Prithu Adhikari

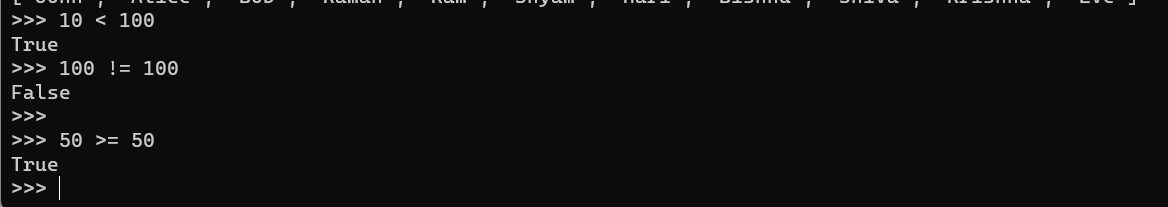
Week 3

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

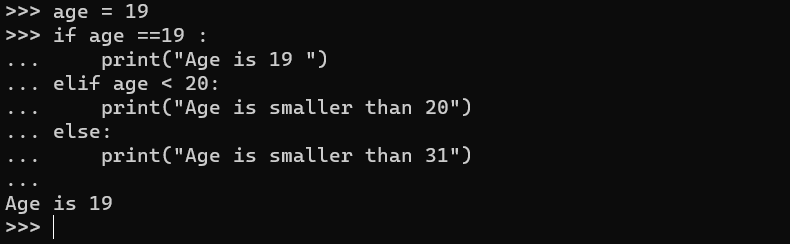
10 < 100

100 != 100

50 >= 50



**TASK**: Input a program that defines a variable called ‘age’ that is initialised to your own age. Then type several boolean expressions that compare the variable to see whether it is less than ‘18’, ‘21’ then ‘31’.

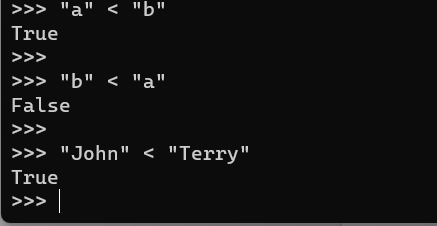


**TASK**: Try inputting the following code and note the results.

"a" < "b"

"b" < "a"

"John" < "Terry"



**ASK**: 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"

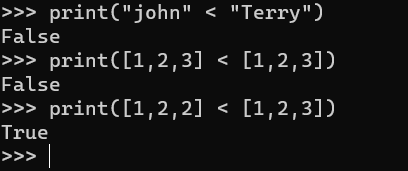
It is even possible to apply relational operators to lists (but less commonly seen), e.g.

>>> [1,2,3] < [1,2,3]

False

>>> [1,2,2] < [1,2,3]

True

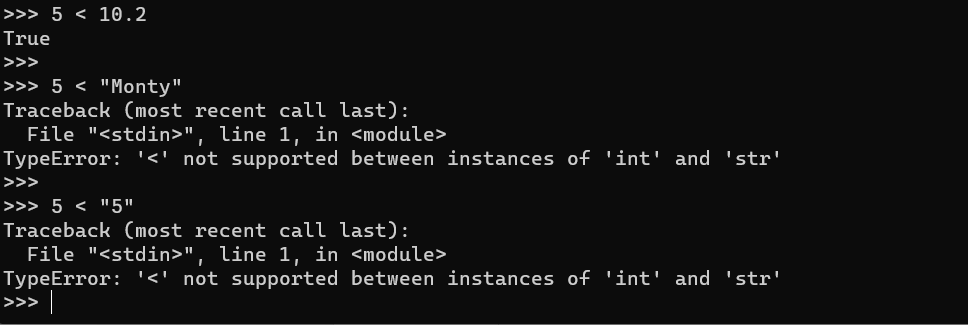


**TASK**: Try inputting the following code and note the results.

5 < 10.2

5 < "Monty"

5 < "5"



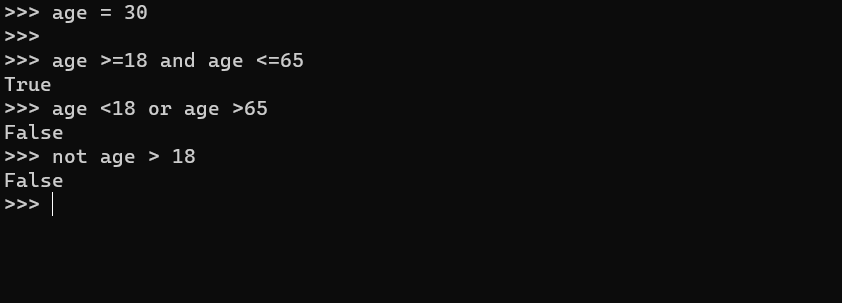
**TASK**: Try inputting the following code and examine the results.

age = 30

age >=18 and age <=65

age <18 or age >65

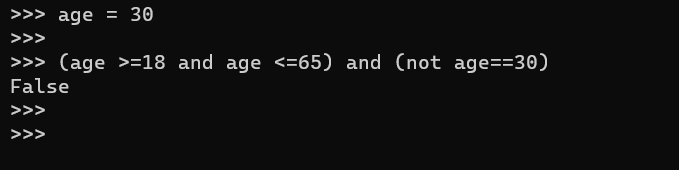
not age > 18



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

age = 30

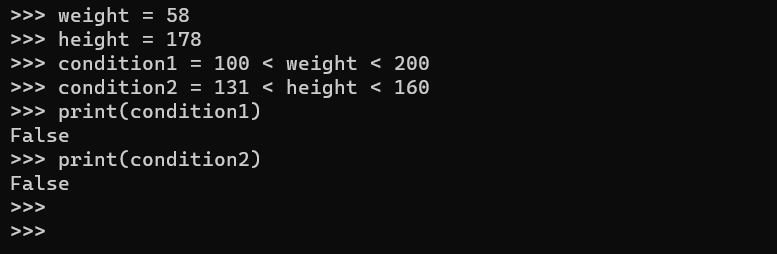
(age >=18 and age <=65) and (not age==30)



**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)

100 < weight and weight < 200

height > 131 and height < 160



>>> names = ["Terry", "John", "Michael", "Eric", "Terry", "Graham"]

>>> "Eric" in names

True

The not in operator can test whether a value is NOT within the list, for example -

>>> "Mark" not in names

True

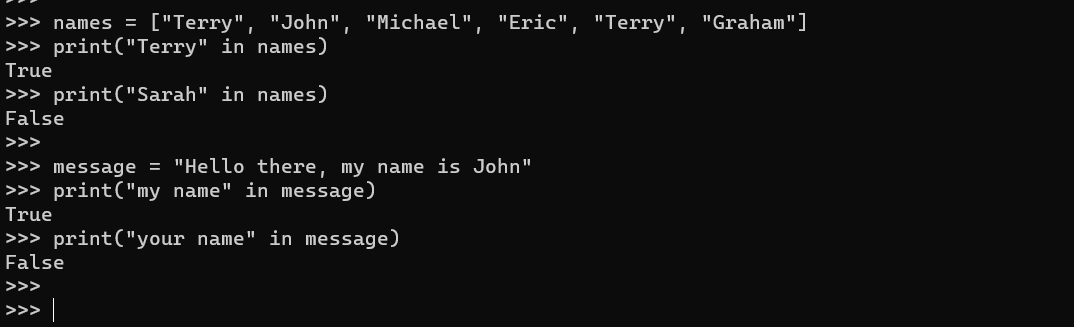
When the right-hand operand value is a string, then membership tests will return True if a substring is present, for example -

>>> message = "Hello there, my name is John"

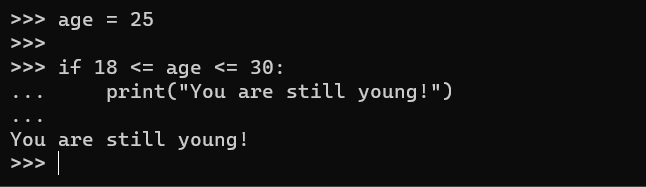
>>> "nam" in message

True

**TASK**: Input the examples above but with alternative operand values, that result in both True and False results.



**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!"



if age > 100:

print("you are very old,")

print("well done!")

elif age > 80:

print("you are fairly old")

print("pretty good!")

elif age > 40:

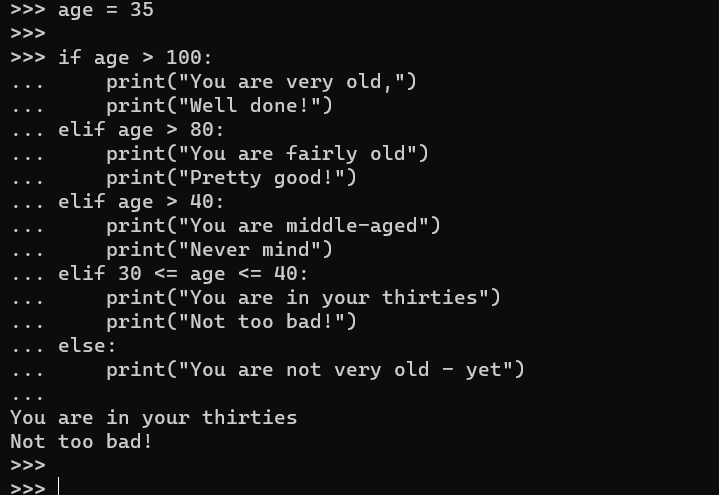
print("you are middle aged")

print("never mind")

else:

print("you are not very old - yet")

**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.



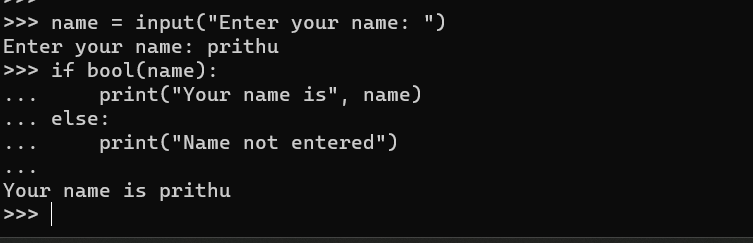
**TASK**: Rewrite the above code that inputs a name then prints a message, but change the condition so it explicitly uses a Boolean expression. Use the example below to help.

if total != 0:

print("Total is non-zero")

else:

print("Total is zero")



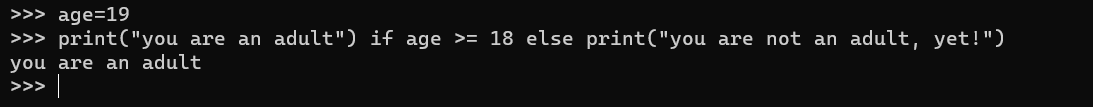
**TASK**: Rewrite the code shown below as a single line Ternary expression.

if age >= 18:

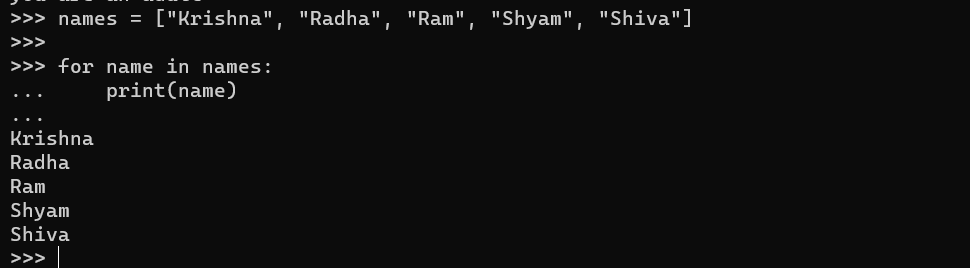
print("you are an adult")

else:

print("you are not an adult, yet!")



**TASK**: Input and execute a for loop that iterates over a list of four names, printing each of them to the screen.



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

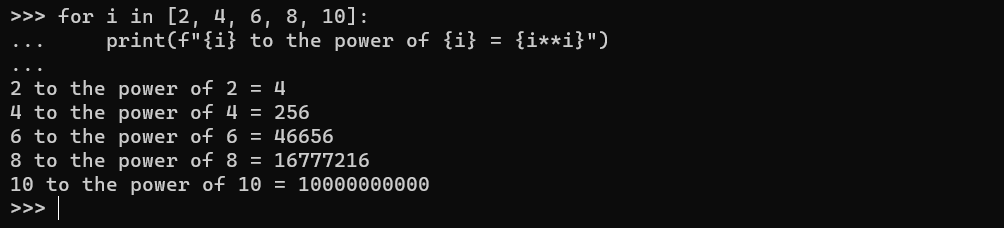
2 to the power of 2 = 4

4 to the power of 4 = 256

6 to the power of 6 = 46656

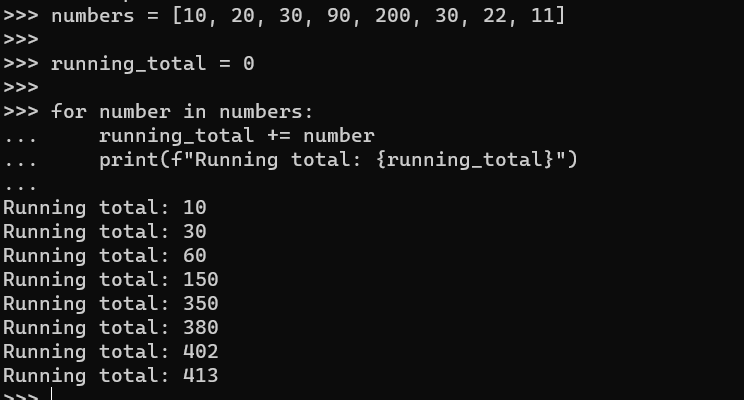
8 to the power of 8 = 16777216

10 to the power of 10 = 10000000000

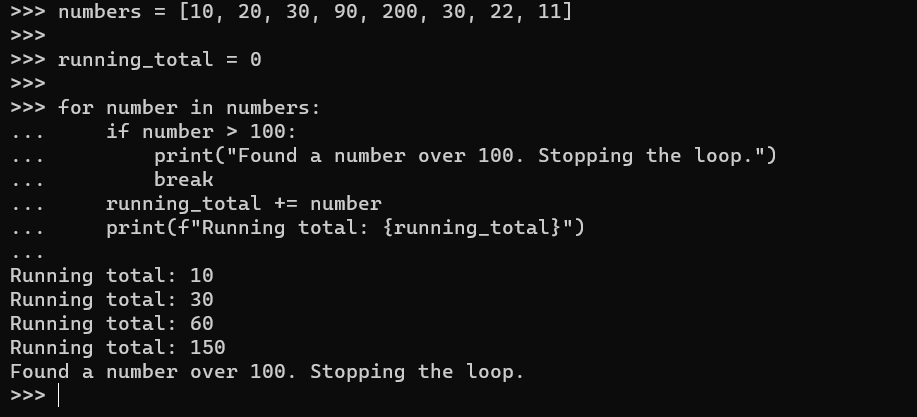


**TASK**: Input code containing a for loop that iterates over a list of numbers, printing a running total during each iteration. You may wish to first define the numbers list as follows:

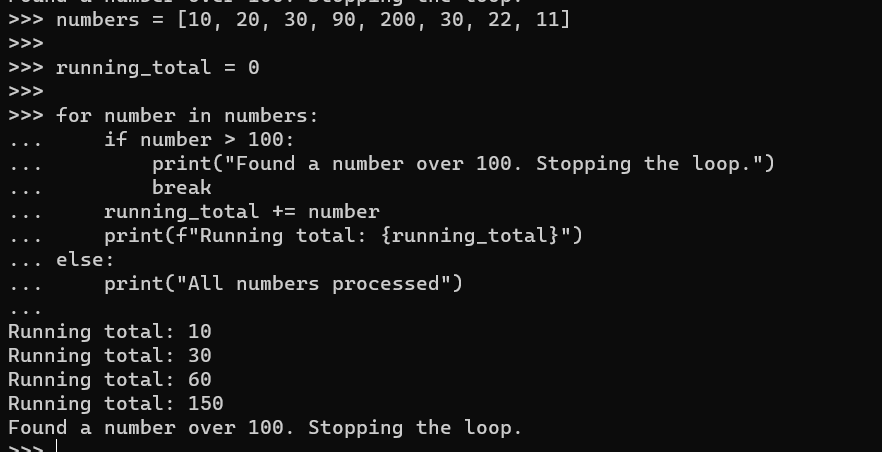
>>> numbers = [10, 20 , 30, 90, 200, 30, 22, 11]



**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

\

**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)



**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
* Relational Operator
* Logical Operator
* Operator Chaining
* Ternary Operator
* Iteration
* Nested Loop

Boolean Expression: An expression that evaluates to either True or False. It often involves comparisons or logical operations.

Relational Operator: An operator used to compare two values. Common relational operators include == (equal), != (not equal), < (less than), > (greater than), <= (less than or equal to), and >= (greater than or equal to).

Logical Operator: An operator used to combine or manipulate Boolean values. Common logical operators include and, or, and not.

Operator Chaining: The practice of using multiple operators in a single expression. For example, x < y < z is an example of operator chaining.

Ternary Operator: An operator that takes three operands and evaluates a Boolean expression. It is often used as a shorthand for an if-else statement. In Python, the ternary operator is expressed as a if condition else b.

Iteration: The process of repeatedly executing a set of statements. It is often associated with loops, where a block of code is executed multiple times.

Nested Loop: A loop inside another loop. In programming, it refers to having one loop structure (like for or while) inside another loop structure.