Hyperion Development

Task: Logical Programming - Operators

www.hyperiondev.com

Introduction

Welcome to The Operators Task!

In a programming language, an operator is a symbol that tells the compiler or interpreter to perform specific operations, whether it be mathematical, relational or logical, and produce a final result. This task will introduce you to the different types of operators and show you how to use them.

Connect with your mentor

CONNECT

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

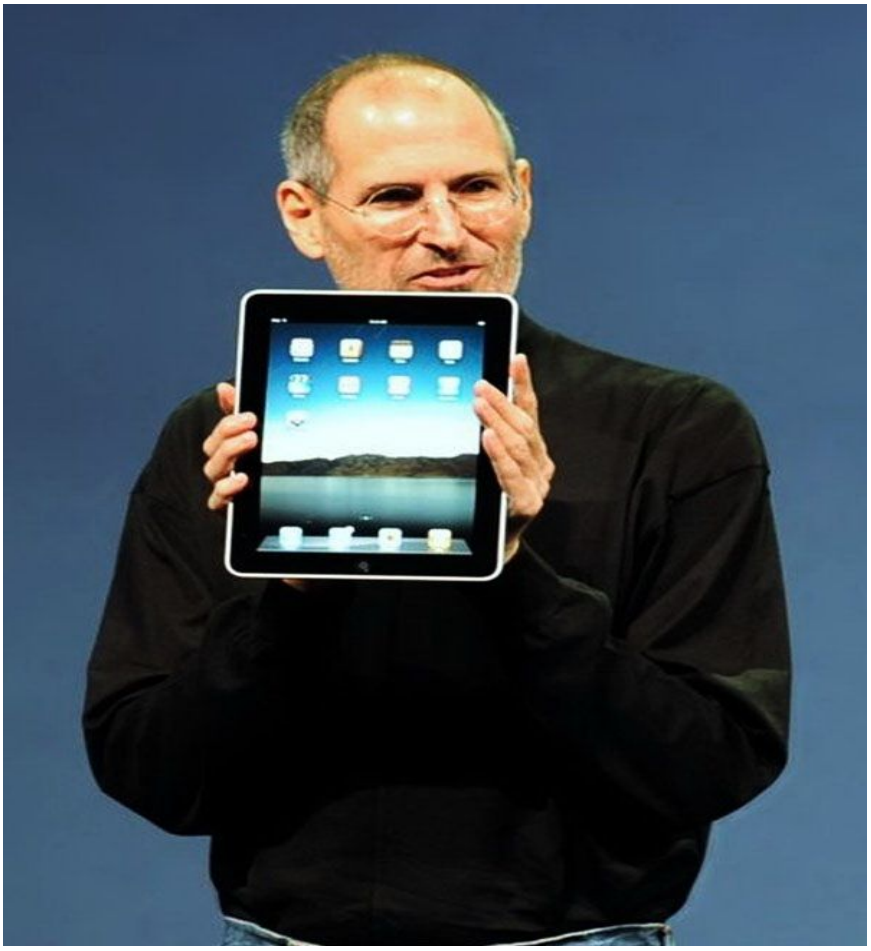
The best way to get help is to login to www.hyperiondev.com/support to start a chat with your mentor. You can also schedule a call or get support via email.

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...

The iPad combines many of the popular capabilities of the iPhone, such as built-in high-definition camera, access to the iTunes Store, and audio-video capabilities, but with a nine-inch screen and without the phone.

Apps, games, and accessories helped spur the popularity of the iPad and led to its adoption in thousands of different applications from movie making, creating art, making music, inventory control and point-of-sale systems, to name but a few.



-The Hyperion Team

What are Operators?

Operators are symbols that tell the computer which mathematical calculations to perform or which comparisons to make.

Comparison Operators

As a programmer, it's important to not 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 to control what goes on in our programs.

The four basic comparative operators are:

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

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

- greater than or equals to `>=`
- less than or equals to `<=`
- not equals to `!=`

Comparing Strings

```
myName = "Tom"

if myName == "Tom":
        print "I was looking for you"
```

Comparing Numbers

```
num1 = 10
num2 = 20
```

```
if num1 >= num2:                        # The symbol for ‘greater than or equal to’ is >=
       print "It's not possible that 10 is bigger than or equal to 20."

elif num1 <= num2:                      # The symbol for 'less than or equal to' is <=
       print "10 is less than or equal to 20."

elif num1 != num2:                      # The symbol for 'not equal to' is !=

       print "This is also true since 10 isn't equal to 20, but the elif statement
before comes first and is true so Python will execute that!"

elif num1==num2 :                       # The symbol for 'equal to' is ==

       print "Will never execute this print statement..."
```

The program will check the first part of the if statement (is num 1 bigger than or equal to num 2?).
If it is not, then it goes into the first 'elif' statement and checks if num1 is less than or equal to num2.
If it is not then it goes into the next 'elif' statement...etc.

Logical Operators

What if the program needs to test two or more conditions? We can have multiple conditions, it is quicker than writing a nested if statements.

Let's take this real life situation. When buying items at a store two criteria need to be met. The item needs to be in stock and you need enough money to pay for the item. This is an example of a conjunction operation where both conditions need to be true for the whole statement to be true. This is called a 'and' operation.

If you were to receive a good mark at school it’s because you either very bright or studied hard. In this instance either one of the options can be true or even both can be true but at least one needs to be true. This is a disjunction operation where at least one of the conditions needs to be met for the whole statement to be true. This is also called the 'or' operation.

Example of an AND condition:

```
team1Score = 3
team2Score = 2
game = "Over"

if (team1Score > team2Score) and (game == "Over"):
    print "congratulations you have won the match!"
```

Example of an OR condition:

```
speed = int(raw_input("How many killometers per hour are you travelling at?"))
belt = raw_input("Are you wearing a safety belt?")

if (speed > 80) or (belt != "Yes"):
    print "Sorry Sir but I have to give you a traffic fine"
```

Arithmetic Operators

The arithmetic operators in Python are as follows:

- + (Addition) - Adds values on either side of the operator
- - (Subtraction) - Subtracts the value on the right of the operator from the value on the left
- * (Multiplication) - Multiplies values on either side of the operator
- / (Division) - Divides the value on the left of the operator by the value on the right
- % (Modulus) - Divides the value on the left of the operator by the value on the right and returns the remainder
- ** (Exponent) - Performs exponential calculation

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 Notepad++') or 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.

- You are not required to read the entirety of Additional Reading.pdf, it is purely for extra reference.

Compulsory Task

Follow these steps:

- Create a new Python file in this folder called "Colours.py."
- Design a program that determines the award a person competing in a triathlon will receive.
- Your program should read in the times in minutes for all three events of a triathlon, namely swimming, cycling and running and then calculate and display the total time taken to complete the triathlon.
- Your program should also read in the position the participant places in the triathlon. (e.g. 1, 2, 3, 4, ...)
- The award a participant receives is based on the total time taken to complete the triathlon. Display the award the participant will receive based on the following criteria:
    - If the time taken for a participant to complete the triathlon is less than or equal to 100 minutes and they placed first (i.e. 1) they receive **Provincial Colours.**
    - If the time taken for a participant to complete the triathlon is less than or equal to 110 minutes and they placed first or second they receive **Provincial Half Colours.**
    - If the time taken for a participant to complete the triathlon is is less than or equal to 115 minutes they receive a **Provincial Scroll.**
    - If the time taken for a participant to complete the triathlon is less than or equal to 120 minutes they receive a **Provincial Certificate**
    - If the time taken for a participant to complete the triathlon is greater than 120 minutes they do not receive an award.

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.

Give 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.