Hi, welcome to Basic Program Structure.

☼Part D1: More on Selection (Branching)

☼As mentioned earlier, the intelligence that computers appear to have come from their ability to make billions of tiny decisions every second. Each of those decisions is based on a choice between the truth and falsity of an expression. In this lesson, we will explore more on the selection. At the end of this lesson, you should be able to apply the following selection (branching) structures:

IF-ELSE statement

Nested IF statement

Besides that, you should be able to Use Logical/ Boolean Operators to chain expressions.

☼ In this lesson, we will go through them in details one by one.

☼ In scenario 3, we only have one coffee shop. Let's add one more coffee shop to our scenario. Now, we have two coffee shops to choose from based on distances. We need to calculate the distances traveled from home to each of the coffee shops, and compare **them to Find the Nearer one.**

☼Here's an improved flow chart. Besides adding a process to calculate travel distance to location B, a decision box and two more processes are added.

"<=" is the relational operator to compare two numbers to return a Boolean value of either True or False.

If the conditional expression " Is distanceA <= distanceB?" is true, select coffee shop A . Or else, select coffee shop B. In this example, 6<=2 is false, so select coffee shop B.

This simple decision represents a choice of doing one of two things: doing one thing or the other.

☼A variation on the basic if statement adds the ability to execute one suite of statements if the decision Boolean is True or to execute a different suite of statements if the Boolean expression is False. Note that there are two statements and two suits here: the suit associated with the if and the suite associated with the else. Also, note that the if and else are at the same indentation; the else is not part of the if suite, but together they create an if-else compound statement. After this if-else statement, the program continues the sequence by executing the next instruction.

☼Let's visualize it using flowchart.

Evaluate condition, that is, a Boolean expression to yield True or False.

If the condition yields True,

then execute suite A

and continue with rest of the program.

If the condition yields False,

then execute suite B,

and continue with rest of the program.

☼Adding more features to scenario 5 we have scenario 6. After finding the Nearer Coffee Shop, we shall decide Transportation to the selected destination. If the distance is longer than 3km, take a taxi, or else, ride a scooter.

The task becomes more complex. Let's work out the flowchart first.

Find distance to location A Find distance to location B

Is distanceA <= distanceB? if yes, Is distanceA < 3 km?

if yes, Scooter to location A end of the program

if distanceA is not less than 3 km then Taxi to location A

end of the program

If distanceA <= distanceB is false, go to location B

Then same transportation rule applies to location B.

Let's look at our example (Scenario animation and flowchart path highlighted).

Find distance to location A, which is 6. Find distance to location B, which is 2.

Is distanceA <= distanceB? False. Then go to location B

Is distanceB < 3 km? yes, Scooter to location B end of the program

How to implement the flow chart? It looks quite complex.

☼Let's recall the "if statement". It is our first control statement and, simple as it is, it is quite powerful. It is a critical part of enabling computers to perform complex tasks.

☼First, we can use Logical operators to make the conditional expression more powerful and complex. What are Logical operators?

Logical operators connect Boolean values and expressions and return a Boolean value as a result.

The basic logical operators are: not, and, or.

The 'not' operator flips the value of the Boolean operand. That is, it converts true to False, and false to True.

The 'and' operator requires both A and B to be True for the whole expression to be true. Otherwise, the value of the expression is false.

The 'or' operator only requires at least one of A and B to be true for the whole expression to be True. Therefore, the expression is false when neither A nor B is true.

☼Let's have a quick check on relational operators and logical operators.

What is the value of the given expression? (not(10>=20) and not(10<30)

☼ The answer is False. Let's analyze.

- 10 >= 20 is False. So not 10>=20 is True
- 10<30 is True. So not 10<30 is False
- True and False, Since not both values are True, hence, (not(10>=20) and not(10<30)) is **False**.

☼Besides that, can there be another IF statement(s) inside the True Block? Yes. Within the suite of any control structure, we can insert another control structure.

☼**Can there be another IF statement(s) inside the False Block?** Yes.

☼**Can there be another IF statement(s) inside the True Block of a True Block?**

Yes. Within the inserted control construct/kən'strʌkt/, we can insert yet another one. We can nest one within another with no limit (in reality there usually is a limit, but if you reach it, your code is unreadable and should be restructured).

☼Can there be another IF-ELSE statement(s) inside the True Block? It’s not an issue at all.

☼quick summary:

In this lesson, we have revisited if statement and introduced two more selection structures: if-else and nested if Statements With these and the previous concepts learnt, you are now equipped to understand basic programs.