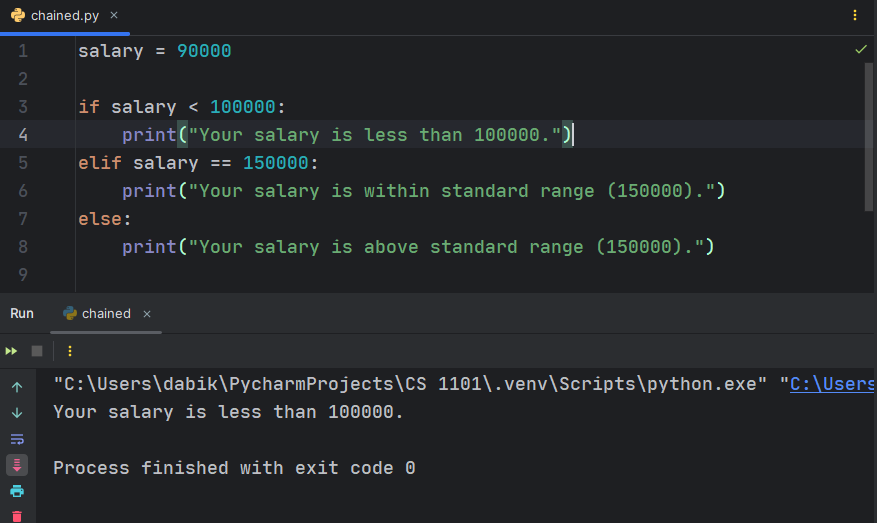
1. Describe the difference between a **chained conditional** and a **nested conditional**. Give your own example of each

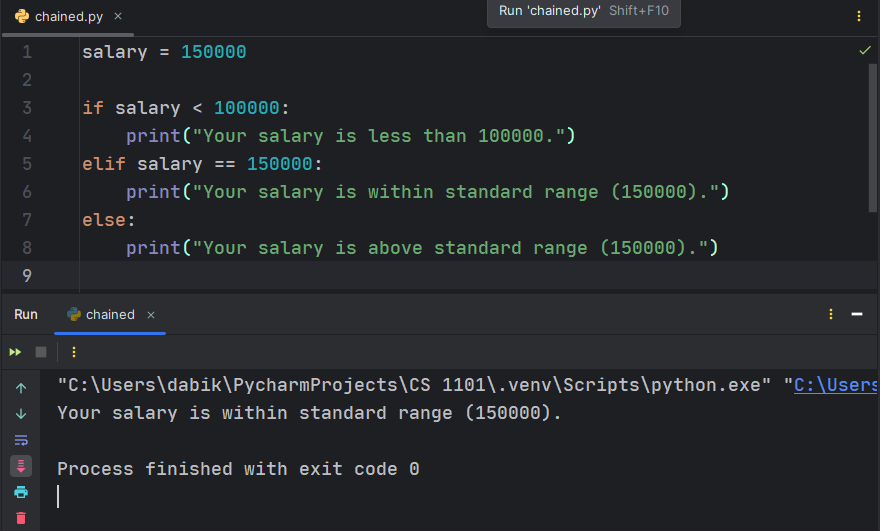
Both chained and nested conditionals are ways to structure conditional statements in programming, but they have different structures and uses.

* 1. In a chained conditional, multiple conditions are checked sequentially, and the corresponding code blocks are executed based on the evaluation of each condition. If one condition is true, its associated code block is executed, and the rest are not evaluated. Example: show image results of all three condition (Allen Downey, 2015) (kjdElectronics, 2018)

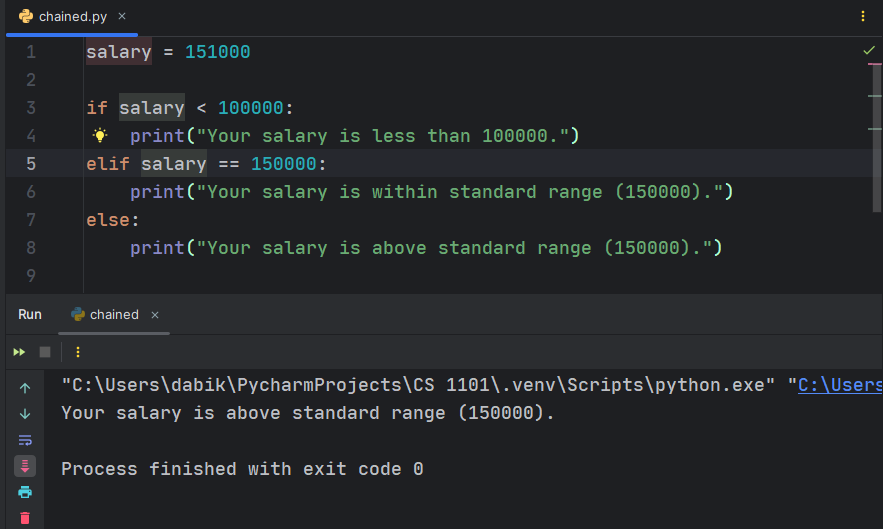
Results when salary is < 100,000



Results when salary is = 150,000



Results when salary is > 150,000

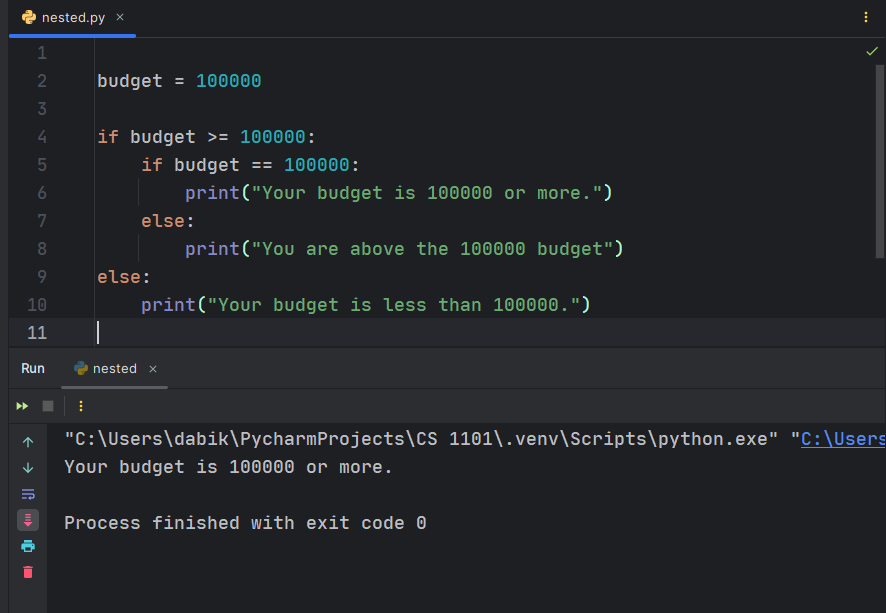


Explanation

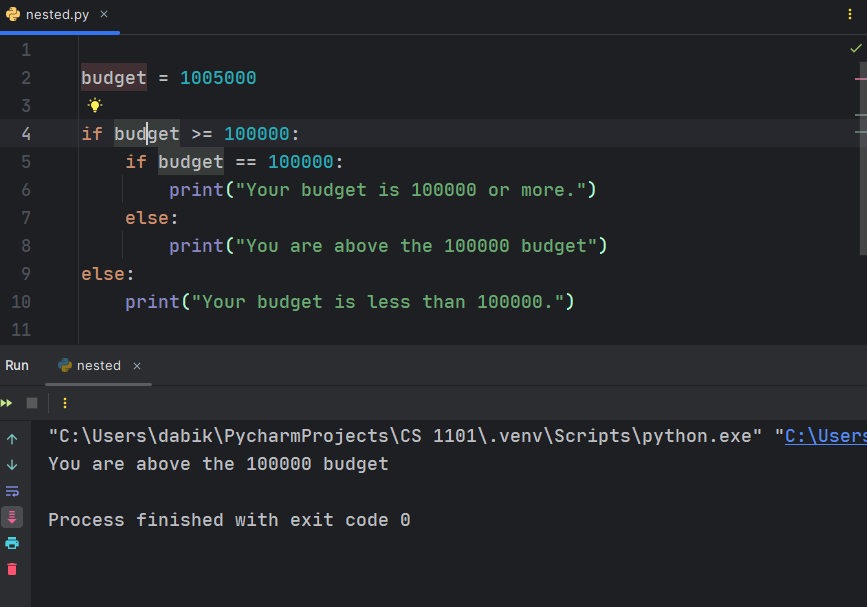
In my example, if salary is less than 100,000, the first condition is true, so "Your salary is less than 100000" will be printed. If salary is not less than 100,000, but equal to 150,000, "Your salary is within standard range (150000)" will be printed. If neither of these conditions is true, "Your salary is above standard range (150000)" will be printed.

* 1. In a nested conditional, one or more conditionals are nested within another code block. This means one or more conditions are checked only if the outer condition is true. Example: show image results of all conditions (Allen Downey, 2015).

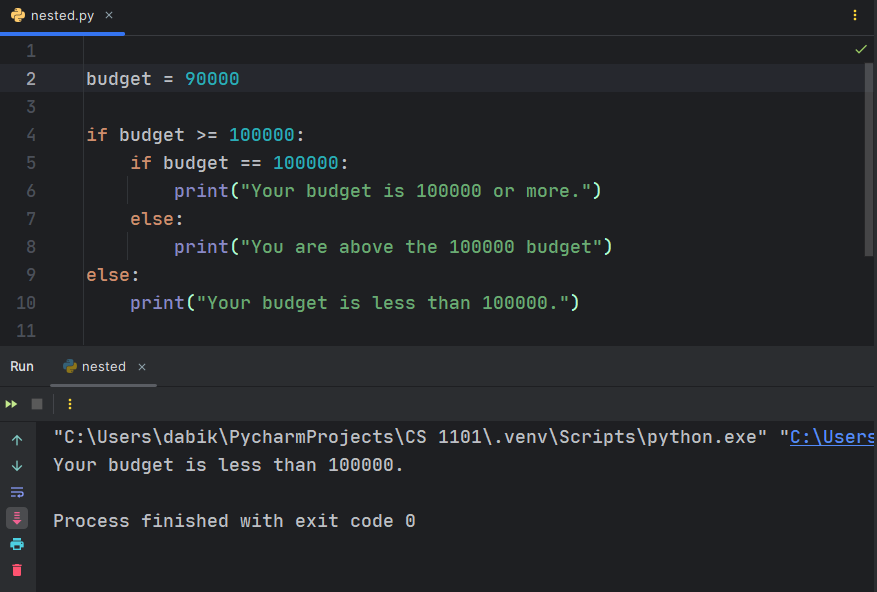
When budget is = 100,000



When budget is > 100,000



When budget is < 100,000



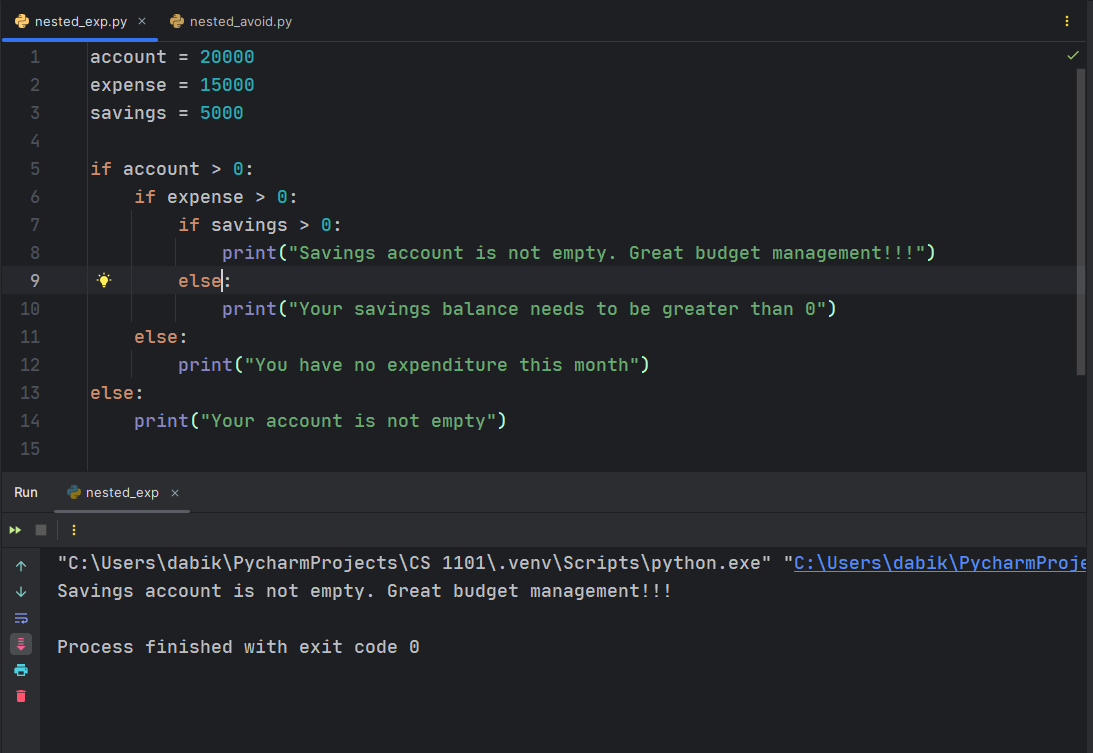
Explanation.

Based on the example above, the outer condition checks if budget is greater than or equal to 100000. If it is, the inner conditional checks whether budget is equal to 100000. If budget is not equal to 100000, the inner else block is executed, printing "You are above the 100000 budget". If the outer condition is false, "Your budget is less than 100000." is printed.

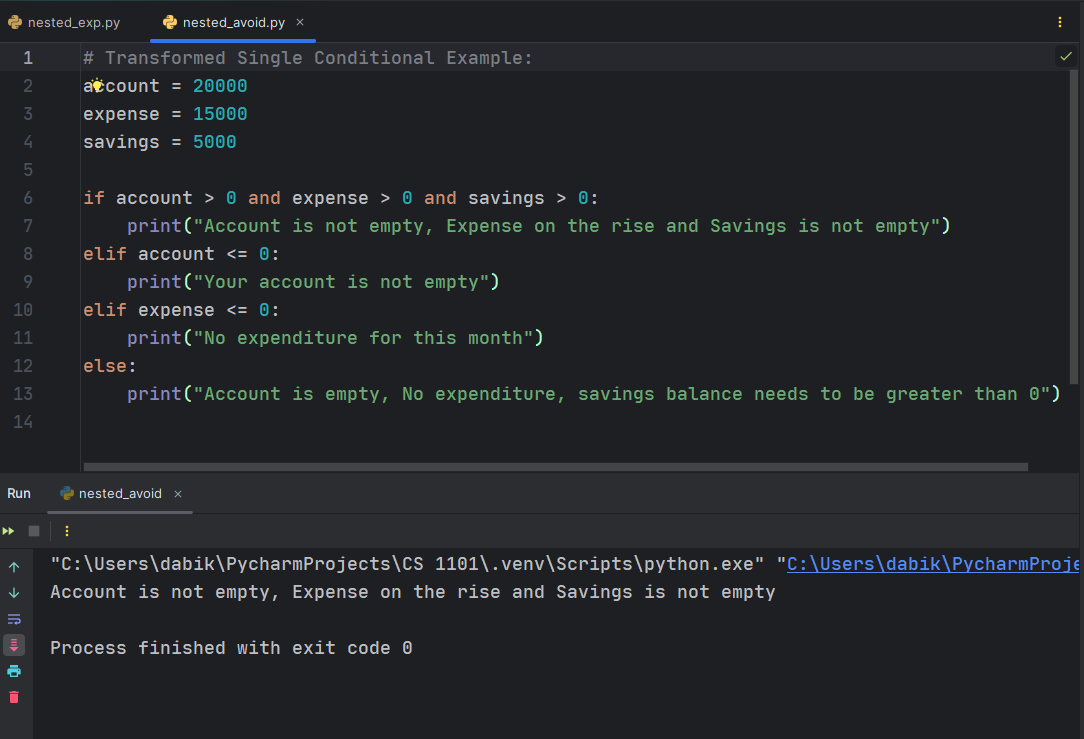
However, a chained conditional evaluates multiple conditions sequentially, while a nested conditional contains conditionals within conditionals, allowing for more complex logic to be implemented.

1. Deeply nested conditionals can become difficult to read. Describe a strategy for avoiding nested conditionals. Give your own example of a nested conditional that can be modified to become a single conditional and show the equivalent single conditional.
   1. One effective strategy for avoiding deeply nested conditionals is to use logical operators and compound conditions to combine multiple conditions into a single conditional statement. By carefully structuring the conditions, you can often simplify the code and make it more readable.

Nested Conditional example



Nested Transformed Single Conditional:

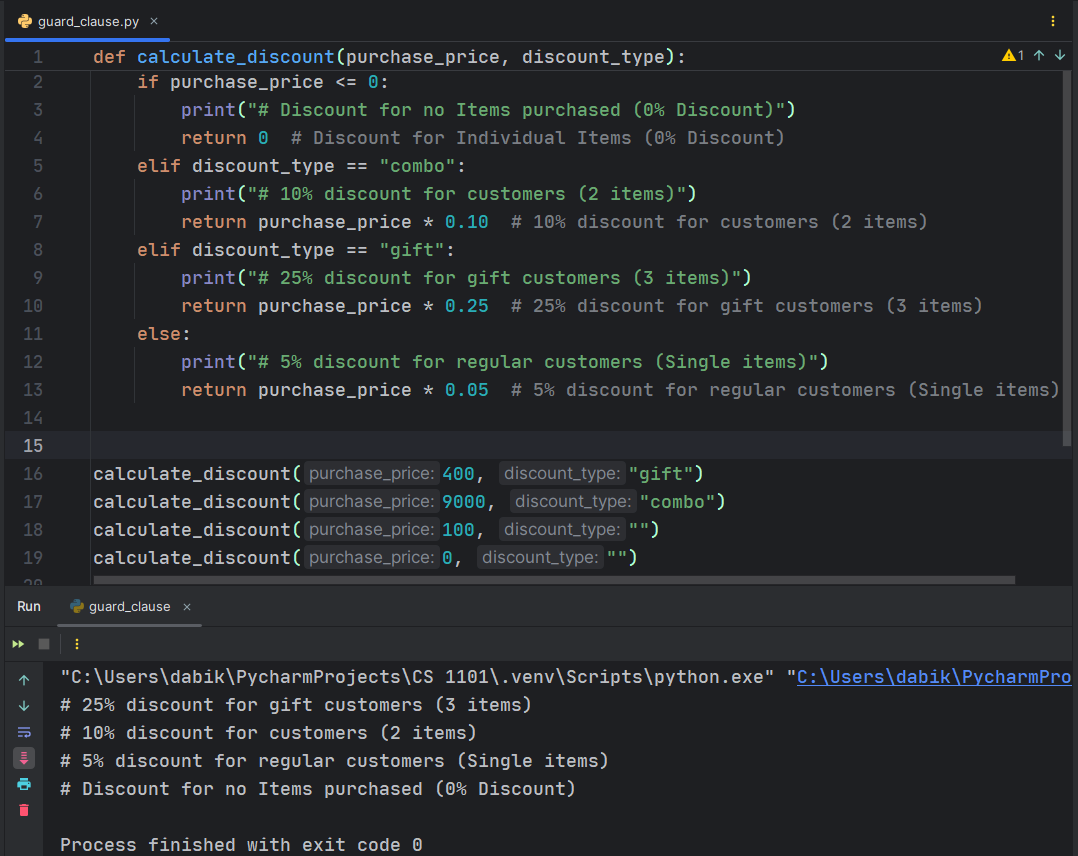


Explanation

In the example above, I have combined the conditions "account > 0", "expense > 0", and "savings > 0" into a single conditional using the and logical operator. These single conditional checks if accounts, expense and savings are not 0. If not, it evaluates subsequent conditions to determine which is <=0. This approach eliminates the need for nested conditionals and makes the code more concise and easier to understand (Allen Downey, 2015).

* 1. Another strategy for avoiding nested conditionals is to use guard clauses or early returns. This approach involves checking for conditions that would result in the function exiting early and returning a result. By handling exceptional cases first, you can reduce the need for nested conditionals and make the code easier to read and understand (Olsen, 2024).

Example



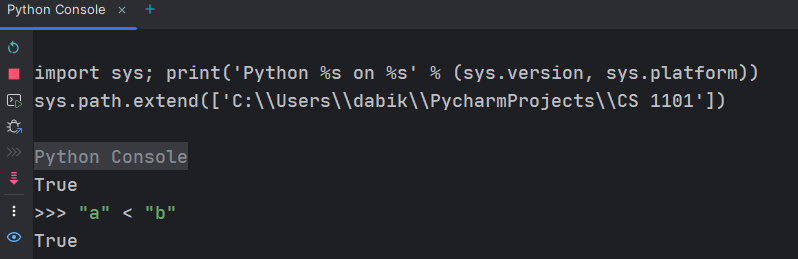
Explanation:

In this example, I have handled exceptional cases first: if the purchased price is non-positive, or if the discount type is "combo" or "gift", we return the appropriate discount immediately. This eliminates the need for nested conditionals to handle these cases.

Using guard clauses or early returns in this way can make the code more readable, maintainable, and easier to reason about, especially when dealing with multiple conditional branches.

Question

Using the logical operator “<,> or =” to compare alphabets is possible. See the example below.



How is this possible?

# References

Allen Downey, G. T. (2015). *Think Python: How to Think Like a Computer Scientist.* Needham, Massachusetts: Green Tea Press.

kjdElectronics. (2018). *Python Beginner Tutorial 5 - Booleans and Conditionals (Reposted w/ Zoom)*. Retrieved from Yutube: https://www.youtube.com/watch?v=E4wbrwDpnIg

Olsen, D. (2024, February 7). *Avoiding Nested If Statements with Guard Clauses*. Retrieved from LinkedIn: https://www.linkedin.com/pulse/avoiding-nested-statements-guard-clauses-danny-olsen-a1e8f/