

# M2 Practice Problems

Week 1 | Day 1 | **Python List Comprehensions**

# How to Write a List Comprehension



```
def multiply_by_two(numbers):  
    results = []  
    for num in numbers:  
        results.append(num * 2)  
    return results
```

Diagram illustrating the transformation of a traditional loop into a list comprehension:

- The `for num in numbers:` loop is mapped to the `for num in numbers` part of the list comprehension.
- The `results.append(num * 2)` operation is mapped to the `num * 2` expression in the list comprehension.
- The `return results` statement is mapped to the `return` keyword in the list comprehension.

```
def multiply_by_two(numbers):  
    return [num * 2 for num in numbers]
```



```
def multiply_by_two(numbers):  
    1 STEP results = []  
    for num in numbers:  
        results.append(num * 2)  
    return results
```

```
def multiply_by_two(numbers):  
    1 STEP return [num * 2 for num in numbers]
```



## Step One:

- Define the data structure being returned:
  - Is it a list? [ ]
  - Is it a list of dictionaries? [ { }, { } ]



```
def multiply_by_two(numbers):  
    results = []  
    for num in numbers:  
        results.append(num * 2)  
    return results
```

```
def multiply_by_two(numbers):  
    return [num * 2 for num in numbers]
```

2  
STEP



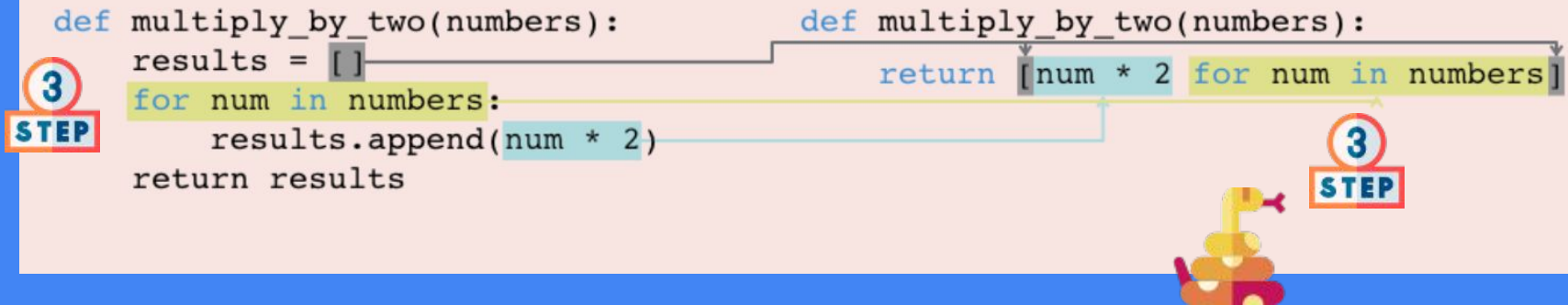
2  
STEP



Step Two:

Define the operation that is being iterated on each item in the list





3  
STEP



## Step Three:



- Afterwards, add the iterable expression (for loop)



# List Comprehensions



- **Syntactic Sugar:** elegant one-liner
  - Doesn't really change the underlying for loop mechanism or time complexity
- **Returns a new list**
  - Old list is unchanged
- **Python comprehensions work on any iterable:**
  - List, Object, Tuple, Set, range( ), etc.

