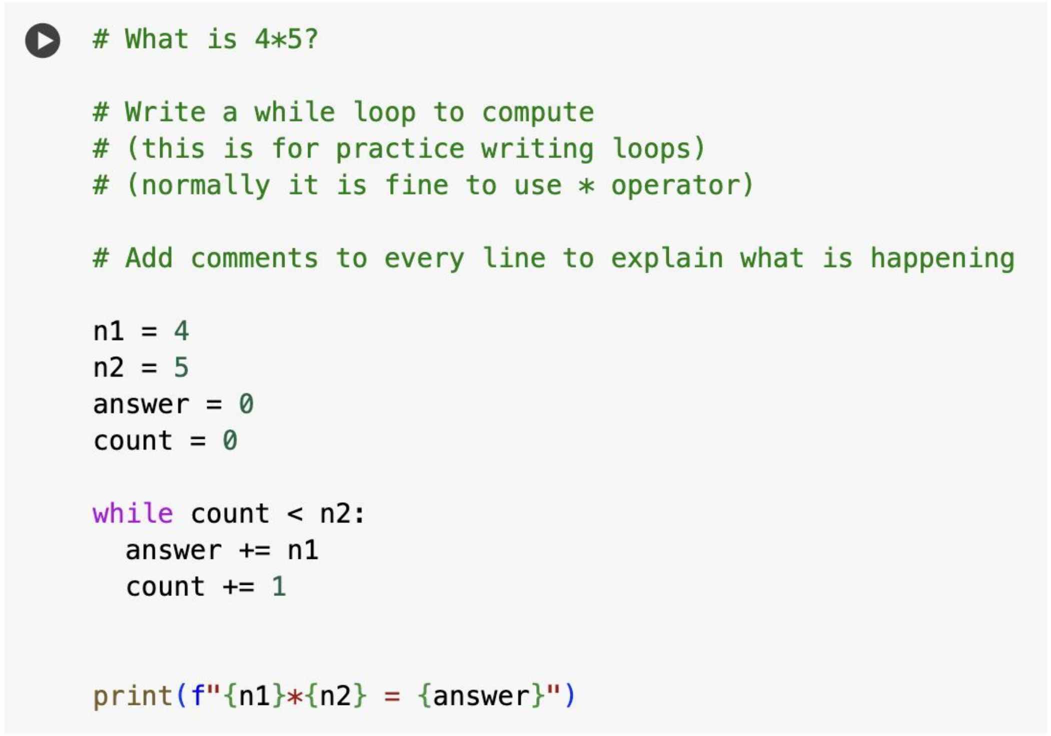
# Python Foundation

Writing functions

## From Last Week

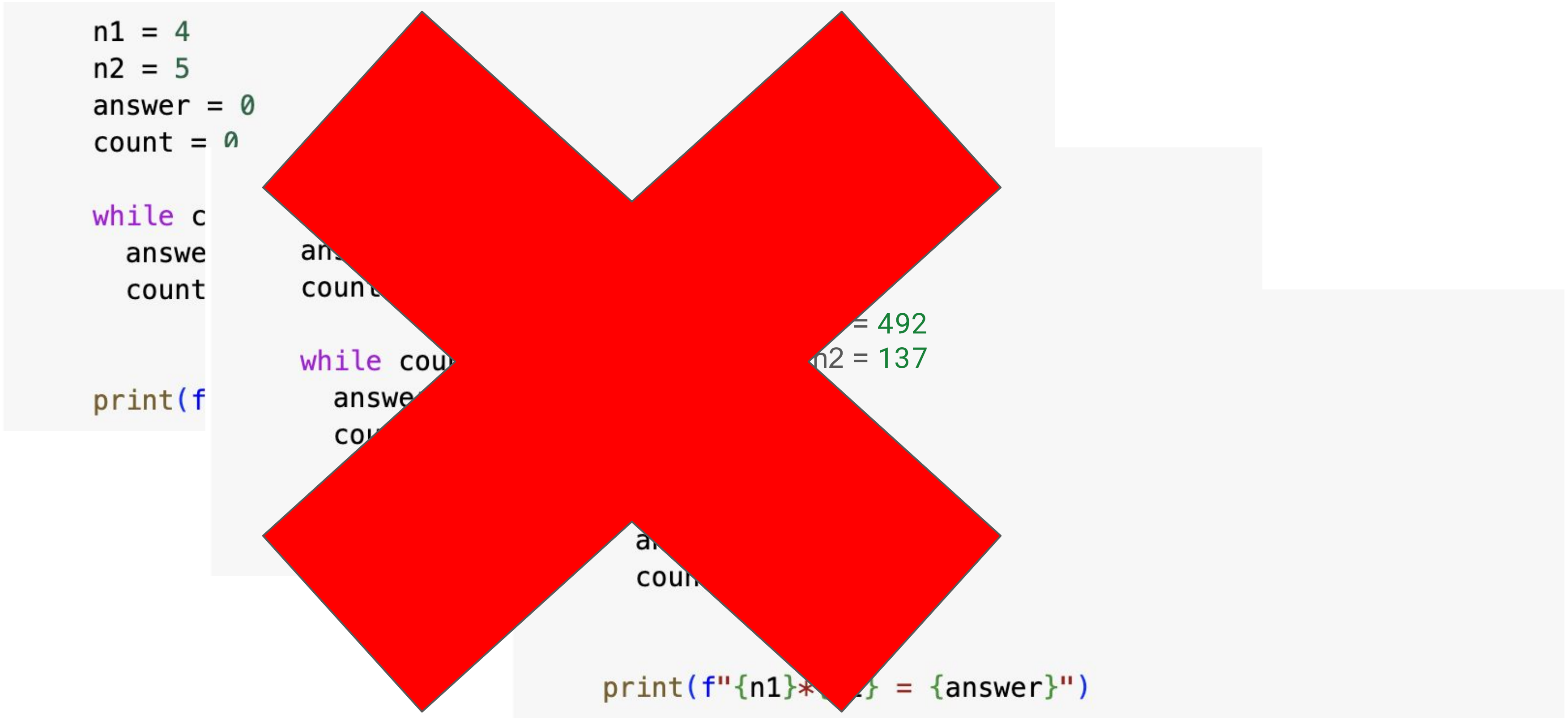


```
# What is 4*5?

# Write a while loop to compute
# (this is for practice writing loops)
# (normally it is fine to use * operator)

# Add comments to every line to explain what is happening

n1 = 4
n2 = 5
answer = 0
count = 0

while count < n2:
  answer += n1
  count += 1


print(f"{n1}*{n2} = {answer}")
```

What does this code do?

What values would I change to solve this problem: 7*12?

What values would I change to solve this problem: 492*137?



```
n1 = 4
n2 = 5
answer = 0
count = 0

while c
  answe
  count


print(f
```

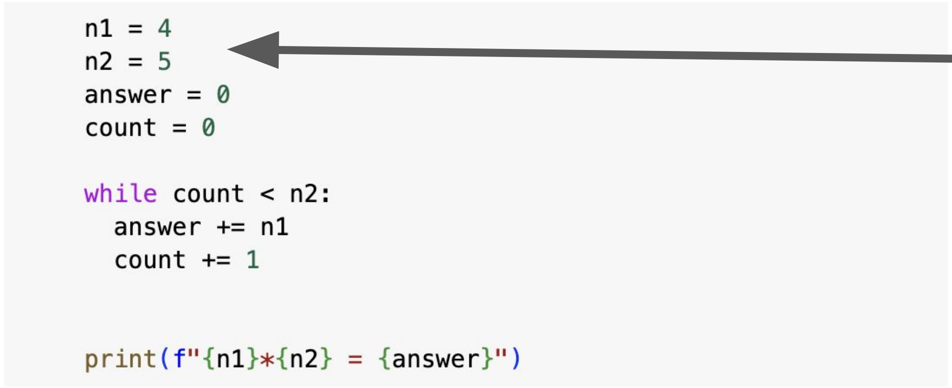
```
an
count

while cou
  answe
  co
```

```
= 492
n2 = 137

  a
  coun


print(f"{n1}*      } = {answer}")
```

**Copying or duplicating code should be avoided**

# Write a function

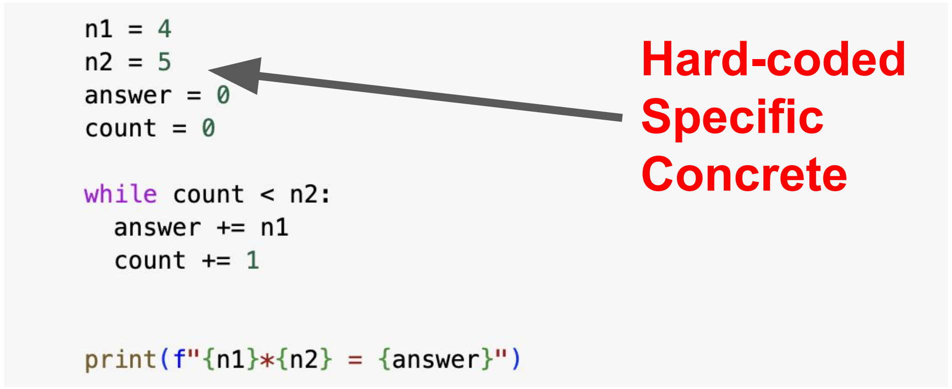
- N.B. coding jargon
- Functions turn code into **generalizable** instructions about a certain task
- possible tasks:
  - multiply two numbers using addition in a while loop
  - multiply two numbers together using addition in a for loop



```python
n1 = 4
n2 = 5
answer = 0
count = 0

while count < n2:
  answer += n1
  count += 1


print(f"{n1}*{n2} = {answer}")
```

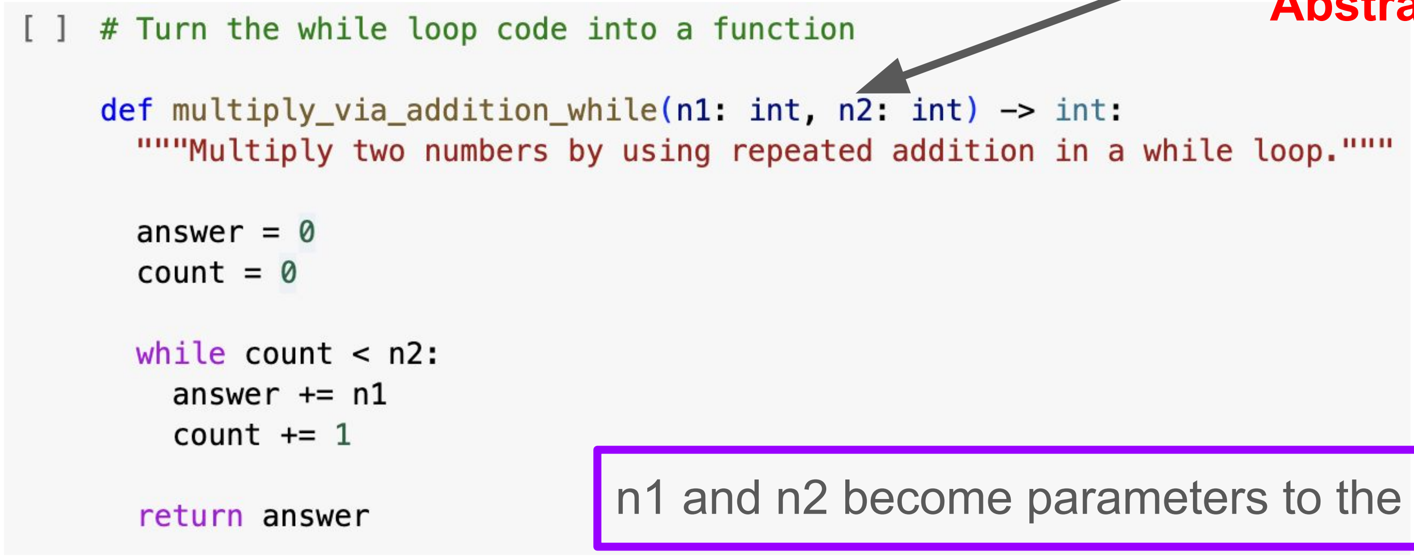
**Hard-coded!**

## Compare



```
n1 = 4
n2 = 5
answer = 0
count = 0

while count < n2:
  answer += n1
  count += 1


print(f"{n1}*{n2} = {answer}")
```

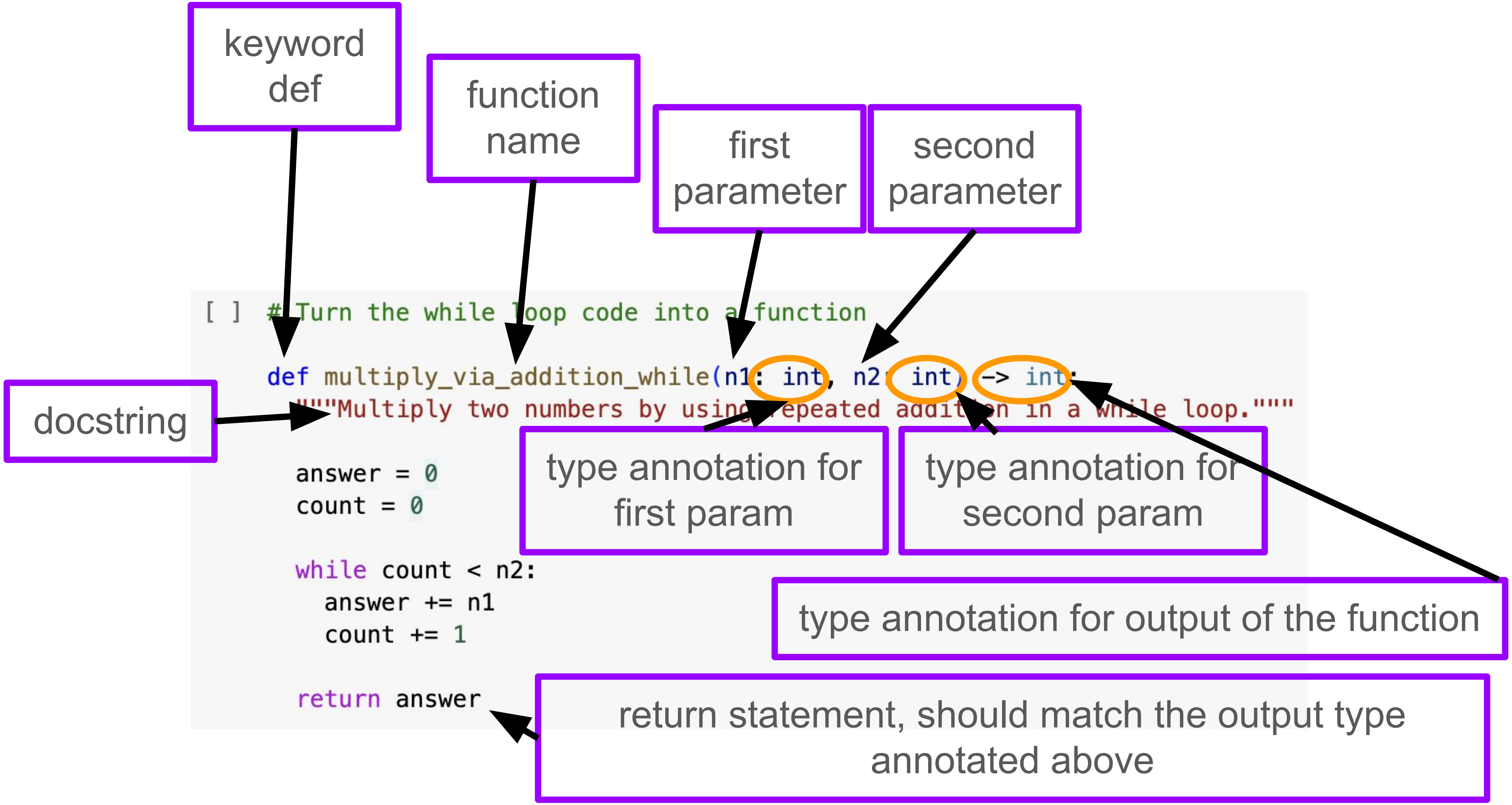
**Hard-coded**
**Specific**
**Concrete**



```
[ ]  # Turn the while loop code into a function

     def multiply_via_addition_while(n1: int, n2: int) -> int:
       """Multiply two numbers by using repeated addition in a while loop."""

       answer = 0
       count = 0

       while count < n2:
         answer += n1
         count += 1

       return answer
```

**Generic,**
**Unspecified**
**Abstract**

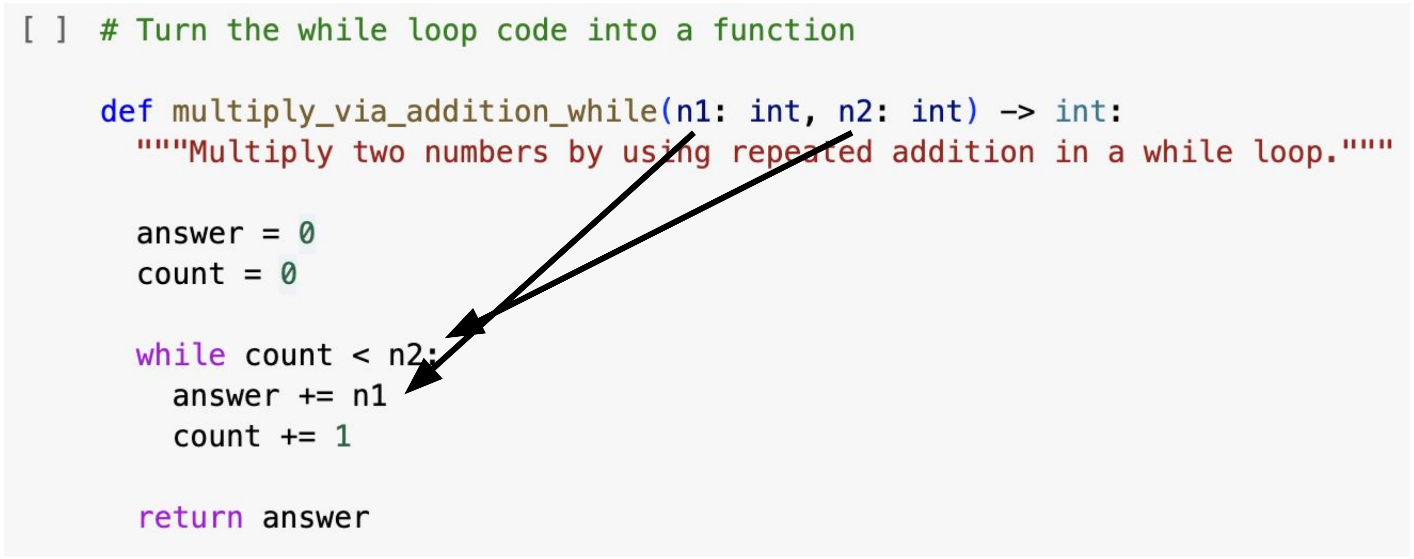
n1 and n2 become parameters to the function


keyword def
function name
first parameter
second parameter
docstring
type annotation for first param
type annotation for second param
type annotation for output of the function
return statement, should match the output type annotated above

```
[ ] # Turn the while loop code into a function

    def multiply_via_addition_while(n1: int, n2: int) -> int:
      """Multiply two numbers by using repeated addition in a while loop."""

      answer = 0
      count = 0

      while count < n2:
        answer += n1
        count += 1

      return answer
```

Parameters are <u>used</u> inside the function

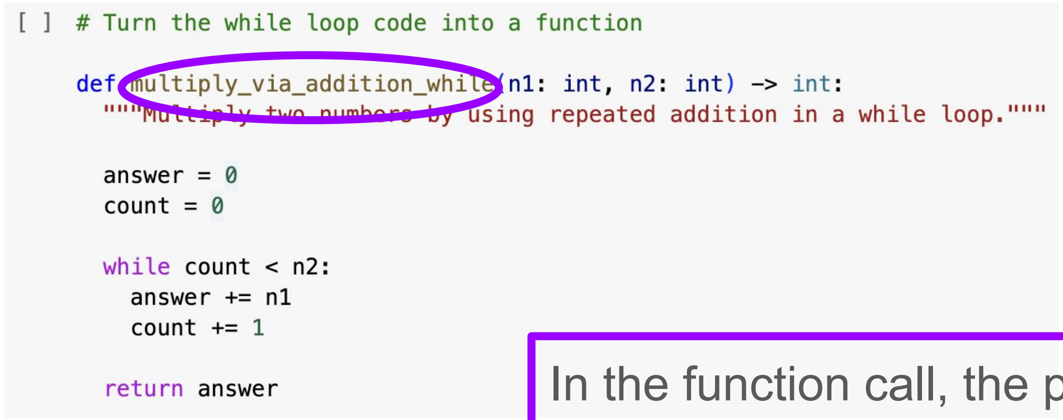
Parameters are <u>not</u> <u>created</u> inside the function



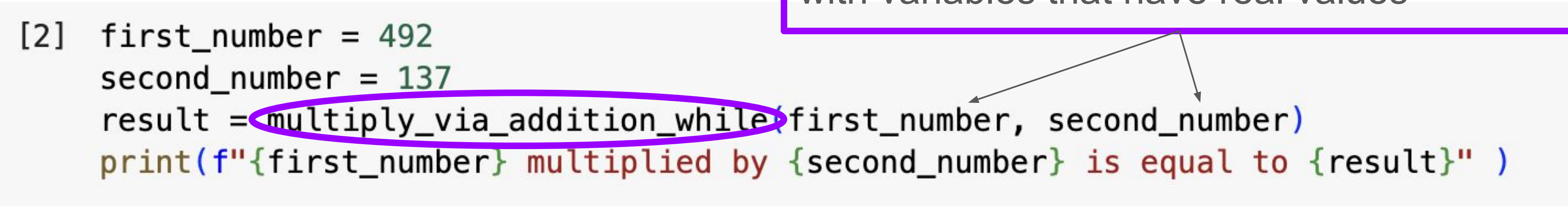
```
[ ]  # Turn the while loop code into a function

     def multiply_via_addition_while(n1: int, n2: int) -> int:
       """Multiply two numbers by using repeated addition in a while loop."""

       answer = 0
       count = 0

       while count < n2:
         answer += n1
         count += 1

       return answer
```

## Calling a function

- Functions do nothing on their own. They are a set of instructions
- Functions must be "called" in order for the instructions to get executed



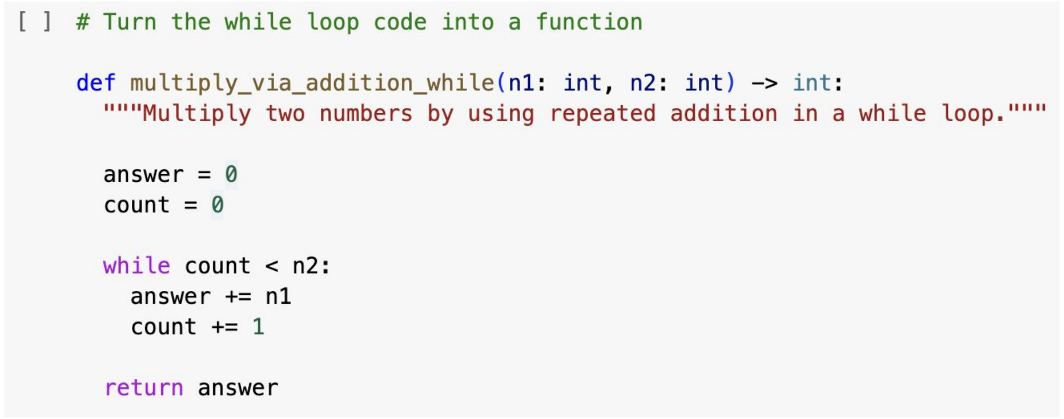
```
[ ] # Turn the while loop code into a function

    def multiply_via_addition_while(n1: int, n2: int) -> int:
      """Multiply two numbers by using repeated addition in a while loop."""

      answer = 0
      count = 0

      while count < n2:
        answer += n1
        count += 1

      return answer
```



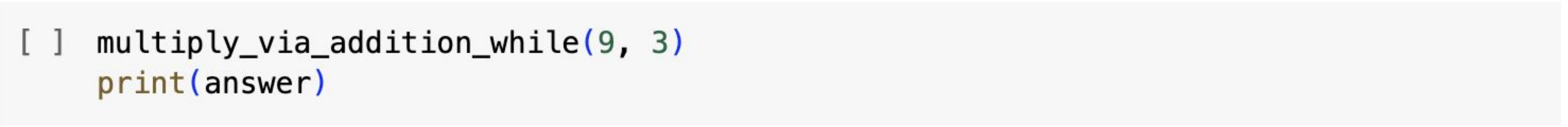
```
[2] first_number = 492
    second_number = 137
    result = multiply_via_addition_while(first_number, second_number)
    print(f"{first_number} multiplied by {second_number} is equal to {result}" )
```

In the function call, the parameters are filled with variables that have real values

## What is wrong with the following code?

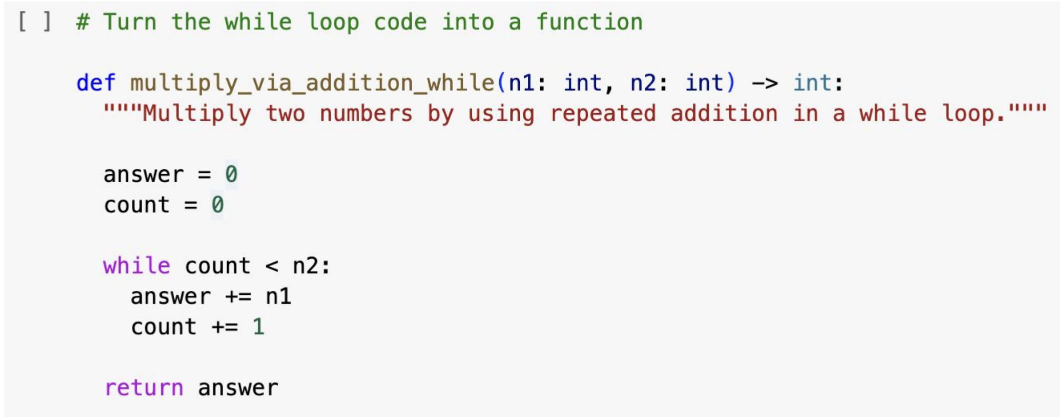


```
[ ]  # Turn the while loop code into a function

     def multiply_via_addition_while(n1: int, n2: int) -> int:
       """Multiply two numbers by using repeated addition in a while loop."""

       answer = 0
       count = 0

       while count < n2:
         answer += n1
         count += 1

       return answer
```

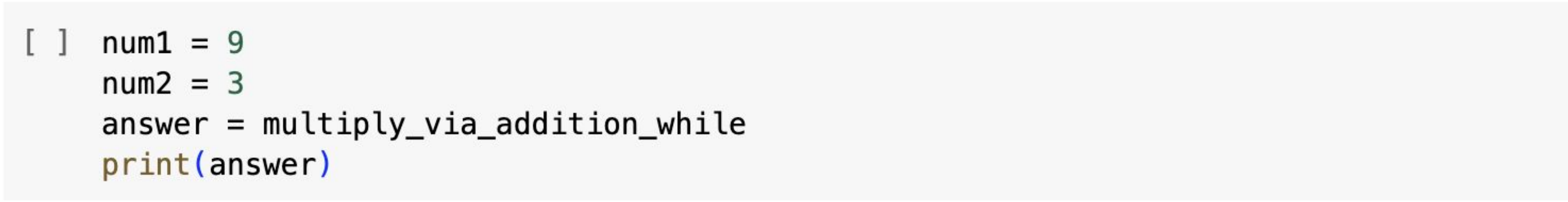


```
[ ]  multiply_via_addition_while(9, 3)
     print(answer)
```

## What is wrong with the following code?

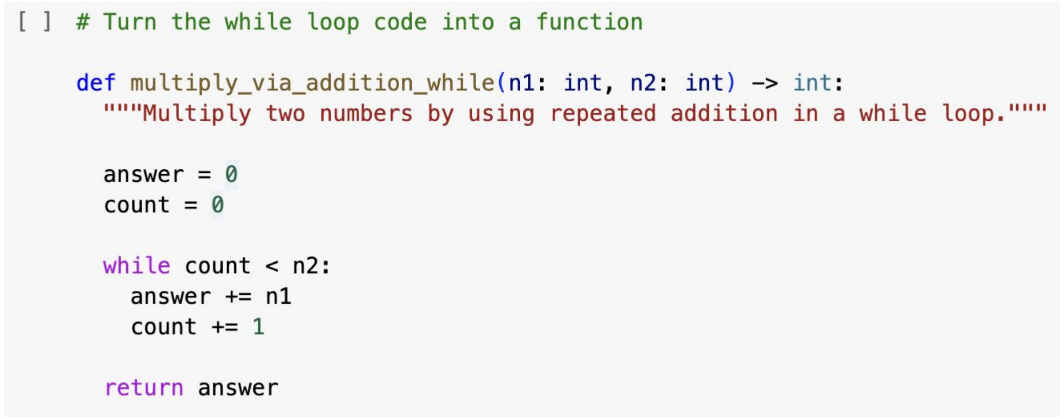


```python
# Turn the while loop code into a function

def multiply_via_addition_while(n1: int, n2: int) -> int:
  """Multiply two numbers by using repeated addition in a while loop."""

  answer = 0
  count = 0

  while count < n2:
    answer += n1
    count += 1

  return answer
```

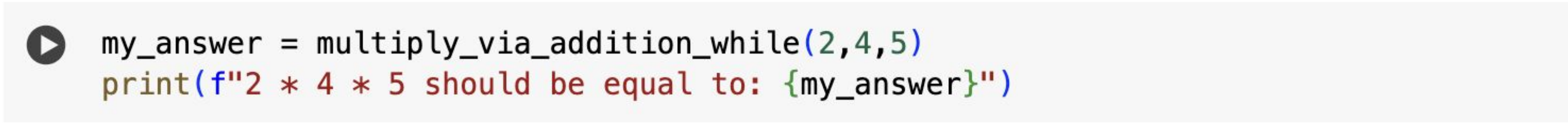


```python
num1 = 9
num2 = 3
answer = multiply_via_addition_while
print(answer)
```

## What is wrong with the following code?



```python
# Turn the while loop code into a function

def multiply_via_addition_while(n1: int, n2: int) -> int:
  """Multiply two numbers by using repeated addition in a while loop."""

  answer = 0
  count = 0

  while count < n2:
    answer += n1
    count += 1

  return answer
```



```python
my_answer = multiply_via_addition_while(2,4,5)
print(f"2 * 4 * 5 should be equal to: {my_answer}")
```

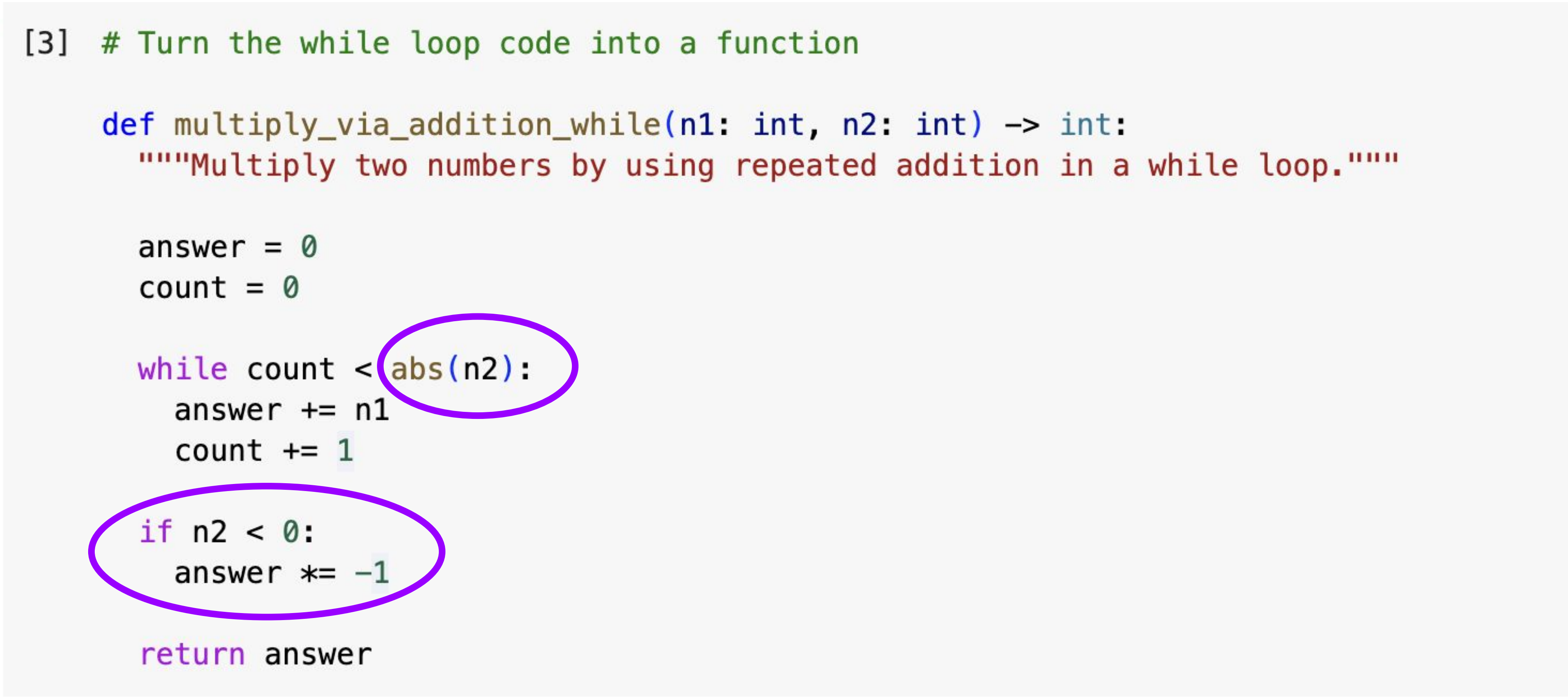
## Activity: Write and call a function

- Write a function that does multiplication of two numbers via addition inside a for loop
- Assume all inputs will be integers
- Annotate the function accordingly
- Don't forget the docstring

- After coding the function, call the function with two inputs of your choice
- Assign the return value into a variable
- Print out the previous variable

Required Check: https://forms.gle/9N91F8RGYpWpDuAZ9

## Why have these changes been made?



```
[3] # Turn the while loop code into a function

    def multiply_via_addition_while(n1: int, n2: int) -> int:
      """Multiply two numbers by using repeated addition in a while loop."""

      answer = 0
      count = 0

      while count < abs(n2):
        answer += n1
        count += 1

      if n2 < 0:
        answer *= -1

      return answer
```