Digital Skills and Programming Introduction (DSPI)

Course overview

Lecturers: Joost Broekens, Jonne Goedhart, Christos Athanasiadis

TA's: Jakub, Prem , Eliza, Matei

Course overview and overall objective

- This course introduces students to the fundamentals of computer science through two main pillars: **Digital Skills** and an **Introduction to Programming**.

- Students will develop foundational knowledge and practical skills to **navigate digital environments** and **write basic programs**.

- The course emphasizes understanding key concepts, applying learned skills, and describing and explaining core principles.

The computer: your tool!

- As a Data scientist, AI specialist and computer scientist, the **computer is your tool**!

- You must **master your tool** before you can do cool things,
  - develop a new algorithm
  - understand buggy ChatGPT code
  - develop larger software systems
  - use and customize AI code.
  - develop data preprocessing code
  - writing reports, making graphs

- https://www.youtube.com/watch?v=BZcVzfRBdQI (29 to 33 Minutes)

Course goal 1: Theory: understand the basics of a computer system

- **Number systems and data representation**
  - Understand Binary and hexadecimal number systems
  - Understand logical gates, and simple Boolean logic
  - Understand basic encodings including bits, bytes, words, floats.
- **Files and files systems**
  - Understand the file system structure: directories, files, links, file and folder permissions (read, write, execute).
  - Understand the difference between file types (executables/data)
  - Understand text/character encoding (ASCII, UTF-8, Unicode)
  - Explain basic lossless file compression principle.
- **Basic computer hardware concepts**
  - Understand the basic idea behind the von Neuman architecture.
  - Know the following components: RAM, ROM(firmware/BIOS), CPU, GPU, Data bus, I/O Peripherals (e.g. Disk, USB, Network adapter).
  - Understand – at a high level - their roles and how they interact in data transfer and processing.
  - Understand the system boot process: what happens when you power up the computer in terms of BIOS/UEFI boot steps
- **Operating system fundamentals**
  - Understand the role of the OS in hardware and software management
  - User interaction: the GUI (graphical user interface) and CLI (command line interface)
  - Understand the difference between kernel code, drivers, and user applications (apps)
  - Understand basic resource management: memory management, process management, application management
- **Networking and Cloud Fundamentals**
  - Understand the TCP/IP network model including IP address hierarchy, routing, ports, and the 4 network layers.
  - Explain the Client-Server model.
  - Understand remote storage and computation on cloud computing.
  - Know the most common IP based protocols: HTTP (web), FTP (file transfer), SSH.

Course goal 2: Practice: Apply Basic Digital Skills necessary for Computer Science

- **Number systems and data representation**
  - Perform basic calculations using the binary and hexadecimal number systems
  - Perform simple data representation tasks using bits, bytes and floats with and without sign bits.
  - Perform simple Boolean logic (AND, OR, NOT, XOR).
- **Files and files systems**
  - Execute programs through the CLI and GUI
  - Identify the available disks and drives in your system through GUI and CLI.
  - Navigate the file system using the CLI and GUI.
  - Identify and modify file and folder permissions (read, write, execute).
  - View and interpret raw file content in a text editor or hex viewer to identify encoding or format.
- **Basic computer hardware concepts**
  - Identify and list hardware components through CLI and GUI: RAM, ROM, CPU, GPU, Bus, I/O Peripherals (e.g. Disk, USB, Network adapter).
  - Analyze their activity through a system monitoring tool through CLI and GUI.
  - Inspect and understand the boot sequence in your own BIOS or UEFI, create a boot stick and boot from it.
- **Operating system fundamentals**
  - Install programs (edu VPN, pycharm, python, notepad++, git) and find device drivers and inspect where they are located.
  - Install Python through anaconda (and understand where it is installed to!)
  - List and monitor active processes of the system through CLI and GUI.
  - Identify the executable file that belongs to an active process.
- **Networking and Cloud Fundamentals (Practice)**
  - Find your device’s IP address and network information using CLI tools.
  - Use ping to check server availability.
  - Use curl to make a basic request to a web server and inspect the data.
  - Use SSH to log on to a server and inspect the file system through the CLI
  - Create a git repo and use it to store your course files, understand that it’s stored remotely by inspecting the repo through the browser.

Course goal 3: Theory: Understand basic programming concepts

- **Computational thinking, specification, coding and the basic programming pipeline**
    - Understand the basics of a Turing Machine
    - Understand the process: problem → computational thinking → (formal) specification → program code (source file) → parse tree (syntax) → compiled code (executable binary) or interpreting (line-by-line execution).
    - Understand the difference between syntax and semantics
- **Variable-related concepts**
    - Understand types, variables and instances, constants, assignment (by value/reference/copy).
- **Operations and expressions**
    - Understand the difference between expression and statements
    - Understand that operators and custom functions are both functions with parameters and return values.
    - Understand arithmetic, logical and comparison operators.
- **Control flow**
    - Understand the line-by-line execution nature of code
    - Understand the (conditional) jump, and how if/else/while/break/when (event-based) relate to this
    - Understand function calls (in imperative languages), and why functions exist.
- **Variable scope, namespace and lifetime.**
    - Understand that types may contain attributes and functions.
    - Understand variable scope, in particular the difference between global, function and local (variable instance).
    - Understand variable visibility and lifetime
    - Understand parameter passing (function arguments).
- **Basic algorithmic designs**
    - Understand the basics of iteration and recursion

Course goal 4: Apply basic programming concepts in *Start* pseudocode

- **Computational thinking, specification, coding and the basic programming pipeline**
  - Add the Start pseudocode package using the CLI and PIP install.
  - With a Start pseudocode example, inspect the syntax and compile to Python code and run as a python file through the CLI.
  - Write a small Turing machine program using
- **Variable-related concepts**
  - Define variables and instantiate (new) instances of such types.
  - Construct simple data types for common data structures such as a bank record or an array
  - Correctly perform assignments by value, reference or copy.
- **Operations and expressions**
  - Write simple expressions and statements to perform simple mathematical operations and manipulations on numbers and vectors.
  - Write simple logical operations that implement Boolean logic, and use the outcomes in comparisons
- **Control flow**
  - Predict and trace program behavior by simulating code execution manually.
  - Write simple conditional processing based in user input data (Input function)
  - Write simple algorithms that process lists and simple data types in which looping is essential, such as sorting or finding an element
  - Write a custom helper function to perform a compact operation and use this in another loop or conditional branching.
- **Variable scope, namespace and lifetime.**
  - Write solutions to simple puzzles that make use of global, local/instance and function scope.
  - Construct simple solutions to puzzles that play with parameter passing by reference and what it means to the underlying memory and variable values
  - Trace variable state through simple print-based debugging lines
- **Basic algorithmic designs**
  - Define a type for a simple data structure (e.g. a phonebook entry) that involves attributes and several functions.
  - Use this type to instantiate a small database (e.g. 10 entries) and implement a simple search or sort algorithm

Teaching methods

- Three (3) blocks of two hours per week (2+2 and 2) mandatory attendance
- Week structure block consists of
  - Block 1 and 2
    - 1 hour lecture
    - 1 hour instruction plenary, and work on assignments
  - Block 3
    - 1 hour Q and A and mini exam
    - 1 hour instruction plenary, and finalize assignments + submit in BS.
- Every week has 2 (sets of) practical assignments (Course Goal 2 and 4)
  - Deadline end of the week.
  - Non-graded but mandatory to submit in Brightspace at end of last instruction hour.
- End of week mini exam at the end of last lecture
  - Mandatory submit in Brightspace
- End of course digital exam (100% of grade)
  - MC knowledge and understanding questions on the Theory (see above Goal 1 and 3)
  - Pseudocode in Start (Goal 4)
- Bonus: assignments and mini exams all done well = right to an oral assessment in case of 2x written exam failure.

Additional material

- https://missing.csail.mit.edu/