Python List Comprehensions, Zip, and Enumerate

Contents

Time required: 30 minutes

List Comprehension

A list comprehension is used for creating new lists from other iterables like tuples, strings, arrays, lists, etc. A list comprehension consists of brackets containing the expression, which is executed for each element along with the for loop to iterate over each element.

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

- **new_list** : The list to be created.
- **expression** : The operation or transformation to apply to each item.
- **item** : The variable representing elements from the iterable.
- **iterable** : The source of data (e.g., a list, tuple, or range).

Let's square each number in a list.

```
numbers = [1, 2, 3, 4, 5]
squared_numbers = [x**2 for x in numbers]
print(squared_numbers)
```

Example run:

```
[1, 4, 9, 16, 25]
```

Let's find the even numbers in a list.

```
numbers = [1, 2, 3, 4, 5]
even_numbers = [x for x in numbers if x % 2 == 0]
print(even_numbers)
```

Example:

```
[2, 4]
```

We can split a string into a list with two different methods, one with a loop, one with a list comprehension.

```
# Split a string into a list using a for loop
name = "Darwin"
name_list = []
for n in name:
    name_list.append(n)
print(name_list)

# Split a string using a list comprehension
name_list = [n for n in name]
print(name_list)
```

Each method comes up with the same result.

```
['D', 'a', 'r', 'w', 'i', 'n']
['D', 'a', 'r', 'w', 'i', 'n']
```

Numbers work very well in list comprehensions.

```
list_of_numbers = []
for n in range(10):
    list_of_numbers.append(n)
print(list_of_numbers)

# All numbers
list_of_numbers = [n for n in range(10)]
print(list_of_numbers)

# Even numbers
list_of_numbers = [n for n in range(10) if n % 2 == 0]
print(list_of_numbers)
```

Example run:

```
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
[0, 2, 4, 6, 8]
```

The following example compares a for loop with a list comprehension.

```
"""
    Name: list_comprehension_square.py
    Author:
    Created:
    Description: Square each number in a list,
    return a list of the squares
"""

# ----------------- FOR LOOP ------------------- #
numbers = [1, 2, 3, 4, 5]
print(f"Original list")
print(numbers)
squares = []
print("For loop squared")
for x in numbers:
    squares.append(x * x)
print(squares)

# ------------ LIST COMPREHENSION -------------- #
numbers = [1, 2, 3, 4, 5]
# A list comprehension returns a list
# The result of the expression is automatically
# appended to the list
# [<expression> for <element> in <iterable>]
squares = [x * x for x in numbers]
print("List comprehension squared")
print(squares)
```

Replit of [list_comprehension_square]

1. expression: x * x
2. element: x
3. iterable: numbers

Example run:

```
[1, 4, 9, 16, 25]
[1, 4, 9, 16, 25]
```

The following function takes a list as an argument, then uses a list comprehension to find all numbers greater than 0.

```
def positive(numbers):
    # [<expression> for <element> in <iterable> if <condition>]
    positive_numbers = [num for num in numbers if num > 0]
    return positive_numbers

numbers = [21, -23, 4, -2, 34, -8]
print(positive(numbers))
```

Example run:

```
[21, 4, 34]
```

As you can see by the example, the list comprehension only takes one line compared to 3 lines of code.

Tutorial 1: Temperature List Comprehension

Create a Python program named **temperature_list_comprehension.py**

Enter the following code.

```python
"""
    Name: temperature_list_comprehension.py
    Author:
    Created:
    Purpose: Compare for loop and list comprehension
    to operate on each element of a list
"""
#
temps = [233, 453, 112, 332, 331, -9999]

# Divide each item in the list by 10
# -------------- Use for each loop ---------------------- #
new_temps = []
for temp in temps:
    new_temps.append(temp / 10)
print(f"For loop: {new_temps}")

# --------------- Use list comprehension --------------- #
# For every temp in temperature,
# divide by 10, store in new list
# [<expression> for <element> in <iterable>]
new_temp = [temp / 10 for temp in temps]
print(f"List Comprehension: {new_temp}")

# Do not process temp if it is -9999
# [<expression> for <element> in <iterable> if <condition>]
new_temp = [temp / 10 for temp in temps if temp != -9999]

print(f"If valid data: {new_temp}")

# Substitute 0 if number is -9999
# [<expression> for <element> in <iterable> if <condition> else <condition>]
new_temp = [temp / 10 if temp != -9999 else 0 for temp in temps]

print(f"If -9999: {new_temp}")
```

Example run:

```
For loop: [23.3, 45.3, 11.2, 33.2, 33.1, -999.9]
List Comprehension: [23.3, 45.3, 11.2, 33.2, 33.1, -999.9]
If valid data: [23.3, 45.3, 11.2, 33.2, 33.1]
```

Cheatsheet: List Comprehensions

A list comprehension is an expression that creates a list by iterating over another container.

A basic list comprehension:

```
[i * 2 for i in [1, 5, 10]]
```

Output: [2, 10, 20]

List comprehension with if condition:

```
[i * 2 for i in [1, -2, 10] if i > 0]
```

Output: [2, 20]

List comprehension with an if and else condition:

```
[i * 2 if i > 0 else 0 for i in [1, -2, 10]]
```

Output: [2, 0, 20]

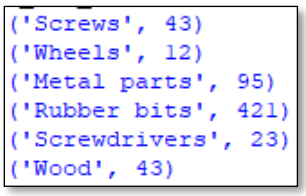
Zip and Enumerate

The zip method is a way to work with parallel lists.

```
# Filename: zip_and_enumerate.py
inventory_names = [
    'Screws', 'Wheels', 'Metal parts',
    'Rubber bits', 'Screwdrivers', 'Wood'
]
inventory_numbers = [
    43, 12, 95, 421, 23, 43
]

# The zip method packs two lists into a tuple of lists
for item in zip(inventory_names, inventory_numbers):
    print(item)
```

Example run:



```
inventory_names = ['Screws', 'Wheels', 'Metal parts',
                   'Rubber bits', 'Screwdrivers', 'Wood']
inventory_numbers = [43, 12, 95, 421, 23, 43]

# zip
# Unpack each tuple into name and number
for name, number in zip(inventory_names, inventory_numbers):
    print(f'{name} current inventory: {number}')

# enumerate function - get the current index
for index, name in enumerate(inventory_names):
    print(f'{index}: {name}')
    if index == len(inventory_names) // 2:
        print('halfway done!')

# Exercise
# combine zip and enumerate to get 'Screws [id: 0] - inventory: 43'
for index, inventory_tuple in enumerate(zip(inventory_names,
inventory_numbers)):
    print(
        f'{inventory_tuple[0]} [id: {index}] - inventory:
{inventory_tuple[1]}')
```

Assignment Submission

1. Attach the pseudocode.
2. Attach the program files.
3. Attach screenshots showing the successful operation of the program.
4. Submit in Blackboard.