Hyperiondev

TASK

Logical Programming — Operations

Visit our website

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.

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.

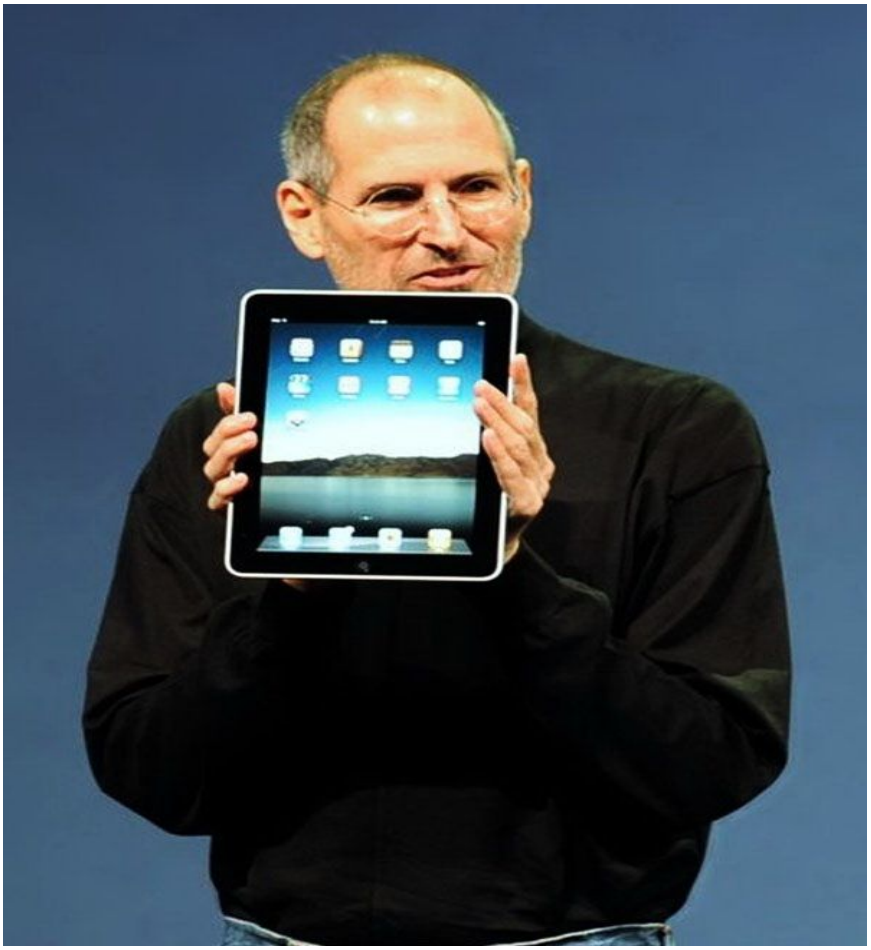
The best way to get help is to login to **www.hyperiondev.com/portal** to start a chat with your mentor. You can also schedule a call or get support via email.

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

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.



What are Operators?

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

Comparison Operators

Here is a quick reminder of our comparison operators from Task 7:

The four basic comparative operators are:

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

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 !=

Comparing Strings:

```
my_name = "Tom"

if my_name == "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 *num1* bigger than or equal to *num2*?).

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

A **logical operator** is another type of operator that is used to control the flow of a program. Logical operators are usually found as part of a control statement such as an *if statement*. Logical operators basically allow a program to make a decision based on multiple conditions. For example, if you would like to test more than one condition in an *if statement*.

The three logical operations are:

Operator	Explanation
and	both conditions need to be true for the whole statement to be true (also called the conjunction operation)
or	at least one condition needs to be true for the whole statement to be true (also called the disjunction operation)
not	the statement is true if the condition is false (only requires one condition)

The 'and' and 'or' operators require two operands, while the 'not' operator requires one.

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 to have 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 the **and** operation.

One could receive a good mark at school either because they are very bright or because they studied hard. In this instance, either one of the options can be true, or 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:

```
team1_score = 3
team2_score = 2
game = "Over"

if (team1_score > team2_score) and (game == "Over"):
   print("Congratulations you have won the match!")
```

Example of an OR condition:

```
speed = int(input("How many kilometres  per hour are you travelling at?"))
belt = 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:

Arithmetic Operations	Symbol used in Python
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. E.g. *4 % 2 == 0* but *5 % 2 == 1*.	%
Exponent: Performs exponential calculation, i.e. calculates the answer of the value on the left to the power of the value on the right.	**
Plus-equals: Adds 1 to the variable for each iteration.	+=

E.g. *n += 1* is shorthand for *n = n + 1*. (This is particularly useful when using loops, as you will see in Task 13.)	
Minus-equals: Minuses 1 from the variable for each iteration. Eg. *n -= 1* is shorthand for *n = n - 1*.	-=

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 IDLE (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 1

Follow these steps:

- Create a new Python file in this folder called **task1.py**.
- Declare three variables called *num1*, *num2* and *num3*. Assign each variable a number value.
- Now, compare *num1* and *num2* and display the larger value.
- Next, determine whether *num1* is odd or even and display the result.
- Next, write a conditional statement to sort the three numbers from largest to smallest.

Compulsory Task 2

Follow these steps:

- Create a new Python file in this folder called **task2.py**.
- This program will be used to calculate the area that the foundation of a building covers. This program should:
    - Ask the user to enter the shape of the building (square, rectangular or round).
    - Based on the user's input, prompt for the appropriate dimensions. See what dimensions you need in the list of formulae below.
    - Calculate and display the area that will be taken up by the foundation of the building.
- Formulae for area:
    - Area of square = length of the square to the power of two (length^2).
    - Area of rectangle = length x width.
    - Area of circle = pi X radius squared (πradius^2).

Compulsory Task 3

Follow these steps:

- Create a new Python file in this folder called **task3.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.
- The award a participant receives is based on the total time taken to complete the triathlon. The qualifying time for awards is 100 minutes. Display the award the participant will receive based on the following criteria:

Total time	Award
Within qualifying time.	Provincial Colours

Within 5 minutes of qualifying time.	Provincial Half Colours
Within 10 minutes of qualifying time.	Provincial Scroll
More than 10 minutes of qualifying time.	No award

Compulsory Task 4

Follow these steps:

- Create a new Python file in this folder called **task4.py**.
- Create a program that asks the user to enter an integer and determine if it is:
    - divisible by 2 and 5,
    - divisible by 2 or 5,
    - not divisible by 2 or 5
- Display your result.

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.