**Chapter 3 – Week 5 – Exercises**

Exercises #1 – page 70

1. **Write the outputs of the following loops:**

a) for count in range(5):

print(count + 1, end =” “)

**1 2 3 4 5**

b) for count in range(1,4):

print(count + 1, end =” “)

**1 2 3**

c) for count in range(1,6,2):

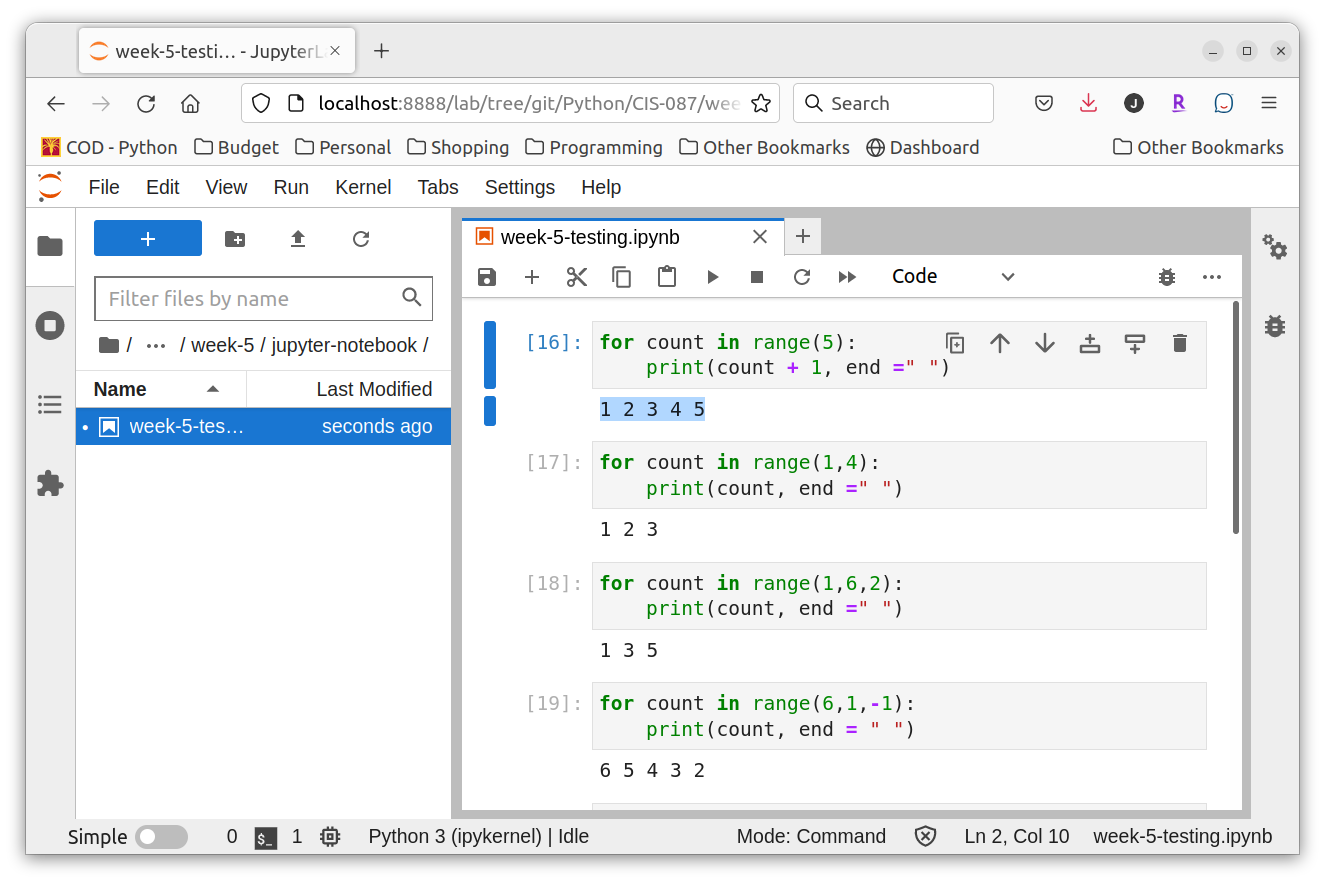
print(count + 1, end =” “)

**1 3 5**

d) for count in range(6,1,-1):

print(count + 1, end =” “)

**6 5 4 3 2**

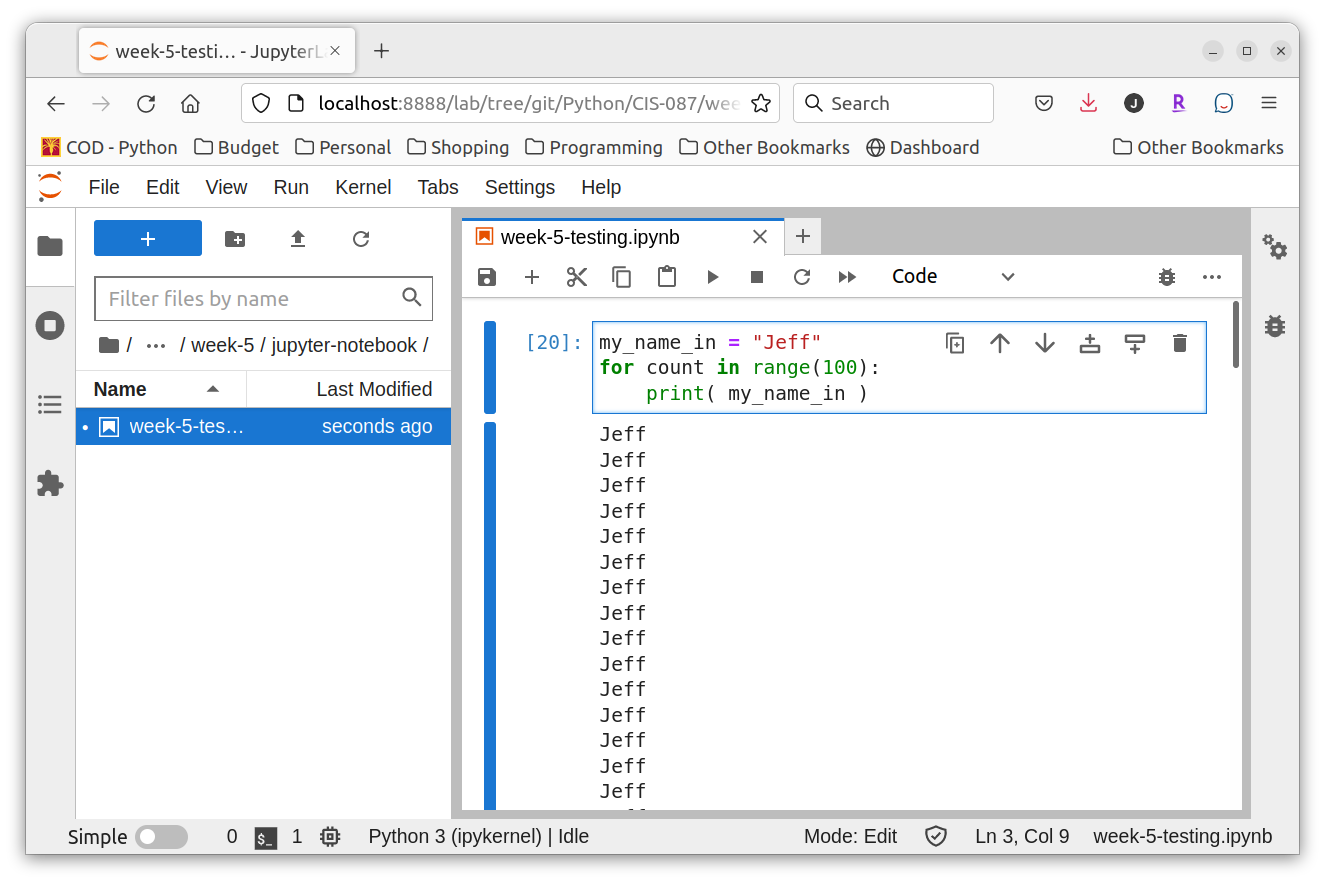


1. **Write a loop that prints your name 100 times. Each output should begin on a new line.**

my\_name\_in = "Jeff"

for count in range(100):

print( my\_name\_in )



1. **Explain the role of the variable in the header of a for loop.**

The variable in the header of a for loop is know as the **loop control variable**. In a for loop. The loop control variable is used to track the progress of the loop allowing it to conclude at the proper time. In the case of a simple loop that loops through a range of numbers, the variable also allow the programmer access to the number of times the loop has been executed.

1. **Write a loop that prints the first 128 ASCII values folowed by the corresponding characters.**

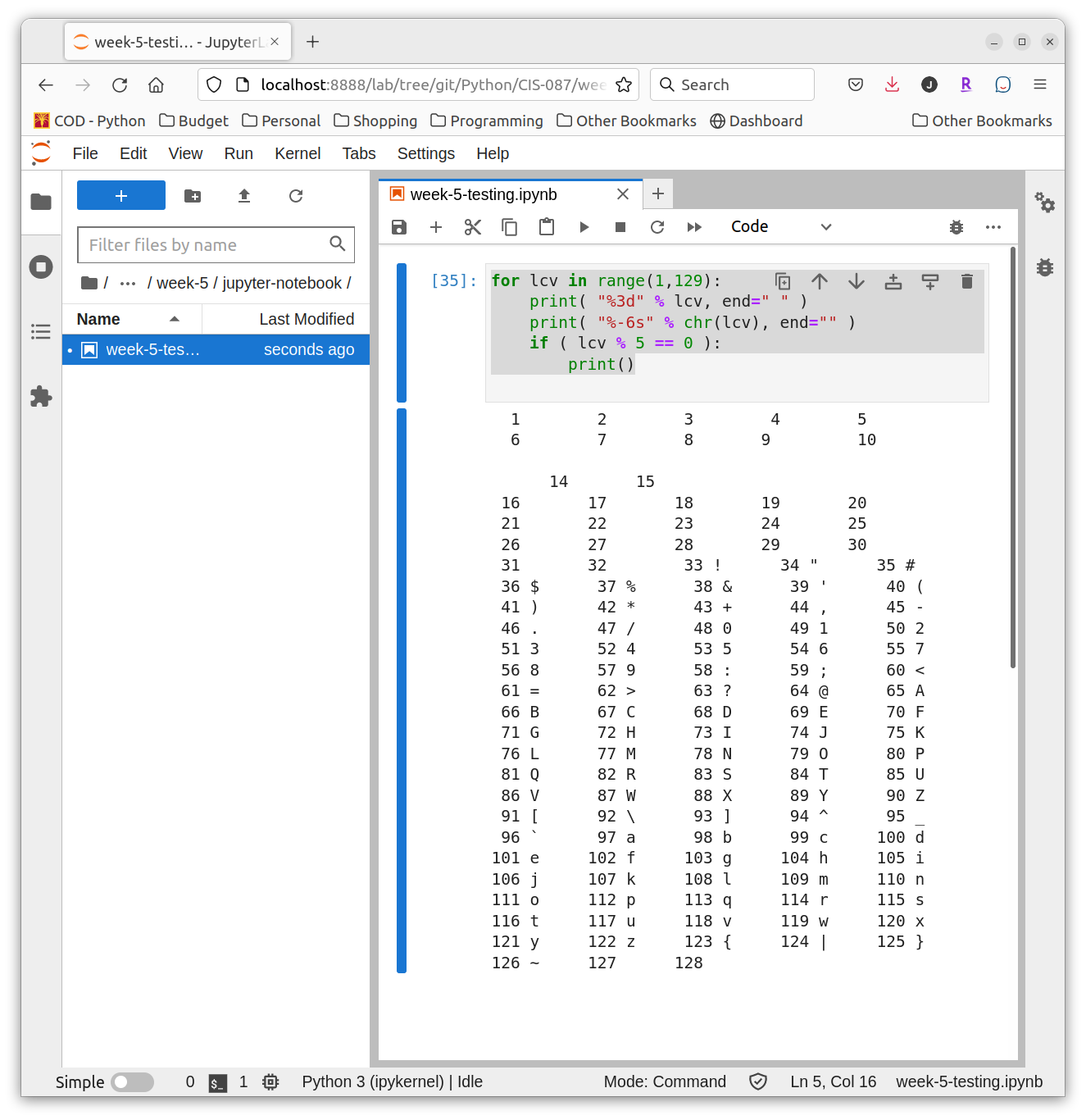
for lcv in range(1,129):

print( "%3d" % lcv, end=" " )

print( "%-6s" % chr(lcv), end="" )

if ( lcv % 5 == 0 ):

print()

****

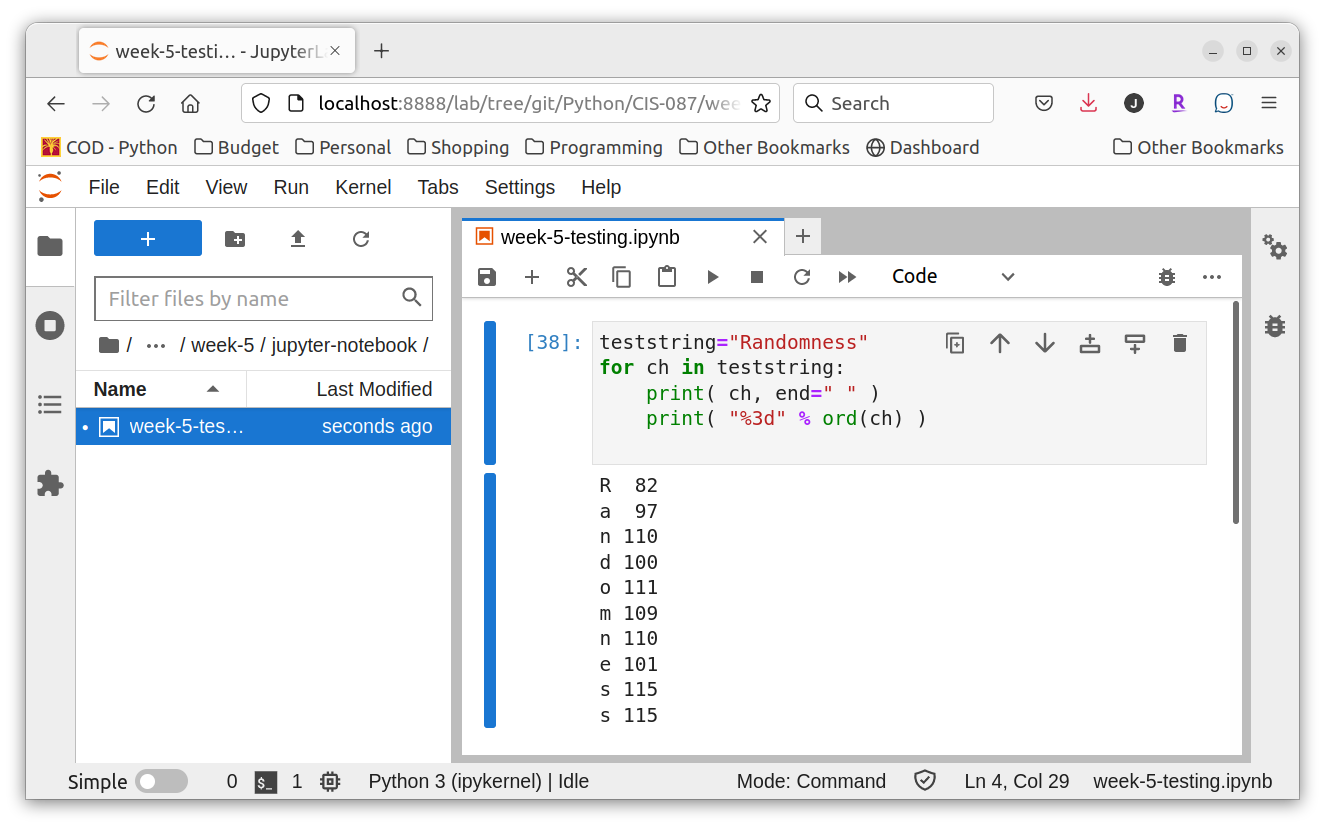
1. **Assume the variable teststring refers to a string. Write a loop that prints each character in this string followed by its ASCII value.**

teststring="Randomness"

for ch in teststring:

print( ch, end=" " )

print( "%3d" % ord(ch) )

****

Exercises #2 – page 72

1. **Assume that the variable amount refers to 24.325. Write the outputs of the following statements:**

a) print(“Your salary is $%0.2f” % amount)

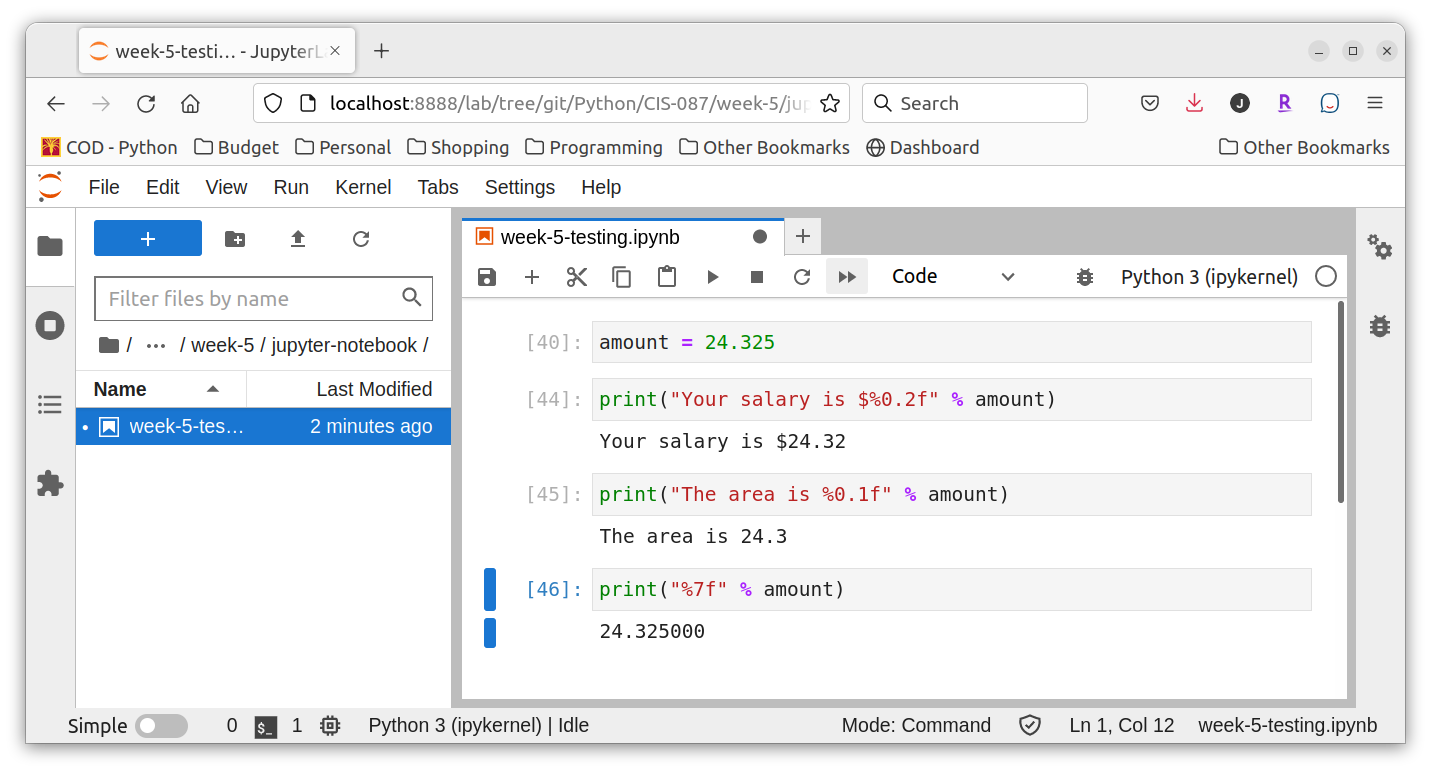
**Your salary is $24.32**

b) print(“The area is %0.1f” % amount)

**The area is 24.3**

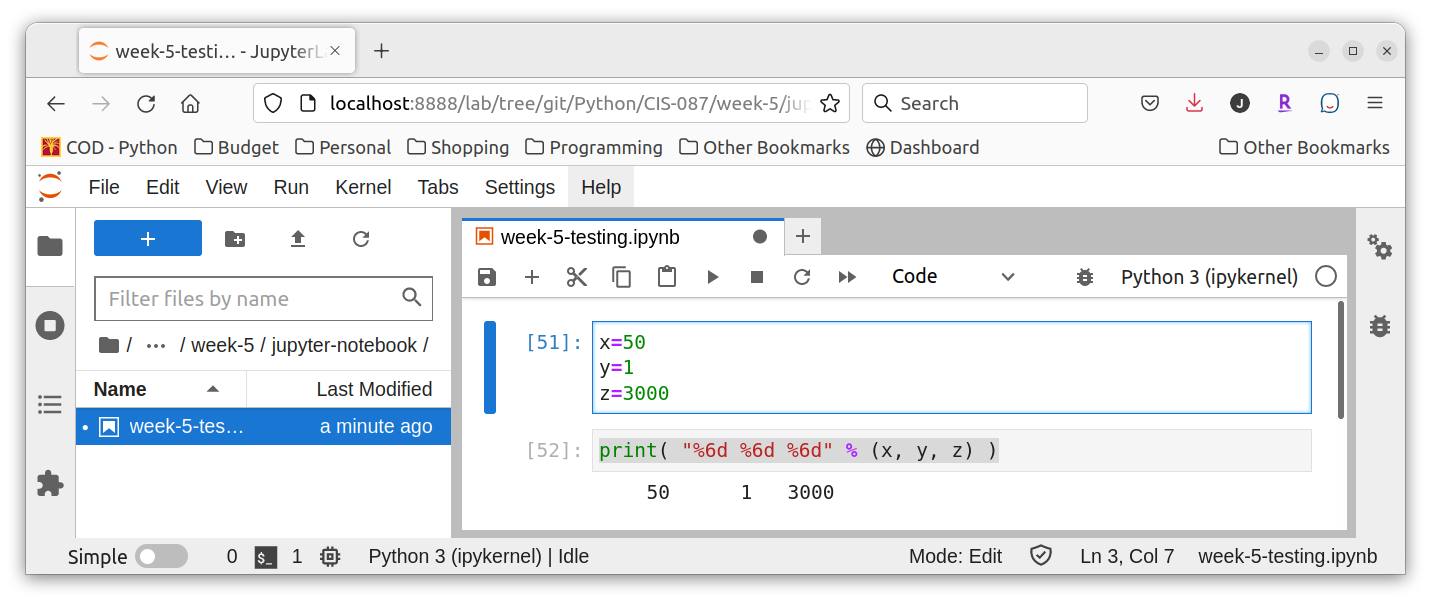
c) print(“%7f % amount)

**24.325000**



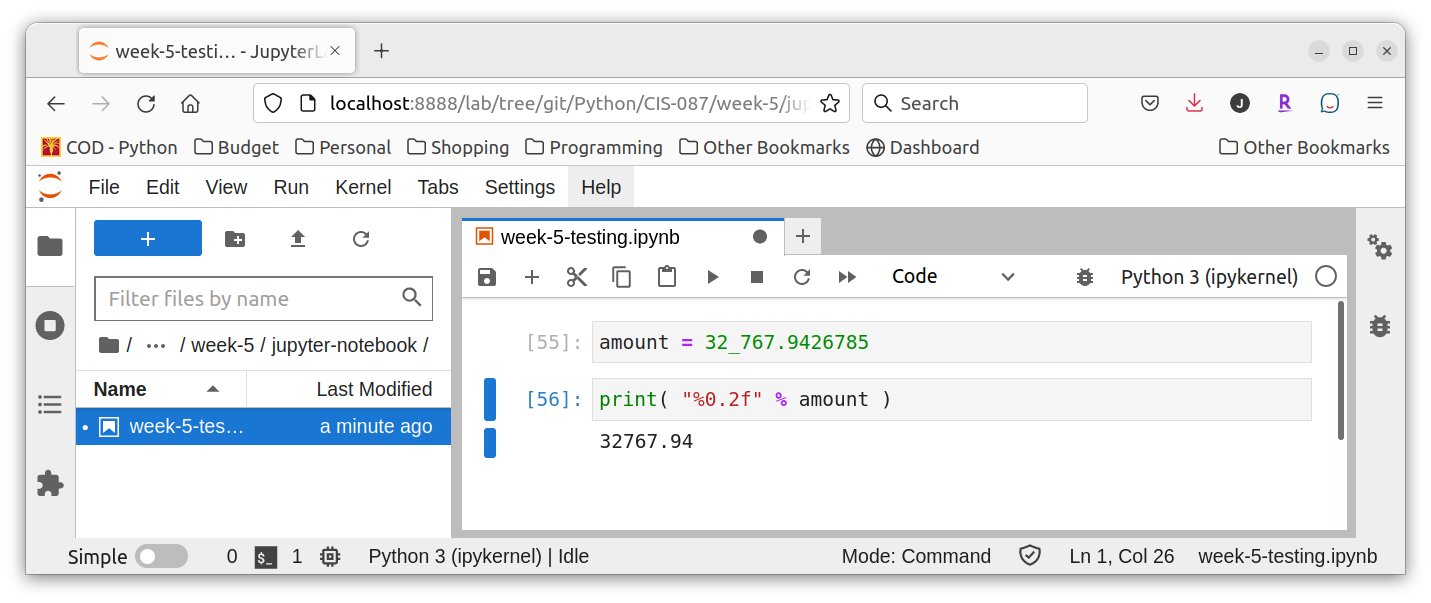
1. **Write a code segment that displays the values of the integers x, y, and z on a single line, such that each value is right-justified with a field width of 6.**

print( "%6d %6d %6d" % (x, y, z) )



1. **Write a format operation that builds a string for the float variable amount that has exactly two digits of precision and a field width of zero.**

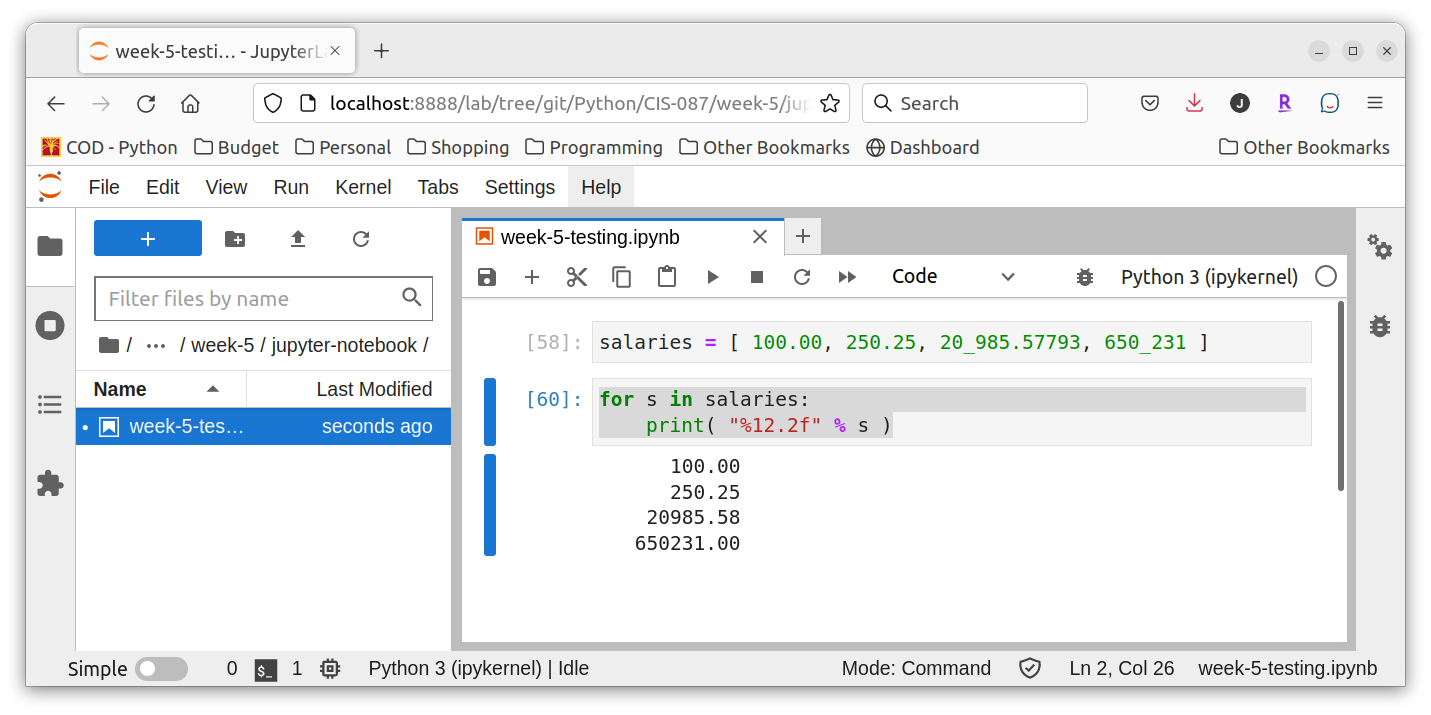
print( "%0.2f" % amount )



1. **Write a loop that outputs the numbers is a lists names salaries. The outputs should be formated in a column that is right-justified, with a field width of 12 and a precision of 2.**

for s in salaries:

print( "%12.2f" % s )



Exercises #3 – page 85

1. **Assume that x is 3 and y is 5. Write the values of the following expressions:**

a) x==7

**False**

b) x > y – 3

**True**

c) x <= y – 2

**True**

d) x == y or x > 2

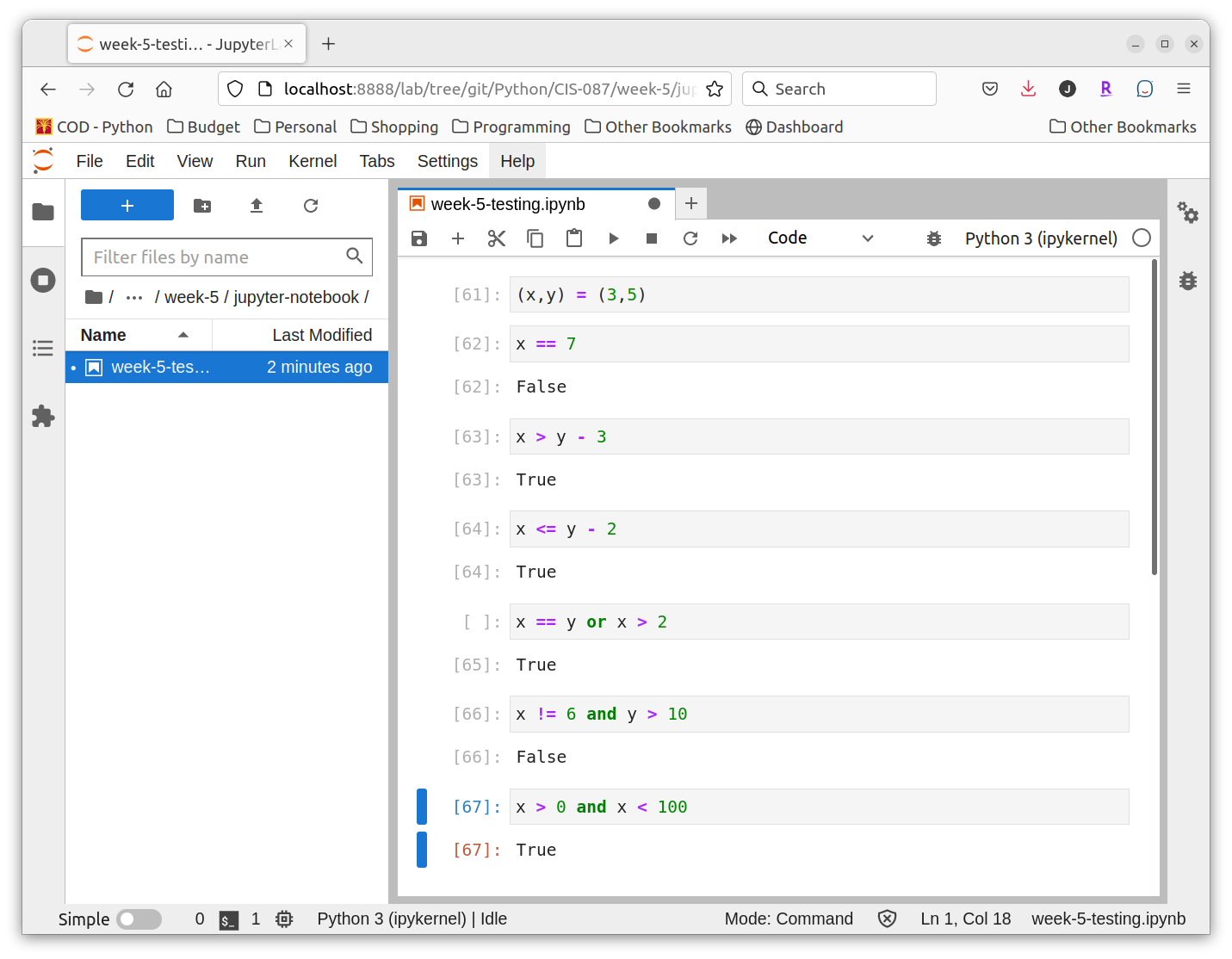
**True**

e) x != 6 and y > 10

**False**

f) x > 0 and x < 100

**True**



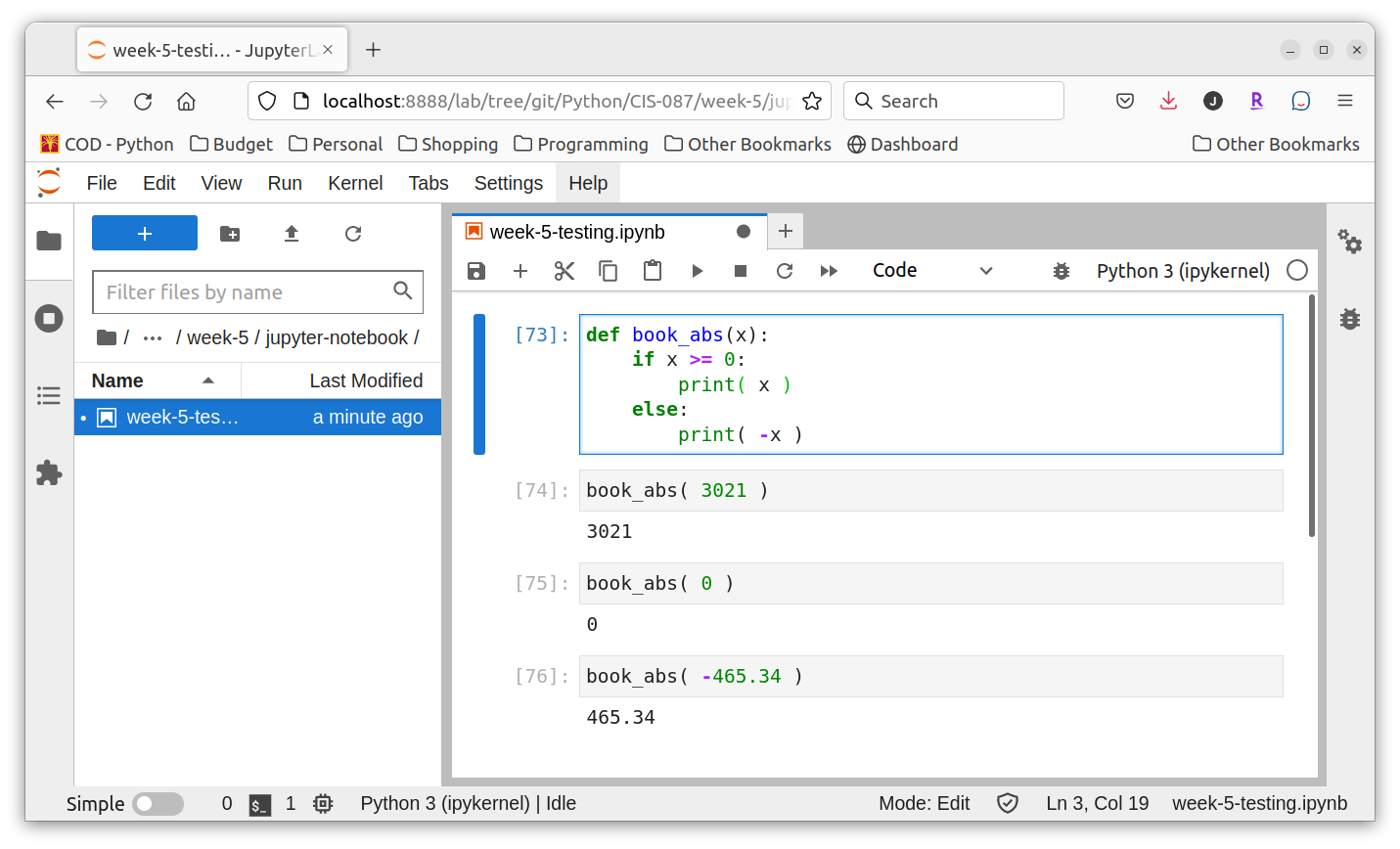
1. **Assume THAT X REFERS TO A NUMBER. Write a code segment that prints the number’s absolute value without using Python’s abs function.**

if x >= 0:

print( x )

else:

print( -x )



1. **Write a loop that counts the number of space characters in a string. Recall that the space character is represented as ‘ ‘.**

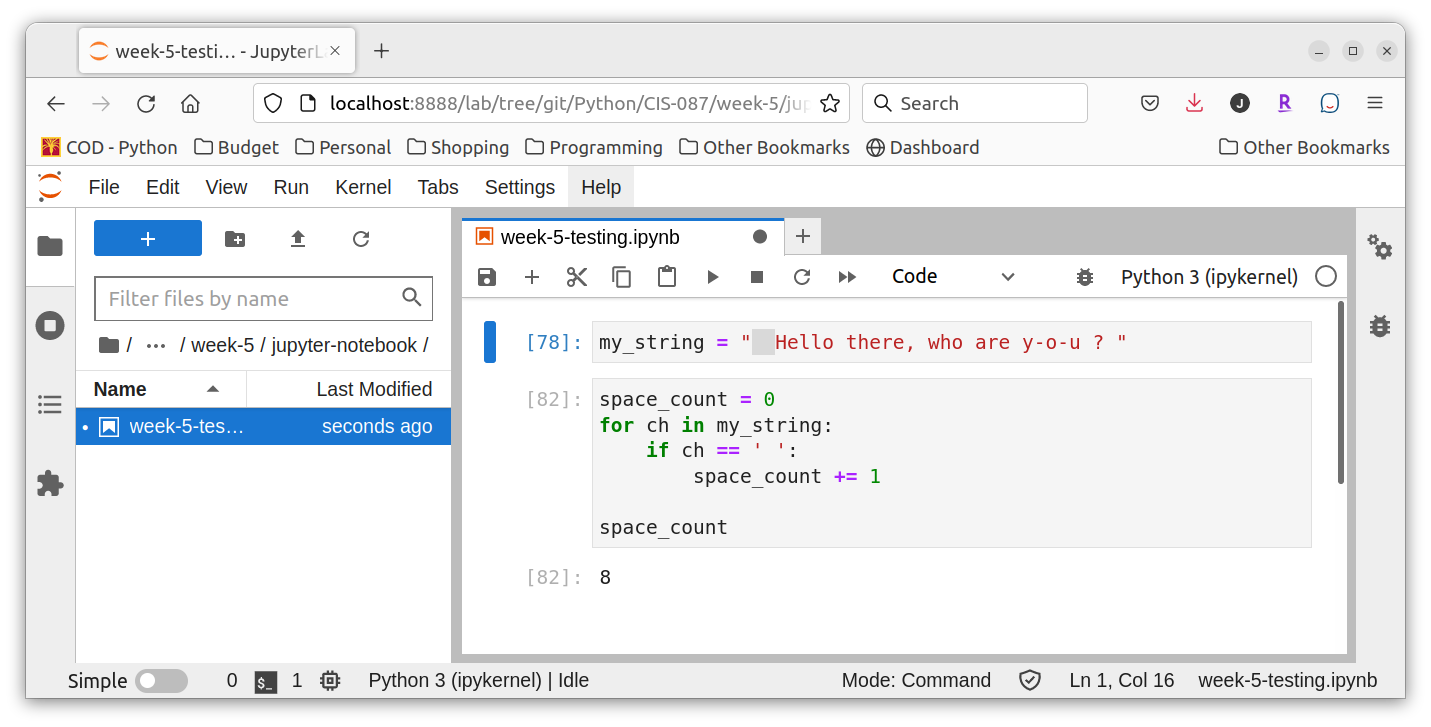
**space\_count = 0**

for ch in my\_string:

if ch == ' ':

space\_count += 1

space\_count



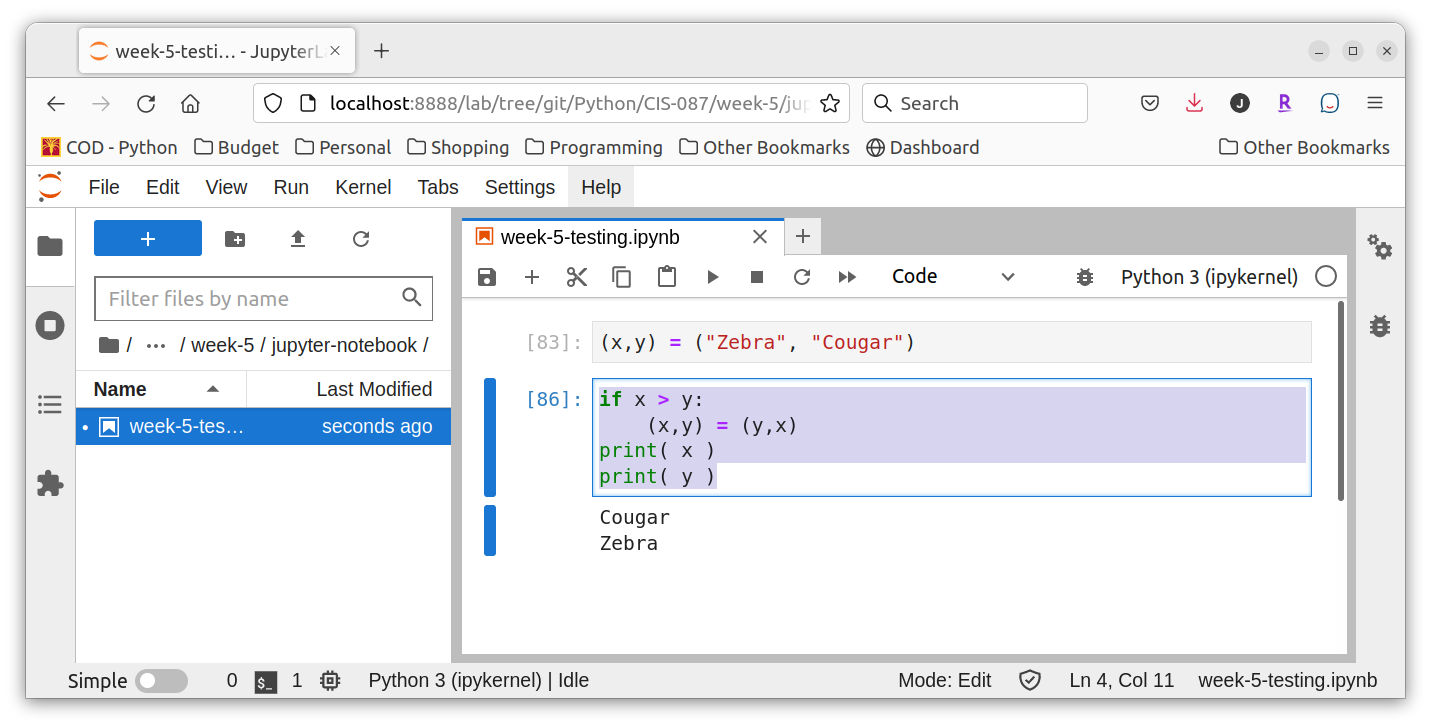
1. **Assume that the variables x and y refer to strings. Write a code segment that prints those strings in alphabetical order. You should assume that they are not equal.**

if x > y:

(x,y) = (y,x)

print( x )

print( y )



1. **Explain how to check for an invalid input number and prevent if being used in a program. You may assume that the user enters a number.**

As long as the user has entered a numeric value that Python can easily convert from a string into a numeric data type, you can simply use an **if** statement to compare the supplied number (that you have presumably stored in a variable) with the legal range of values. Unless the situation is unusual, this will be something like

if my\_var < 0:

<error code>

else:

<valid code>

1. **Construct truth tables for the following Boolean expressions:**

a) **not (A or B)**

|  |  |  |
| --- | --- | --- |
| **A** | **B** | **Result** |
| True | True | False |
| True | False | False |
| False | True | False |
| False | True | True |

b) **not A and not B**

|  |  |  |
| --- | --- | --- |
| 1. **A** | 1. **B** | 1. **Result** |
| 1. True | 1. True | 1. False |
| 1. True | 1. False | 1. False |
| 1. False | 1. True | 1. False |
| 1. False | 1. True | 1. True |

1. **Explain the role of the trailing else part of an extended if statement.**

The extended if statement, which I presume to mean an **if** statement using one or more **elif** blocks, can conclude with an **else** block just as a simple if statement can. This else will be executed if none of the boolean conditions specified in the if or elif statements evaluate to true. This is the “catch-all” case that is executed if nothing else was.

1. **The variables x and y refer to numbers. Write a code statement that prompts the user for an arithmetic operator and prints the value obtained by applying that operator to x and y.**

**operator = input( "Please enter of of the following: +, -, /, \*, or ^: " )**

answer = 0

if operator == "+":

answer = x + y

elif operator == "-":

answer = x - y

elif operator == "/":

answer = x / y

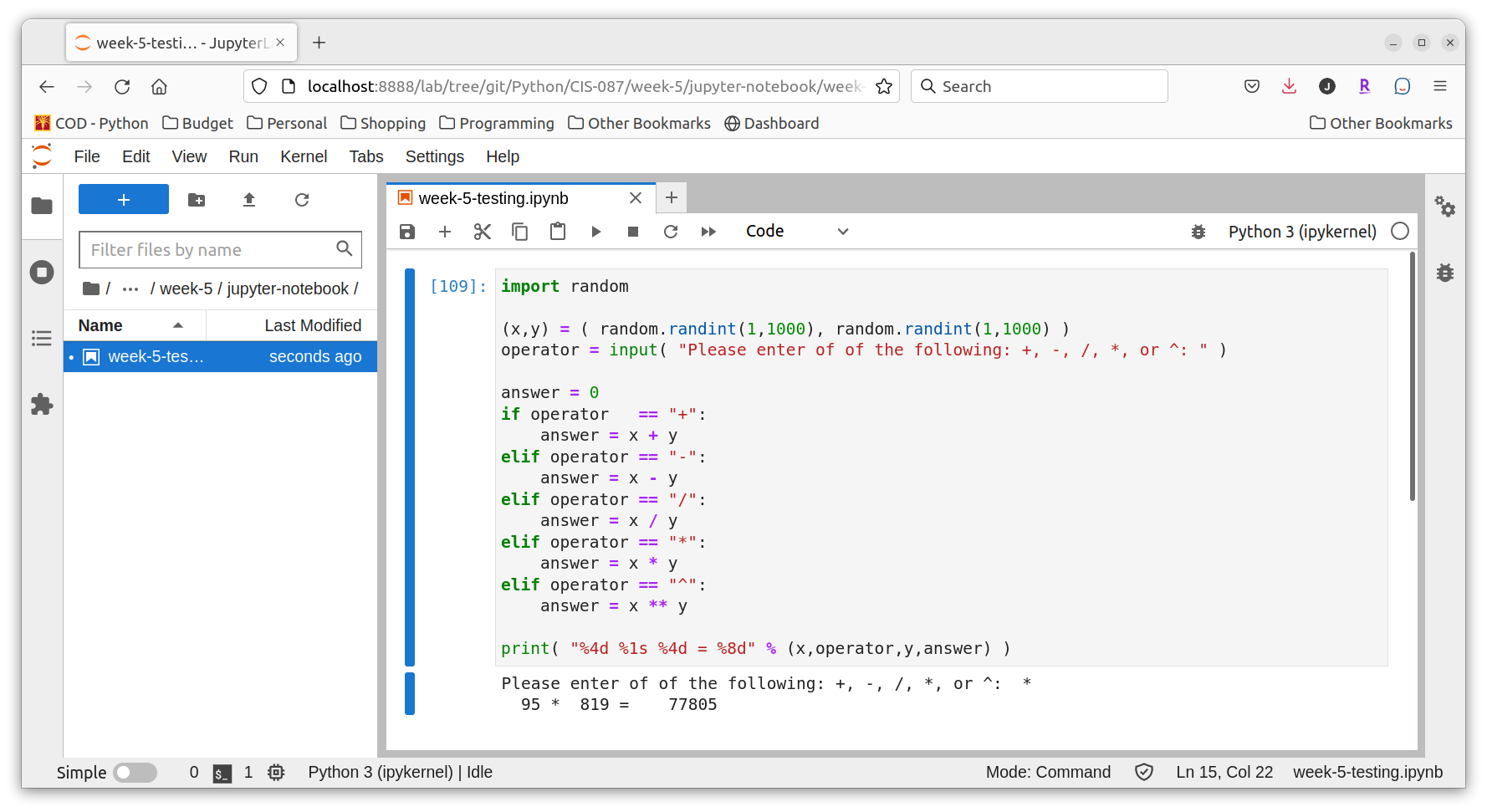
elif operator == "\*":

answer = x \* y

elif operator == "^":

answer = x \*\* y

print( "%4d %1s %4d = %8d" % (x,operator,y,answer) )



1. **Does the Boolean expression count > 0 and total // count > 0? If not, why not?**

The potential error in question is division by 0. If total // count is evaluated, it will throw a ZeroDivisionError. However, since Python like most modern languages, short circuits the evaluation of boolean statements once in knows what the result will be, no error is possible with this statement. This is because the condition before the and will be false if count == 0. Once this portion of the boolean expression is false, it is certain that the entire expression will be false due to the nature of false. As such, the remainder of the expression is not evaluated avoiding the potential of dividing by zero.

Exercises #4 – page 92

1. **Translate the following for loops to equivalent while loops:**

(a) for count in range(100):

print(count)

**lcv = 0**

**while lcv < 100:**

**print(lcv)**

**lcv += 1**

(b) for count in range(1,101):

print(count)

**lcv = 1**

**while lcv < 101:**

**print(lcv)**

**lcv += 1**

(c) for count in range(100,0,-1):

print(count)

**lcv = 100**

**while lcv > 0:**

**print(lcv)**

**lcv -= 1**

1. **The factorial of an integer *N* is the product of the integers between 1 and *N*, inclusive. Write a while loop that computes the factorial of a given integer *N*.**

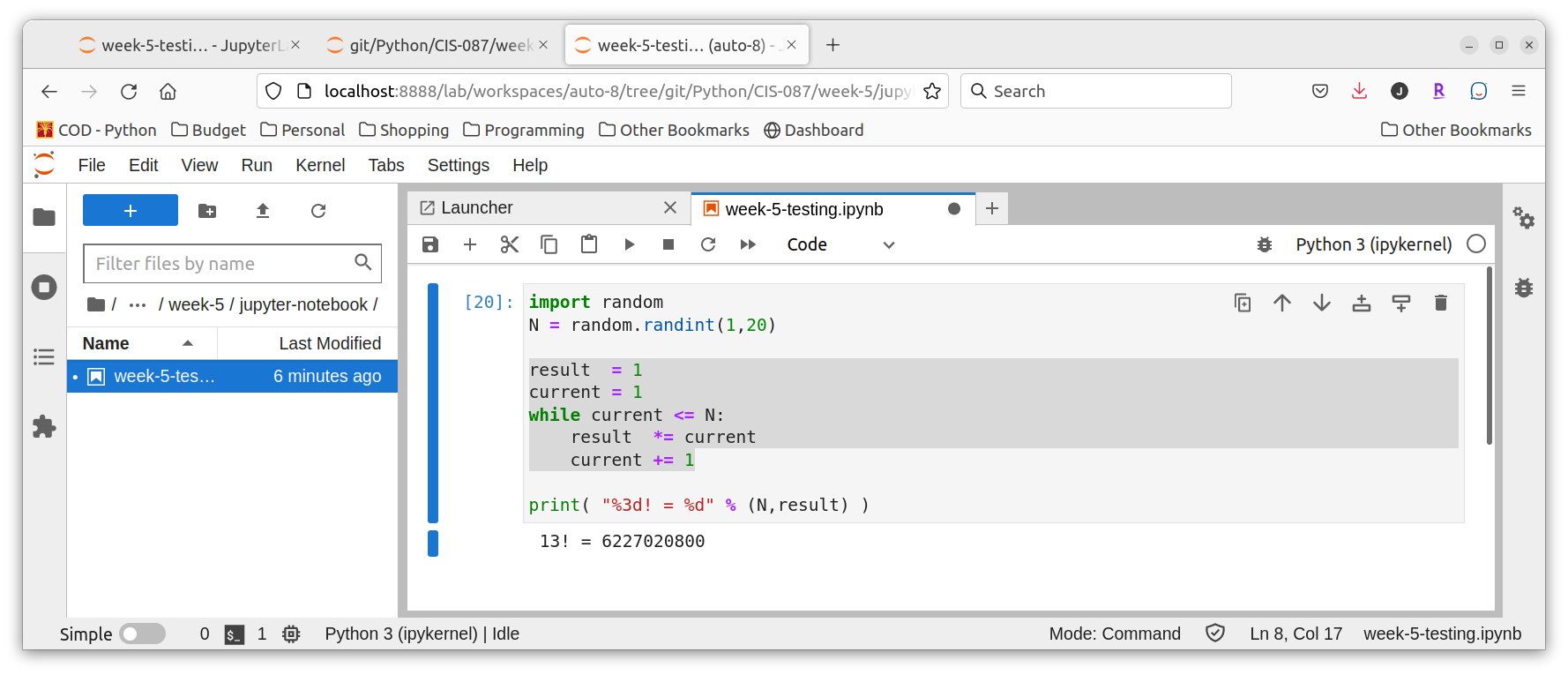
result = 1

current = 1

while current <= N:

result \*= current

current += 1

****

1. **The log2 of a given number N is given by M in the equation N=2M. Using integer arithmetic, the value of M is approximately equal to the number of times N can be evenly divided by 2 until it becomes 0. Write a loop that computes this approximation of the log2 of a give number N. You can check your code by importing the math.log function and evaluating the expression round(math.log(N,2)) (note that the math.log function returns a floating-point value).**

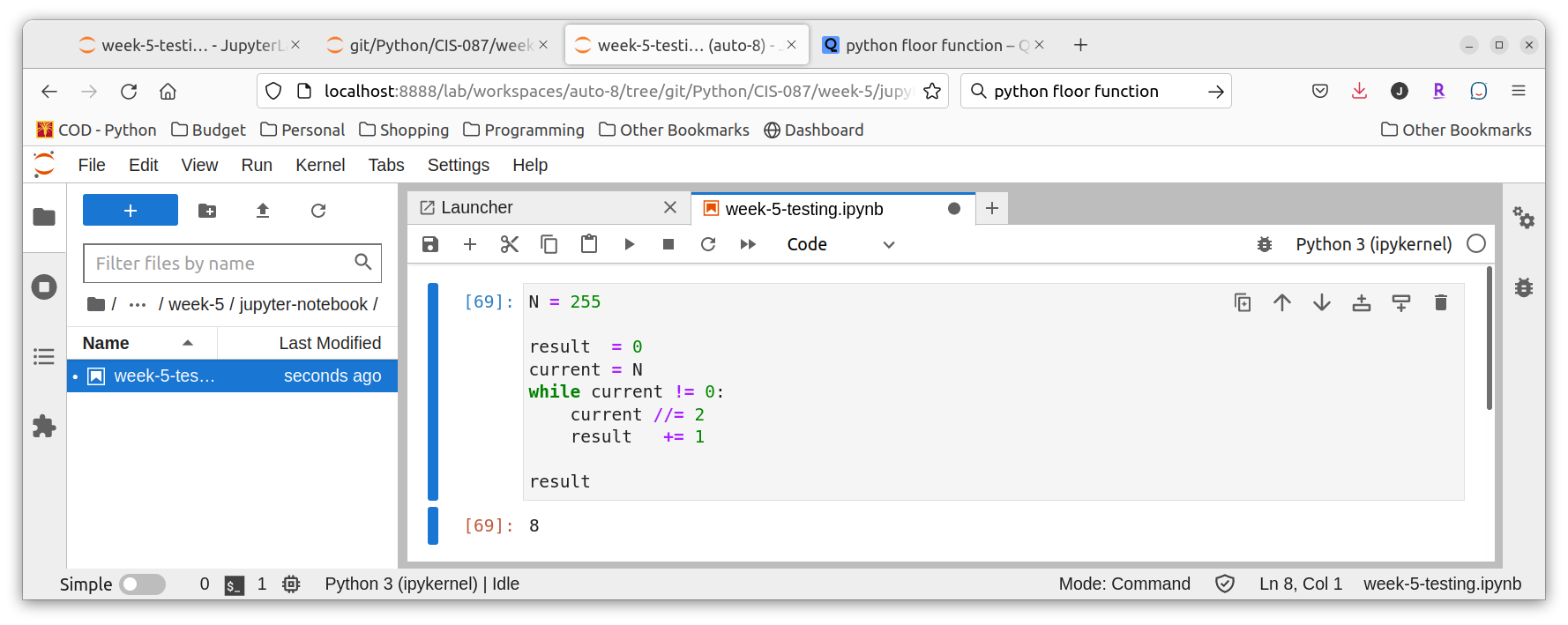
result = 0

current = N

while current != 0:

current //= 2

result += 1

****

1. **Describe the purpose of the break statement and the type of problem for which it is well suited.**

The **break** statement will exit a loop immediately and continue to the Python statement following the loop. The author of the text believes this allows us to write code that easier to understand and debug when we write certain **while** loops. It is often difficult to get the condition with which to exit the loop correct in the loop header but easier to create the condition that should end the loop somewhere within the loop. Under these conditions the author prefers to write an infinite (while True:) loop that ends with a break once the body of the loop determines it is time to exit the loop. This often also makes the code easier to read and, thus, maintain.

1. **What is the maximum number of guesses necessary to guess correctly a given number between the numbers N and M?**

log2 (M – N + 1) page 91 paragraph 1 of the text.

1. **What happens when the programmer forgets to update the loop control variable in a while loop?**

This will nearly always end up causing an **infinite loop**.