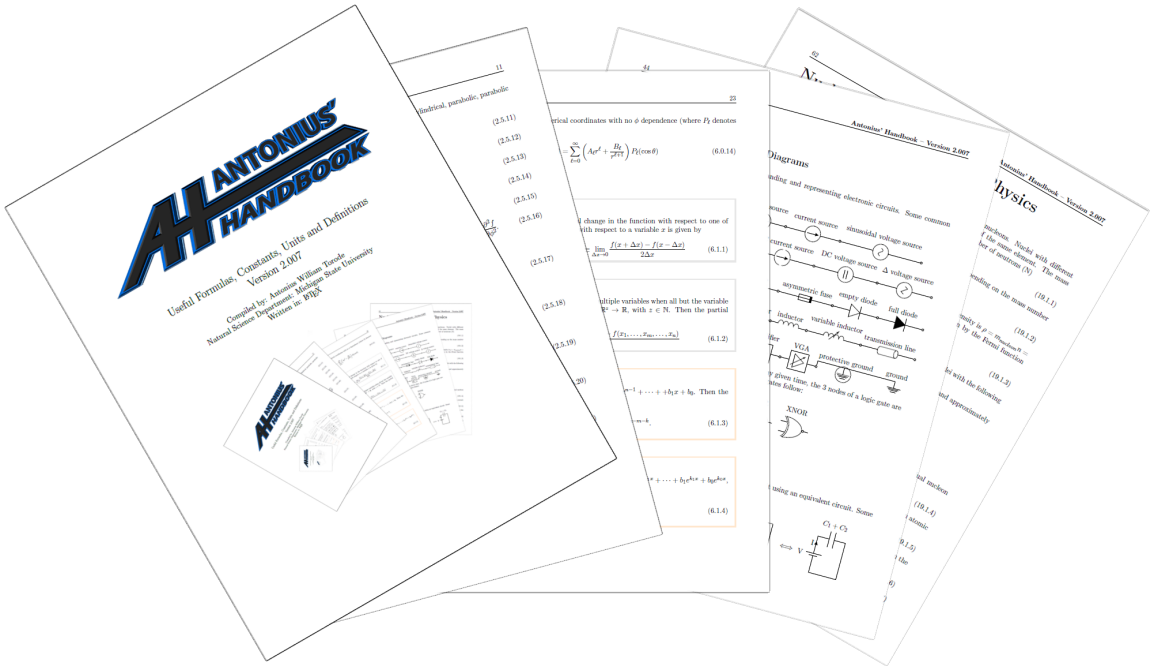
# ANTONIUS' HANDBOOK

Useful Formulas, Constants, Units and Definitions
Volume II - Programmers Paradise
Version 0.000

Compiled by: Antonius William Torode
Natural Science Department: Michigan State University
Written in: LaTeX



The original maintainer of this work is: Antonius Torode.

The current maintainer of this work is: Antonius Torode.

Published by Antonius Torode.

Hosted at: https://msu.edu/~torodean/AHandbook.html

Github Repository: https://github.com/torodean/Antonius-Handbook

| | |
|---|---|
| First Personal Release (Version 0.000): | June 2017 |
| First Public Release (Version 1.000): | N/A |
| Most Current Revision Date (Version 0.000): | June 27, 2017 |

Torode, A.
Antonius' Handbook.
Michigan State University –
Department of Physics & Astronomy.
2016, Student.
Volume II.
Version: 0.000

## Preface

This document is a compilation of useful programming formulations, definitions, constants, and general information used throughout my own schooling and research as a reference while furthering education. It's purpose is to provide a complete 'encyclopedia' per say of various codes, syntax and significant ideas used often. The idea and motivation behind it is to be a quick reference providing easily accessible access to necessary information for either double checking or recalling proper formulations or algorithms for use in various situations due to my own shortcomings in matters of memorization. All the material in this document was either directly copied from one of the references listed at the end or derived from scratch. On occasion typos may exist due to human error but will be corrected when discovered.

The version number is updated every time the document is distributed, printed, or referred to. This ensures that there is no two copies with different information and similar version numbers. The latest update date is automatically set to the current date each time the document is edited. Please refrain from distributing this handbook without permission from the original author/compiler. This book is formatted for printing.

For more information about this book or details about how to obtain your own copy please visit:

https://msu.edu/~torodean/AHandbook.html

## Disclaimer

This book contains codes, formulas, definitions, and theorems that by nature are very precise. Due to this, some of the material in this book was taken directly from other sources. This is only such in cases where a change in wording or codes could cause ambiguities or loss of information quality. Following this, all sources used are listed in the references section.

*This page intentionally left blank.*
*(Yes, this is a contradiction.)*

# Contents

# Introduction

This document is still under the initial formatting stages and useful information will be added soon. When it has sufficient information to be ready for distribution the version will be updated to 1.000.

# C

# C++

## 3.1 Basic Input and Output

To output text via a terminal you can use:

```
std::string text = "Hello World!";
std::cout << text << std::endl; //std::endl is equivalent to the new-line character.
```

To get input as a user in the type of a std::string, you can use:

```
std::string input = "";
std::cout << "Enter some text: ";
std::getline(std::cin, input);
```

## 3.2 Converting Between Types

### std::string to int

To convert a string to an integer you can use:

```
std::string text = "31415";
int number = std::stoi(text);
```

## 3.3 Mathematical Commands

### Prime Number

A simple brute for method to determines if a number of type long is prime or not.

```
bool isPrime(long num) {
    int c = 0;    //c is a counter for how many numbers can divide evenly into num
    if (num == 0 || num == 1 || num == 4) {
        return false;
    }
    for (long i = 1; i <= ((num + 1) / 2); i++) {
        if (c < 2) {
            if (num % i == 0) {
                c++;
            }
        } else {
            return false;
        }
    }
    return true;
}
```

# C#

# CSS

# HTML

# Linux

## 7.1 System Related Commands

How to display the processes that are currently running.

```
ps aux
```

To search the results of a command for a string of characters one can use the grep command. For example:

```
ps aux | grep "firefox"
```

Restore power/battery icon if it disappears.

```
/usr/lib/x86_64-linux-gnu/indicator-power/indicator-power-service &disown
```

Restore volume icon/control button if it disappears.

```
gsettings set com.canonical.indicator.sound visible true
```

Reset wifi services in case the connection gets lost.

```
sudo systemctl restart network-manager.service
```

Turn off LCD display.

```
xset dpms force off
```

# PHP

# Python

Import floating point division which allows python 2 compatibility when using division with doubles. Include this at the beginning of the script.

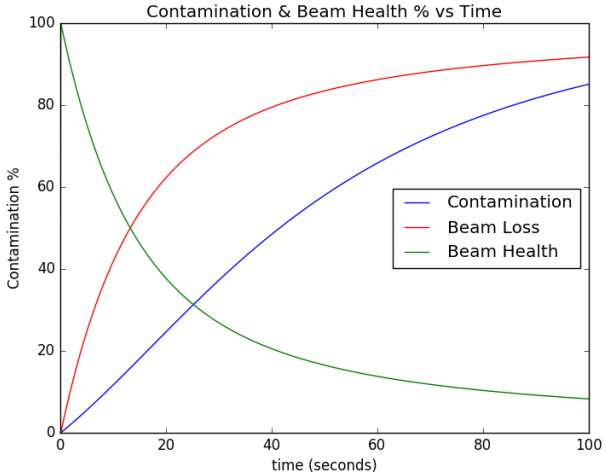
```
from __future__ import division
```

## 9.1 Plotting and Graphs

A nicely formatted plot with a legend using the pylab package.

```
import pylab as plt #Imports the correct packages for plotting.

plt.title('Contamination & Beam Health % vs Time') # Creates a title.

plt.plot(t, Contamination, '-b', label='Contamination') #Plots Contamination in blue.
plt.plot(t, Beam_loss, '-r', label='Beam Loss') #Plots Beam_loss in red.
plt.plot(t, Beam_health, '-g', label='Beam Health') #Plots Beam_health in green.
#plt.plot(x,y,'-color', label='Legend Label') #Template

plt.xlabel("time (seconds)") #Creates a x-axis label
plt.ylabel("Contamination %") #Creates a y-axis label

plt.legend(loc='center right') #Creates a legend with the labels set above.
#Other locations include upper/lower/center left/right

plt.show() #Displays plot.
```

This code would display a graph such as the one below such that the proper values are input.



# ROOT

# References