COSC 1306 - Programming for Non- Majors

An Introduction

Dr. Mohan

McMurry University

August 19, 2023

Welcome to Class!

- I am Prof. Mohan a.k.a, **Dr. Mo**
- This is the **COSC-1306** Programming for Non-Majors.
- Taken for exploring CS in general education and by (Math & Physics) Majors.
- **No Prerequiste.**

How to access the lecture slides?

- **Access** class materials through the website `https://amohan.mcm.edu/`
- **Download** the Lesson-1 materials.
- **Double Click** the downloaded file (lesson1.pdf) to open in Adobe Acrobat for Windows or Preview for Mac.

About Me ...

- A brief background about my **teaching**:
  - What do I love about teaching the course?
  - How long have I been teaching?
  - What is my favorite thing about teaching?

About Me ...

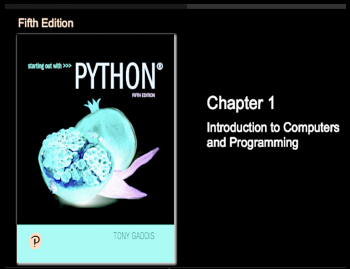
- A brief background about my **research**:
  - My research area is in the field of **Big Data and Cloud Computing!**
  - I love developing data models, algorithms, and tools to solve problems in computing large and unstructured data and identify & apply strategies to make it scalable using the Cloud.

Icebreaker ...

- Okay, I like us to spend a few minutes during this class period to Introduce and **get to know** each other. So, tell us about you by answering the following questions.
  - What is your Name?
  - Tell us something about you. For example your hobbies, interests, favourite food, movie, etc.
  - What do you love about Computer Science?

Lesson Topics Overview

- Introduction
- Motivation
- Basics of Computers & Programming
- Course Info
- To Do's



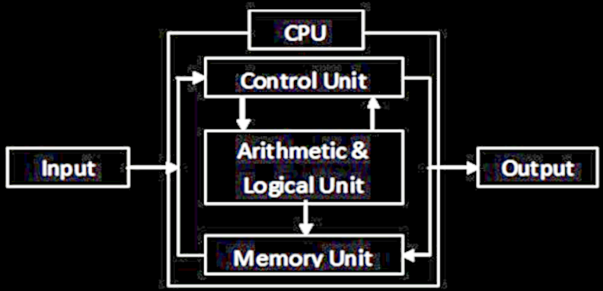
Introduction

- Computers can be programmed and designed to do any job that a **program** tells them to.
- Program is a set of **instructions** that a computer follows to perform a task. Commonly referred to as Software
- Programmer is a **person** who can design, create, and test computer programs. Also known as software developer.

Introduction

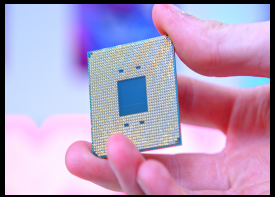
- What is a **Computer?**

  Computer is a system composed of several **components** that all work together. The typical major components are Central processing unit, Main memory, Secondary storage devices, and Input/Output devices.



Introduction

- What is a **CPU?**
  - Central processing unit (CPU) is the part of the computer that actually runs programs.
    - Most important component.
    - Without it, cannot run software.
    - Used to be a huge device.
  - Microprocessors are CPUs located on small chips.



Introduction

- What is a **Main memory?**
  - Main memory is where computer stores a program while program is running, and data used by the program.
  - Known as Random Access Memory or RAM.
    - CPU is able to quickly access data in RAM.
    - Volatile memory used for temporary storage while program is running.
    - Contents are erased when computer is off.

Introduction

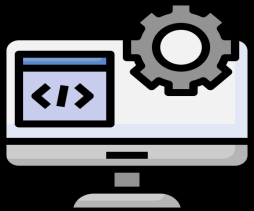
- What is a **Secondary storage?**
  - Secondary storage can hold data for long periods of time. Programs normally stored here and loaded to main memory when needed.
  - Types of secondary memory:
    - Disk drive: magnetically encodes data onto a spinning circular disk.
    - Solid state drive: faster than disk drive, no moving parts, stores data in solid state memory.
    - Flash memory: portable, no physical disk.

- What is an **Input Device?**
  - Input is the data the computer collects from people and other devices. Input device are the component that collects the data.
    - Examples: keyboard, mouse, touchscreen, scanner, camera.
    - Disk drives can be considered input devices because they load programs into the main memory.

- What is an **Output Device?**
  - Output is the data produced by the computer for other people or devices. Can be text, image, audio, or bit stream
  - Output device formats and presents output.
    - Examples: video display, printer
    - Disk drives and USB drives can be considered output devices because data is sent to them to be saved.

Introduction

- What is a **Software**?
  - Software is defined as the programs and other operating information used by a computer. In other words, everything the computer does is controlled by software.



Introduction

- General **categories of Software**:
  - **Application software:** programs that make computer useful for every day tasks. Examples: word processing, email, games, and Web browsers.
  - **System software:** programs that control and manage basic operations of a computer.
    - Operating system: controls operations of hardware components.
    - Utility Program: performs specific task to enhance computer operation or safeguard data.
    - Software development tools: used to create, modify, and test software programs.

Introduction

- What is a **Hardware**?
  - Hardware is a **physical** devices that make up a computer. The term hardware distinguishes the tangible aspects of a computing device from software, which consists of written, machine-readable instructions or programs that tell physical components what to do and when to execute the instructions.

Introduction

- General **categories of Hardware**:
  - **Internal Components:** collectively process or store the instructions delivered by the program or Operating System.
    Examples: Motherboard, CPU, RAM, Hard drive, Graphics processing unit, Network Interface Card (NIC), USB ports, power supplies, transistors and chips.
  - **External Components:** are those items that are often externally connected to the computer to control either input or output functions.
    Examples: Mouse, Keyboard, Microphone, Camera, Scanner, USB stick, Memory Card, Monitor, Printer, Speakers, Headphones.

Introduction

- What is **Programming**?
  - Process of creating software that directs the electronic digital hardware operations to perform a desired task.
  - To recap, Hardware consists of the physical electronic components and typically have very specialized and limited operations.
  - To recap, Software is a set of instructions used by the hardware to enable it to perform some useful function and typically have a specific purpose.
  - Problem solving technique that transforms raw data into useful information.

Motivation

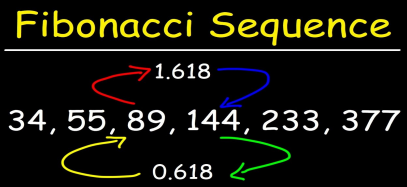
- Today's **computers**, like laptops and mobile devices, are incredibly fast and have enormous amounts of storage but the real power comes from the **software**, launch a different application and the computer performs a different task!
- **Usefulness** of the computer is limited only by the creativity of programmers!
- This course is all about learning **how to program** computers!

How Computers Store Data?

- All data in a computer is stored in sequences of 0s and 1s.
- Byte is a storage that is just enough memory to store letter or small number.
- Divided into eight bits.
  - Bit is an electrical component that can hold positive or negative charge, like on/off switch.
  - The on/off pattern of bits in a byte represents data stored in the byte.

Storing Numbers

- Bit represents two values, 0 and 1.
- Computers use binary numbering system.
  - Position of digit j is assigned the value 2j-1.
  - To determine value of binary number sum position values of the 1s.
- Byte size limits are 0 and 255. 0 is all bits off. 255 is all bits on.
- To store larger number, use several bytes.

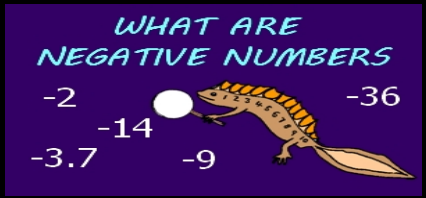


Storing Characters

- Data stored in computer must be stored as binary number.
- Characters are converted to numeric code, numeric code stored in memory.
- Most important coding scheme is ASCII.
- ASCII is limited and defines codes for only 128 characters.
- Unicode coding scheme becoming standard. Compatible with ASCII. Can represent characters for other languages.

Advanced Number Storage

- To store negative numbers and real numbers, computers use binary numbering and encoding schemes.
- Negative numbers encoded using two's complement - Real numbers encoded using floating-point notation.



Other Types of Data

- Digital describes any device that stores data as binary numbers.
- Digital images are composed of pixels.
- To store images, each pixel is converted to a binary number representing the pixel's color.
- Digital music is composed of sections called samples.
- To store music, each sample is converted to a binary number.

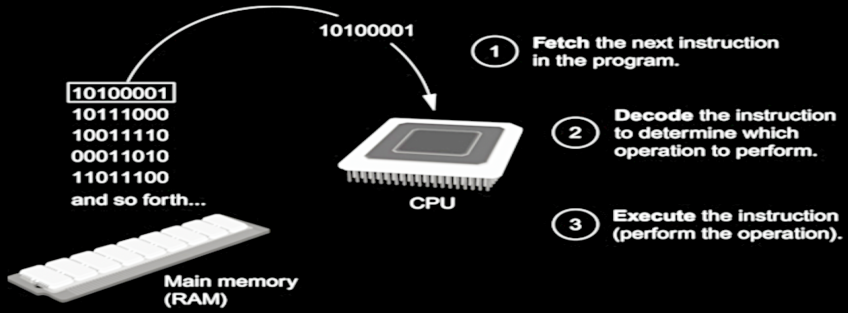
How a Program Works? (1 of 3)

- CPU designed to perform simple operations on pieces of data. Examples: reading data, adding, subtracting, multiplying, and dividing numbers.
- Understands instructions written in machine language and included in its instruction set. Each brand of CPU has its own instruction set.
- To carry out meaningful calculation, CPU must perform many operations.

How a Program Works? (2 of 3)

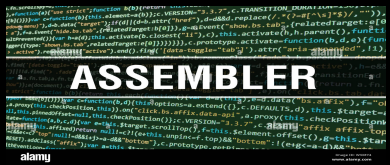
- Program must be copied from secondary memory to RAM each time CPU executes it.
- CPU executes program in cycle:
  - Fetch: read the next instruction from memory into CPU.
  - Decode: CPU decodes fetched instruction to determine which operation to perform.
  - Execute: perform the operation.

How a Program Works? (3 of 3)



From Machine Language to Assembly Language

- Impractical for people to write in machine language.
- Assembly language uses short words (mnemonics) for instructions instead of binary numbers. Easier for programmers to work with.
- Assembler: translates assembly language to machine language for execution by CPU.



High level Language

- Low-level language: close in nature to machine language. Example: Assembly language.
- High-Level language: allows simple creation of powerful and complex programs.
- No need to know how CPU works or write large number of instructions. More intuitive to understand.

Keywords, Operators, and Syntax: an Overview

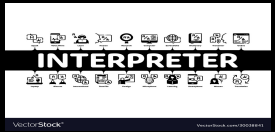
- Keywords are predefined words used to write program in high-level language. Each keyword has specific meaning.
- Operators are used to perform operations on data. Example: math operators to perform arithmetic.
- Syntax is a set of rules to be followed when writing programs.
- Statement are individual instructions used in high-level language.

Compilers and Interpreters (1 of 3)

- Programs written in high-level languages must be translated into machine language to be executed.
- Compilers translates high-level language program into separate machine language program.
- Machine language program can be executed at any time.

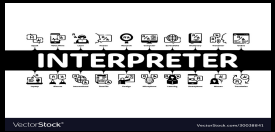
Compilers and Interpreters (2 of 3)

- Interpreter translates and executes instructions in high-level language program.
- Used by Python language.
- Interprets one instruction at a time
- No separate machine language program.
- Source code are statements written by programmer.
- Syntax error prevents code from being translated.

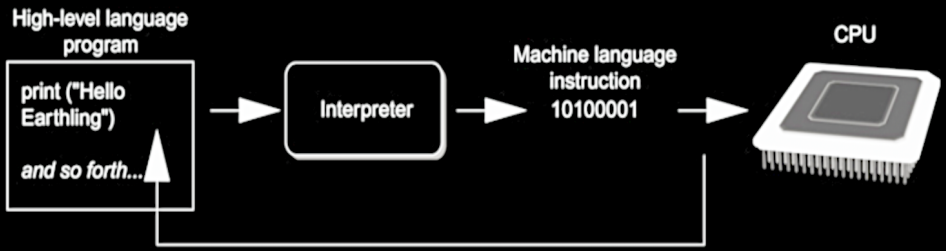


Compilers and Interpreters (2 of 3)

- Interpreter translates and executes instructions in high-level language program.
- Used by Python language.
- Interprets one instruction at a time
- No separate machine language program.
- Source code are statements written by programmer.
- Syntax error prevents code from being translated.



Compilers and Interpreters (3 of 3)



The interpreter translates each high-level instruction to its equivalent machine language instructions then immediately executes them.

This process is repeated for each high-level instruction.

Using Python

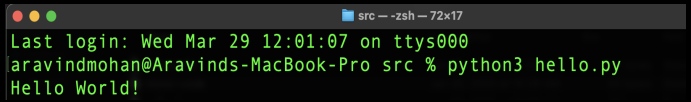
- Python must be installed and configured prior to use. One of the items installed is the Python interpreter.
- Python interpreter can be used in two modes:
  - Interactive mode: enter statements on keyboard.
  - Script mode: save statements in Python script.

Interactive Mode

- When you start Python in interactive mode, you will see a prompt.
- Indicates the interpreter is waiting for a Python statement to be typed.
- Prompt reappears after previous statement is executed.
- Error message displayed If you incorrectly type a statement.
- Good way to learn new parts of Python. But not a standard way of doing Programming.

Writing Python Programs and Running Them in Script Mode

- Statements entered in interactive mode are not saved as a program.
- To have a program use script mode.
- Save a set of Python statements in a file.
- The filename should have the .py extension.
- To run the file, or script, type python filename at the operating system command line.



Course Objectives

- **Identify** and **describe** programming concepts and techniques.
- **Demonstrate** ability to design and implement algorithms to solve selected problems.

Course Information

Course Objectives

- **Demonstrate** ability to use proper programming techniques.
- **Demonstrate** ability to use an Integrated Development Environment to write and test program code and debug programs.

Course Information

Course Topics

- Introduces **fundamental concepts** of programming as a **problem-solving methodology** through the following topics:
  - Data types, variables, and basic data operations.
  - Operators and Expressions.
  - Decision and repetition control structures.
  - Basic input and output including use of data files.
  - Programmer-defined functions.
  - Working with various Python objects.

Course Information

Course Goals

- At the **end of this course** the student will:
  - Have a working knowledge of the Python programming language with some of its most commonly used features and libraries.
  - Have a foundation in common programming practices and techniques.
  - Be able to design, write, and debug Python computer programs and work with an Integrated Development Environment (IDE).

Course Information

Special Software Requirements

- **JupyterHub**, an interactive computing infrastructure and IDE managed by McM is provided to design, develop, and test programs in this course. All classwork and assignments that contains a programming component should be completed using JupyterHub.

Course Information

Special Software Requirements

- **GitHub** is an industry-standard tool for software development. In this course, GitHub is used to deliver course materials such as lecture slides and completing programming assignments.

Course Information

Special Software Requirements

- **Slack** is an industry-standard tool used to communicate announcements, links to assignments, classwork activities, quizzes & exams, etc. and to facilitate q/a sessions.

Course Information

Special Software Requirements

- Accomodate by setting **Dark Theme** on your laptop including Google Chrome Browser and/or other specific IDE related to this course so that I can see your computer screen.

Course Information

Class Meeting Time

- We will meet **Tuesday's and Thursday's** from 9:30 AM - 10:50 PM in the Main Campus, Cooke, 211.

Course Information

Professor's Office Hours

- Dr. Mo's office is located at **C-206** (Cooke).
- Monday, Wednesday:
  8-10:30 AM, 2:30-3 PM.
- Tuesday, Thursday:
  8-9 AM, 11-12:30 PM, and 2:30-3 PM.

Email to schedule time outside office hours.

Course Information

Professor's Office Hours

To **schedule office hours** time slot, visit my website [teaching page] and click on the **Schedule Meeting** link located on the top right-hand corner to schedule 20 mins slots.

Let us connect with each other and enjoy our time together...

Course Information

Website Details

- Professor's Website:

  `https://amohan.mcm.edu`
- Course Website:

  `https://amohan.mcm.edu/course.php?cid=MTg=`

Course Information

Administrative Stuff!

- Verify if you are correctly registered for the course using McM Portal and Moodle.
- **McM Portal:** `https://mymcm.mcm.edu/ics`
- **Moodle:** `https://moodle.mcm.edu/`
- Make sure to read through the **Syllabus** available in the course webpage, before next class.

Course Information

More Administrative Stuff!

Assignments (4)	40%
Quizzes (4)	10%
Midterm	15%
Final	25%
Attendance & Participation	10%

Read the Syllabus to get a complete overview of the course.

Course Information

Interaction between us...

- All questions are welcome. There is no question which is good or bad. So, **ask** your questions at any time and clarify.
- Interaction is the best way to get rid of long lectures. So, let us try to **interact** more so that the communication is a two way process and the class is not boring.
- Reach out to me to ask for **Accommodations ...**

We will work together to explore C Programming in this course.

Course Information

How to do well in this Class?

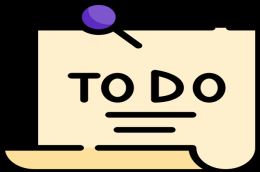
- **Attentively** listen to classes and try to participate in all class discussions.
- Take detailed **notes** during every class period.
- **Clarify** with the Professor, if a lesson is confusing.
- Complete all the **reading** assignments thoroughly.
- Do the **in-class** exercises thoroughly.

Be ready to think, process, and learn programming in this course!

Course Information

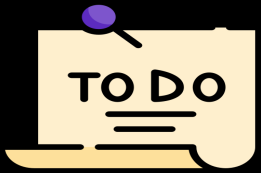
Things to do

- Register for the **Course Slack** Channel using your McM Email.
- You can join the channel by clicking the second icon at the top right hand side of the course webpage!
- Complete this task **before the start** of the next class period.



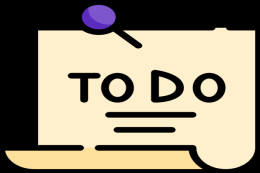
Things to do

- Register in **GitHub.com** using your McM Email.
- You can access the class materials by clicking on the first icon at the top right hand side of the course webpage!
- Complete this task **before the start** of the next class period.



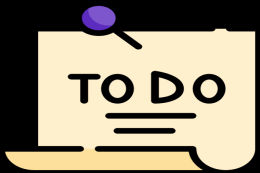
Things to do

- Read the **Syllabus** thoroughly before next class and feel free to reach out to me and ask your questions to clarify.
- Complete this task **before the start** of the next class period.



Things to do

- Complete the **Week-1 Activity** by accepting the link below:
- `https://classroom.github.com/a/Ev-AbyfC`
- The activity should be submitted through GitHub, and the **deadline for submission** is end of the next class period.



Lesson Summary

This lesson covered:

- Introduction
- Motivation
- Basics of Computers & Programming
- Course Info
- To Do's

Questions?

Ask your Questions to clarify!