How to check a list is EMPTY or NOT

In [ ]:
```
list1 = []

if not list1:
    print("List1 Empty")
else:
    print("List1 Non-Empty")


list2 = [3,4,5]

if not list2:
    print("List2 Empty")
else:
    print("List2 Non-Empty")
```

```
List1 Empty
List2 Non-Empty
```

In [ ]:
```
list3 = []

if not any(list3):
    print("List3 Empty")
else:
    print("List3 Non-Empty")


list4 = [3,4,5]

if not any(list4):
    print("List4 Empty")
else:
    print("List4 Non-Empty")
```

```
List3 Empty
List4 Non-Empty
```

In [ ]:
```
list5 = []

if isinstance(list5, list) and not list5:
    print("List5 Empty")
else:
    print("List5 Non-Empty")


list6 = [3,4,5]

if isinstance(list6, list) and not list6:
    print("List6 Empty")
else:
    print("List6 Non-Empty")
```

```
List5 Empty
List6 Non-Empty
```

In [ ]:
```
list7 = []
```

```
if len(list7) == 0:
    print("List7 Empty")
else:
    print("List7 Non-Empty")


list8 = [3,4,5]

if len(list8) == 0:
    print("List8 Empty")
else:
    print("List8 Non-Empty")
```

```
List7 Empty
List8 Non-Empty
```

In [ ]:

```
list7 = []

if list7 == []:
    print("List7 Empty")
else:
    print("List7 Non-Empty")


list8 = [3,4,5]

if list8 == []:
    print("List8 Empty")
else:
    print("List8 Non-Empty")
```

```
List7 Empty
List8 Non-Empty
```

In [ ]:

In [ ]:

Six ways to flatten a List

In [ ]:
```
list1 = [[1,2,3],[4,5,6,7],[8,9,10,11]]
new_list = []

for i in list1:
    print("Len of i = ",len(i))
    print("i",i)
    for j in i:
        print("j",j)
        print(f"Appending value {j} to list")
        new_list.append(j)

print(new_list)
```

```
Len of i =  3
i [1, 2, 3]
j 1
Appending value 1 to list
j 2
Appending value 2 to list
j 3
Appending value 3 to list
Len of i =  4
i [4, 5, 6, 7]
j 4
Appending value 4 to list
j 5
Appending value 5 to list
j 6
Appending value 6 to list
j 7
Appending value 7 to list
Len of i =  4
i [8, 9, 10, 11]
j 8
Appending value 8 to list
j 9
Appending value 9 to list
j 10
Appending value 10 to list
j 11
Appending value 11 to list
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11]
```

In [ ]:
```
list1 = [[1,2,3],[4,5,6,7],[8,9,10,11]]
result = [innerlist for outerlist in list1 for innerlist in outerlist]
print(result)
```

```
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11]
```

In [ ]:
```
from timeit import timeit
print(timeit("[innerlist for outerlist in list1 for innerlist in outerlis
```

```
0.8535068639976089
```

In [ ]:
```
list1 = [[1,2,3],[4,5,6,7],[8,9,10,11]]
new_list = []
```

```
for i in list1:
    print("Len of i = ",len(i))
    print("i",i)
    new_list = new_list + i

print(new_list)
```

```
Len of i =  3
i [1, 2, 3]
Len of i =  4
i [4, 5, 6, 7]
Len of i =  4
i [8, 9, 10, 11]
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11]
```

In [ ]:
```
list1 = [[1,2,3],[4,5,6,7],[8,9,10,11]]
result2 = sum(list1, [])
print(result2)
```

```
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11]
```

In [ ]:
```
from timeit import timeit
print(timeit("sum(list1, [])", "from __main__ import list1"))
```

```
0.3979934620001586
```

In [ ]:
```
list1 = [[1,2,3],[4,5,6,7],[8,9,10,11]]

from functools import reduce
result4 = reduce(lambda x,y : x+y, list1)
print(result4)
```

```
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11]
```

In [ ]:
```
from timeit import timeit
print(timeit("reduce(lambda x,y: x+y, list1)", "from __main__ import list
```

```
0.4760102359978191
```

In [ ]:
```
list1 = [[1,2,3],[4,5,6,7],[8,9,10,11]]

from functools import reduce
from operator import add
result5 =  reduce(add, list1)
print(result5)
```

```
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11]
```

In [ ]:
```
from timeit import timeit
print(timeit("reduce(add, list1)", "from __main__ import list1; from oper
```

```
0.2977846169997065
```

In [ ]:
```
list1 = [[1,2,3],[4,5,6,7],[8,9,10,11]]

from itertools import chain
result3 = list(chain(*list1))
print(result3)
```

```
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11]
```

```
In [ ]: from timeit import timeit
        print(timeit("list(chain(*list1))", "from __main__ import list1; from ite
```

```
0.5779259410010127
```

```
In [ ]: list1 = [[1,2,3],[4,5,6,7],[8,9,10,11]]

        from itertools import chain
        result3 = list(chain.from_iterable(list1))
        print(result3)
```

```
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11]
```

```
In [ ]: from timeit import timeit
        print(timeit("list(chain.from_iterable(list1))", "from __main__ import li
```

```
0.6360882539993327
```

List Comprehension

- List comprehensions have been a feature of the Python programming language since its early versions. They were introduced in Python 2.0, which was released in October 2000. Guido van Rossum, the creator of Python, added list comprehensions as a way to make code more concise and expressive for common tasks involving iterating over and transforming elements in sequences.
- The introduction of list comprehensions was inspired by similar constructs in functional programming languages like Haskell and set-builder notation in mathematics. List comprehensions aimed to provide a more elegant and readable way to create lists by applying transformations and filters in a compact manner.
- List comprehensions were very well received by the Python community and became a fundamental part of the language. They not only simplified code but also promoted a more functional programming style in Python, encouraging developers to focus on the transformation of data rather than explicit looping and mutation.
- Since their introduction, list comprehensions have remained a core feature of Python and have been widely used by developers for various tasks involving sequence processing. Over time, other similar constructs like dictionary comprehensions and set comprehensions were introduced in Python, further extending the concept of comprehensions to other data structures.
- In summary, list comprehensions were introduced in Python 2.0 and have since become a popular and powerful feature, contributing to the readability and conciseness of Python code.

List comprehensions in Python offer several advantages and some potential disadvantages. Let's go through the pros and cons of using list comprehensions:

Pros:

- Readability: List comprehensions can make your code more concise and easier to read, especially for simple operations on iterables.
- Conciseness: List comprehensions allow you to express transformations and filters in a single line of code, reducing the overall code length.
- Performance: In some cases, list comprehensions can be faster than traditional loops because they are optimized internally by Python's interpreter.
- Functional Programming Style: List comprehensions promote a functional programming style by focusing on transformations and avoiding explicit mutation

of data.

*Easy to Understand for Simple Transformations: For straightforward operations, like mapping one set of values to another, list comprehensions are often easier to understand than a multi-line loop.

Cons:

- Readability for Complex Operations: For more complex operations, list comprehensions can become difficult to read and understand, reducing code maintainability.
- Limited Expressiveness: List comprehensions are great for one-liners, but they might not be suitable for more intricate tasks that require multiple steps or conditions.
- Debugging Complexity: When debugging, it can be harder to pinpoint issues within a complex list comprehension compared to a traditional loop.
- Limited to Creating Lists: List comprehensions are specifically designed for creating lists. If you need to create other data structures, like dictionaries or sets, you'll need to use other techniques.
- Less Explicit: While the concise nature of list comprehensions can be an advantage, it might also make your code less self-explanatory, especially for those unfamiliar with the concept.

Basic Syntax

- new_list = [expression for item in iterable if condition]
- expression: The operation you want to perform on each item in the iterable.
- item: The variable that takes on each value in the iterable.
- iterable: The collection of items you're iterating through.
- condition (optional): A filter that determines whether the item should be included in the new list based on a specified condition.

In [ ]:
```
# Without List comprehension
```

In [ ]:
```
num = [1,2,3,4,5,6,7,8,9,10]
results1 = []
for i in num:
    i = i * 9
    results1.append(i)

print(results1)
```

```
[9, 18, 27, 36, 45, 54, 63, 72, 81, 90]
```

In [ ]:
```
# Without List comprehension
```

In [ ]:
```
num = [1,2,3,4,5,6,7,8,9,10]
results2 = []
[results2.append(i * 9) for i in num]

print(results2)
```

```
[9, 18, 27, 36, 45, 54, 63, 72, 81, 90]
```

In [ ]:
```
# With List comprehension
```

In [ ]:
```
num = [1,2,3,4,5,6,7,8,9,10]

results3= [(i * 9) for i in num]
print(results3)
```

```
[9, 18, 27, 36, 45, 54, 63, 72, 81, 90]
```

In [ ]:
```
# Without List comprehension
```

In [ ]:
```
string = ["i","am","a","boy"]
results4 = []
for i in string:
    i = i.upper()
    results4.append(i)
print(results4)
```

```
['I', 'AM', 'A', 'BOY']
```

In [ ]:
```
# With List comprehension
```

In [ ]:
```
string = ["i","am","a","boy"]
results5= [(i.upper()) for i in string]

print(results5)
```

```
['I', 'AM', 'A', 'BOY']
```

In [ ]:
```
# With List comprehension
```

In [ ]:
```
results6 = [(i * 9) for i in [1,2,3,4,5,6,7,8,9,10]]
print(results6)
```

```
[9, 18, 27, 36, 45, 54, 63, 72, 81, 90]
```

In [ ]:
```
# Without List comprehension
```

In [ ]:
```
def square(x):
    return x * x

nums = [1,2,3,4,5,6,7,8,9,10]
results6 = []
for i in nums:
    a = square(i)
    results6.append(a)
print(results6)
```

```
[1, 4, 9, 16, 25, 36, 49, 64, 81, 100]
```

In [ ]:
```
# Without List comprehension
```

In [ ]:
```
nums = [1,2,3,4,5,6,7,8,9,10]
results6 = []
for i in nums:
    a = i * i
    results6.append(a)
print(results6)
```
```
[1, 4, 9, 16, 25, 36, 49, 64, 81, 100]
```

In [ ]:
```
# With List comprehension
```

In [ ]:
```
def square(x):
    return x * x

nums = [1,2,3,4,5,6,7,8,9,10]
results7= [square(i) for i in nums]
print(results7)
```
```
[1, 4, 9, 16, 25, 36, 49, 64, 81, 100]
```

In [ ]:
```
# With List comprehension
```

In [ ]:
```
nums = [1,2,3,4,5,6,7,8,9,10]
results17= [i * i for i in nums]
print(results17)
```
```
[1, 4, 9, 16, 25, 36, 49, 64, 81, 100]
```

In [ ]:
```
# With List comprehension
```

In [ ]:
```
def square(x):
    return x * x

results8= [square(i) for i in [1,2,3,4,5,6,7,8,9,10]]
print(results8)
```
```
[1, 4, 9, 16, 25, 36, 49, 64, 81, 100]
```

In [ ]:
```
# Without List comprehension
```

In [ ]:
```
def square(x):
    return x * x

nums = [1,2,3,4,5,6,7,8,9,10]
results9 = []
for i in nums:
    a = square(i)
    if a> 50:
        results9.append(a)
print(results9)
```
```
[64, 81, 100]
```

In [ ]:
```
# Withour List comprehension
```

In [ ]:
```
def square(x):
    return x * x

nums = [1,2,3,4,5,6,7,8,9,10]
results10 = []
```

```
for i in nums:
    if square(i)>50:
        results10.append(square(i))
print(results10)
```

```
[64, 81, 100]
```

In [ ]:
```
# With List comprehension
```

In [ ]:
```
def square(x):
    return x * x

nums = [1,2,3,4,5,6,7,8,9,10]
results11 = [square(i) for i in nums if square(i) > 50]
print(results11)
```

```
[64, 81, 100]
```

In [ ]:
```
# With List comprehension
```

In [ ]:
```
def square(x):
    return x * x

results12 = [square(i) for i in [1,2,3,4,5,6,7,8,9,10] if square(i) > 50]
print(results12)
```

```
[64, 81, 100]
```

In [ ]:
```
# Without List comprehension
```

In [ ]:
```
dicts = [{"name" : "Dawood"}, {"name" : "Ahmad"}, {"name" : "Salman"}]
results13 = []
for i in dicts:
    a = i["name"]
    results13.append(a)
print(results13)
```

```
['Dawood', 'Ahmad', 'Salman']
```

In [ ]:
```
# Without List comprehension
```

In [ ]:
```
dicts = [{"name" : "David"}, {"name" : "Kapathy"}, {"name" : "Hinton"}]
results14 = []
for i in dicts:
    a = i["name"]
    results14.append(a)
print(results14)
```

```
['David', 'Kapathy', 'Hinton']
```

In [ ]:
```
# Without List comprehension
```

In [ ]:
```
dicts = [{"name" : "David"}, {"name" : "Kapathy"}, {"name" : "Hinton"}]
results15 = []
for i in range(len(dicts)):
    results15.append(dicts[i]["name"])
print(results15)
```

```
['David', 'Kapathy', 'Hinton']
```

```
In [ ]: # With List comprehension
```

```
In [ ]: dicts = [{"name" : "David"}, {"name" : "Kapathy"}, {"name" : "Hinton"}]
        results16 = [dicts[i]["name"] for i in range(len(dicts))]
        print(results16)
```

```
['David', 'Kapathy', 'Hinton']
```

```
In [ ]: # With List comprehension
```

```
In [ ]: dicts = [{"name" : "David"}, {"name" : "Kapathy"}, {"name" : "Hinton"}]
        results17 = [i["name"] for i in dicts]
        print(results17)
```

```
['David', 'Kapathy', 'Hinton']
```

```
In [ ]:
```

```
In [ ]:
```

Removing empty List from a List

In [ ]:
```
list0  = [1,2,3,4,5,6,7,[],[],[1],[],"if","break", ""]
result0 = []
emptylists = []
for x in list0:
    if type(x) == list:
        if len(x) == 1:
            result0.append(x)
        else:
            emptylists.append(x)
    else:
        result0.append(x)

print("Non_Empty = ", result0, "\nEmpty lists = " , emptylists )
```

```
Non_Empty =  [1, 2, 3, 4, 5, 6, 7, [1], 'if', 'break', '']
Empty lists =  [[], [], []]
```

In [ ]:
```
list1  = [1,2,3,4,5,6,7,[],[],"if","break", ""]
result1 = [x for x in list1]
print(result1)
```

```
[1, 2, 3, 4, 5, 6, 7, [], [], 'if', 'break', '']
```

In [ ]:
```
list2  = [1,2,3,4,5,6,7,[],[],"if","break",""]
result2 = [x for x in list2 if x]
print(result2) # empty string will not print
```

```
[1, 2, 3, 4, 5, 6, 7, 'if', 'break']
```

In [ ]:
```
list2  = [1,2,3,4,5,6,7,[],[],"if","break", ""]
result2 = [x for x in list2 if x != []]
print(result2) # empty string will be printed
```

```
[1, 2, 3, 4, 5, 6, 7, 'if', 'break', '']
```

In [ ]:
```
list2  = [1,2,3,4,5,6,7,[],[],"if","break", ""]
result2 = [x for x in list2 if x == []]
print(result2) # just empty lists will be printed
```

```
[[], []]
```

In [ ]:
```
# Using Filter Function
```

In [ ]:
```
list3  = [1,2,3,4,5,6,7,[],[],"if","break", ""]
# list(filter(callable , iterable))
result3 = list(filter(None , list3)) # Will remove empty string too
result3
```

Out[ ]:
```
[1, 2, 3, 4, 5, 6, 7, 'if', 'break']
```

In [ ]:
```
list3  = [1,2,3,4,5,6,7,[],[],"if","break", ""]
# list(filter(callable , iterable))
result3 = list(filter(lambda x : x , list3)) # Will remove empty string t
result3
```

Out[ ]:
```
[1, 2, 3, 4, 5, 6, 7, 'if', 'break']
```

In [ ]:
```
list3  = [1,2,3,4,5,6,7,[],[],"if","break", ""]
# list(filter(callable , iterable))
result3 = list(filter(lambda x : x != [] , list3)) # Will not remove empt
result3
```

Out[ ]:
```
[1, 2, 3, 4, 5, 6, 7, 'if', 'break', '']
```

In [ ]:
```
list3  = [1,2,3,4,5,6,7,[],[],"if","break", ""]
# list(filter(callable , iterable))
result3 = list(filter(lambda x : x == [] , list3)) # Will print only empt
result3
```

Out[ ]:
```
[[], []]
```

In [ ]:

In [ ]: