Learning Activity (2 of 2): Complex Conditions

Prepare

In this activity, you will continue practicing working with **`if`** statements, using even more complicated programs and logic.

Preparation Material

Watch the following videos:

- Direct Link: Complex Condition Checks (4 mins)

- Direct Link: Demo: Complex Conditions (8 mins)

Testing Multiple Conditions

As shown, you can combine two conditions with **`and`** and **`or`**:

```
if x > 5 and y > 10:
    # This happens if both conditions are true

if x > 5 or y > 10:
    # This happens if either one of the conditions (or both!) are true
```

And you can mix and match any expressions that evaluate to true:

```
if x > 5 and y > 10 and word == "taco":
    # This happens if all three conditions are true
```

Just as mathematical operators have precedence rules where **`*`** happens before **`+`**, the **`and`** condition takes precedence over the **`or`** condition, so if you want to mix them and have the or condition happen first, you need to use parentheses:

```
if (x > 5 or x < -5) and y > 10:
    # In this case, x can either be greater than 5 or less than negativ
```

```
    # always be greater than 10

if x > 5 or x < -5 and y > 10:
    # Without parentheses, the x < -5 and y > 10 conditions go togethe
```

Sometimes, these can get a little confusing, so it is a good idea to use parentheses in complicated expressions, to make it clear what you intend to happen.

Testing Multiple Similar Conditions

It is common to want to test the same variable for two conditions, or to test two variables against the same condition. In these cases, you may be tempted to leave part out, but this will not work, anytime you use operators like **>** or **==**, you must provide both a left and a right hand side for it.

```
# To check if x is either 5 or 6, you might be tempted to write:
if x == 5 or 6:
    # INCORRECT: This does not work!

# You must write them both out:
if x == 5 or x == 6:
    # This is the correct way to check
```

```
# To check if either first_name or last_name is Scott you might be temp
if first_name or last_name == "Scott":
    # INCORRECT: This does not work!

# You must write them both out:
if first_name == "Scott" or last_name == "Scott":
    # This is the correct way to check
```

True and False values

In addition to comparing against other variables, you can also set a variable itself to be a **`True`** or **`False`**. To this point, we have learned about the variable types of int, float, and string. Variables that are either true or false are a new data type called Boolean variables (named after a mathematician, George Boole).

You can set a boolean variable directly, such as **`is_hot = True`** or **`is_hot = False`**. Notice that you must always use a capital T on True and a capital F on False.

Because these variables are already either true or false, you don't need to compare them against another value. For example:

```
is_hot = True

# You can check the value of the variable directly
if is_hot:
    print("It's hot")

# It works, but is redundant (and therefore bad practice) to check if i
if is_hot == True:
    print("It's hot")
```

You can check if a boolean variable is not true with the **`not`** keyword:

```
is_hot = True

# Using the "not" keyword
if not is_hot:
    print("It is not hot")

# It works, but is generally avoided to check if it is false:
if is_hot == False:
    print("It is not hot")
```

Boolean variables are typically named with words that indicate that it will only have two values, and help you understand what true means. For example, if you want to have a boolean variable that determines whether someone is an adult, you would typically choose a name of **`is_adult`** rather than **`age`** or even **`adult`**.

Common boolean variable names are **`is_xxx`**, **`has_xxx`**, **`can_xxx`**, etc.

Activity Instructions

For this activity, you will implement a simplistic system to determine if a user can qualify for a loan.

The Problem: Qualifying for a loan

When loaning money to someone, you want to have some level of confidence that they will be able to repay the loan.

There are different characteristics you might base this decision on, or different questions you might ask someone. And depending on their answers to those questions, you might ask other questions. For example, consider the following possible questions:

- Does the person have a job, and if so, how long have they worked there, and how much money do they make?
- Is there good collateral for the loan? (for example, is the loan against a car or home that is worth at least the amount of the loan?)
- Does the person have a good credit score or history of paying back loans?
- How much other debt does the person have?
- How much money does the person have?
- Will the person have a down payment, and if so, what percentage of the loan does it amount to?

Program Specification

Write a program to determine whether to loan money based on the following rules.

First, ask for a rating from 1–10 on the following:

- How large is the loan?
- How good is your credit history?
- How high is your income?
- How large is your down payment?

Then, you will create a boolean variable for whether you should loan the money that will be set to either True or False. Set up a series of **`if`** statements to decide if your decision to loan is "yes" or "no" according to the following rules:

- First, check if the loan size is at least 5. If it is, use the following rules:
    - If the credit history and income are both at least 7, then the decision is "yes"
    - If either the credit history or income is at least 7, then check if the down payment is at least 5, if it is, the decision is "yes", if not, the decision is "no"
    - Otherwise (if neither the credit history nor income is at least 7), the decision is "no"
- Otherwise (if the loan is not 5 or greater), use these rules:

    - If the credit is less than 4, then the decision is "no"
    - Otherwise, check the following:
        - If either the income or the down payment is at least 7, the decision is "yes"
        - If both the income and the down payment are at least 4, then the answer is "yes"
        - Otherwise, the decision is "no"

Finally, display the decision to the user.

Sample Solution

When your program is finished, please view a sample solution of this program to compare your approach to that one.

You should work to complete this checkpoint program first, without looking at the sample solution. However, if you have worked on it for at least an hour and are still having problems, you may feel free to use the sample solution to help you finish your program.

- Sample solution

Testing Procedure

Verify that your program works correctly by following each step in this testing procedure:

Test your program with these values and ensure you arrive at the decision indicated:

1. Loan size: 8, Credit: 7, Income: 8, Down Payment: 1, Decision: "yes"
2. Loan size: 5, Credit: 2, Income: 7, Down Payment: 5, Decision: "yes"
3. Loan size: 8, Credit: 2, Income: 8, Down Payment: 3, Decision: "no"
4. Loan size: 2, Credit: 4, Income: 1, Down Payment: 7, Decision: "yes"
5. Loan size: 4, Credit: 5, Income: 5, Down Payment: 3, Decision: "no"

Submission

You have now completed all of the learning activities for the week!

Make sure to:

- Return to Canvas and submit the associated quiz.

Up Next

- Check Your Understanding

Other Links:

- Return to: Week Overview | Course Home