TASK

Beginner Control Structures: If Else

Visit our website

Introduction

WELCOME TO THE 'BEGINNER CONTROL STRUCTURES: IF ELSE' TASK!

We now come to the part where we start learning about a program's control flow. Control flow is the order in which the program's code executes. A control structure is a block of code that analyses variables and statements and chooses a direction in which to go based on given parameters. In essence, it is a decision-making process in computing that determines how a computer responds when given certain conditions and/or parameters. This involves anticipation of conditions occurring during the execution of the program, and specifying actions taken according to the conditions.

Control structures are very useful as they allow you to specify how the code should execute based on different prior conditions. This is helpful if you want to run a piece a code only if a certain condition is met, or if you want to run a piece of code multiple times if a certain scenario holds.

In real life, we make decisions like these on an everyday basis. For instance, if it is cold outside, you would likely wear a jacket, but if it is not cold you may not find a jacket necessary. This type of 'branching' decision-making can be implemented in Python programming using control structures like 'if' and 'if-else' statements. This is a vital concept in programming. We will essentially be teaching the computer how to make decisions for itself!

Get in touch
Connect for support

Remember that with our courses, you're not alone! You can contact your mentor to get support on any aspect of your course.

The best way to get help is to log in to **www.hyperiondev.com/portal** to start a chat with your mentor. You can also schedule a call or get support via email.

Your mentor is happy to offer you support that is tailored to your individual career or education needs. Do not hesitate to ask a question or for additional support!

IF STATEMENTS

Sometimes, you may need to execute some statements only if a certain condition holds. This is achieved with the help of the **If Statement**.

The *if* statement is able to compare two or more variables or scenarios and perform a certain action based on the outcome of that comparison.

In Python, an *if* statement looks like the following:

```
num = 10

if (num < 12):
    print("the variable num is lower than 12")
```

This *if* statement checks if the variable *num* is less than 12. If yes, then it will print the given sentence, letting us know the same. If num is greater than 12, then the code would not print out that sentence.

As you can see, the *if* statement is pretty limited as is. You can only really make decisions based upon two possible outcomes (either the given condition is true or the given condition is false).

What happens if we have more outcomes to take into account, or we want the decision-making to be more complex? What if one decision will have further ramifications, and we will need to make more decisions to fully solve the problem at hand?

ELSE STATEMENTS

If statements are one of the most fundamental concepts in programming, but on their own, they are quite limited.

Imagine if you were very hungry and you asked your friend to go to the grocery store nearby to buy you some chocolate. When they get to the store, they find no chocolates and come back empty-handed because you did not give them any alternatives ("If you don't find any chocolates, you can get me some chips."). Your friend would have to keep coming back for instructions unless you provide them with clearly specified alternative(s).

The scenario where the given condition is not met can either be tackled with several ‘if’ statements to account for each alternative scenario (very inefficient), or we can add an ‘else’ statement to give us a single alternative that tells the program what to do in case the given condition is not met.

Take a look at the following example.

```python
num = 10

if (num < 12):
    print("the variable num is lower than 12")
```

We are now going to expand on it with an *else* statement.

```python
num = 10

if (num < 12):
    print("the variable num is lower than 12")
else:
    print("the variable num is not less than 12")
```

Now, instead of nothing happening in case the variable `num` is equal to or greater than 12 (that is, in case the condition of the *if* statement is not met), the *else* statement will be executed.

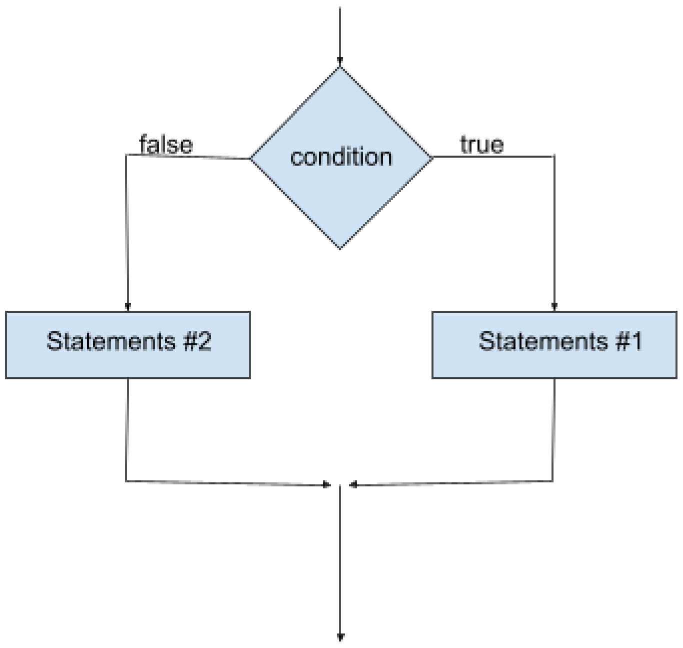
Notice the colon at the end of the expression in the if and else statements, as well as the indentation (4 spaces) before the print function in the next line. **Remember**, **indentation is necessary for Python**, whereas in most other programming languages, indentation is used only to help make the code more readable. The code we wish to execute if the condition is true can be multiple lines long and should be at the same indentation level as the print statements in the example.

Another example of using an *else* statement with an *if* statement can be found below. Do you see that value of the variable `hour` determines what string is assigned to the greeting?

```python
if hour < 12:
     greeting = "Good Morning";
else:
     greeting = "Good Day";
```

THE STRUCTURE OF IF-ELSE STATEMENTS

The basic structure of an *If-Else* statement can be represented by this diagram:



It is mainly used in the case that you want one thing to happen when a condition is True, and something else to happen when it is False.

A note from our coding mentor
Masood

Did you know that many of the people who shaped our digital world started out by coding games for fun? For example, Steve Jobs and Steve Wozniak, the co-founders of Apple, began their coding careers as teenagers when they created the arcade game, Breakout!

Instructions

Before you get started, we strongly suggest you start using Notepad++ or IDLE to open all text files (.txt) and python files (.py). Do not use the normal Windows Notepad or Mac TextEdit as it will be much harder to read.

First, read *example.py* in this task folder by opening it with your IDLE or Notepad++ (right-click the file and select 'Edit with Notepad++').

- The file **example.py** should help you understand some simple Python. Every task will have example code to help you get started. Make sure you read all of example.py and try your best to understand.
- Do run example.py to see the output. Feel free to write and run your own example code before doing the tasks to become more comfortable with Python.
- This may take some time at first as it is different from other (natural) languages, but persevere! Your mentor is here to assist you along the way.

Compulsory Task 1

Follow these steps:

- Create a Python file called **courier.py** in this folder.
- You need to design a program for a courier company to calculate the cost of sending a parcel.
- Ask the user to enter the price of the package they would like to purchase.
- Ask the user to enter the total distance of the delivery in kilometres.
- Now, add on the delivery costs to get the final cost of the product. There are four categories to factor in when determining a parcel's final cost, each with two options based on the customer's delivery preferences. (Use an if-else statement based on the choice they make)
  - Delivery via air ($0.36 per km) or via freight ($0.25 per km)
  - Full insurance ($50.00) or limited insurance ($25.00)
  - Gift option ($15.00) or not ($0.00)
  - Priority delivery ($100.00) or standard delivery ($20.00)

- Write code to work out the total cost of the package based on the options selected in each category.

Completed the task(s)?

Ask your mentor to review your work!

Review work

Things to look out for:

1. Make sure that you have installed and set up all programs correctly. You have setup **Dropbox** correctly if you are reading this, but **Python or Notepad++** may not be installed correctly.
2. If you are not using Windows, please ask your mentor for alternative instructions.

Rate us

Share your thoughts

HyperionDev strives to provide internationally-excellent course content that helps you achieve your learning outcomes.

Think that the content of this task, or this course as a whole, can be improved or think we've done a good job?

Click here to share your thoughts anonymously.