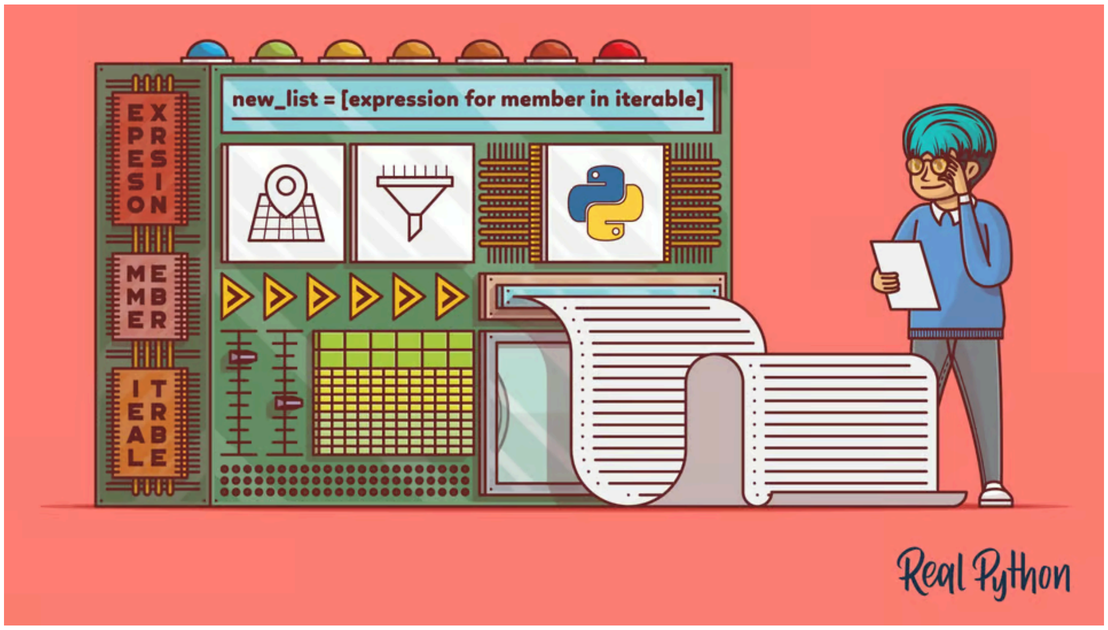
10. List Comprehension

List comprehensions are a concise and powerful way to create lists in Python. They provide a more readable and efficient alternative to traditional loop-based list creation.



List comprehension is a concise way to create lists in Python by combining a loop and conditional statements within a single line of code.

The general syntax of list comprehension is as follows:

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

Here's a breakdown of the different components:

1. expression: This represents the value or operation that will be applied to each item in the iterable. It defines how the items in the resulting list will be calculated or transformed.

2. item: This is a variable that represents each element in the iterable. It can be used in the expression to perform operations or calculations.

3. iterable: This is the sequence, such as a list, tuple, string, or any other iterable object, from which elements will be taken to form the new list.

4. condition (optional): This is an additional conditional statement that can be included to filter elements from the iterable. Only items that satisfy the condition will be included in the resulting list.

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

```
[4, 16]
```

In this example, the list comprehension generates a new list, squared_numbers, by squaring each even number in the numbers list. The expression x**2 calculates the square of each x (the current item in the iteration), and the condition if x % 2 == 0 filters out odd numbers. The resulting list contains the squares of the even numbers, which are [4, 16].

List comprehensions can also be nested, allowing for more complex transformations and filtering

```
In [2]: matrix = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]
        flattened_matrix = [num for sublist in matrix for num in sublist]
        print(flattened_matrix)
```

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

In this example, the nested list comprehension flattens a 2-dimensional matrix into a 1-dimensional list. It iterates over each sublist in the matrix, and for each num in each sublist, it appends the num to the flattened_matrix list. The resulting list is [1, 2, 3, 4, 5, 6, 7, 8, 9].

```
In [3]: words = ['hello', 'world', 'python']
        uppercase_words = [word.upper() for word in words]
        print(uppercase_words)
```

```
['HELLO', 'WORLD', 'PYTHON']
```

```
In [4]: letters = ['a', 'b', 'c']
        numbers = [1, 2, 3]
        pairs = [(letter, number) for letter in letters for number in numbers]
        print(pairs)
```

```
[('a', 1), ('a', 2), ('a', 3), ('b', 1), ('b', 2), ('b', 3), ('c', 1), ('c', 2), ('c',
3)]
```

```
In [5]: words = ['apple', 'banana', 'cherry', 'date', 'elderberry']
        filtered_words = [word for word in words if len(word) > 5]
        print(filtered_words)
```

```
['banana', 'cherry', 'elderberry']
```

```
In [6]: numbers = range(1, 20)
        prime_numbers = [num for num in numbers if all(num % i != 0 for i in range(2, int(num**
        print(prime_numbers)
```

```
[1, 2, 3, 5, 7, 11, 13, 17, 19]
```

The code iterates over each number in the range and checks if it's prime by verifying that it's not divisible by any number from 2 up to its square root. If a number passes this check, it's added to the prime_numbers list. The final list contains the prime numbers within the specified range.

```
In [7]: celsius = [0,10,20.1,34.5]
        fahrenheit = [((9/5)*temp + 32) for temp in celsius ]
        fahrenheit
```

```
Out[7]: [32.0, 50.0, 68.18, 94.1]
```

```
In [8]: lst = [ x**2 for x in [x**2 for x in range(11)]]
        lst
```

```
Out[8]: [0, 1, 16, 81, 256, 625, 1296, 2401, 4096, 6561, 10000]
```

```
In [9]: list_1 = [1,2,3]
        list_2 = [4,5,6]
        list_3 = [7,8,9]

        matrix = [list_1,list_2,list_3]

        first_col = [row[0] for row in matrix]
        first_col
```

```
Out[9]: [1, 4, 7]
```