TASK

If Statements and the Boolean Data Type

Visit our website

Introduction

WELCOME TO THE BEGINNER CONTROL STRUCTURES — IF STATEMENT TASK!

In this task, you will learn about a program's flow control. A control structure is a block of code that analyses variables 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 to certain conditions.

Get in touch

Connect for support

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

The best way to get help is to login to Discord at **https://discord.com/invite/hyperdev** where our specialist team is ready to support you.

Our team 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!

A note from the
Hyperion Team

Let's consider how we make decisions in real life. When you are faced with a problem, you have to see what the problem entails. Once you figure out the crux of the problem, you then follow through with a way to solve this problem. Teaching a computer how to solve problems works in a similar way. We tell the computer to look out for a certain problem and tell it how to solve the problem when faced with it.

What is Computational Thinking?
Computational Thinking is the process of solving a problem using a certain manner of thinking. Computational Thinking is used in the development of computer applications, but it can also be used to solve problems across a wide variety of categories, including math, science, geography and art. After learning about Computational Thinking, you will be able to apply this to work as well as real-life problems.

Simple Daily Examples of Computational Thinking

- Looking up a name in your contact list
- Cooking a gourmet meal

WHAT ARE BOOLEANS?

Booleans were developed by an English mathematician and computer pioneer named George Boole (1815-1864). A boolean data type can only store one of two values, namely *True* or *False*. One byte is reserved for the boolean data type.

Use booleans when checking if one of two outcomes is *True*. For example: is the car insured? Is the password correct? Is the person lying? Do you love me?

Once the information is stored in a variable, it is easy to use *loops* and *if statements* to check an extensive sample of items and base your calculations on the result of a boolean value.

ASSIGNING BOOLEAN VARIABLES

Assigning a boolean variable is very simple. You declare the variable name and then choose its starting value. This value can then be changed as the program runs.

For example:

```
pass_word = False
pass_word = True
```

COMPARISON OPERATORS

Boolean values on their own are valuable, but useless without comparison operators. A comparison operator works on two values and produces a boolean value. These operators work well with *if statements* and *loops* (which we will get to in a later task) to control what goes on in our programs.

The four basic comparative operators are:

- greater than >
- less than <
- equal to ==
- not !

Take note that the symbol we use for equal to is '==', and not '='. This is because '==' literally means 'equals' (e.g. *i == 4* means *i* is equal to 4). We use '==' in conditional statements. On the other hand, '=' is used to assign a value to a variable (e.g. *i = "blue"* means I assign the value of *"blue"* to *i*). It is a subtle but important difference. You're welcome to check back with Task 4 if you feel like you need a refresher on variables.

We can combine the greater than, less than and not operator with the equals operator and form three new operations.

- greater than or equal to >=
- less than or equal to <=
- not equal to !=

IF STATEMENTS

We are now going to learn about a vital concept when it comes to programming. We will be teaching the computer how to make decisions for itself using an *if statement*. As the name suggests, it is essentially a question. It can compare two or

more variables or scenarios and perform a certain action based on the outcome of that comparison.

If statements contain a condition. The condition is the part of the *if statement* in brackets in the example below. Conditions are statements that can only evaluate to True or False. If the condition is True, then the indented statements are executed. If the condition is False, then the indented statements are skipped.

In Python, *if statements* have the following general syntax:

```
if <boolean_value> :
        <indented statements>
```

Here's an example of a Python *if statement*:

```
num = 10
is_less_than_12 = num < 12
if is_less_than_12:
      print("the variable num is lower than 12")
```

The code above checks that the *num* variable is less than 12. It uses the < (less than) operator, which takes two values (*num* and 12) and returns True if *num* is less than 12. This is stored in a variable called *is_less_than_12*. The if-statement will then read the value of this variable and, if true, print the string.

There is a more readable and efficient way of expressing this:

```
num = 10

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

This evaluates the boolean value directly, taking away the need to store it in a variable. It is also a lot more readable and easy to understand.

Notice the following important syntax rules for an *if statement*:

- In Python, the colon is followed by code that can only run if the statement's condition is True.
- The indentation is there to demonstrate that the program's logic will follow the path indicated by it. Every indented line will run if the program's *if statement's* condition is True.
- In Python, when writing a conditional statement such as the *if statement* above, you have the option of putting the condition in brackets (i.e. *if (num < 12):*). While your code will still run without them, when your code gets more

complicated it could help with readability, which is a very important requirement in coding.

As you can see, the *if statement* is pretty limited as is. You can only really make decisions that result in one of two possible outcomes. What happens if we have more options? What if one decision will have further ramifications and we will need to make more decisions to fully solve the problem at hand?

Control structures are not limited to *if statements*. You will soon learn how to expand on an *if statement* by adding an else statement or an *elif statement* (else if).

BOOLEANS IN CONTROL STATEMENTS

Control statements allow you to use booleans to their full potential. As of now we only know how to declare a boolean variable as either *True* or *False*, but how would this benefit us? How would we use it? This is where the *if statement* comes into play. Let's look at a simple decision we might make in our everyday lives.

When you are about to leave your house, do you always take an umbrella? No, you would only take an umbrella if it is raining outside. This is a very rudimentary example of decision making where there are only two outcomes. We can apply these basic principles to create more complex programs.

```
umbrella = "Leave me at home"
is_raining = False
if is_raining:
   umbrella = "Bring me with"
```

Take a look at line 3 in the example above. This statement reads the value of the boolean and finds it to be true (similar to writing `if is_raining == True:`). Because the default of a boolean is True, we only use '==' with boolean conditionals if a condition specifically needs to be False.

A note from our coding mentor
Sarah

Sorry to interrupt but I found this fantastic blog written by MARC CHERNOFF that shows 10 if Statements in "Real Life" that are worth learning. It just shows how important making decisions are to our lives. It also shows you that if you don't follow the desired path based on your decision, the outcome could be detrimental to you.

1. *If you don't understand the product or service, don't buy it until you do.*
2. *If you do not take ownership of your actions, your actions will eventually own you.*
3. *If you are not saving at least 10% of your salary, you are not saving enough.*
4. *If you talk too much, people will stop listening. If you don't talk enough, people will never hear your point of view.*
5. *If you are lazy, you will fail. Laziness will always overshadow your true potential.*
6. *If you hate your job, you also hate half of the time you spend on this planet.*
7. *If you are not investing (120 minus your age) percent of your savings in the stock market, you are giving up thousands of dollars over the course of your lifetime.*
8. *If you don't finish what you start, your success rate will always be zero.*
9. *If you don't consume enough liquids, you will never be healthy.*
10. *If your monthly debt payments exceed 40% of your total income, you will go broke if you don't fix your spending habits promptly.*

Now let's look at how we can put *if statements* into practice using the boolean data type.

Instructions

Before you get started, we strongly suggest you start using an editor such as VS Code or Anaconda to open all text files (.txt) and python files (.py).

First, read **example.py**.

- **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.

- You may run **example.py** to see the output. Feel free to write and run your own example code before doing the Task to become more comfortable with Python.

Compulsory Task 1

Follow these steps:

- Create a Python file called **baby.py** in this folder.
- This program will be used to test if the user is 18 or older and allowed entry into a party.
- Ask the user to enter the year they were born and calculate if they are 18 or older.
- If they are 18 or older, then display a message saying “Congrats you are old enough.”

Compulsory Task 2

Follow these steps:

- Create a Python file called **full_name.py** in this folder.
- This program will be used to validate that a user inputs at least two names when asked to enter their full name.
- Ask the user to input their full name.

- Do some validation to check that the user has entered a full name. Give an appropriate error message if they haven't. One of the following messages should be displayed based on the user's input:
    - "You haven't entered anything. Please enter your full name."
    - "You have entered less than 4 characters. Please make sure that you have entered your name and surname."
    - "You have entered more than 25 characters. Please make sure that you have only entered your full name."
    - "Thank you for entering your name."

If you are having any difficulties, please feel free to contact our specialist team **on Discord** for support.

Optional Bonus Task 1

Follow these steps:

- Create a Python file called **optional_task1.py** in this folder.
- This program should test whether a number is odd or even.
- Ask the user to enter an integer.
- Test whether the number entered is odd or even. *Hint: use the modulus operator.*
- Print out the number and whether it is odd or even.

Optional Bonus Task 2

Follow these steps:

- Create a Python file called **optional_task2.py** in this folder.
- This program will help the user decide what to wear. To determine what to wear, you need to determine whether the temperature is greater than 20 degrees, whether it is the weekend, and whether it is sunny. Declare boolean variables for each one of these characteristics.
- Then, ask the user a series of "yes or no" questions for each variable, and change the boolean variable to *True* or *False* based on their answer.
- If the temperature is greater than 20 degrees, then the user should wear a short-sleeved shirt.
- If the temperature is less than 20 degrees, then the user should wear a long-sleeved shirt.
- If it is the weekend, the user should wear shorts.
- If it is a weekday, the user should wear jeans.
- If it is sunny, the user should wear a cap.
- If it is not sunny, the user should wear a raincoat.
- Display a sentence that fully describes what the user's outfit should be.

Thing(s) to look out for:

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

Rate us

Share your thoughts

Hyperion 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.