



TASK

Thinking Like a Programmer — Pseudo Code I

Visit our website

Introduction

WELCOME TO THE PSEUDO CODE TASK!

The bootcamp you are starting today is structured as a series of 'Tasks'. Tasks include a lesson component designed to teach you the theory needed to develop your skills, as well as a Compulsory Task component designed to give you the platform to apply your newly-gained knowledge by completing practical exercises.

This particular task will introduce the concept of pseudo code. Although pseudo code is not a formal programming language, it will not only ease your transition into the world of programming but offer a useful tool that even very experienced programmers use to tackle complex problems. Pseudo code makes creating programs easier because, as you may have guessed, they can sometimes be complex and long — so preparation is key. It is difficult to find errors without understanding the complete flow of a program. This is the reason why you will begin your venture into programming with pseudo code.



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 www.hyperiondev.com/portal to start a chat with a reviewer. You can also schedule a call or get support via email.

The reviewers are 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

Chances are, if you're taking this course, you're brand new to the Python programming language. You may even be new to programming altogether. Either way, you've come to the right place!

The key to becoming a competent programmer is utilising all available resources to their full potential. Occasionally, you may stumble upon a piece of code that you cannot wrap your head around. So, knowing which platforms to go to for help is important!

HyperionDev knows what you need, but there are also other avenues you can explore for help. One of the most frequently used by programmers around the world is [Stack Overflow](#); it would be a good idea to bookmark this for future use!.

Also, [our blog](#) is a great place to find detailed articles and tutorials on concepts into which you may want to dig deeper. For instance, when you get more comfortable with programming, if you want to [read up about machine learning](#), or learn [how to deploy a machine learning model with Flask](#), you could find all this information and more on our blog.

The blog is updated frequently, so be sure to check back from time to time for new articles or subscribe to updates through our [Facebook page](#).

INTRODUCTION TO PSEUDO CODE

What is pseudo code? And why do you need to know this? Well, being a programmer means that you will often have to visualise a problem and know how to implement the steps to solve that particular conundrum. This is where pseudo code comes in.

Pseudo code is a detailed yet informal description of what a computer program or algorithm must do. It is intended for human reading rather than machine reading. Therefore, it is easier for people to understand than conventional programming language code. Pseudo code does not need to obey any specific syntax rules, unlike conventional programming languages. Hence it can be understood by any programmer, irrespective of the programming languages they're familiar with. **As a programmer, pseudo code will help you better understand how to**

implement an algorithm, especially if it is unfamiliar to you. You can then translate your pseudo code into your chosen programming language.

Pseudo code is often used in computer science textbooks and scientific publications in the description of algorithms so that all programmers can understand them, irrespective of their knowledge of specific programming languages. It is also used by programmers in the planning of computer program development to initially sketch out their code before writing it in its actual language.

For this course, pseudo code can also help guide the comments that you make in your program to explain your code from Task 3 onwards, but don't worry too much about that for now.

WHAT IS AN ALGORITHM?

Now, what exactly is an algorithm? Simply put, an algorithm is a step by step method of solving a problem. To understand this better, it might help to consider an example that is not algorithmic. When you learned to multiply single-digit numbers, you probably memorised the multiplication table for each number (say n) all the way up to $n \times 10$. In effect, you memorised 100 specific solutions. That kind of knowledge is not algorithmic. But along the way, you probably recognised a pattern and made up a few tricks. For example, to find the product of n and 9 (when n is less than 10), you can write $n - 1$ as the first digit and $10 - n$ as the second digit (e.g. $7 \times 9 = 63$). This trick is a general solution for multiplying any single-digit number by 9. That's an algorithm! Similarly, the techniques you learned for addition with carrying, subtraction with borrowing, and long division are all algorithms.

One of the characteristics of algorithms is that they do not require any intelligence to execute. Once you have an algorithm, it's a mechanical process in which each step follows from the last according to a simple set of rules (like a recipe). However, breaking down a hard problem into precise, logical algorithmic processes to reach the desired solution is what requires intelligence or computational thinking.



A note from the **Hyperion Team**

Did you know that [Ada Lovelace](#) (who was an English mathematician and writer born in 1815) is regarded as the first computer programmer? Her notes on an early mechanical, general-purpose computer (the Analytical Engine) is recognised as the first algorithm (pseudocode) to be carried out by a machine.



Ada Lovelace

Also, did you know the word 'algorithm' comes from the mediaeval Latin word 'algorism' and the Greek word 'arithmos' (ἀριθμός)? The word 'algorism' (and therefore, the derived word 'algorithm') comes from Al-Khwārizmī (Persian: خوارزمی, c.780–850), a Persian mathematician, astronomer, geographer and scholar. The English language adopted the French term, but it wasn't until the late 19th century that 'algorithm' took on the meaning that it has in modern English.

In English, it was first used in about 1230 and then by Chaucer in 1391. Another early use of the word is from 1240 in a manual titled *Carmen de Algorismo* composed by Alexandre de Villedieu.

It begins thus:

*Haec algorismus ars praesens dicitur, in qua / Talibus Indorum fruimur bis
quinque figuris.*

which translates as:

Algorism is the art by which at present we use those Indian figures, which number two times five.

The poem summarises the art of calculating with the new style of Indian dice, or Talibus Indorum, or Hindu numerals or al-adad al-Hindi.

PSEUDO CODE SYNTAX

There is no specific convention for writing pseudo code, as a program in pseudo code cannot be executed. In other words, pseudo code itself is not a programming language. It is a model of all programming languages which one uses to make eventual coding a little simpler.

Pseudo code is easy to write and understand, even if you have no programming experience. You simply need to write down a logical breakdown of what you want your program to do. Therefore, it is a good tool to use to discuss, analyse and agree on a program's design with a team of programmers, users and clients before coding the solution.

Example 1

Problem: Write an algorithm that prints out "passed" or "failed" depending on whether a student's grade is greater than or equal to 50 could be written in pseudocode as follows:

Pseudo code solution:

```
get grade
if grade is equal to or greater than 50
    print "passed"
else
    print "failed"
```

Example 2

Problem: Write an algorithm that asks a user to enter their name and prints out "Hello, World" if their name is "John":

Pseudo code solution:

```
request user's name  
  
if the inputted name is equal to "John"  
    print "Hello, World"
```

Example 3

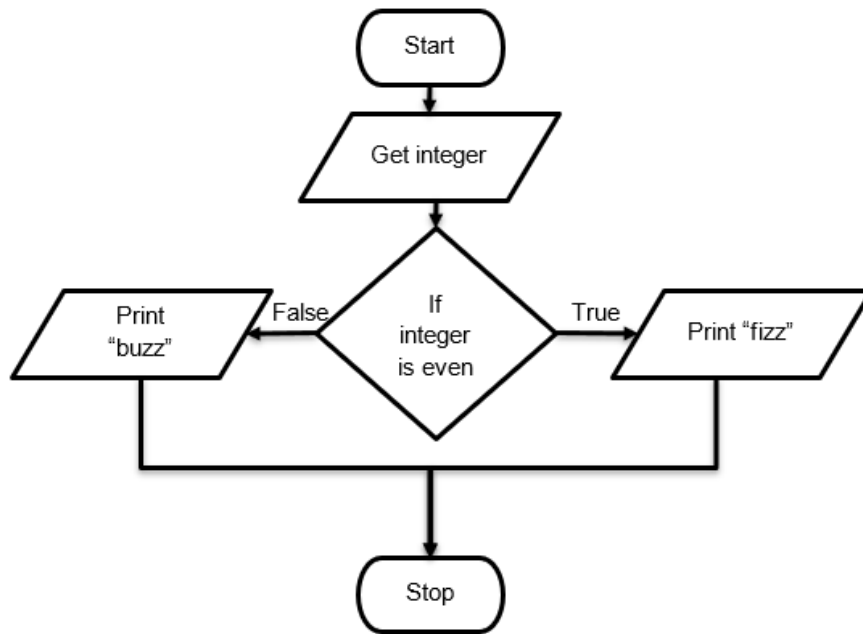
Problem: Write an algorithm that requests an integer from the user and prints "fizz" if the number is even or "buzz" if it is odd:

Pseudo code solution:

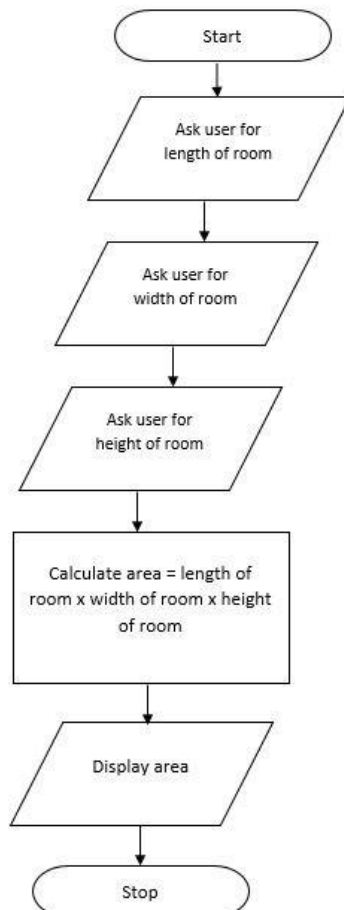
```
request an integer from the user  
  
if the integer is even  
    print "fizz"  
  
else if the integer is odd  
    print "buzz"
```




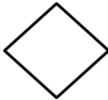
Flowcharts can be thought of as a graphical implementation of pseudo code. This is because it is more visually appealing and probably more readable than a "text-based" version of pseudo code. An example of a flowchart that would visualise the algorithm for "Example 3" above, is shown in the image below:

Another example for an algorithm that calculates the area of a room based on user input could look something like this:



There are a number of shapes or symbols used with flowcharts to denote the type of instruction or function you are using for each step of the algorithm. The most commonly used symbols used for flowcharts are shown in the table below:

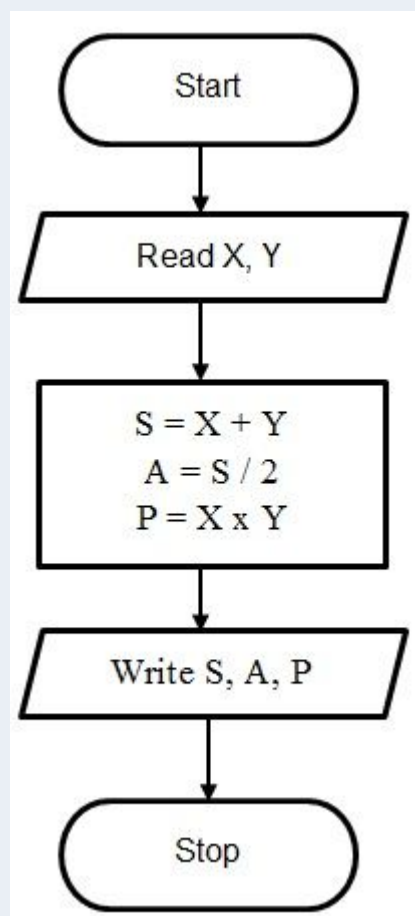


Symbol	Use
	Indicates the start and end of an algorithm.
	Used to get or display any data involved in the algorithm.
	Indicates any processing or calculations that need to be done.
	Indicates any decisions in an algorithm.

Compulsory Task

Follow these steps:

- Use a plain text editor like **Notepad++**. Create a new text file called **pseudo.txt** inside the folder for this task in Dropbox.
- Inside **pseudo.txt**, write pseudo code for each of the following scenarios:
 - An algorithm that asks a user to enter a positive number repeatedly until the user enters a zero value, then determines and outputs the largest of the numbers that were input.
 - An algorithm that reads an arbitrary number of integers and then returns their arithmetic average.
 - An algorithm that reads a grocery list and prints out the products (in alphabetical order) that are still left to buy.
 - An algorithm for the flowchart below:



Optional Bonus Task

Follow these steps:

- Create a new text file called **optional_task.txt** inside this folder.
- Inside **optional_task.txt**, write pseudocode for the following scenario:
 - An algorithm that asks the user to enter four numbers, sorts those numbers from smallest to largest and then prints the sorted numbers.

Thing(s) to look out for:

1. Make sure that you have installed and set up all programs correctly. You have set up **Dropbox** correctly if you are reading this.



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.

