

# Tutorial #1 - What You Need to Know About the Tutorials

Habib Ghaffari

2022-09-19 Mon

## Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
<b>2</b>	<b>Objectives of this tutorial</b>	<b>2</b>
2.1	Learn How to Work With the Course's website . . . . .	2
2.1.1	Resolve your problems . . . . .	2
2.2	How to Install <b>Python</b> Interpreter: . . . . .	4
2.2.1	Windows . . . . .	4
2.2.2	Mac OS . . . . .	4
2.2.3	Linux (Ubuntu) . . . . .	5
2.3	IDE for <b>Python</b> Coding . . . . .	5
2.4	Implement Our First <b>Python</b> Program . . . . .	5
2.5	Simple Language . . . . .	5

## 1 Introduction

My name is **Habib Ghaffari Hadigheh**. It is a mouthful name so you can simply call me **Habib** whenever you see me on campus. I am a fourth-year Ph.D. student under the supervision of Dr. Christopher Anand. My main area of research is Machine Learning (ML), Operations Research (Or) and Signal Processing. You can find information about how to contact me via my website at the following address [Here](#).

ghaffh1

Today we are going to learn how to work with the course's website, how to install **Python** interpreter, and how to write our first program in python to make sure our interpreter is working properly. I am also going to introduce

an IDE helps you to do coding much easier. If we have time I am also going to start working on a very simple language and try to write some programs based on it and see how our example program going to be evaluated.

## 2 Objectives of this tutorial

At the end of this session you should be able to :

- Learn to work with the website.
- Install the python interpreter in different operating systems.
- Install IDE for coding.
- Write our first Python program.
- Try to define a very simple language.

### 2.1 Learn How to Work With the Course's website

Here is the website address:

<http://www.cas.mcmaster.ca/~franek/courses/cs3mi3/>

On this website, you can find the course outline, a page for announcements and a link to sign in to your page. You should receive an email around a week ago with your username and password.

#### 2.1.1 Resolve your problems

- As TAs, we do not have any office hours, but we are available via email and you can schedule online/in-person meetings with any of us by emailing us.
- Problems with marking may happen in any course. If you think there is any problem with your marking or the comments/feedback you received is unclear, please contact me. I will figure out who marked your assignment/exam and make sure they provide you with more clarification about your mark. If you are still not satisfied I will ask another TA to mark your submission again. The final step would be for either me or Dr. Franek to look at your solution and make the final decision.






## COMPSCI 3MI3 -- Principles of Programming Languages




[Course Outline](#)  
[Course Announcements History](#)  
[Login](#)





Welcome to [COMPSCI 3MI3](#), a course intended to introduce the students to the fundamental principles of programming languages, and corresponding programming paradigms. This website contains information related to the class, such as the course schedule, test dates and location, assignments, and other general information. Please let the instructor, Prof. F.Franek, know if there is something you can suggest to improve it.

It is each student's responsibility to be aware of the information on the course Web pages, and to regularly check for announcements and course news.

*The instructor and the university reserve the right to modify elements of the course during the term. The university may change the dates and deadlines for any or all courses in extreme circumstances. If either type of modification becomes necessary, reasonable notice and communication with the students will be given with explanation and the opportunity to comment on the changes. It is the responsibility of the student to check his/her McMaster email and course websites daily during the term and to note any changes.*

Last revised: Aug 29, 2022  
 Comments to: Prof. F. Franek  
[Dept. of Computing and Software](#)  
[Faculty of Engineering](#)  
[McMaster University](#)  
[Hamilton, Ontario](#)  
[Canada L8S 4K1](#)  
 email: [franek@mcmaster.ca](mailto:franek@mcmaster.ca)  
[administrator](#)  
[TA](#)

Figure 1: Course Page

## CS 3MI3 (2022/23, Term I) Start login

[CS 3MI3 home](#)

enter your student number  
 (leading zeros can be omitted)

and press

WARNING: It is illegal and not permitted by McMaster regulations to use a student number of any other person than yourself.

Figure 2: Login Page (Put your user name)



## CS 3MI3 (2022/23, Term I) Login

[CS 3MI3 home](#)

*You need a valid course password to view your personal information (such as marks and comments), and to view or submit assignments or projects.*

enter your  
course password  ←

and press  OR

↑

Figure 3: Login Page (Type your password)

## 2.2 How to Install Python Interpreter:

### 2.2.1 Windows

There are a few methods to install Python interpreter on your machine. The straightforward method is to use Python website.

<https://www.python.org/>

Another method is to use Chocolatey. You can install Chocolatey using the following link:

<https://chocolatey.org/install>

Then you can simply use the following link to install a version of Python on your machine:

<https://community.chocolatey.org/packages/python/3.9.1>

Another method is to use Anaconda:

<https://www.anaconda.com/>

### 2.2.2 Mac OS

Mac OS should have an in-built Python version pre-installed on the Machine. However, you can easily install a version based on your preferences:

The straightforward method again is to use the website:

<https://www.python.org/downloads/macos/>

But the most preferable method is to use Homebrew. You can install Homebrew on your machine using the following link:

<https://brew.sh/>

The next step is to use the following instruction to install any version of python you want:

```
brew install python
```

### 2.2.3 Linux (Ubuntu)

Installing Python on a Linux (Ubuntu) machine is very easy. Just type the following command in your terminal

```
sudo apt-get install python
```

## 2.3 IDE for Python Coding

You can use almost any editor to implement python programs. However, the best well-known IDE is Pycharm:

<https://www.jetbrains.com/pycharm/>

## 2.4 Implement Our First Python Program

Let's implement our Hello World program in Python.

```
print("Hello World!!")
```

## 2.5 Simple Language

Let's have a look at the BNF as you know it:

```
<postal-address> ::= <name-part> <street-address> <zip-part>
```

```
    <name-part> ::= <personal-part> <last-name> <opt-suffix-part> <EOL> | <personal
```

```
    <personal-part> ::= <initial> "." | <first-name>
```

```
    <street-address> ::= <house-num> <street-name> <opt-apt-num> <EOL>
```

```
    <zip-part> ::= <town-name> ", " <state-code> <ZIP-code> <EOL>
```

```
    <opt-suffix-part> ::= "Sr." | "Jr." | <roman-numeral> | ""
```

```
    <opt-apt-num> ::= <apt-num> | ""
```

This translates into English as:

- A postal address consists of a name-part, followed by a street-address part, followed by a zip-code part.
- A name-part consists of either: a personal-part followed by a last name followed by an optional suffix (Jr., Sr., or dynastic number) and end-of-line, or a personal part followed by a name part (this rule illustrates the use of recursion in BNFs, covering the case of people who use multiple first and middle names and initials).
- A personal-part consists of either a first name or an initial followed by a dot.
- A street address consists of a house number, followed by a street name, followed by an optional apartment specifier, followed by an end-of-line.
- A zip-part consists of a town-name, followed by a comma, followed by a state code, followed by a ZIP-code followed by an end-of-line.
- An opt-suffix-part consists of a suffix, such as "Sr.", "Jr." or a roman-numeral, or an empty string (i.e. nothing).
- An opt-apt-num consists of an apartment number or an empty string (i.e. nothing).

The language we are going to define contains just a handful of syntactic forms: the boolean constants **true** and **false**, conditional expressions, the numeric constant 0, the arithmetic operators **succ** (successor) and **pred** (predecessor), and a testing operation **iszero** that returns **true** when it is applied to 0 and **false** when it is applied to some other number.

<b>&lt;t&gt; ::=</b>	<b>Term</b>
<b>T</b>	<b>Constant True</b>
<b>F</b>	<b>Constant False</b>
<b>0</b>	<b>Zero</b>
<b>if t then t else t</b>	<b>Conditional Expression</b>
<b>succ t</b>	<b>Successor</b>
<b>pred t</b>	<b>Predecessor</b>
<b>iszero t</b>	<b>Zero Test</b>

A program in the present language is just a term built from the forms given by the grammar above. Here are some examples of programs, along with the results of evaluating them:

```
if (F) then 0 else (succ 0);  
> succ 0
```

```
iszero (pred (succ 0));  
> T
```

What is the evaluation result of the following programs?

```
> if (succ 0) then 0 else pred (succ (succ (0)));  
> if 0 then T else F;  
> iszero (T);
```

Exercise:

- How could we evaluate the above expression?

Now let's define the above language using **Haskell** programming language:

```
data term =  
    T -- True Constant  
  | F -- False Constant  
  | Zero -- Zero Constant  
  | Succ term -- Successor  
  | Pred term -- Predecessor  
  | IfThenElse term term term -- Conditional Expression  
  | IsZero term -- Zero Test
```

Exercise:

- How could we define the language rules using **Haskell**?
- Is it possible to define this language in **Python**? If yes then how?