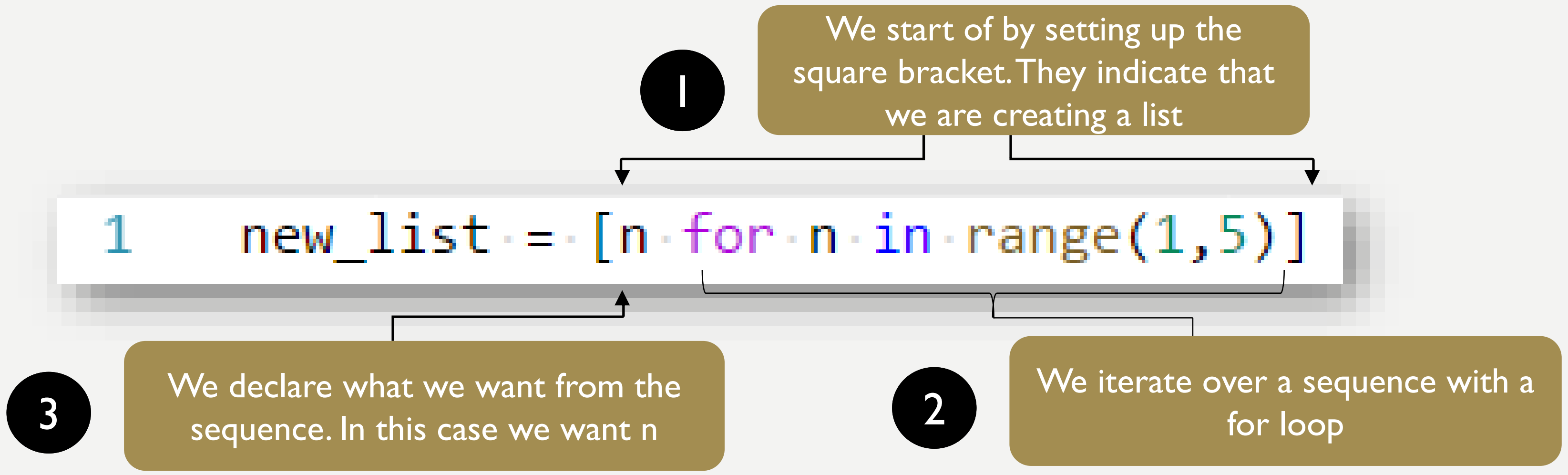
LIST COMPREHENSION

LIST COMPREHENSION

- List comprehension is one of Python's greatest strengths
  - It is a compact way to create lists



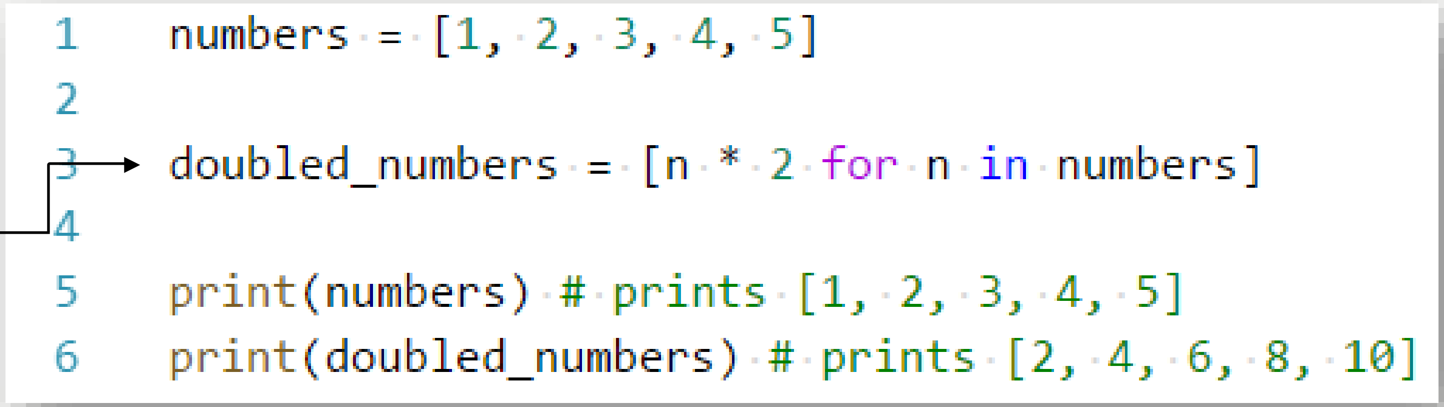
```
new_list = [n for n in range(1,5)]
```

1
We start of by setting up the square bracket. They indicate that we are creating a list
2
We iterate over a sequence with a for loop
3
We declare what we want from the sequence. In this case we want n

LIST COMPREHENSION

- It is very important to remember that list comprehension creates a new list!

Here we create a new list with all numbers from the list numbers doubled!

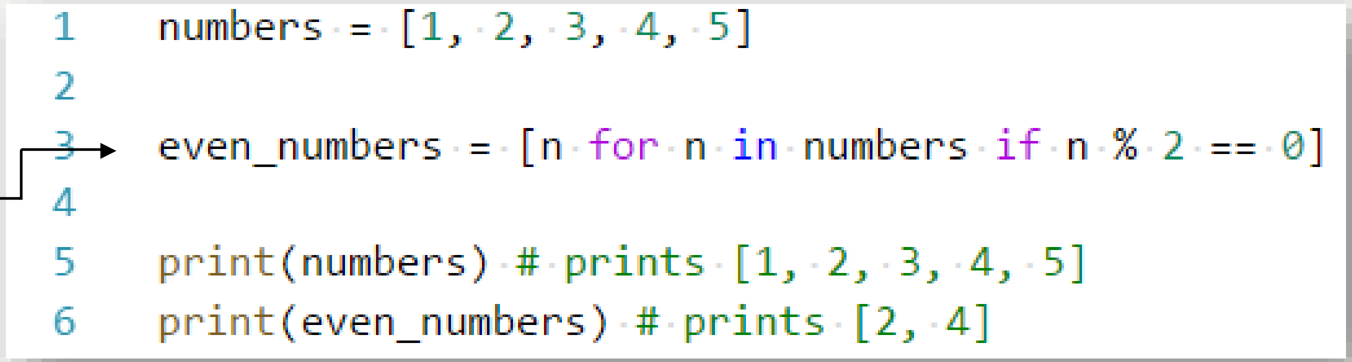


```python
numbers = [1, 2, 3, 4, 5]

doubled_numbers = [n * 2 for n in numbers]

print(numbers) # prints [1, 2, 3, 4, 5]
print(doubled_numbers) # prints [2, 4, 6, 8, 10]
```

LIST COMPREHENSION

- We can filter the values we want to have in our resulting list with an if statement

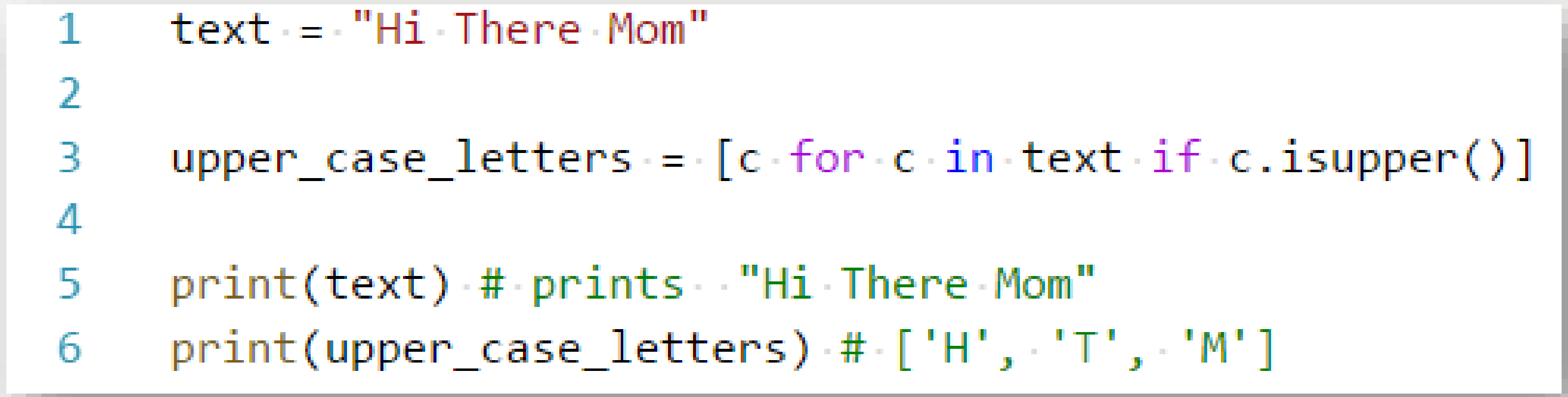
The list even numbers will only contain the even numbers for the list numbers because we use the if statement



```
numbers = [1, 2, 3, 4, 5]

even_numbers = [n for n in numbers if n % 2 == 0]

print(numbers) # prints [1, 2, 3, 4, 5]
print(even_numbers) # prints [2, 4]
```

LIST COMPREHENSION

- Here is another example of how we can use the if statement inside a list comprehension
  - Do note that now we are iterating over a string and filtering the upper case letters and storing them in the list upper_case_letters



```python
text = "Hi There Mom"

upper_case_letters = [c for c in text if c.isupper()]

print(text) # prints  "Hi There Mom"
print(upper_case_letters) # ['H', 'T', 'M']
```

TUPLES

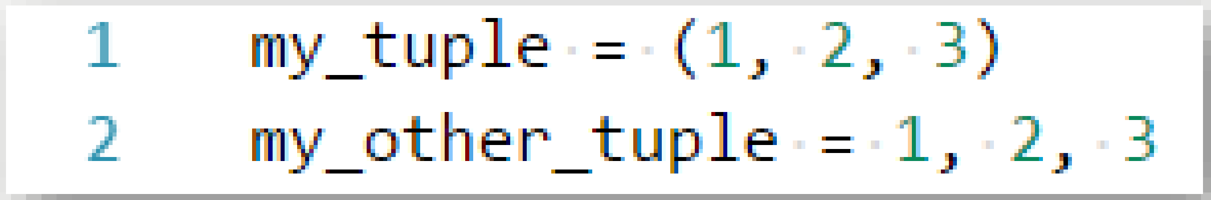
TUPLES

- Tuples are basically immutable lists
  - That means they cannot change!
- Lists use the square brackets but tuples use the parentheses
- Values of tuples are seperated by a comma just like values of a list

TUPLES

- We can create tuples like this

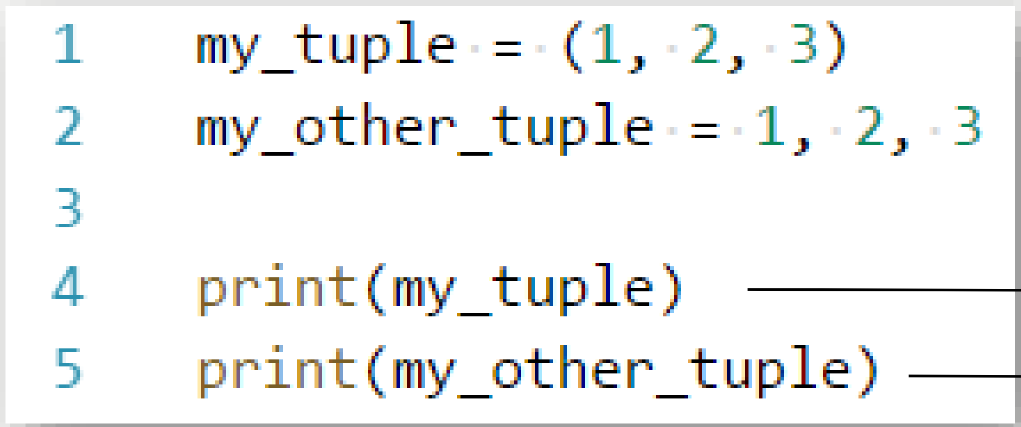
We can use the parantheses to indicate that the variable stores a tuple or we can omit the parentheses and



```
my_tuple = (1, 2, 3)
my_other_tuple = 1, 2, 3
```

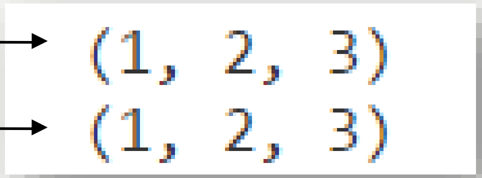
TUPLES

- Printing tuples



```python
my_tuple = (1, 2, 3)
my_other_tuple = 1, 2, 3

print(my_tuple)
print(my_other_tuple)
```

This is how a tuple will be displayed when printed



```
(1, 2, 3)
(1, 2, 3)
```

TUPLES

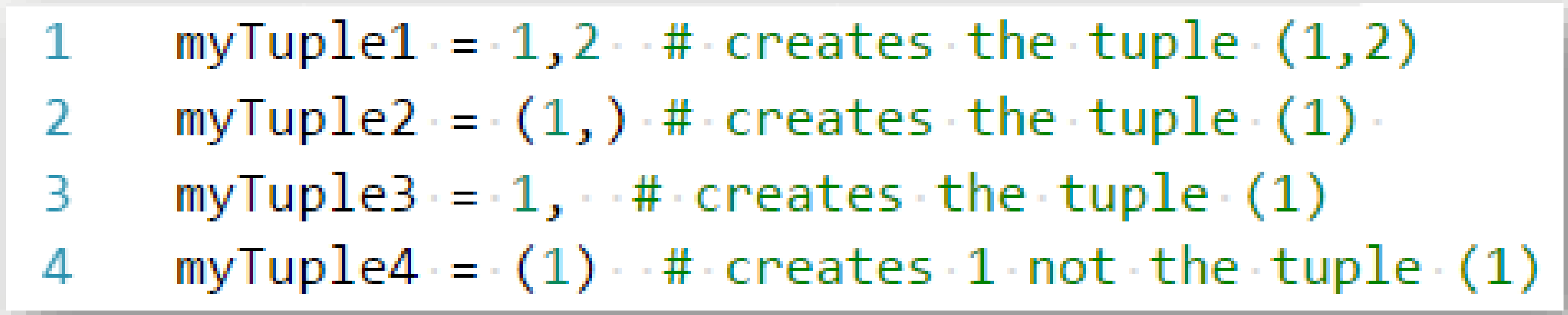
- Why have an immutable list(a tuple) an a mutable list as separate types?
  - An immutable list(a tuple) gives you a data structure with some integrity, some permanent-ness if you will
  - You have a guarantee that you will not accidentally change one

TUPLES

- Everything that works with a list works with a tuple ***except*** methods that modify the tuple
- That means that **indexing, slicing, len and print** all work as expected
- However, ***none*** of the mutable methods work
  - `append`
  - `insert`
  - `extend`
  - `del`
  - `pop`

TUPLES

- Take a look at the figure below, it shows various ways of creating tuples
  - Pay special attention to how to create a tuple with a single value



```
myTuple1 = 1,2  # creates the tuple (1,2)
myTuple2 = (1,) # creates the tuple (1)
myTuple3 = 1,  # creates the tuple (1)
myTuple4 = (1)  # creates 1 not the tuple (1)
```

TUPLES

- Tuples are :
  - Efficient with respect to us (some algorithm)
  - Efficient with respect to the amount of space used
  - Efficient with respect to the time it takes to perform some operations