VARIABLE SCOPE

The **variable scope** is the region of the code where the variable was assigned

1. **Local scope** – variables created inside of a function
    - *These cannot be referenced outside of the function*
2. **Global scope** – variables created outside of functions
    - *These can be referenced both inside and outside of functions*

```
def concatenator(*words):
    global sentence
    sentence = ''
    for word in words:
        sentence += word + ' '
    last_word = words[-1]
    return sentence.rstrip(), last_word

concatenator('Hello', 'world!', 'How', 'are', 'you?')
print(sentence)
```

```
Hello world! How are you?
```

Since the variable 'sentence' is assigned inside of the concatenator function, it has local scope

Trying to print this variable outside of the function will then return a NameError

CHANGING VARIABLE SCOPE

You can **change variable scope** by using the *global* keyword

```
def concatenator(*words):
    global sentence
    sentence = ''
    for word in words:
        sentence += word + ' '
    last_word = words[-1]
    return sentence.rstrip(), last_word

concatenator('Hello', 'world!', 'How', 'are', 'you?')
print(sentence)
```

```
Hello world! How are you?
```

By declaring the variable 'sentence' as global, it is now recognized outside of the function it was defined in

CHANGING VARIABLE SCOPE

You can **change variable scope** by using the *global* keyword

```
def concatenator(*words):
    sentence = ''
    for word in words:
        sentence += word + ' '
    last_word = words[-1]
    global sentence
    return sentence.rstrip(), last_word

concatenator('Hello', 'world!', 'How', 'are', 'you?')
print(sentence)
```

```
SyntaxError: name 'sentence' is assigned to before global declaration
```

*Note that the variable must be declared as global **before it is assigned a value**, or you will receive a SyntaxError*

PRO TIP: While it might be tempting, declaring global variables within a function is considered bad practice in most cases – imagine if you borrowed this code and it overwrote an important variable! Instead, use 'return' to deliver the values you want and assign them to local variables