# **Fundamentals of Computer Programming**



Submitted by Submitted to

Sumni Limbu Krishna Sir

Section E

ID No 9659

Bsc(Computing)

**Introduction to Programming**

Lab Worksheet

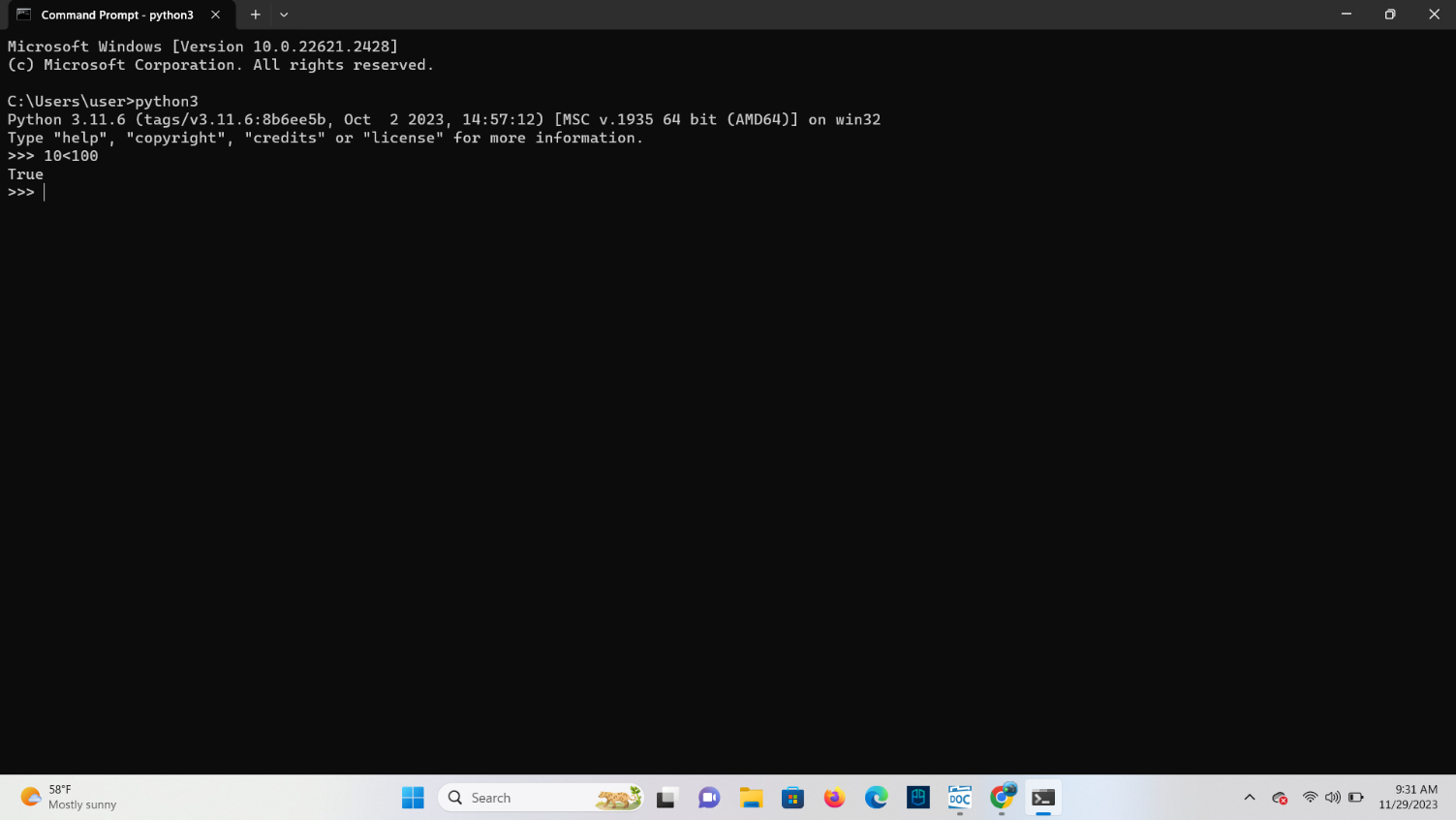
Week 3

**Boolean Expressions**

**TASK**: Start the Python Interpreter and input the following expressions, noting each result.

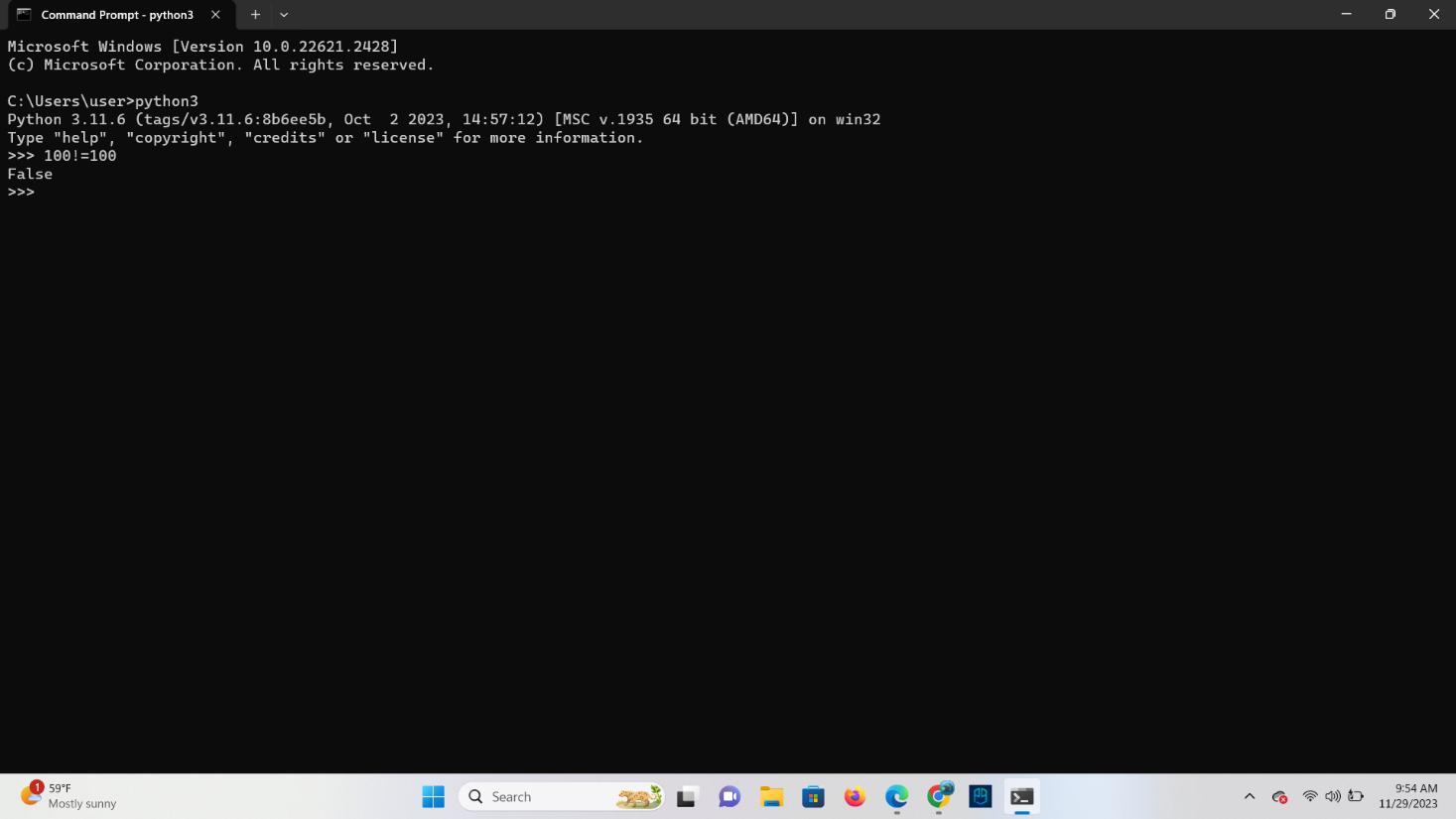
10<100

**Ans**



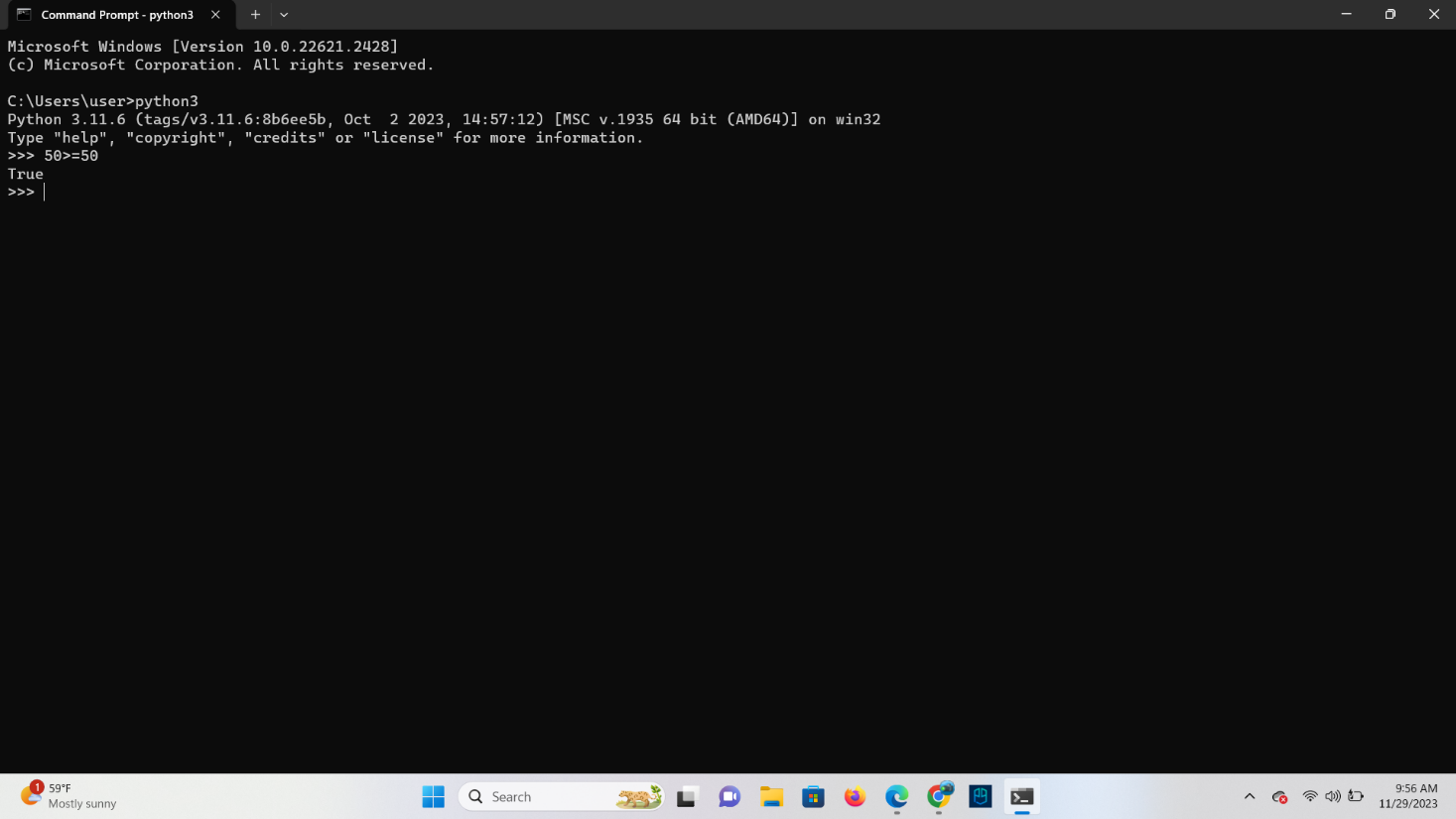
100!=100

**Ans**



50>=50

**Ans**

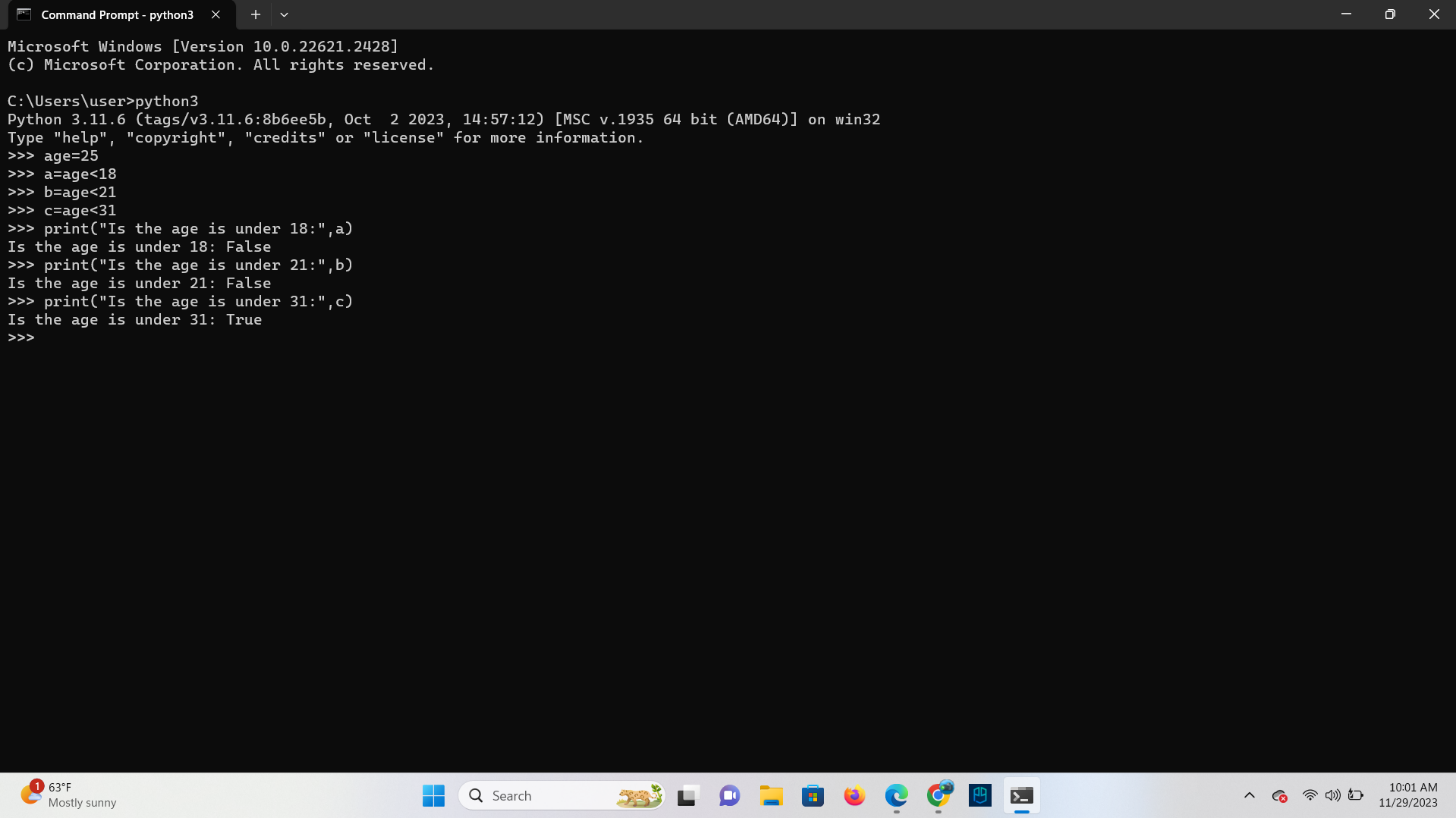


**TASK**: Input a program that defines a variable called ‘age’ that is initialized to your own age.

Then type several Boolean expressions that compare the variable to see whether it is less

than ‘18’, ‘21’ then ‘31’. Boolean expressions do not have to compare just numeric type values, they can also be used to compare other types.

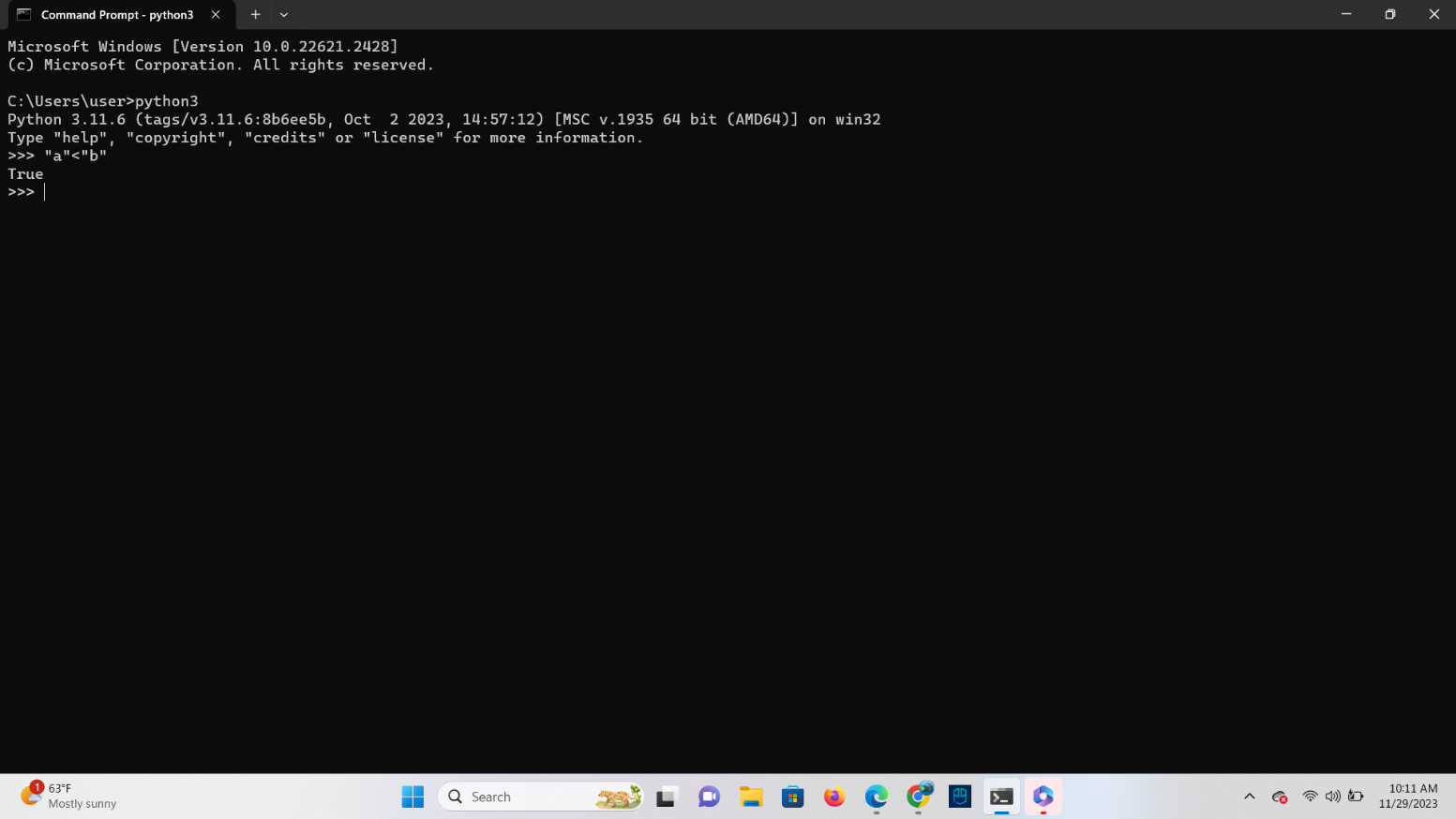
**Ans**



**TASK**: Try inputting the following code and note the results.

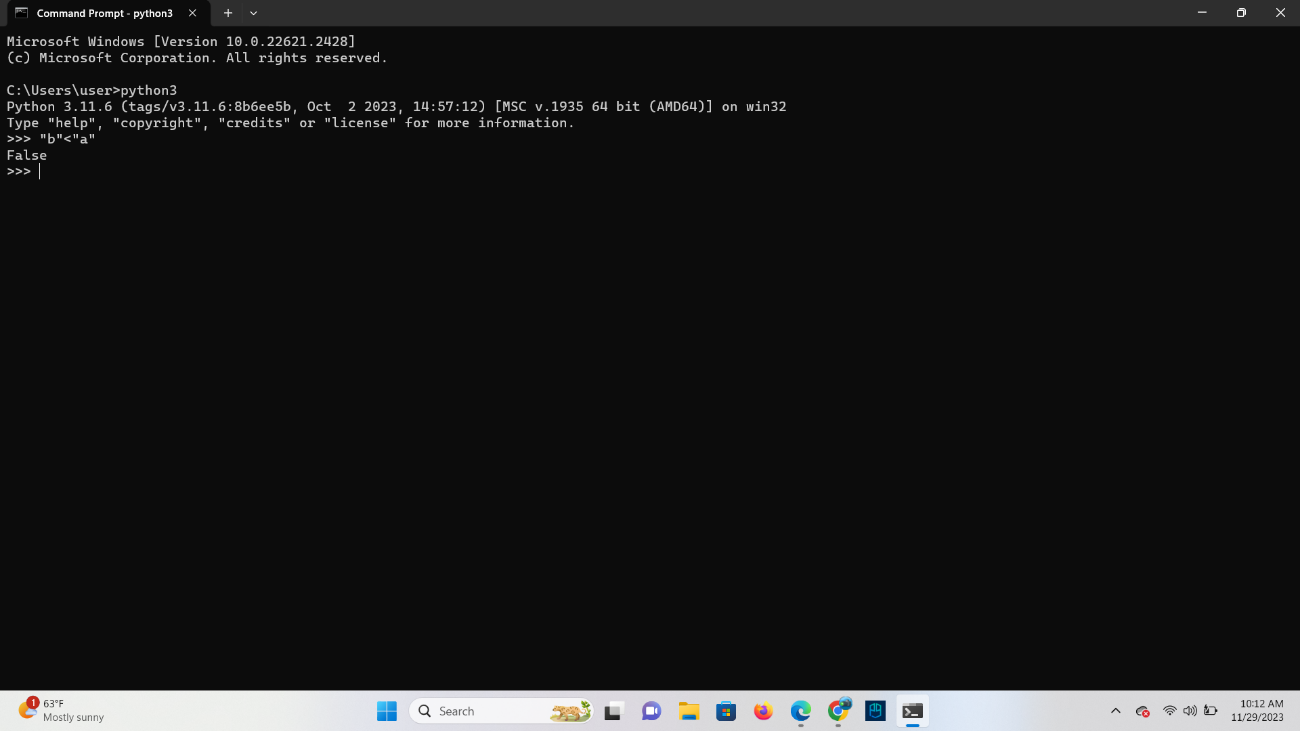
“a” <” b”

**Ans**



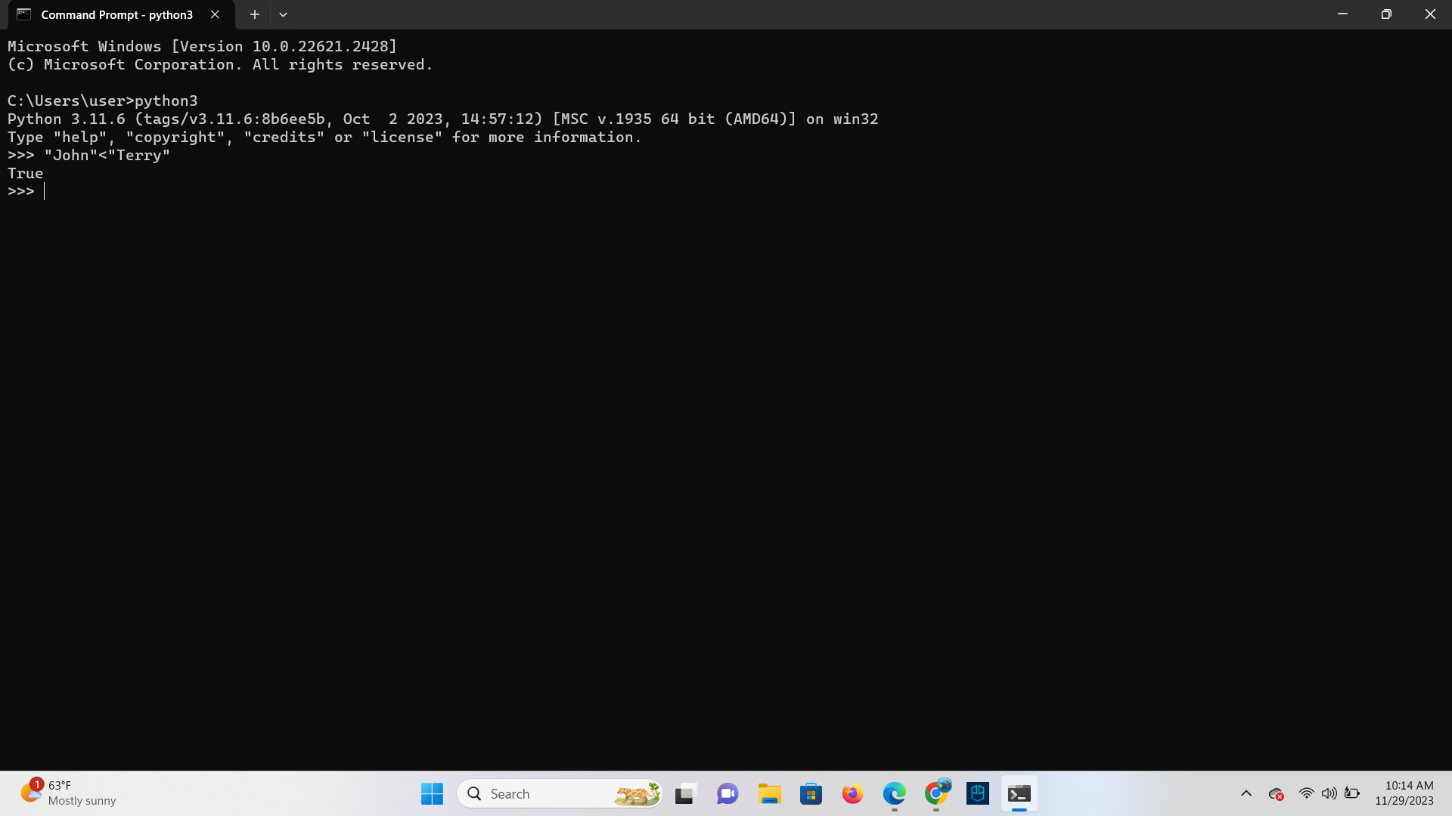
“b” < “a”

**Ans**



“John” < “Terry”

Ans

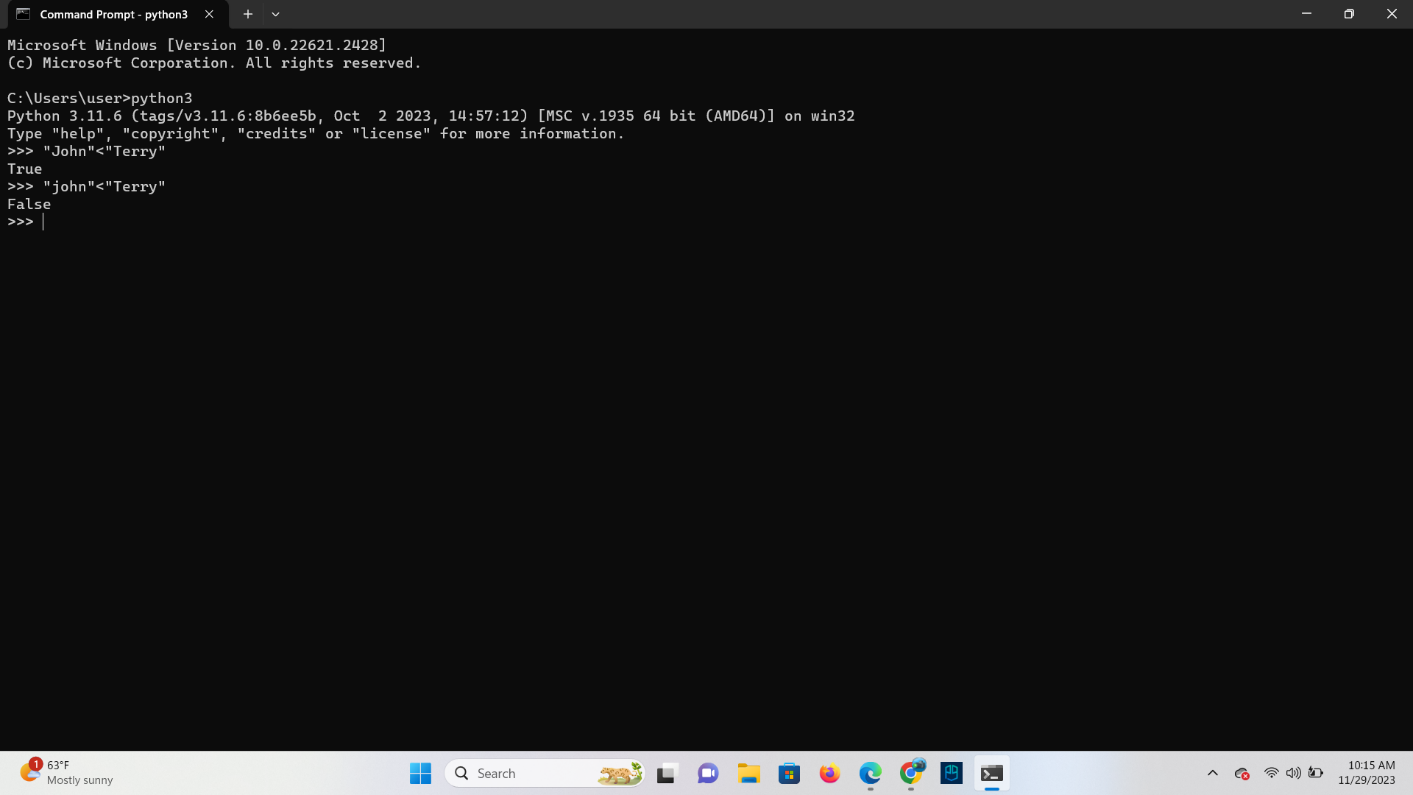


**TASK**: Try inputting the following code and note the result. Try to work out why the answer is

different from the previous expression (look carefully, it is different).

“john” < “Terry”

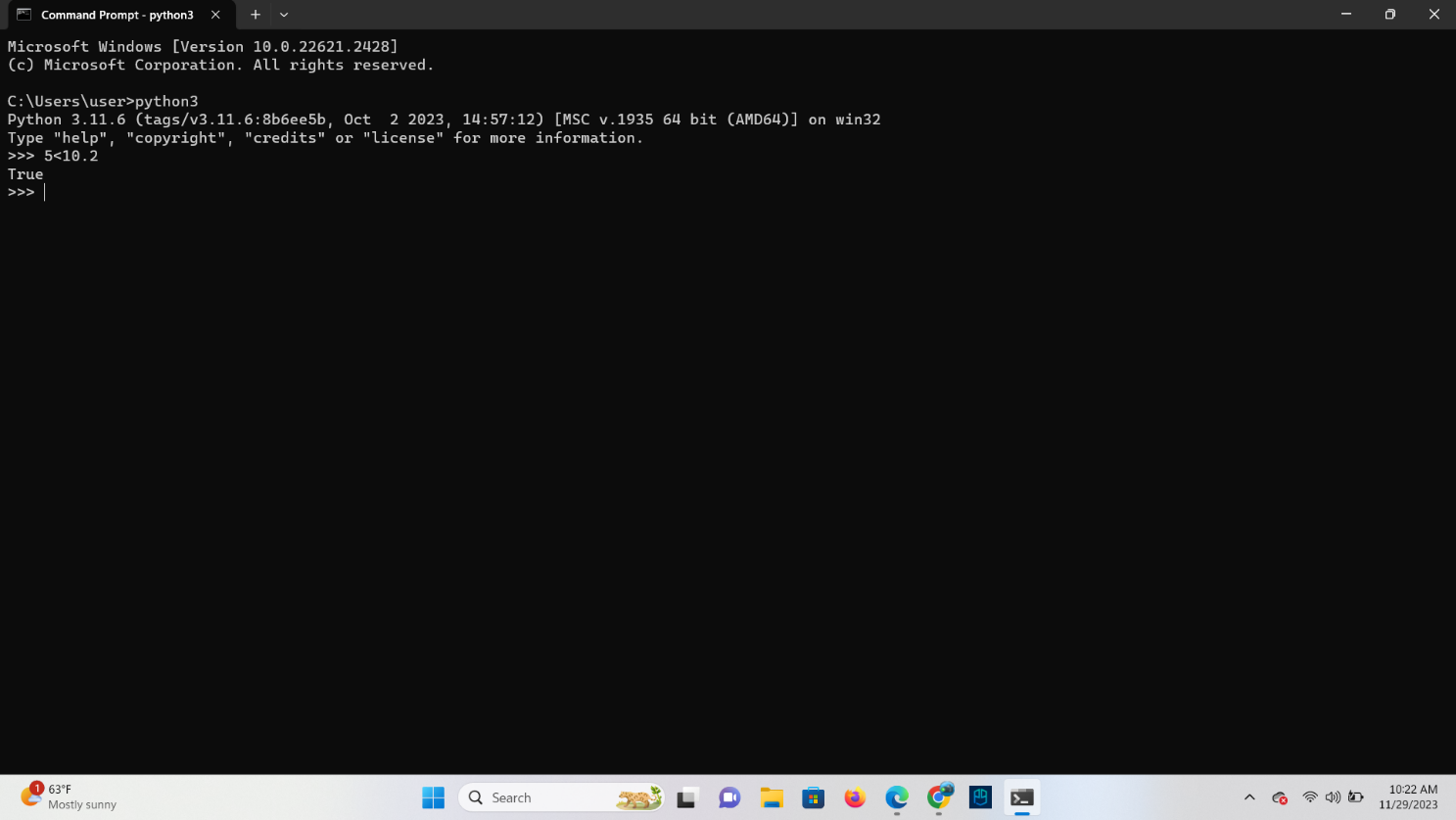
**Ans**  The answer is different as when comparing strings in Python, it is case-sensitive. In the ASCII character set, lowercase letters have higher numerical values than uppercase letters. Therefore, "john" is considered greater than "Terry".



**TASK**: Try inputting the following code and note the results.

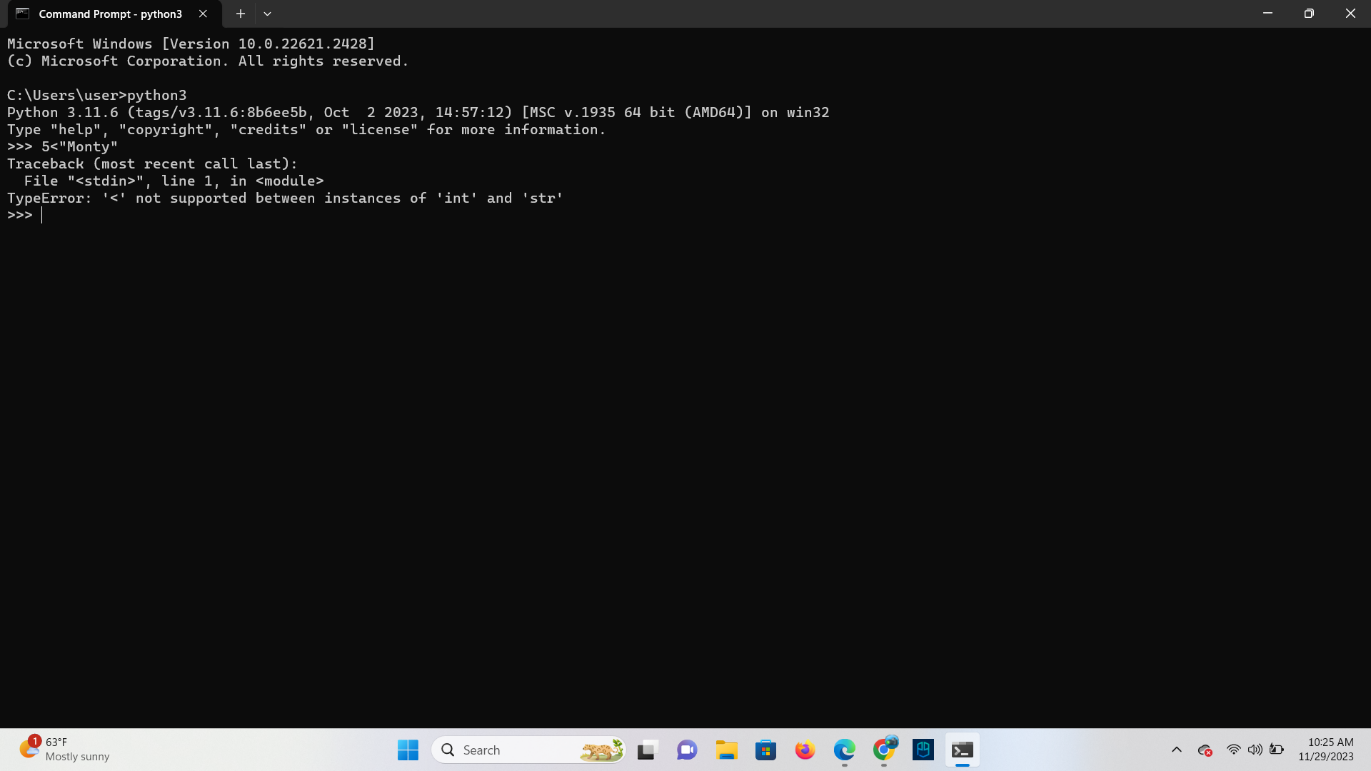
5 < 10.2

**Ans**



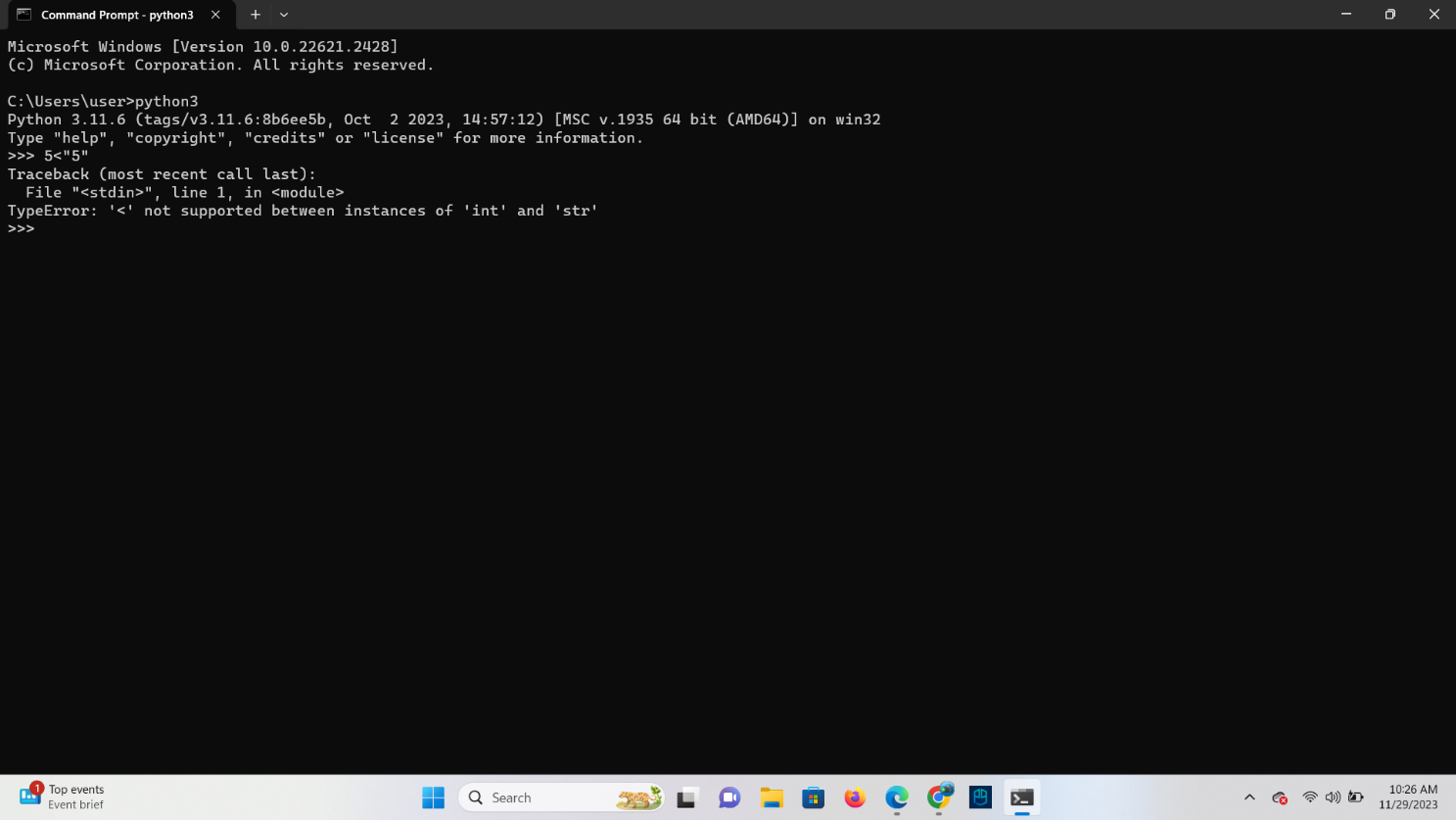
5 < “Monty”

**Ans**



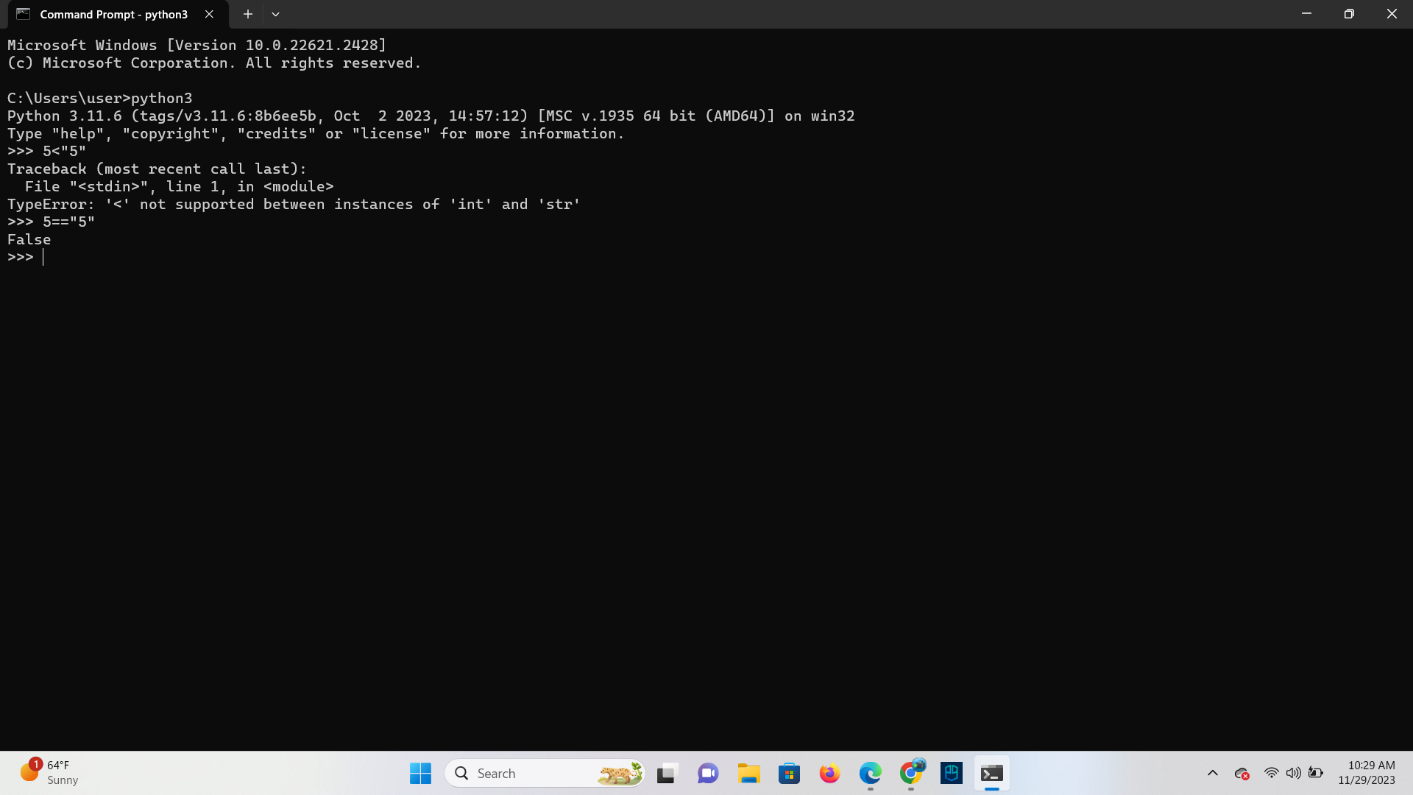
5 < “5”

**Ans**



5 == “5”

**Ans**



**Logical Operators within Expressions**

**TASK**: Try inputting the following code and examine the results.

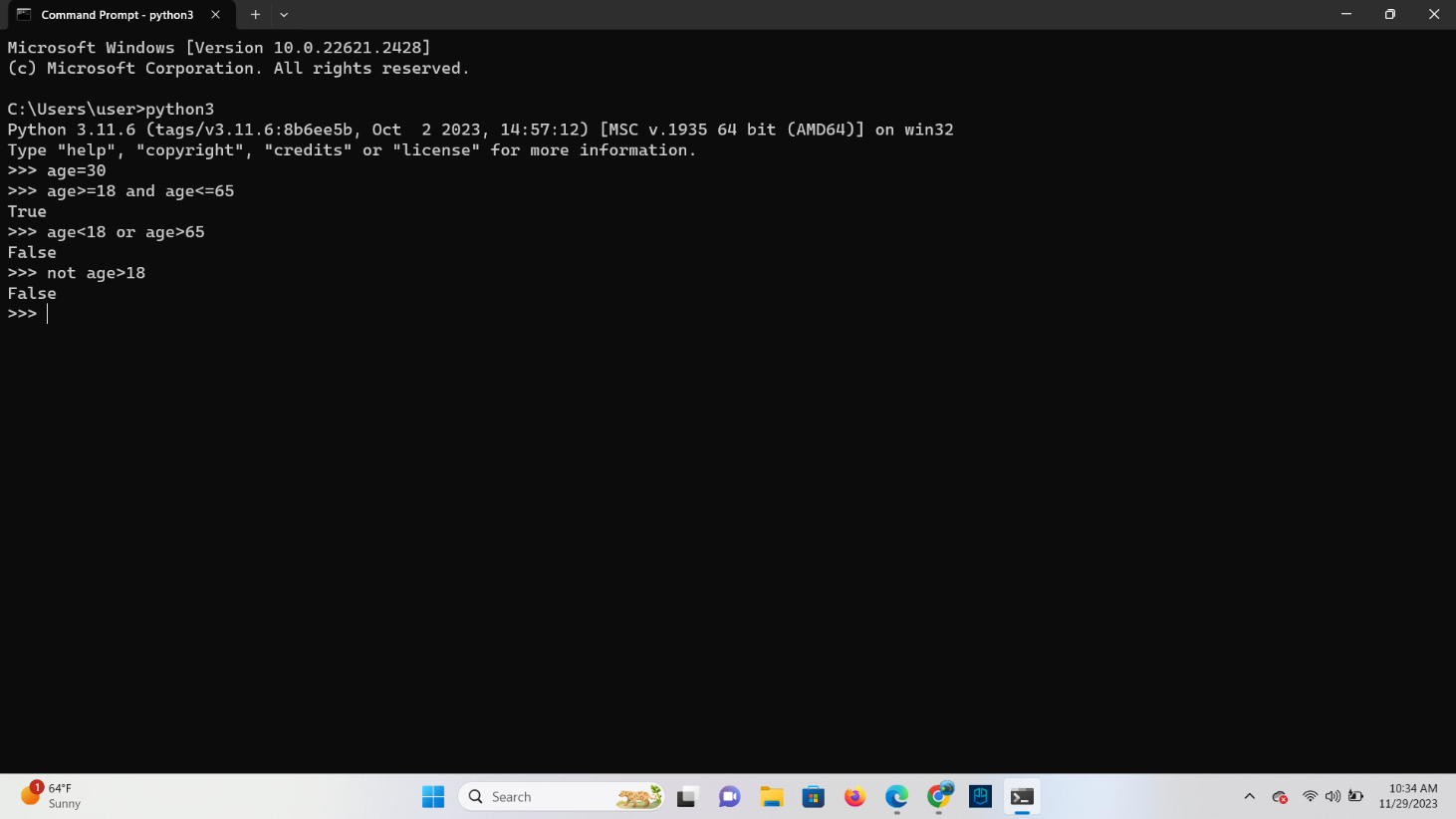
age = 30

age >=18 and age <=65

age <18 or age >65

not age < 18

**Ans**

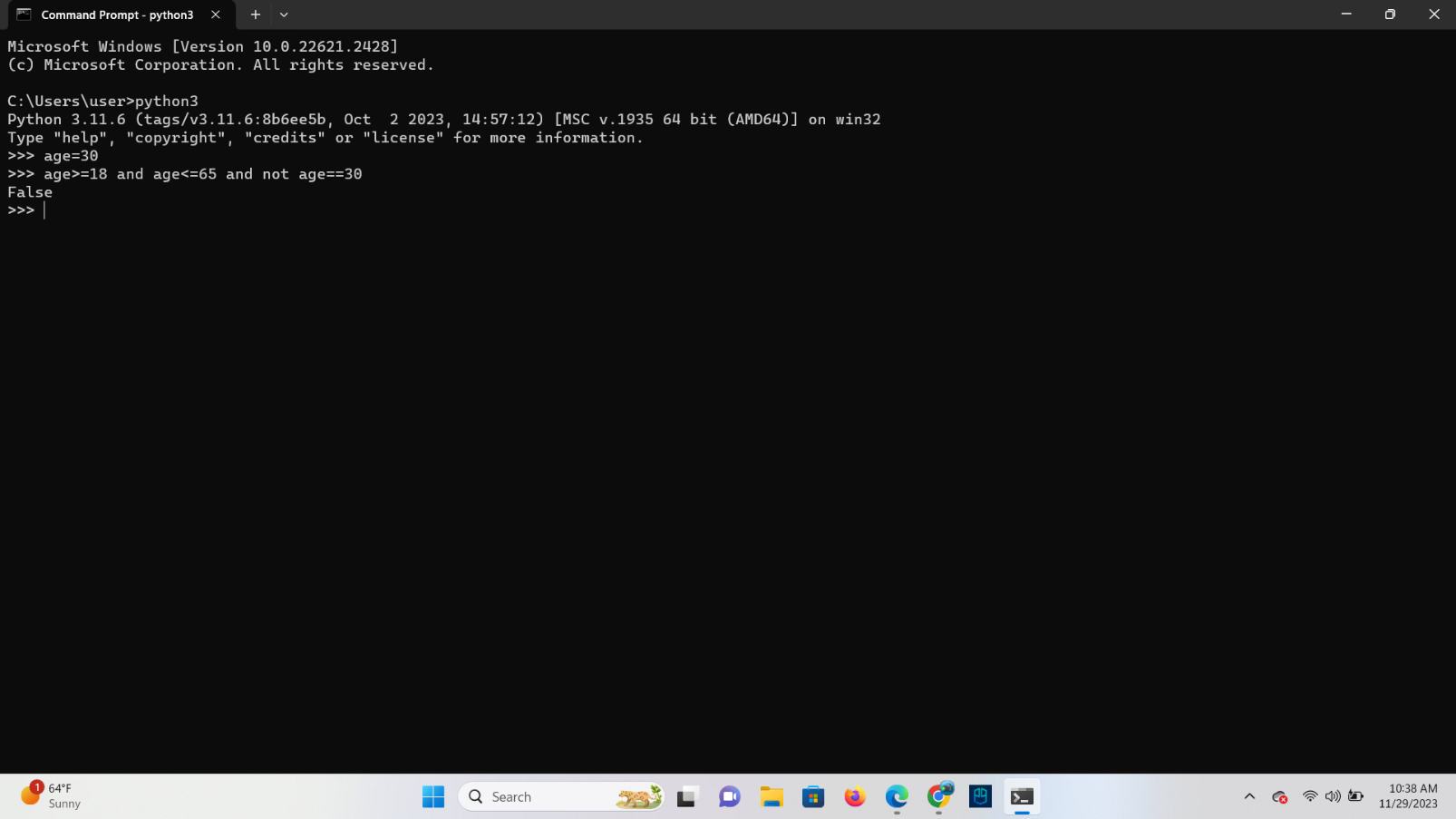


**TASK**: Try inputting the following code and examine the result.

age = 30

(age >=18 and age <=65) and (not age==30)

**Ans**



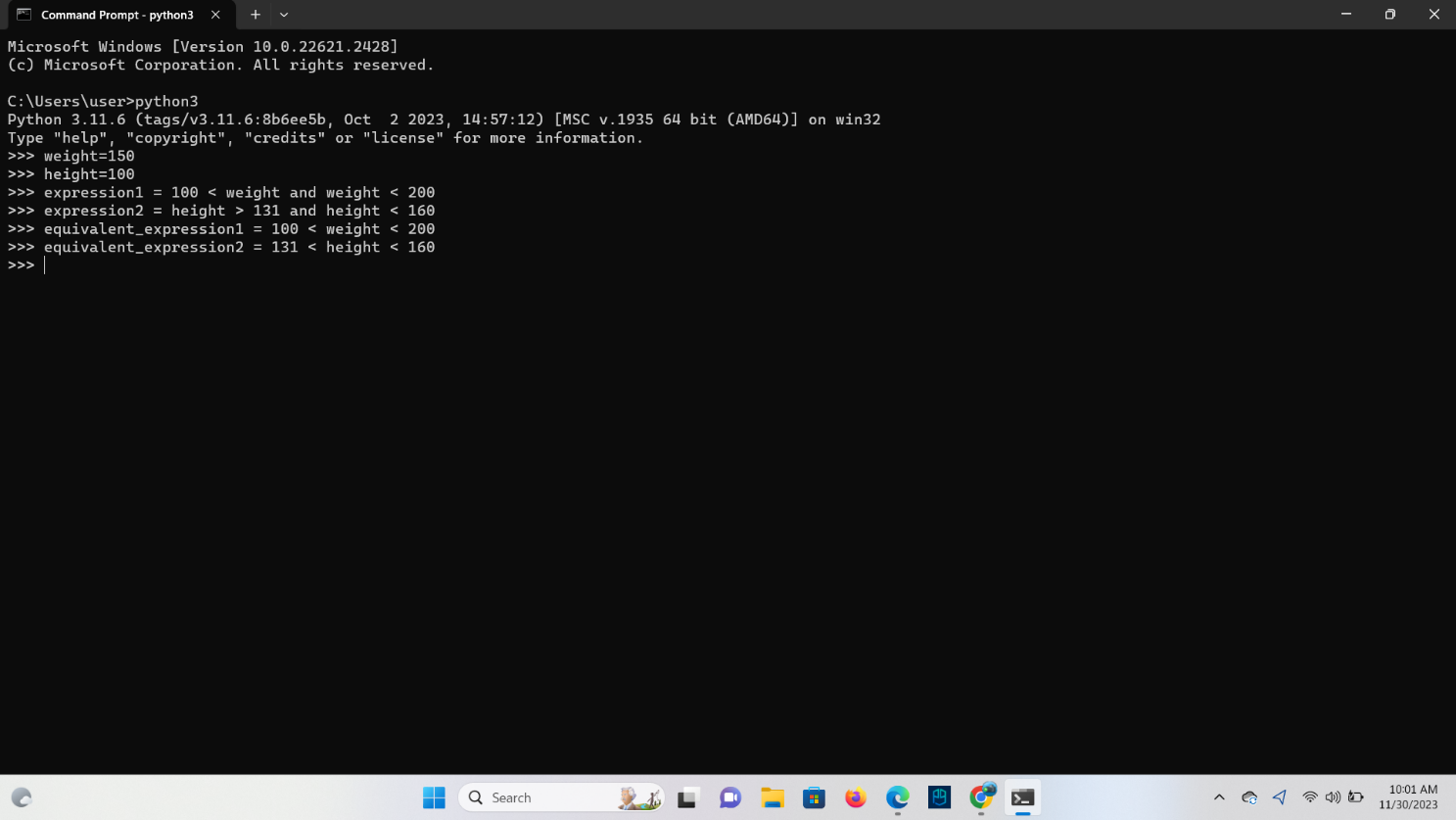
**Chaining relational operators**

**TASK**: Try inputting two expressions that use operator chaining and are equivalent to the

two expressions shown below. (note: you may first want to first assign values to the

‘weight’ and ‘height’ variables for testing purposes)

**Ans**



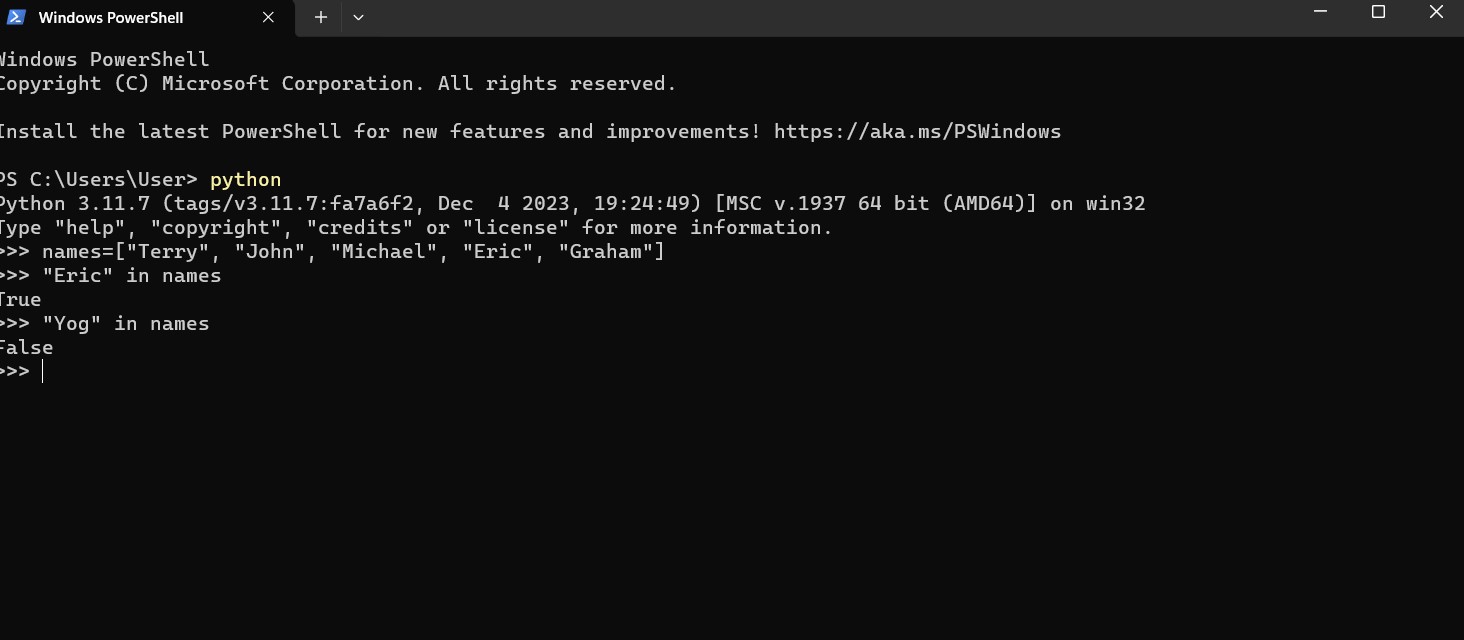
**Membership Testing**

**TASK**: Input the examples above but with alternative operand values, that result in both

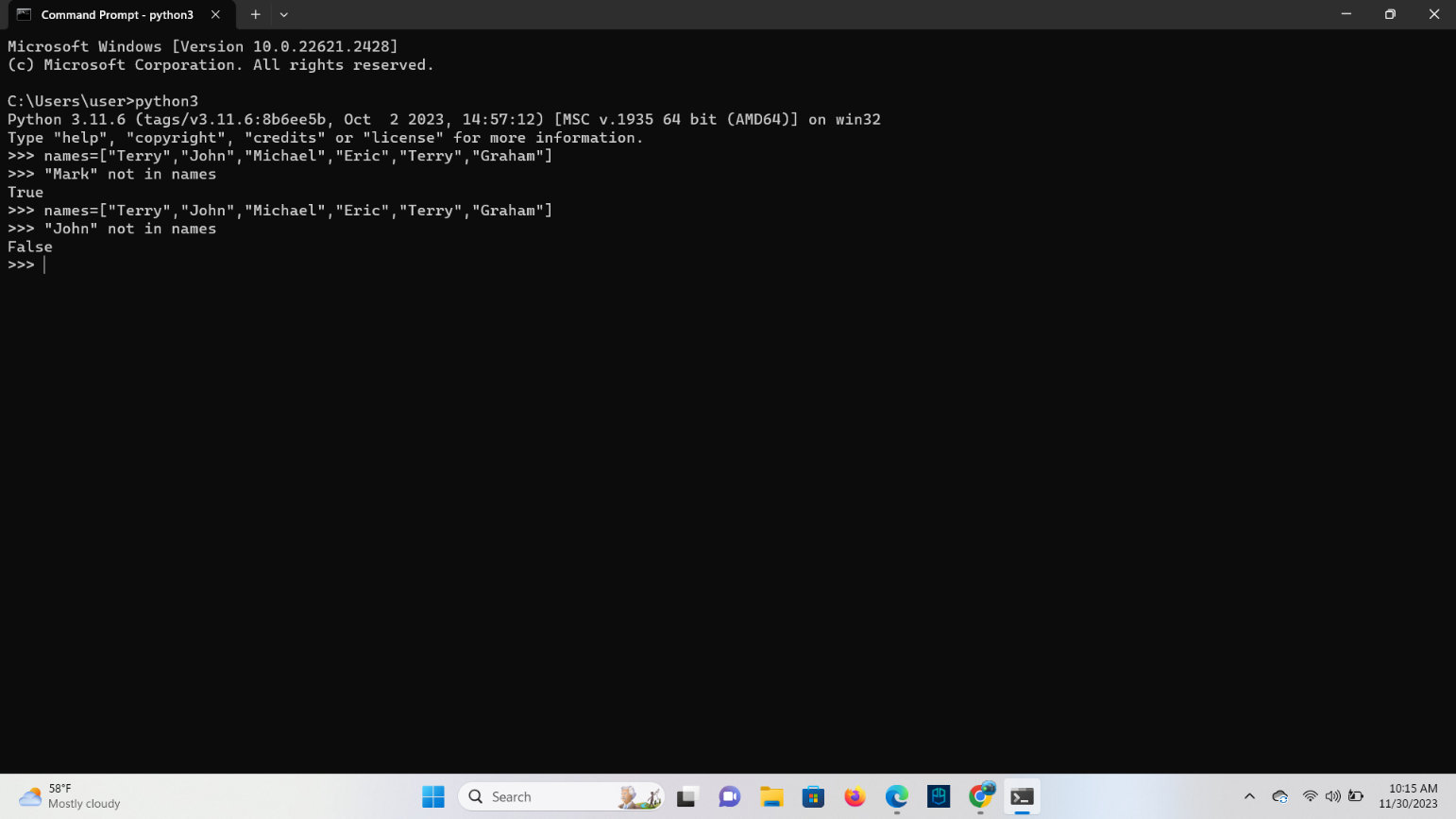
True and False results.

**Ans**

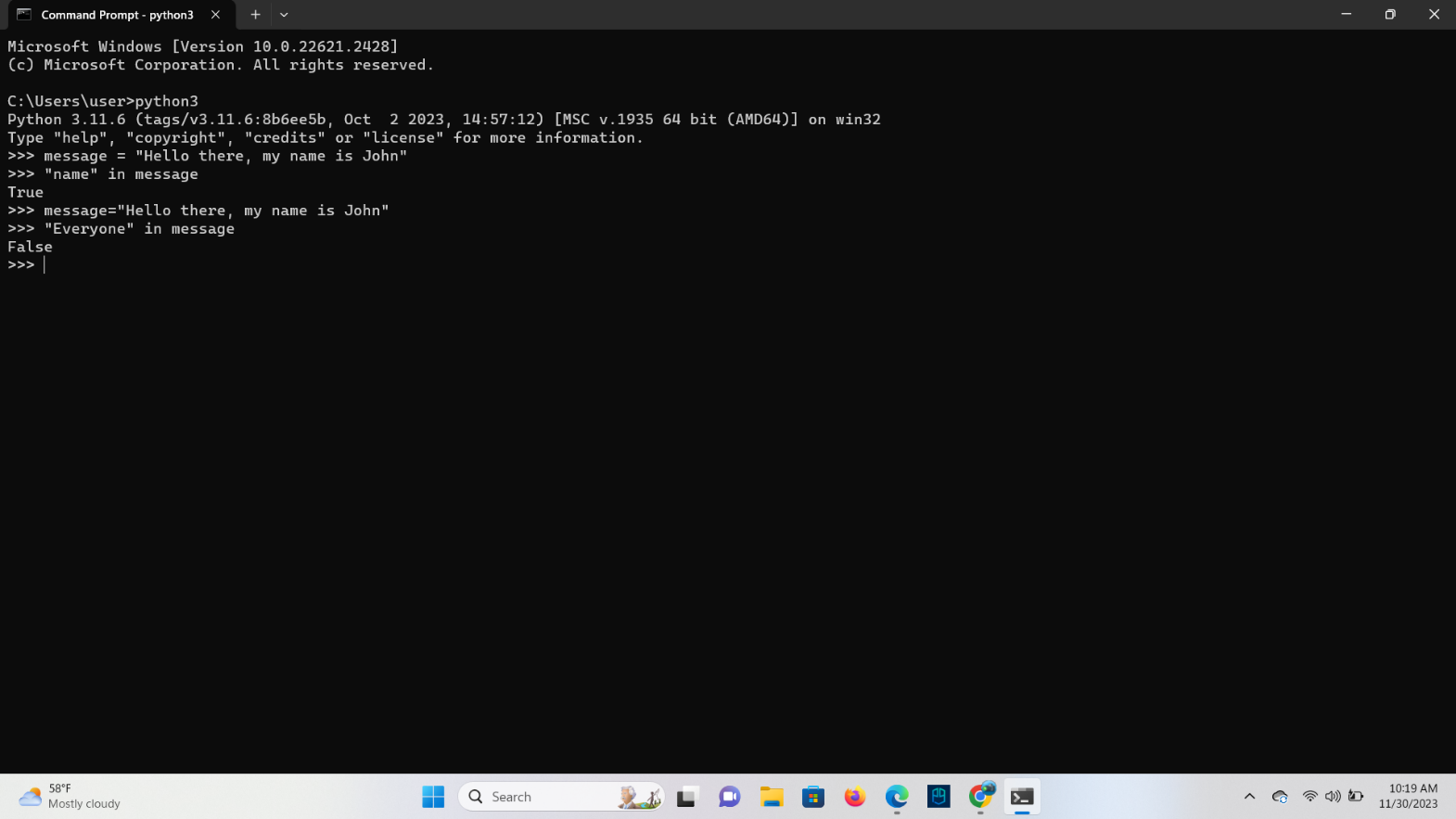
Using the ‘in’ operator:



Using the ‘not in’ operator:



Using the **in** operator with strings:



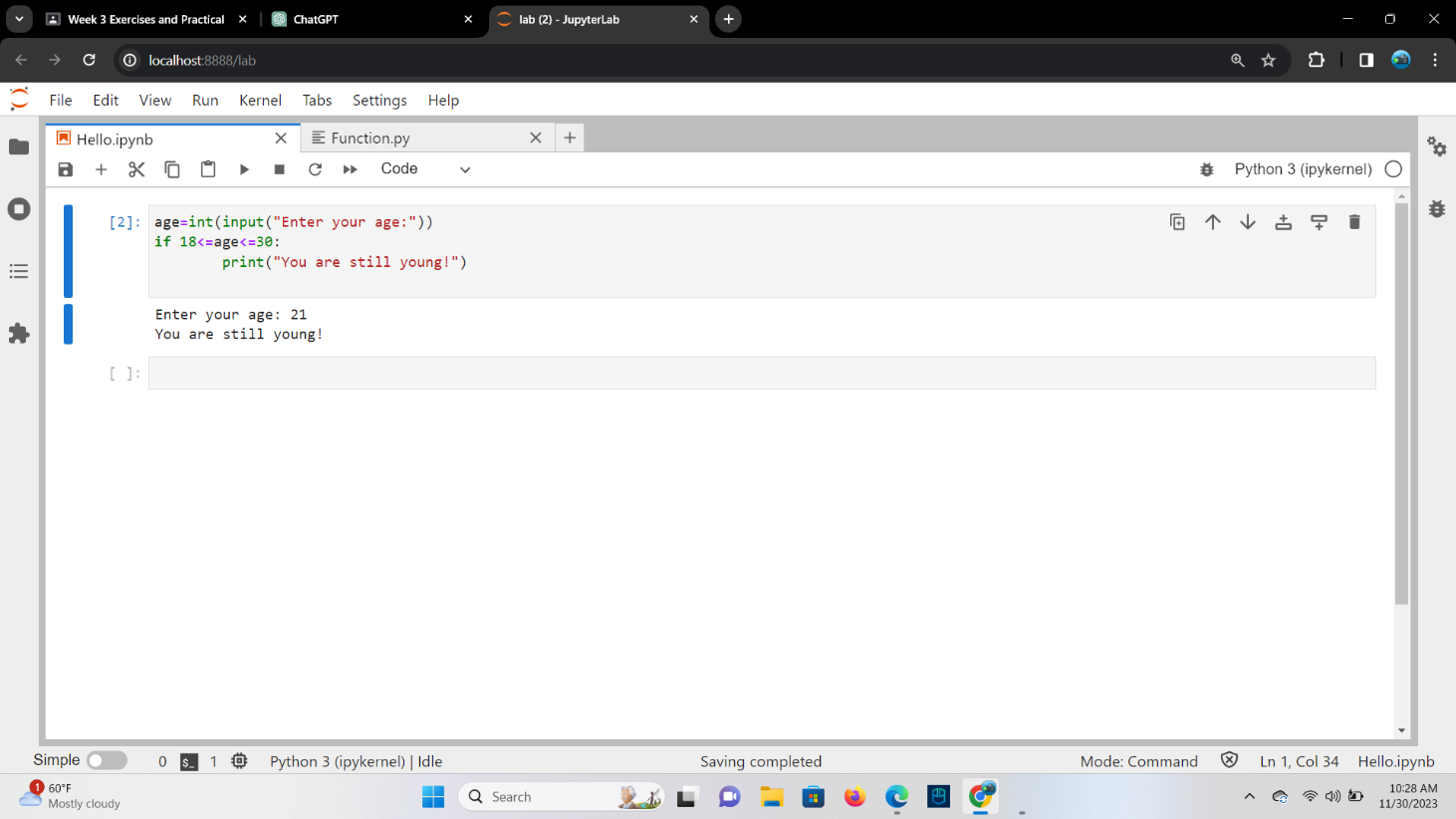
**Expressions: Points to note**

**The ‘if’ statement**

**TASK**: Try writing an if statement that checks if someone is between the ages of 18 and 30

inclusive. If they are, then print a message saying, “you are still young!”

**Ans**



**Using the ‘else’ clause**

For example, an if statement with the else clause may look like the following

if age >100:

print(“you are very old,”)

print(“well done!)

else:

print(“you are not very old – yet")

**Ans** if the age is assigning the value 21 then the output will come:

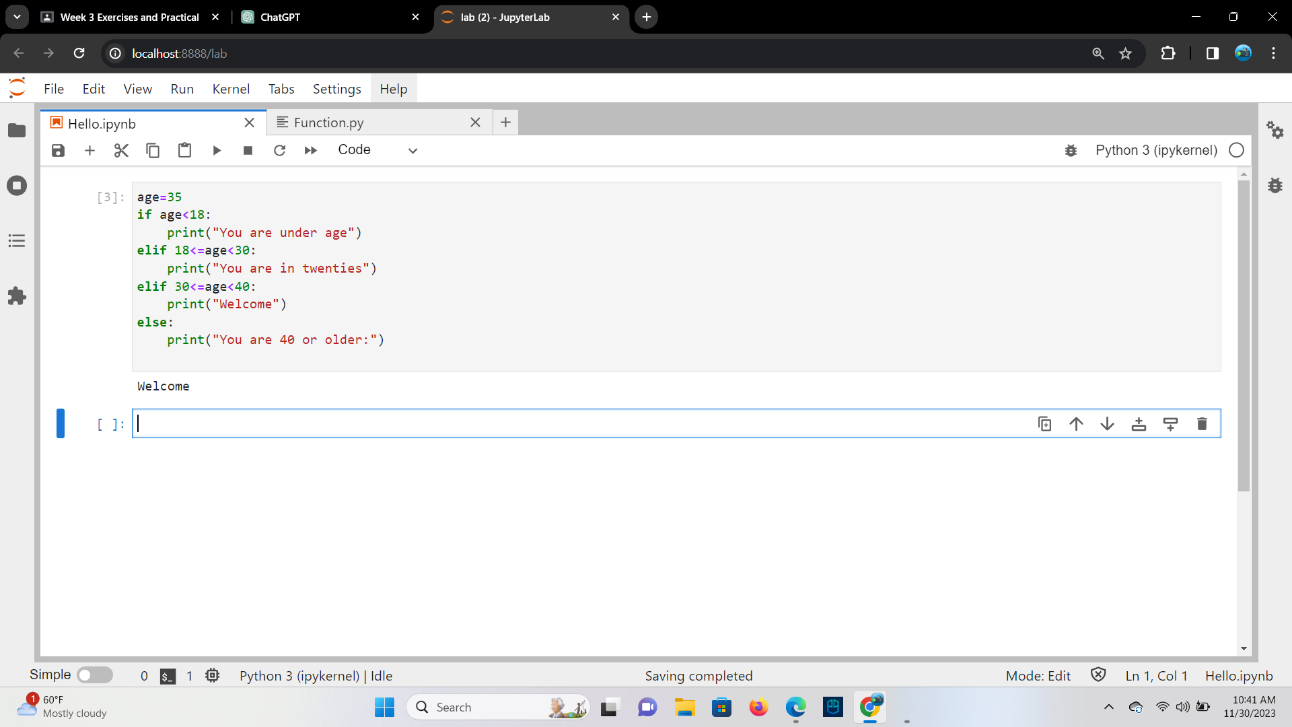
print (“you are not very old-yet")

**Using the ‘elif’ clause**

**TASK**: Try writing an if statement similar to the last example that includes an extra elif

clause to check ages between 30-40. Print a suitable message in the associated code block.

**Ans**

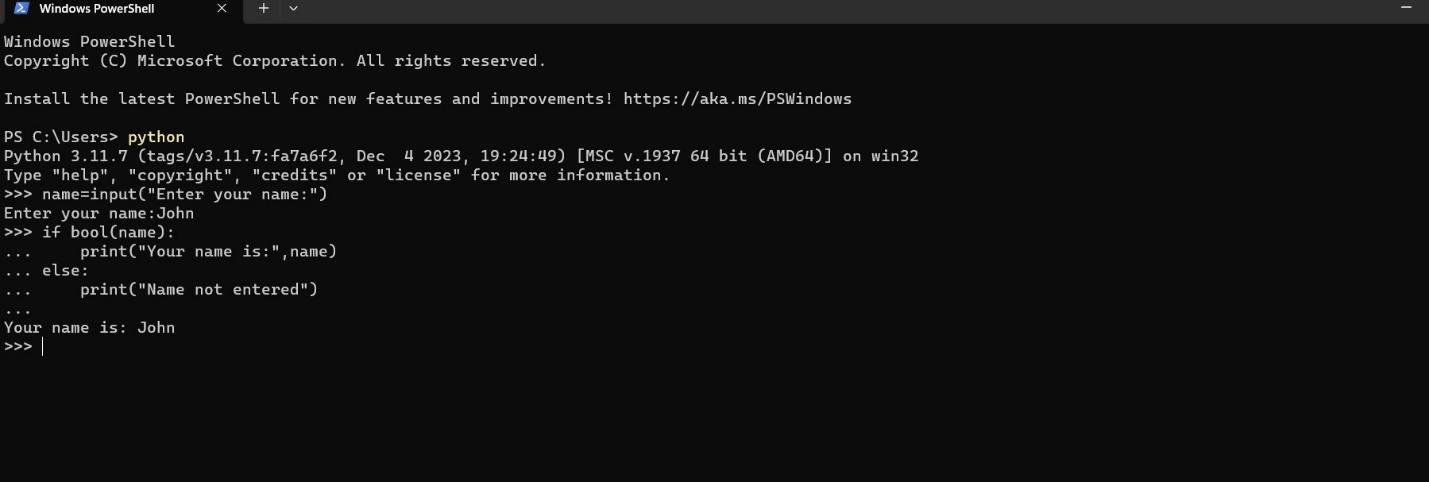


**Non-Boolean conditions**

**TASK**: Rewrite the above code that inputs a name then prints a message, but change the

condition so it explicitly uses a Boolean expression. Use the example below to help.

**Ans**



**The Ternary Operator**

**TASK**: Rewrite the code shown below as a single line Ternary expression.

if age >= 18:

print(“you are an adult”)

else:

print(“you arenot an adult, yet!”)

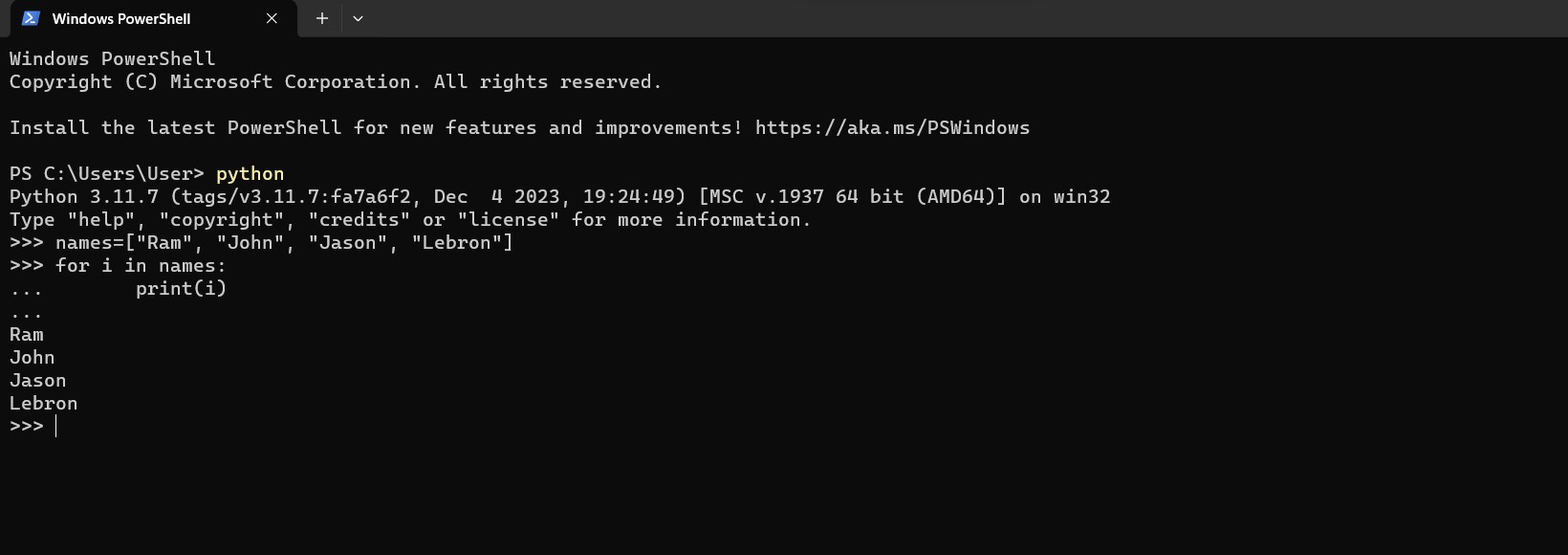
**Ans** print("you are an adult") if age >= 18 else print("you are not an adult, yet!")

**Using ‘while’ and ‘for’ loops**

**TASK**: Input and execute a for loop that iterates over a list of four names, printing each of

them to the screen.

**Ans**



**The range() function**

**TASK**: Input and execute a for loop that uses a range() function to generate the following

output:

2 to the power of 2 = 4

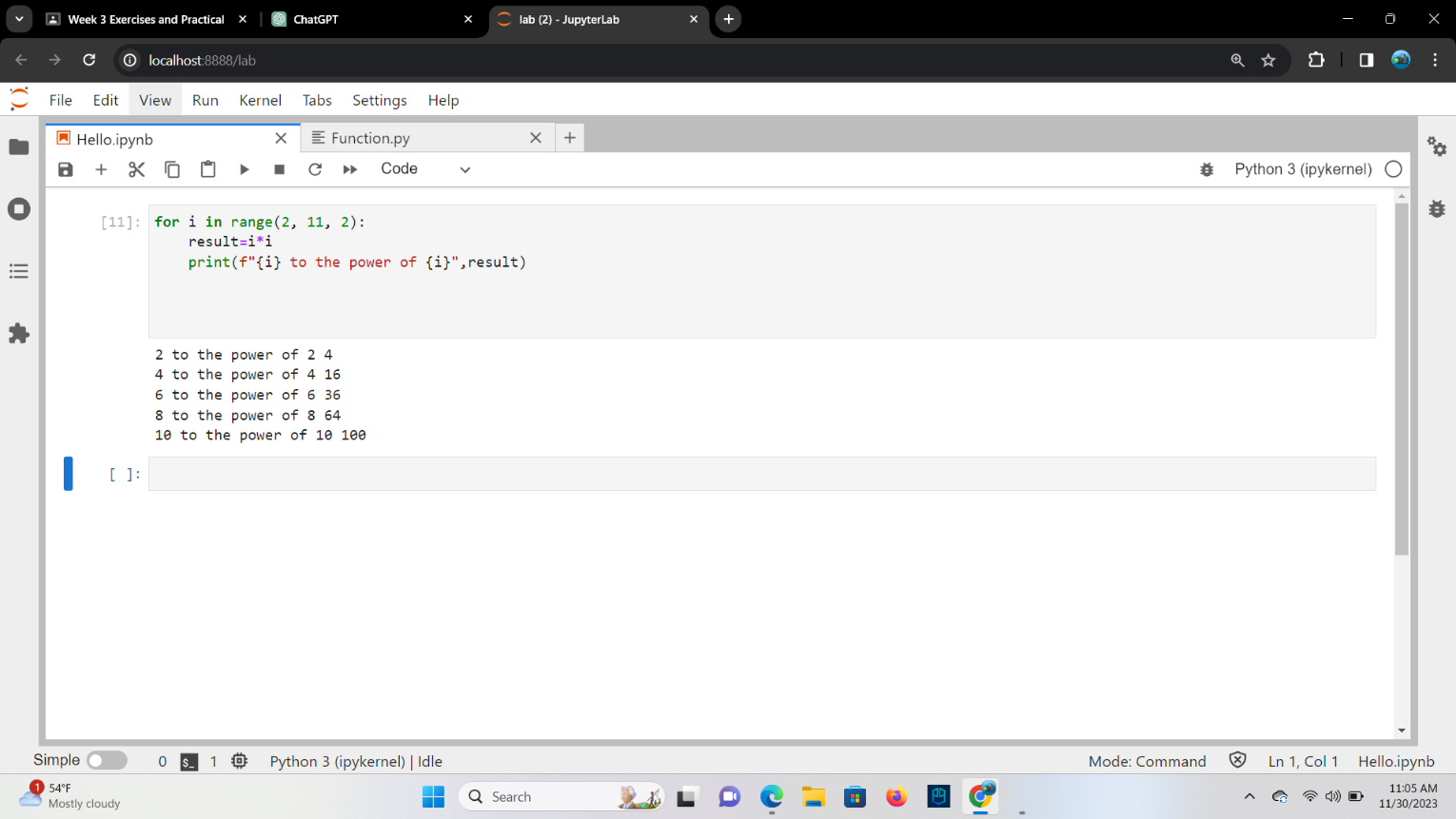
4 to the power of 4 = 256

6 to the power of 6 = 46656

8 to the power of 8 = 16777216

10 to the power of 10 = 10000000000

**Ans**



**Using ‘break’ within a loop**

value = int(input(“enter a number: “))

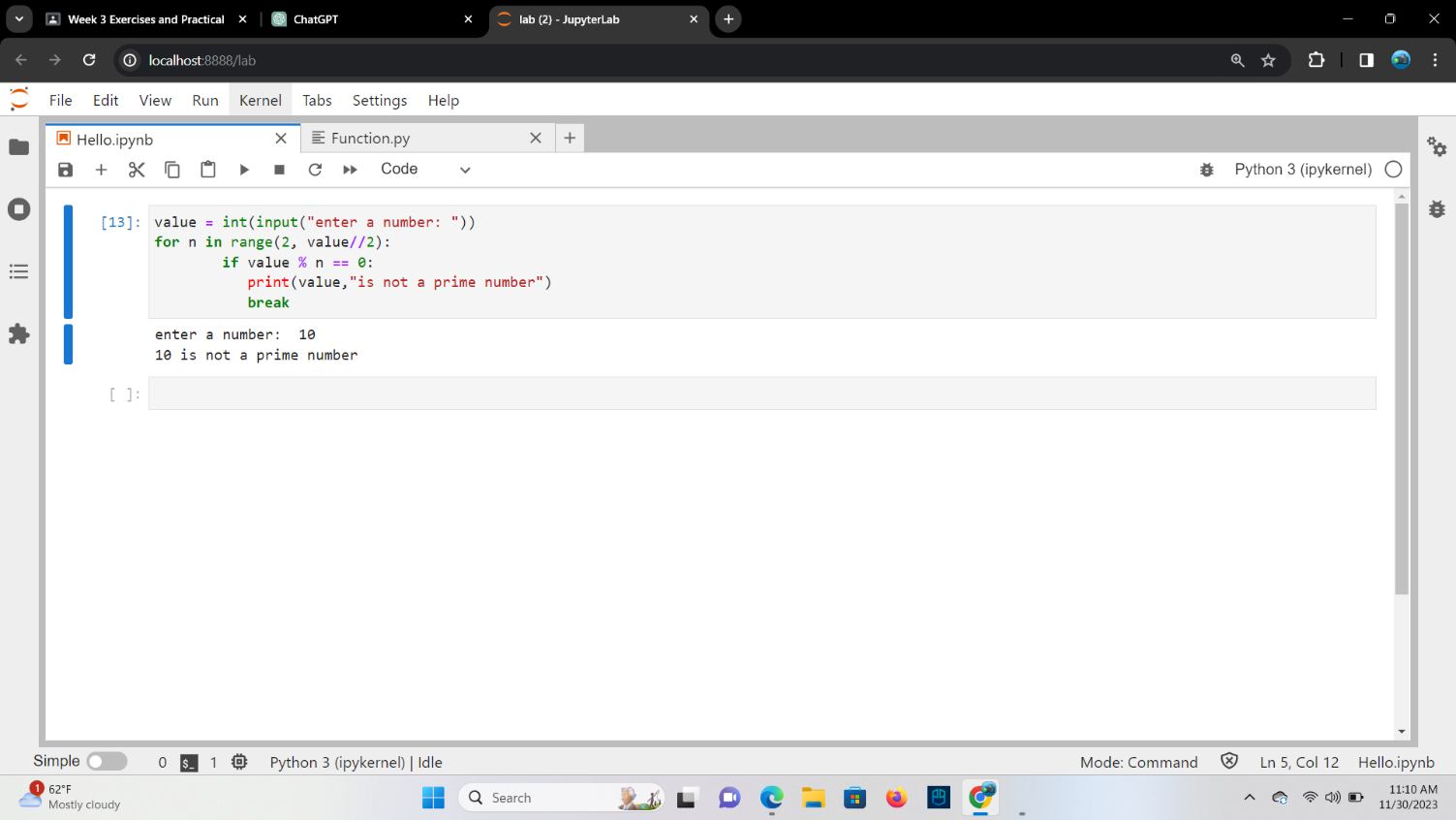
for n in range(2, value//2):

if value % n == 0:

print(value,”is not a prime number”)

break

**Ans**



value = int(input(“enter a number: “))

for n in range(2, value//2):

if value % n == 0:

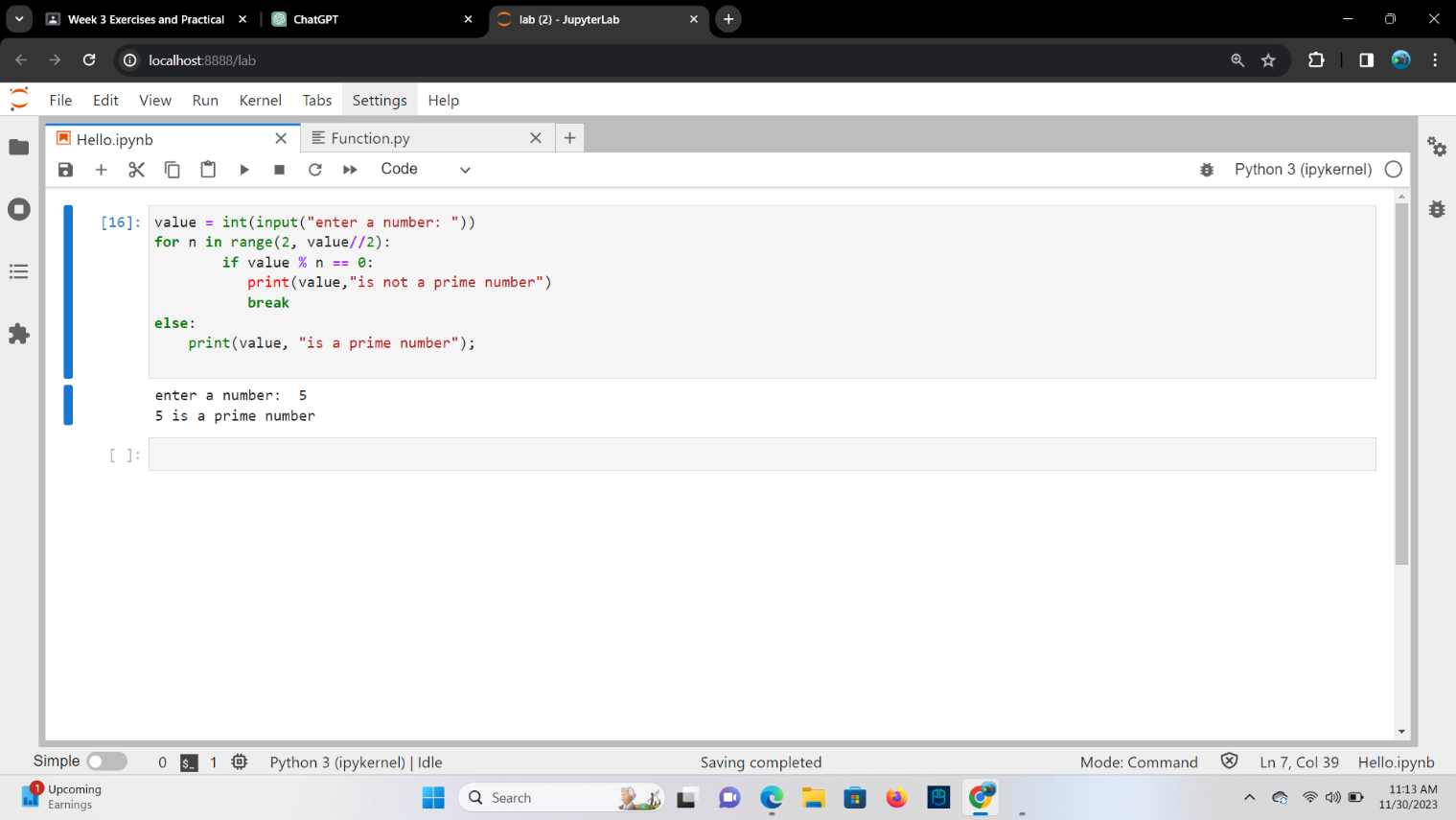
print(value,”is not a prime number”)

break

else:

print(value,”is not a prime number”)

**Ans**



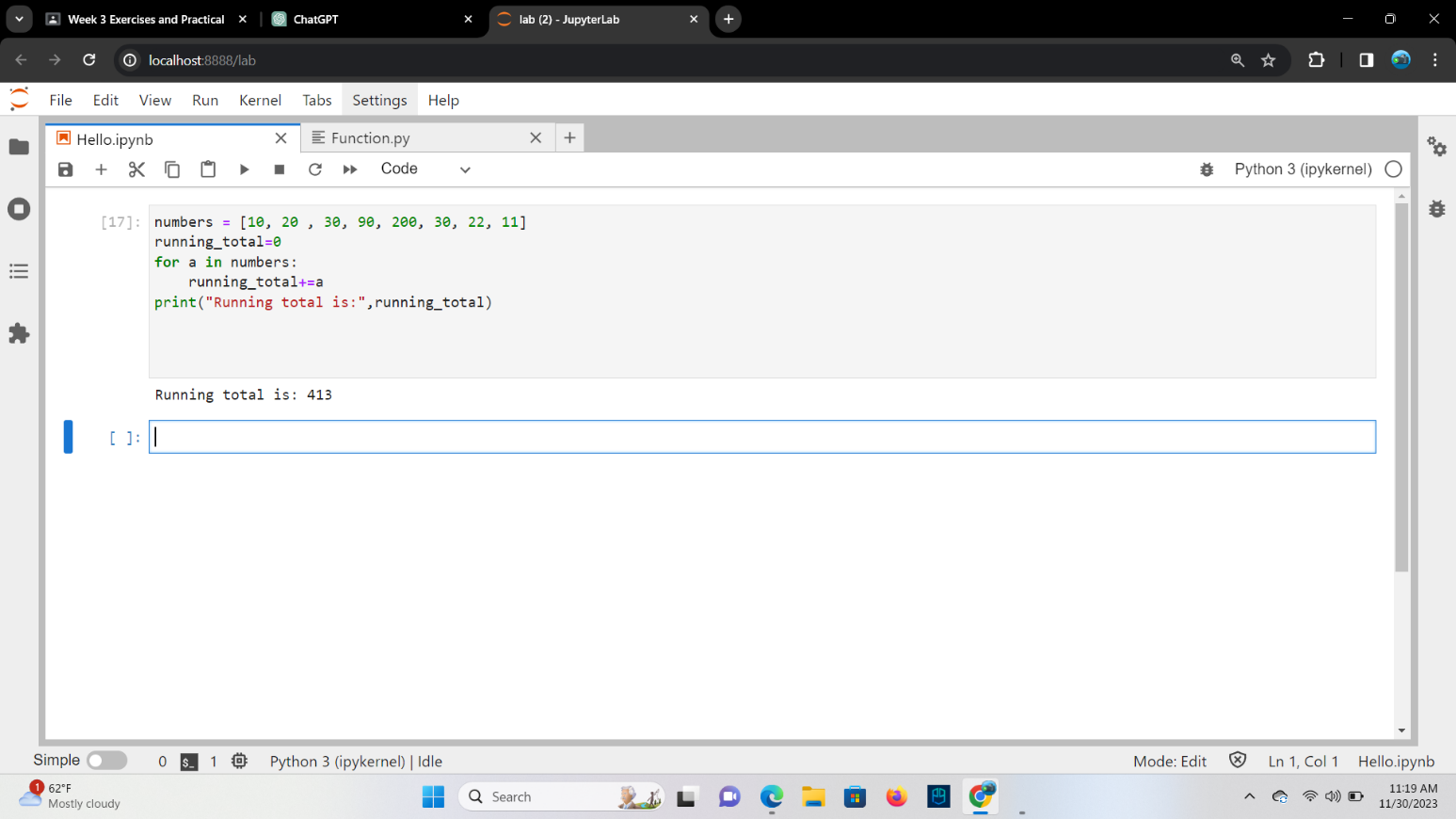
**Using ‘continue’ within a loop**

**TASK**: Input code containing a for loop that iterates over a list of numbers, printing a

running total during each iteration. You may wish to first define the numbers list as follows:

>>> numbers = [10, 20, 30, 90, 200, 30, 22, 11]

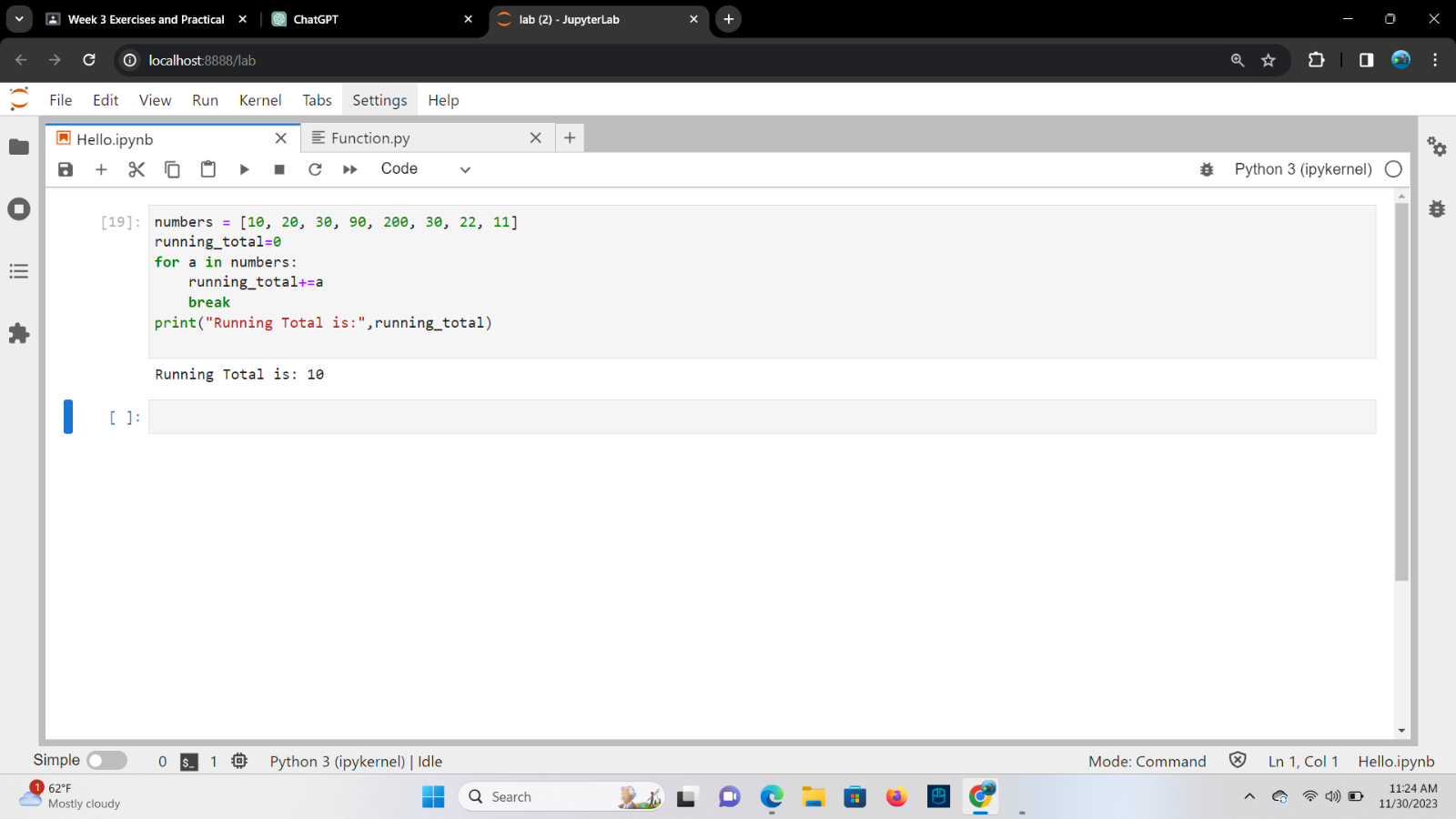
**Ans**



**TASK**: Amend your previous solution so that if any value within the list is found to be over

100 then the loop should break immediately.

**Ans**

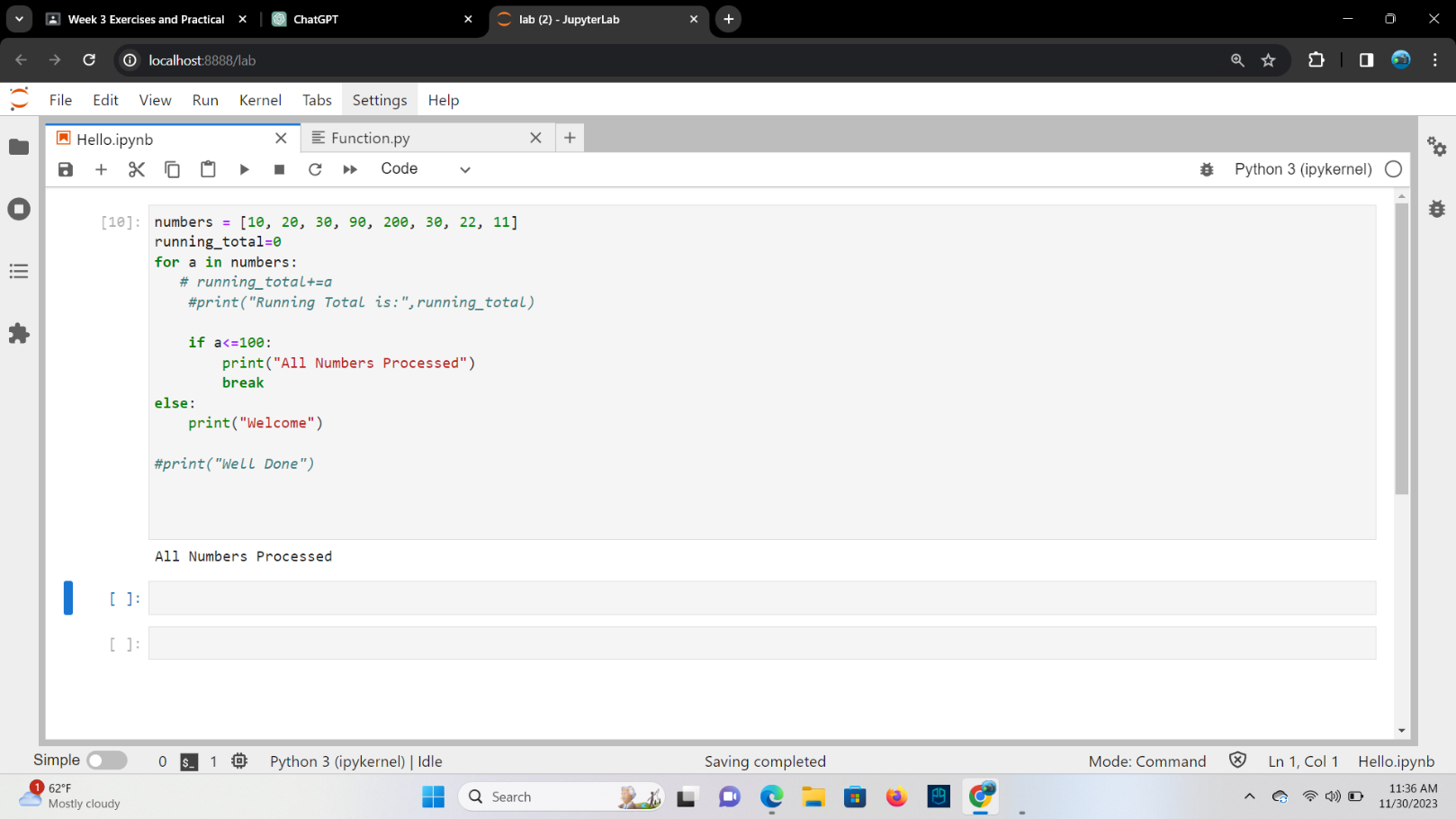


**TASK**: Amend your previous solution once again, so that the message “all numbers

processed” is printed when the loop completes, but only if all values were less or equal to

100 (i.e. the loop did not break early)

**Ans**



**Key Terminology**

**TASK**: Look at each of the phrases below and ensure you understand what each of these

means. For any that you do not understand, do a little research to find a definition of each

term. This research may involve looking back over these notes, or the associated lecture

notes. It may also involve searching for these terms on the Internet.

● Boolean Expression

**Ans** A Boolean expression in Python is an expression that evaluates to either True or False.

● Relational Operator

**Ans** In Python, relational operators are used to compare values and determine the relationship between them.

● Logical Operator

**Ans** In Python, logical operators are used to perform logical operations on Boolean values.

● Operator Chaining

**Ans** It refers to the ability to use multiple operators consecutively in a single expression.

● Ternary Operator

**Ans** It evaluates a condition and returns one of two values depending on whether the condition is true or false.

● Iteration

**Ans** Repetitive execution of the same block of code over and over is referred to as iteration.

● Nested Loop

**Ans** A nested loop refers to a loop within a loop, an inner loop within the body of an outer one.