TASK

Beginner Control Structures — if Statements

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 your mentor to get support on any aspect of your course.

The best way to get help is to login 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!

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

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 condition :
    indented statements

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

```
num = 10

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

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

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.

Comparison Operators

You may have also noticed the less than (<) symbol in line 1. As a programmer, it's important not to forget the basic logical commands. We use comparison operators to compare values or variables in programming. These operators work well with *if statements* and *loops* (which we will get to in Task 13) 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 the '==' 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 !=

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).

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.*

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 as it will be much harder to read.

First, read **example.py**, open it using Notepad++ (Right-click the file and select 'Edit with IDLE).

- **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

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 enters 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."

Optional Bonus Task

Follow these steps:

- Create a Python file called **optional_task.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.

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 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

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

Think 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.