functions / Συναρτήσεις all() και any()

all() Function / Συνάρτηση all()

English: The `all()` function returns `True` if all elements in an iterable are true. If any element is false, it returns `False`.

Ελληνικά: Η συνάρτηση `all()` επιστρέφει `True` αν όλα τα στοιχεία σε μια επαναλήψιμη δομή είναι αληθή. Αν έστω και ένα στοιχείο είναι ψευδές, επιστρέφει `False`.

```
# Examples of all() / Παραδείγματα της all()
print(all([True, True, True]))     # True
print(all([True, False, True]))    # False
print(all([1, 2, 3]))              # True
print(all([1, 0, 3]))              # False
print(all([]))                     # True (empty iterables return True)

# Practical example / Πρακτικό παράδειγμα
numbers = [2, 4, 6, 8, 10]
all_even = all(num % 2 == 0 for num in numbers)
print(all_even)  # True - all numbers are even
```

any() Function / Συνάρτηση any()

English: The `any()` function returns `True` if at least one element in an iterable is true. If all elements are false, it returns `False`.

Ελληνικά: Η συνάρτηση `any()` επιστρέφει `True` αν τουλάχιστον ένα στοιχείο σε μια επαναλήψιμη δομή είναι αληθές. Αν όλα τα στοιχεία είναι ψευδή, επιστρέφει `False`.

```
# Examples of any() / Παραδείγματα της any()
print(any([False, False, True]))   # True
print(any([False, False, False]))  # False
print(any([0, '', False, 1]))      # True
print(any([0, '', False]))         # False
print(any([]))                     # False (empty iterables return False)

# Practical example / Πρακτικό παράδειγμα
numbers = [1, 3, 4, 7, 9]
```

```python
has_even = any(num % 2 == 0 for num in numbers)
print(has_even)  # True - at least one number is even
```

Falsy Values in Python / Ψευδείς Τιμές στην Python

English: In Python, certain values are considered "falsy", meaning they evaluate to `False` in logical conditions. Here are the main falsy values:

Ελληνικά: Στην Python, ορισμένες τιμές θεωρούνται "ψευδείς", που σημαίνει ότι αποτιμώνται ως `False` σε λογικές συνθήκες. Εδώ είναι οι κύριες ψευδείς τιμές:

False / Ψευδής

```python
# Boolean False / Λογική τιμή False
x = False
if x:
    print("This won't print")  # This won't execute / Αυτό δεν θα εκτελεστεί
```

Zero (0) / Μηδέν (0)

```python
# Zero in any numeric type / Μηδέν σε οποιονδήποτε αριθμητικό τύπο
x = 0
if x:
    print("This won't print")  # This won't execute / Αυτό δεν θα εκτελεστεί

# Works with float too / Λειτουργεί και με δεκαδικούς
y = 0.0
if y:
    print("This won't print")  # This won't execute / Αυτό δεν θα εκτελεστεί
```

Empty String ('') / Κενή Συμβολοσειρά ('')

```python
# Empty string / Κενή συμβολοσειρά
x = ''
if x:
    print("This won't print")  # This won't execute / Αυτό δεν θα εκτελεστεί

# Empty string with double quotes / Κενή συμβολοσειρά με διπλά εισαγωγικά
y = ""
if y:
    print("This won't print")  # This won't execute / Αυτό δεν θα εκτελεστεί
```

None / Τίποτα

```python
# None value / Τιμή None
x = None
if x:
    print("This won't print")  # This won't execute / Αυτό δεν θα εκτελεστεί
```

Practical Examples / Πρακτικά Παραδείγματα

Combining all() and Falsy Values / Συνδυασμός all() και Ψευδών Τιμών

```python
# Check if all strings are non-empty / Έλεγχος αν όλες οι συμβολοσειρές
είναι μη-κενές
strings = ['hello', 'world', '']
print(all(strings))  # False (empty string is falsy)
```

```
# Check if all numbers are non-zero / Έλεγχος αν όλοι οι αριθμοί είναι μη-μηδενικοί
numbers = [1, 2, 0, 4]
print(all(numbers))  # False (zero is falsy)
```

Combining any() and Falsy Values / Συνδυασμός any() και Ψευδών Τιμών

```
# Check if any string has content / Έλεγχος αν κάποια συμβολοσειρά έχει περιεχόμενο
strings = ['', '', 'hello', '']
print(any(strings))  # True ('hello' is truthy)

# Check if any number is non-zero / Έλεγχος αν κάποιος αριθμός είναι μη-μηδενικός
numbers = [0, 0, 5, 0]
print(any(numbers))  # True (5 is truthy)
```

Using None in Conditions / Χρήση του None σε Συνθήκες

```
# Function that might return None / Συνάρτηση που μπορεί να επιστρέψει None
def get_value(index):
    values = [1, 2, None, 4]
    if index < len(values):
        return values[index]
    return None

# Checking return value / Έλεγχος επιστρεφόμενης τιμής
result = get_value(5)
if result is None:
    print("No value found")  # This will print / Αυτό θα εκτυπωθεί
```

In [ ]:

Python ord() Function and ASCII / Συνάρτηση ord() και ASCII στην Python

The ord() Function / Η Συνάρτηση ord()

English: The `ord()` function returns the Unicode code point of a single character. For ASCII characters (0-127), this matches the ASCII value. It's particularly useful for character processing and validation.

Ελληνικά: Η συνάρτηση `ord()` επιστρέφει τον κωδικό Unicode ενός μεμονωμένου χαρακτήρα. Για χαρακτήρες ASCII (0-127), αυτό ταιριάζει με την τιμή ASCII. Είναι ιδιαίτερα χρήσιμη για επεξεργασία και επικύρωση χαρακτήρων.

Basic Usage / Βασική Χρήση

```
print(ord('A'))  # 65
print(ord('a'))  # 97
print(ord('0'))  # 48
print(ord('9'))  # 57
```

ASCII Table Overview / Επισκόπηση Πίνακα ASCII

English: Important ASCII ranges:

- Digits: 48-57 (0-9)
- Uppercase letters: 65-90 (A-Z)
- Lowercase letters: 97-122 (a-z)

Ελληνικά: Σημαντικά εύρη ASCII:

- Ψηφία: 48-57 (0-9)
- Κεφαλαία γράμματα: 65-90 (A-Z)
- Πεζά γράμματα: 97-122 (a-z)

Implementing is_numeric / Υλοποίηση του is_numeric

Method 1: Traditional Implementation / Παραδοσιακή Υλοποίηση

```
def is_numeric(input):
    """
    Check if string contains only numeric characters.
    Έλεγχος αν η συμβολοσειρά περιέχει μόνο αριθμητικούς χαρακτήρες.
    """
    if not isinstance(input, str):
        print('Only strings accepted')
        return

    for char in input:
        if not (48 <= ord(char) <= 57):
            print(f'Not an int: {char}')
            return False
    return True

# Test cases / Δοκιμαστικές περιπτώσεις
print(is_numeric('123'))     # True
print(is_numeric('12.3'))    # False (prints: Not an int: .)
print(is_numeric('12a3'))    # False (prints: Not an int: a)
print(is_numeric(123))       # None (prints: Only strings accepted)
```

Method 2: Using all() and List Comprehension / Χρήση all() και List Comprehension

```
def is_numeric_compact(input):
    """
    Check if string contains only numeric characters using list
comprehension.
    Έλεγχος αν η συμβολοσειρά περιέχει μόνο αριθμητικούς χαρακτήρες με list
comprehension.
    """
    if not isinstance(input, str):
        print('Only strings accepted')
        return

    return all(48 <= ord(char) <= 57 for char in input)

# Test cases / Δοκιμαστικές περιπτώσεις
print(is_numeric_compact('123'))     # True
print(is_numeric_compact('12.3'))    # False
```

```
print(is_numeric_compact('12a3'))    # False
print(is_numeric_compact(123))       # None (prints: Only strings accepted)
```

More Examples with ord() / Περισσότερα Παραδείγματα με την ord()

Check for Uppercase Letters / Έλεγχος για Κεφαλαία Γράμματα

```
def is_uppercase(char):
    return 65 <= ord(char) <= 90

# Using with all() / Χρήση με all()
def all_uppercase(text):
    return all(65 <= ord(char) <= 90 for char in text)

print(is_uppercase('A'))     # True
print(is_uppercase('a'))     # False
print(all_uppercase('ABC'))  # True
print(all_uppercase('ABc'))  # False
```

Check for Lowercase Letters / Έλεγχος για Πεζά Γράμματα

```
def is_lowercase(char):
    return 97 <= ord(char) <= 122

# Using with any() / Χρήση με any()
def has_lowercase(text):
    return any(97 <= ord(char) <= 122 for char in text)

print(is_lowercase('a'))     # True
print(is_lowercase('A'))     # False
print(has_lowercase('aBC'))  # True
print(has_lowercase('ABC'))  # False
```

Practical Example: Password Validator / Πρακτικό Παράδειγμα: Επικύρωση Κωδικού

```
def validate_password(password):
    """
    Check if password has at least one uppercase, one lowercase, and one
number.
    Έλεγχος αν ο κωδικός έχει τουλάχιστον ένα κεφαλαίο, ένα πεζό και έναν
αριθμό.
    """
    has_upper = any(65 <= ord(char) <= 90 for char in password)
    has_lower = any(97 <= ord(char) <= 122 for char in password)
    has_digit = any(48 <= ord(char) <= 57 for char in password)

    return has_upper and has_lower and has_digit

print(validate_password('abc123'))    # False (no uppercase)
print(validate_password('Abc123'))    # True
print(validate_password('ABCDEF'))    # False (no lowercase, no numbers)
```

In [ ]: