Hyperiondev

TASK

Beginner Control Structures: elif Statements

Visit our website

Introduction

WELCOME TO THE BEGINNER CONTROL STRUCTURES: ELIF STATEMENTS TASK!

In this task, you will learn further about a program's flow control. Earlier we got introduced to *if* statement and *else* statements. For a quick revision, *if* statements are used to run code if a certain condition holds. An *else* statement follows an *if* statement and contains code that is executed when the *if* statement is false.

We can further chain else and if statements together to run through a series of possibilities (not just a couple of outcomes).

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!

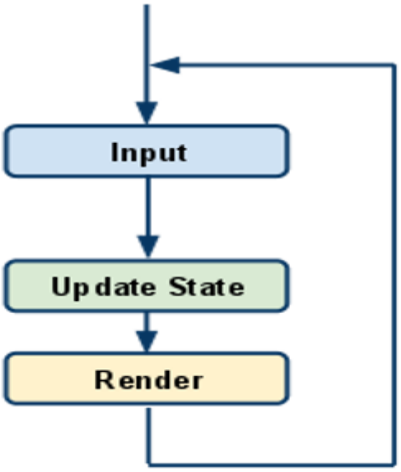
A note from the

HyperionDev Team

Programming games is a great way to learn how to code. Look at any game and think to yourself: How can I do that? What functions will I need? What data types and structures would I use?

For example, using control structures, you could perhaps make a turn-based strategy game. What about games like Sudoku or Hangman even?

The workings of a game can be represented as follows:



The above diagram represents a 'game loop'.

This loop is executed many times a second. Inside the loop, the game logic is executed and the screen is updated and rendered accordingly.

Even if you do not have all the programming tools at your disposal yet, it is often useful to train your mind to think in this way.

Using the pygame module is one way to make games in Python. Visit the ***pygame website*** *to find out more information about pygame.*

ELIF STATEMENTS

The last piece of the puzzle is the *elif* statement. This elif stands for 'Else If'.

What this does is give us more options in our scenarios so that it is not limited to two outcomes (a single condition being true and false). We can add more questions to the *if* statement so that we can test multiple parameters in the same statement.

Look at the following example:

```
num = 10

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

What happens if we want to test multiple conditions? Well, this is where the elif statement comes in. Consider the example below and contrast it with the example above:

```
num = 10

if (num >12):
      print("the variable num is greater than 12")
elif (num > 10):
      print("the variable num is greater than 10")
elif (num < 5):
      print("the variable num is less than 5")
else:
      print("the variable num is 10")
```

Remember that you can also combine if, else and elif into one big statement. This is what we refer to as a **Conditional Statement**.

Some points to note on the syntax of if-elif-else Statements:

- Make sure that the if/elif/else statements end with a colon (the : symbol);
- Ensure that your indentation is done correctly (i.e. statements that are part of a certain control structure's 'code block' need the same indentation);
- To have an elif you must have an if above it;
- To have an else you must have an if or elif above it;

- You can't have an else without an if (think about it!);
- You can have as many ‘elif’ under an ‘if’, but only one ‘else’ right at the bottom. The last ‘else’ is like the fail-safe statement that executes if the other if/elif statements fail.

Now that you know more about Python, it’s time to write a small useful program. You may wonder how you could possibly write a program with the few concepts that we have covered so far, but let’s not get ahead of ourselves. Baby steps! With that in mind, let’s see what we CAN do.

A note from our coding mentor
Jared

Have you heard about Margaret Hamilton? She is the engineer who took us to the moon. She wrote the code for the Apollo 11’s on-board flight software, and as a result of her work, she received NASA’s Exceptional Space Act Award. If that’s not enough, she is also credited with coining the term “software engineering”.

In the picture below, you will see a young Margaret standing next to the actual code she wrote herself to take humanity to the moon!

Margaret Hamilton

Instructions

Feel free at any point to refer back to previous material if you get stuck. Remember that if you require more assistance, our mentors are always willing to help you!

- As always, open the **example.py** file using Notepad++ (Right-click the file and select 'Edit with Notepad++') or IDLE to read them.

- example.py has recap content for you to aid with the Compulsory Tasks and remind you of key concepts.

- As a bonus treat (!) open the **gameExample.py** file in the game folder.
    - **gameExample.py** contains a very simple game coded with pygame. The player can move up and down with the arrow keys and must avoid the programming challenges to win.
    - Do not worry if you do not understand some of the concepts yet as they will be covered in upcoming tasks.
    - Most of the concepts should, however, be familiar from the previous tasks.
    - This should show you how powerful the tools are that you have learnt thus far.
    - You will need to download pygame. Instructions on how to do so are inside the gameExample.py file.
    - You will get to add more functionality and complexity to the game in later tasks as you learn more programming concepts. For example, more enemies, lives and cool effects**.**

Before you proceed, a little information about one of the tasks you are going to tackle.

Notes on Compulsory Task 2

We are going to create an investment calculator to see what returns you will get if you were to put money into an investment at a certain interest rate for a certain amount of time. If you are not very familiar with the concept of investments, rate of interest, and the like, don't worry. Below is a short refresher:

A rate of interest is a factor in most financial dealings, whether it be a loan (which ends up with you paying more than the principal amount to the bank), or an investment (which ends up with you hopefully earning more than the money you put in).

There are two main streams of interest: compound and simple interest.

Simple interest is continually calculated on the initial amount invested and is only calculated once per year. This interest amount is then added to the amount that you initially invested (known as the principal amount).

Example: If you invest $1000 at 10%, in the first year you will earn $100 interest ($1000 * 0.10) giving you $1100. In the next year, the interest is still calculated on the principal amount ($1000) giving you another $100, making your earnings a grand total of $1200.

Compound interest is different in that the interest is calculated on the current accumulated amount (initial principal plus accumulated interest thereon).

To use the above example, imagine you invest $1000 at 10% compounded once a year. The first year you will earn $100, giving you an accumulated amount of $1100. The second year, imagine if you reinvested your gains as well, your accumulated amount will be $1100 and you will now earn interest on the accumulated amount (1100 * 0.10) to earn $110 interest, giving you $1210 in total.

If this doesn't make too much sense, don't worry as the above is just a brief background to what you will be doing in the compulsory tasks. If you'd like to find out more, feel free to read up more online. The needed formulae are given along with the task details.

Compulsory Task 1

Follow these steps:

- Create a Python file called **control.py** in this folder.
- This is going to expand on the first control structure task we created.
- Write code to take in a user's age using `input()` and store their age in an integer variable called age.
- Then check if the user's age is over 18. If the user is over 18, print out the message "You are old enough!" else if they are over 16 print "Almost there", otherwise print "You're just too young!" You should use one if, elif and else statement to do this.

Compulsory Task 2

Follow these steps:

- Create a new Python file in this folder called **investment_calculator.py**.
- At the top of the file include the line:
- import math
- Ask the user to input:
    - The amount that they are depositing, stored as 'P'.
    - The interest rate (as a percentage), stored as 'i'.
    - The number of years of the investment. stored as 't'.
    - Then ask the user to input whether they want "simple" or "compound" interest, and store this in a variable called 'interest'.
- Only the number of the interest rate should be entered - don't worry about having to deal with the added '%', e.g. The user should enter 8 and not 8%
- Depending on whether they typed "simple" or "compound", output the appropriate amount that they will get after the given period at the interest rate. Look below in "additional information" for the formulae to be used.
- Print out the answer!
- Try enter 20 years and 8 (%) and see what a difference there is depending on the type of interest rate!

Additional Information:

- Don't forget to cast the input to ints for the calculations!
- 'r' below is the interest entered above divided by 100, e.g. if 8% is entered, then r is 0.08.
- Simple interest rate is calculated as follows:
    - A = P(1 + r * t)
    - The Python equivalent is very similar:
        - A =P*(1+r*t)
- Compound interest rate is calculated as follows:
    - A = P(1 + r) ^ t
    - The Python equivalent is slightly different:
        - A = P* math.pow((1+r),t)

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 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

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.