THINK PYTHON

CHAPTER THREE (26/04/2020)

trazoM’s Assignment

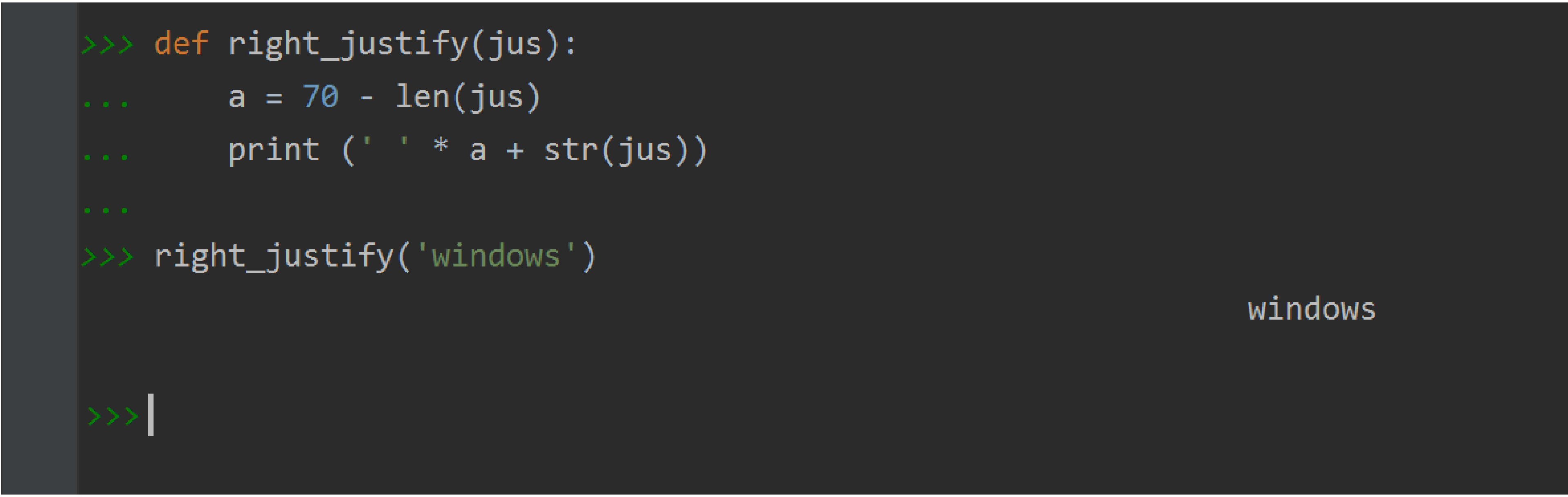
Exercise 3 - 1

1. Write a function named right_justify that takes a string named s as a parameter and prints the string with enough leading spaces so that the last letter of the string is in column 70 of the display:

```
>>> right_justify('monty')
                                                                 monty
```

Hint: Use string concatenation and repetition. Also, Python provides a built-in function called len that returns the length of a string, so the value of len('monty') is 5.

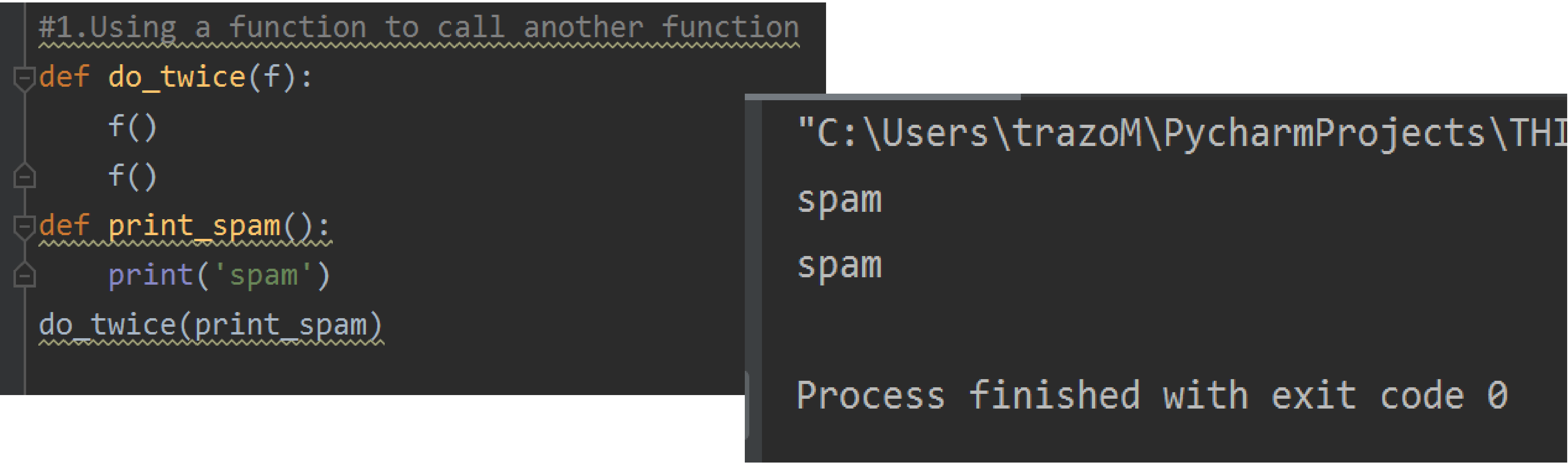
Solution:



```
>>> def right_justify(jus):
...     a = 70 - len(jus)
...     print (' ' * a + str(jus))
...
>>> right_justify('windows')
                                                               windows

>>> 
```

Exercise 3-2.

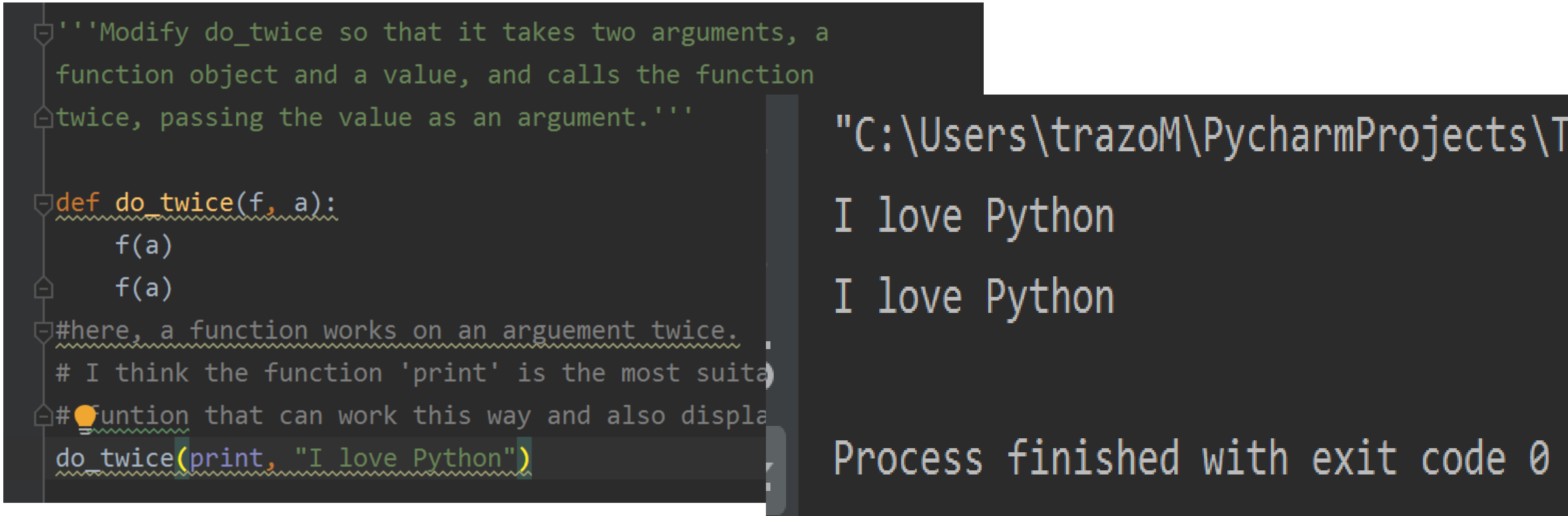
1. Using a funtion to call another function twice.



```
#1.Using a function to call another function
def do_twice(f):
    f()
    f()
def print_spam():
    print('spam')
do_twice(print_spam)
```

```
"C:\Users\trazoM\PycharmProjects\THI
spam
spam

Process finished with exit code 0
```

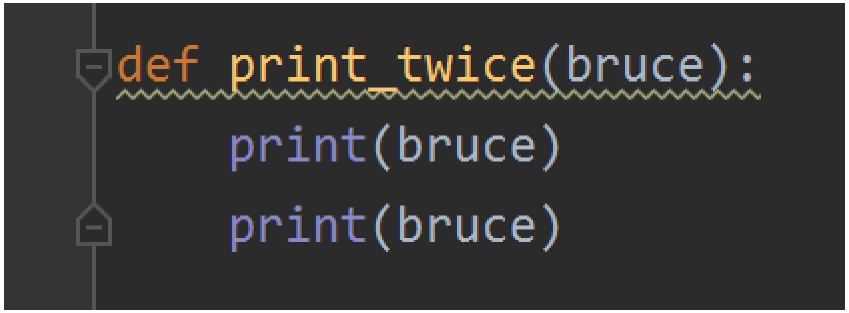
2. Modify do_twice so that it takes two arguments, a function object and a value, and calls the function twice, passing the value as an argument.



```
'''Modify do_twice so that it takes two arguments, a
function object and a value, and calls the function
twice, passing the value as an argument.'''

def do_twice(f, a):
    f(a)
    f(a)
#here, a function works on an arguement twice.
# I think the function 'print' is the most suita
#funtion that can work this way and also displa
do_twice(print, "I love Python")
```

```
"C:\Users\trazoM\PycharmProjects\T
I love Python
I love Python

Process finished with exit code 0
```

3. Copy the definition of `print_twice` from earlier in this chapter to your script.



```
def print_twice(bruce):
    print(bruce)
    print(bruce)
```

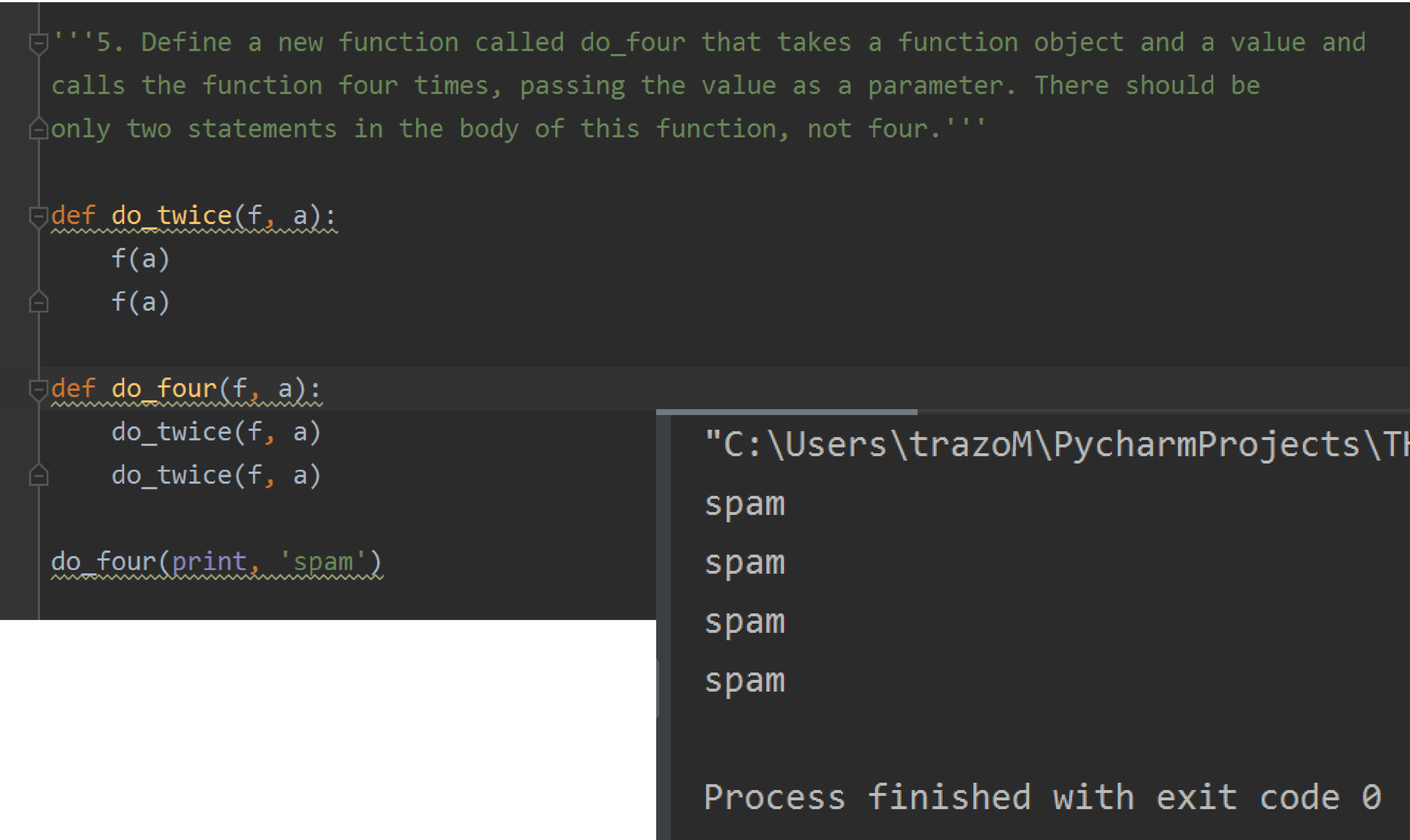
4. Use the modified version of `do_twice` to call `print_twice` twice, passing 'spam‘ as an argument.

This is not possible because;
`do_twice` needs a function and an argument and `print_twice` also needs an argument.

Passing a function (`print_twice`) which requires an argument (`bruce`) into another function (`do_twice`) which requires a function but not a function with an argument will not work. Only the function `print` can work because `print` in itself remains a function even without an argument.

Hence, to achieve this, a fresh function needs to be created. The author has called it `do_four`.

5. Define a new function called `do_four` that takes a function object and a value and calls the function four times, passing the value as a parameter. There should be only two statements in the body of this function, not four.



```
'''5. Define a new function called do_four that takes a function object and a value and
calls the function four times, passing the value as a parameter. There should be
only two statements in the body of this function, not four.'''

def do_twice(f, a):
    f(a)
    f(a)

def do_four(f, a):
    do_twice(f, a)
    do_twice(f, a)

do_four(print, 'spam')
```

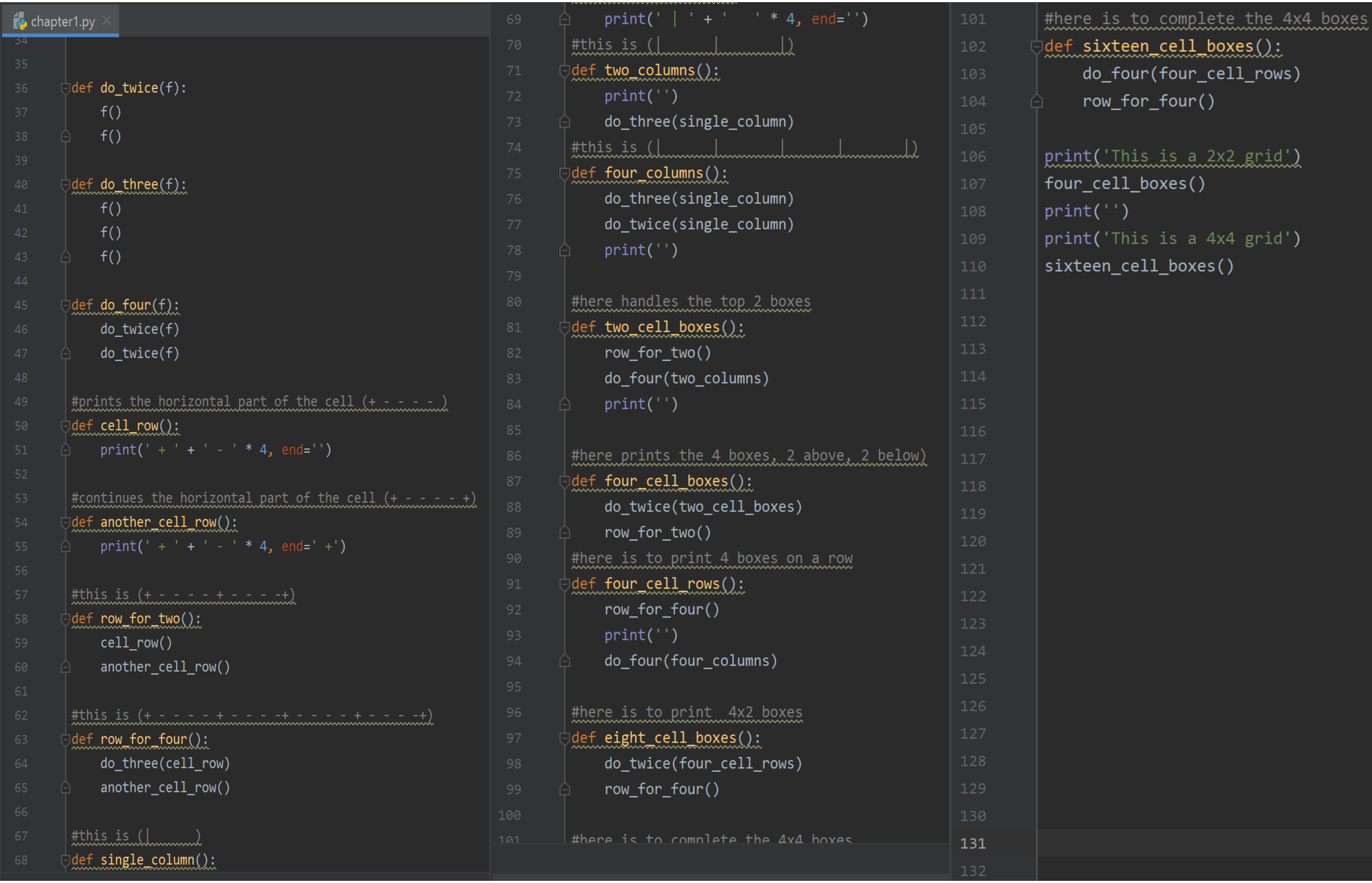
```
"C:\Users\trazoM\PycharmProjects\T
spam
spam
spam
spam

Process finished with exit code 0
```

Exercise 3.3

1. Write a function that draws a grid like the following:

```
+ - - - - + - - - - +
|         |         |
|         |         |
|         |         |
|         |         |
+ - - - - + - - - - +
|         |         |
|         |         |
|         |         |
|         |         |
+ - - - - + - - - - +
```

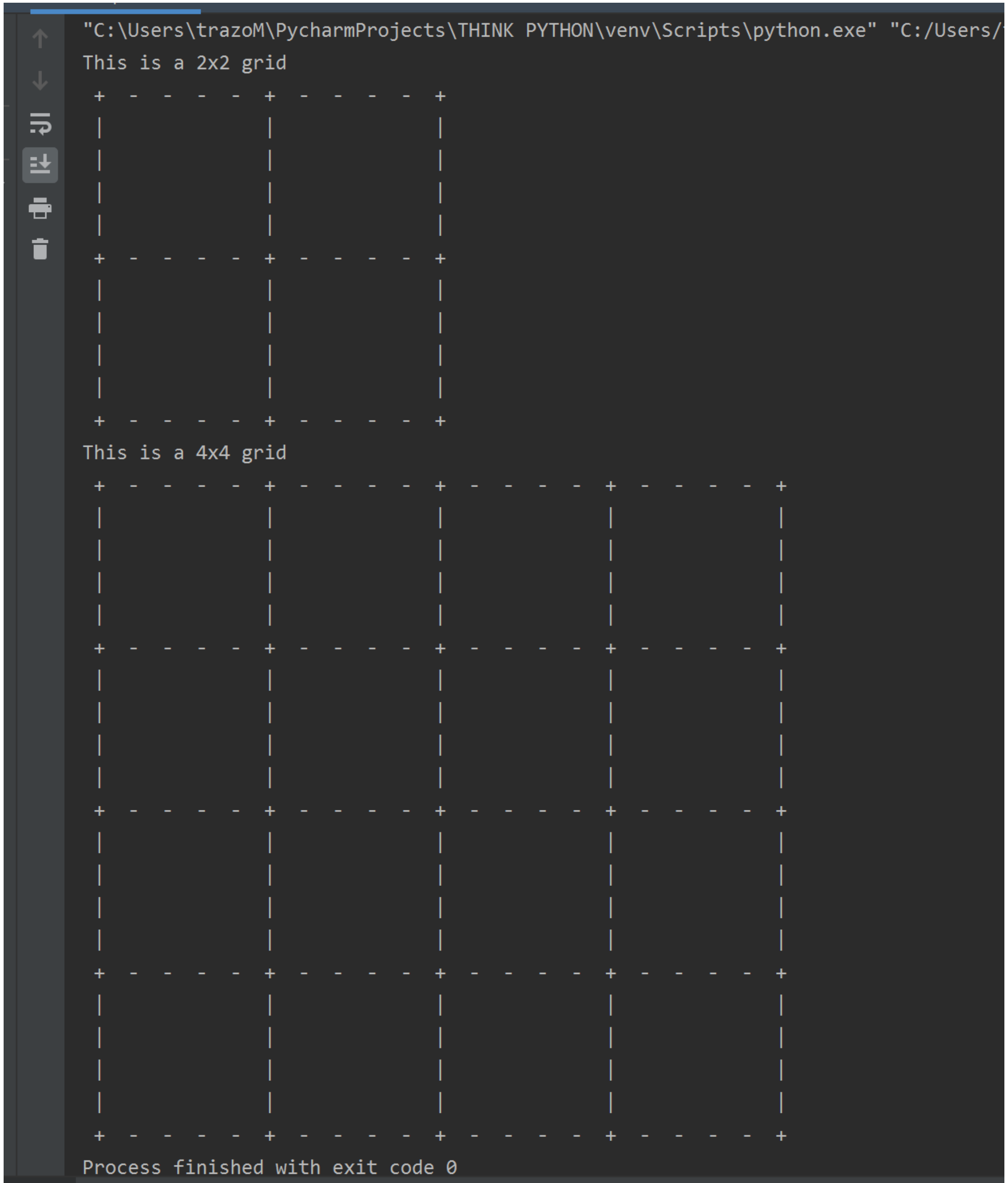
Drawing grids



```python

def do_twice(f):
    f()
    f()

def do_three(f):
    f()
    f()
    f()

def do_four(f):
    do_twice(f)
    do_twice(f)

#prints the horizontal part of the cell (+ - - - - )
def cell_row():
    print(' + ' + ' - ' * 4, end='')

#continues the horizontal part of the cell (+ - - - - +)
def another_cell_row():
    print(' + ' + ' - ' * 4, end=' +')

#this is (+ - - - - + - - - -+)
def row_for_two():
    cell_row()
    another_cell_row()

#this is (+ - - - - + - - - -+ - - - - + - - - -+)
def row_for_four():
    do_three(cell_row)
    another_cell_row()

#this is (|______)
def single_column():
    print(' | ' + '    ' * 4, end='')
#this is (|______|_______|)
def two_columns():
    print('')
    do_three(single_column)
#this is (|______|_______|______|_______|)
def four_columns():
    do_three(single_column)
    do_twice(single_column)
    print('')

#here handles the top 2 boxes
def two_cell_boxes():
    row_for_two()
    do_four(two_columns)
    print('')

#here prints the 4 boxes, 2 above, 2 below)
def four_cell_boxes():
    do_twice(two_cell_boxes)
    row_for_two()
#here is to print 4 boxes on a row
def four_cell_rows():
    row_for_four()
    print('')
    do_four(four_columns)

#here is to print  4x2 boxes
def eight_cell_boxes():
    do_twice(four_cell_rows)
    row_for_four()

#here is to complete the 4x4 boxes
def sixteen_cell_boxes():
    do_four(four_cell_rows)
    row_for_four()

print('This is a 2x2 grid')
four_cell_boxes()
print('')
print('This is a 4x4 grid')
sixteen_cell_boxes()
```

You can copy/download this code here and try to run it yourself.

My result



```
"C:\Users\trazoM\PycharmProjects\THINK PYTHON\venv\Scripts\python.exe" "C:/Users/
This is a 2x2 grid
 + - - - - + - - - - +
 |         |         |
 |         |         |
 |         |         |
 |         |         |
 + - - - - + - - - - +
 |         |         |
 |         |         |
 |         |         |
 |         |         |
 + - - - - + - - - - +
This is a 4x4 grid
 + - - - - + - - - - + - - - - + - - - - +
 |         |         |         |         |
 |         |         |         |         |
 |         |         |         |         |
 |         |         |         |         |
 + - - - - + - - - - + - - - - + - - - - +
 |         |         |         |         |
 |         |         |         |         |
 |         |         |         |         |
 |         |         |         |         |
 + - - - - + - - - - + - - - - + - - - - +
 |         |         |         |         |
 |         |         |         |         |
 |         |         |         |         |
 |         |         |         |         |
 + - - - - + - - - - + - - - - + - - - - +
 |         |         |         |         |
 |         |         |         |         |
 |         |         |         |         |
 |         |         |         |         |
 + - - - - + - - - - + - - - - + - - - - +
Process finished with exit code 0
```