Azebo Digital Foundation
Detailed Lesson Plan

Hiluf Abay and Hailu Kelayu

June 10, 2025

Prepared for high school students, this document outlines a 6-week interleaved summer program designed to introduce participants to technology through foundational training in computer programming with Python, basic computer skills, introductory electronics, and effective communication.

Contents

Program Overview

This document presents the comprehensive lesson plan for the High School Azebo Digital Foundation. Structured in an interleaved format, the program runs for six weeks, with sessions held three days per week, both in the morning and afternoon. The curriculum introduces students to key technological foundations, including Computer Programming with Python, Basic Electronics, Computer Literacy, and Essential Communication Skills with a focus on speaking and writing.

The program consists of 36 sessions, each divided into a 1-hour instructional lecture followed by a 2-hour hands-on practical session, fostering both conceptual understanding and experiential learning. Each subject culminates in a capstone project, enabling students to apply their skills in a meaningful and creative way.

Program Schedule: 6-Week Interleaved Format

Week	Monday AM	Monday PM	Wednesday AM	Wednesday PM	Friday AM	Friday PM
(9am-12pm)						
1	CS100 S1: Intro to Computers	PY101 S1: Intro to Python	CS100 S2: Touch Typing	PY101 S2: Operators & Input	EL100 S1: Intro to Electricity	**Seminar 1: James Webb Space Telescope(JSWT)**
2	CM101 S1: Communication Fund.	PY101 S3: Conditionals	CS100 S3: File Management	EL100 S2: Breadboarding	PY101 S4: Loops	**Seminar 2: SpaceX's Reusable Rockets**
3	CM101 S2: Written Communication	PY101 S5: Functions	CS100 S4: Internet & Email	EL100 S3: Series/-Parallel	EL100 S4: Capacitors	**Seminar 3: OpenAI's Sora**
4	CM101 S3: Tech Documentation	PY101 S6: Data Structures 1	CS100 S5: Cybersecurity	EL100 S5: Microcontrollers	PY101 S7: Data Structures 2	**Seminar 4: Boston Dynamics' Atlas Robot**
5	CM101 S4: Presentation Skills	PY101 S8: Project Intro	CS100 S6: Productivity & Cloud	EL100 S6: Simple Sensors	EL100 S7: More Sensors	**Seminar 5: CRISPR Gene Editing**
6	CM101 S5: Presentation Workshop	EL100 S8: Interactive Projects	Seminar 6: Entrepreneurship	PY101 S9: Project Work	EL100 S9: Project Work	**Seminar 6: Neuralink Brain-Computer Interface**

Detailed Course Modules

1. CS100 - Basic Computer Skills

Module Overview

This module equips students with essential computer literacy skills required to thrive in todays technology-driven world. It introduces fundamental concepts in computer hardware and software, operating system navigation, and cybersecurity best practices. Students will also develop practical competencies in touch typing through structured drills, along with hands-on experience in modern productivity applications and cloud-based collaboration tools. Emphasis is placed on consistent practice and real-world application to build confidence and efficiency in digital environments.

Total Sessions: 6

1.1. S1: Introduction to Computers & Operating Systems

Learning Objectives

- **Distinguish** between various categories of computer hardware and software components.
- **Identify and articulate** the function of core computer hardware components, including the Central Processing Unit (CPU), Random Access Memory (RAM), and different types of storage devices (HDD, SSD).
- **Explain** the fundamental role and purpose of an Operating System (OS) as an intermediary between user applications and hardware.
- **Effectively navigate** and interact with the Graphical User Interface (GUI) of contemporary operating systems (e.g., Windows 11, macOS Sonoma).
- **Perform** basic OS operations such as launching and closing applications, accessing system information, and customizing desktop environments.

Materials

- **Hardware:** Desktop/laptop computers (1 per student), projector, interactive whiteboard.
- **Software:** Windows operating systems, common applications (web browser, text editor).
- **Teaching Aids:** Presentation slides, physical examples, animation on YouTube video, and detailed diagrams of computer internals (motherboard, CPU, RAM sticks), handouts with GUI navigation exercises.

Lecture (1 hour)

- **Introduction to Computer Systems:** Definition of a computer, its evolution, and its ubiquitous role in modern society.
- **Hardware Components:** Detailed explanation of CPU (processing power), RAM (volatile memory for active tasks), Storage (persistent data storage HDDs vs. SSDs), input/output devices. Visual aids will be used to deconstruct a computer.
- **Software Classification:** Distinction between System Software (Operating Systems, Utility Software) and Application Software (productivity, entertainment, specialized).
- **Operating Systems:** Core functions of an OS (resource management, process schedul-

ing, memory management, file management, user interface). Comparison of popular OS types (Windows, macOS, Linux) and their use cases.
- **GUI Navigation:** Live demonstration of basic GUI elements: desktop, icons, taskbar/-dock, start menu/applications folder, window management (minimize, maximize, close), basic settings access.

Interactive Session (2 hours)

- **Guided System Exploration (45 minutes):** Students will engage in a "system scavenger hunt" using a provided checklist. Tasks include: Locating and documenting their computer's CPU type, installed RAM, and primary storage capacity; Identifying the version of their operating system; Opening and closing at least three different pre-installed applications; Accessing and changing desktop background or theme settings; Connecting to a Wi-Fi network.
- **OS Comparison & Discussion (45 minutes):** In small groups, students will be assigned different hypothetical scenarios (e.g., graphic design, gaming, software development) and discuss which operating system might be most suitable for each, justifying their choices. This will be followed by a class-wide discussion on the pros and cons of different OS environments.
- **Basic Troubleshooting Scenarios (30 minutes):** Instructor-led walk-through and student practice with simple common issues, such as force-quitting unresponsive applications using Task Manager (Windows) or Force Quit (macOS).
- **Assessment:** Completion of the scavenger hunt checklist, active participation in group discussions.

1.2. S2: Touch Typing Fundamentals

Learning Objectives

- **Demonstrate** correct finger placement on the home row keys (ASDF JKL;) without visual aid of the keyboard.
- **Apply** ergonomic principles for optimal typing posture, hand, and wrist positioning to prevent strain and injury.
- **Practice and improve** typing accuracy and speed for basic words and sentences using a touch-typing methodology.
- **Identify** and correct common typing errors.

Materials

- **Hardware:** Computers with functional keyboards (standard QWERTY layout).
- **Software:** Access to reputable online typing tutors (TypingClub, Keybr.com, Ratatype and Typing.com).
- **Teaching Aids:** Diagrams illustrating correct ergonomic posture and hand placement, pre-designed practice sheets for offline reference.

Lecture (1 hour)

- **Importance of Touch Typing:** Discussion on how touch typing significantly enhances productivity, reduces cognitive load, and promotes healthier work habits for digital professionals.
- **Ergonomics for Typists:** Detailed instruction and demonstration of proper seating posture, screen distance, arm and wrist positioning, and keyboard/mouse placement. Explanation of common repetitive strain injuries (RSIs) and how to prevent them.
- **Home Row Discipline:** Introduction to the QWERTY keyboard layout. Emphasis on the home row (ASDF JKL;) and the concept of returning fingers to these anchor keys.
- **Finger Mapping:** Demonstrating which fingers are responsible for which keys, expanding beyond the home row to adjacent keys.
- **Guided Practice Introduction:** Explanation of the chosen online typing platform's interface and progression.

Interactive Session (2 hours)

- **Initial Setup & Ergonomic Check (15 minutes):** Students set up their workstations according to ergonomic guidelines. Instructors will circulate to provide individual adjustments and feedback on posture.
- **Guided Home Row Drills (45 minutes):** Students will begin with guided exercises on the chosen online platform, focusing exclusively on mastering the home row keys (ASDF JKL;) and the spacebar. They will practice recognizing key sounds and developing muscle memory.
- **Expanding Key Coverage (45 minutes):** Exercises will progressively introduce keys from the top row (QWERTY UIO) and bottom row (ZXCVBNM), emphasizing the correct finger assignment for each key. Timed drills will be incorporated to encourage speed development.
- **Accuracy Challenges & Self-Correction (30 minutes):** Students will engage in exercises designed to improve accuracy, with immediate feedback from the typing tutor. They will be encouraged to self-correct errors and avoid looking at the keyboard.
- **Typing Games/Freestyle Practice (15 minutes):** A short period for students to engage in typing games on the platform or practice typing short paragraphs from a provided text, applying all learned techniques.
- **Assessment:** Students will be assessed through instructor observation (posture and finger placement), and performance metrics from an online typing tutor (accuracy and WPM). A first marked assessment worth 5% will gauge early proficiency. A Fast Typer Competition will also be held to motivate progress and recognize top performers.

1.3. S3: File Management & Organization

Learning Objectives

- **Comprehend** the concept of a hierarchical file system and its importance for data organization.
- **Execute** fundamental file operations: creating, renaming, moving, copying, and deleting files and folders within an operating system's file explorer.
- **Identify and explain** the purpose of common file extensions (e.g., .docx, .pdf, .jpg,

.py, .zip).
- **Differentiate** between local storage and cloud storage solutions, understanding their respective advantages and disadvantages.
- **Implement** logical naming conventions for files and folders to enhance searchability and maintainability.

Materials

- **Hardware:** Computers.
- **Software:** File Explorer (Windows), and cloud storage client (Google Drive).
- **Teaching Aids:** Presentation slides, pre-configured "disorganized" digital folders containing various file types, checklists for organizational tasks.

Lecture (1 hour)

- **Digital Filing Cabinet Analogy:** Introduction to the concept of a file system as an organized structure for digital information, similar to physical filing cabinets.
- **Directories and Subdirectories:** Explanation of hierarchical structures (root, drives, folders, subfolders) and their role in logical data grouping.
- **File Operations:** Detailed demonstration of creating new folders, creating new files (e.g., text documents), renaming, cutting, copying, pasting, and deleting operations using the operating system's native tools. Emphasis on drag-and-drop vs. keyboard shortcuts.
- **File Extensions:** Explanation of common file extensions and their association with specific application types and data formats. Discussion on the importance of showing file extensions.
- **Local vs. Cloud Storage:** Comparative analysis of local hard drives vs. cloud-based solutions (Google Drive). Discussion of benefits (accessibility, backup, sharing) and considerations (security, privacy).

Interactive Session (2 hours)

- **Organizational Challenge (60 minutes):** Students are provided with a pre-configured, intentionally chaotic digital folder containing a mix of documents, images, videos, and various file types. Their task is to: Design and create a logical hierarchical folder structure; Move and rename existing files; Delete redundant files; Create at least three new files of different types.
- **Cloud Storage Practice (45 minutes):** Students will be guided to log into a pre-assigned or personal cloud storage account. They will: Upload a selection of files; Create a new folder; Share a specific file or folder with a partner.
- **File Extension Scavenger Hunt (15 minutes):** Students will be given a list of uncommon file extensions and tasked with using a search engine to find out what software typically opens them and what type of data they contain.
- **Assessment:** Instructor review of the students' organized file structures, successful upload/sharing of files to cloud storage, and completion of the file extension scavenger hunt.

1.4. S4: Internet & Email Essentials

Learning Objectives

- **Introduce** students to the basics of using the internet and web browsers, including navigating websites, understanding URLs, and using browser tools effectively.
- **Execute** effective search engine queries utilizing advanced operators (e.g., quotation marks for exact phrases, `site:` for domain-specific searches, `filetype:` for specific document types).
- **Critically evaluate** the credibility and reliability of online sources based on established criteria (e.g., authoritativeness, currency, objectivity, accuracy).
- **Compose and send** professionally formatted emails, ensuring clear subject lines, appropriate greetings and closings, concise body paragraphs, and functional email signatures.
- **Identify** and avoid common pitfalls in online research and email communication.

Materials

- **Hardware:** Computers with internet access.
- **Software:** Web browser (Chrome, Edge), email client (Gmail).
- **Teaching Aids:** Presentation slides, checklist for source evaluation, examples of professional and unprofessional emails, scenarios for research tasks.

Lecture (1 hour)

- **Internet Architecture Basics:** A simplified overview of how the internet works (clients, servers, IP addresses, DNS).
- **Effective Search Strategies:** Beyond basic keywords: Exact phrase searches (" "), Exclusion of terms (-), Site-specific searches (`site:domain.com`), File type searches (`filetype:pdf`), Using `OR` for alternative terms, Boolean operators and wildcards.
- **Evaluating Online Sources:** Introduction to criteria for assessing credibility: author, publication date, domain type, bias, supporting evidence, cross-referencing. Discussion of fake news and misinformation.
- **Anatomy of a Professional Email:** Breakdown of essential components: Recipient (To, Cc, Bcc), Subject Line (concise and informative), Greeting, Body (clear, concise, direct), Call to Action (if any), Closing, Signature Block (professional contact info).
- **Email Etiquette:** Discussion on appropriate tone, urgency, attachment handling, and reply discipline.

Interactive Session (2 hours)

- **Typing Practice Integration (30 minutes):** Dedicated time for students to continue practicing their touch-typing skills using the online platform.
- **Advanced Search Challenge (45 minutes):** In small groups, students will be given 2-3 specific research topics. For each topic, they must: Formulate at least three different search queries using advanced operators; Identify and select two credible online sources, justifying their credibility; Identify one non-credible source and explain why.

- **Professional Email Drafting (45 minutes):** Each student will draft two distinct emails: 1) A formal email to the instructor summarizing research findings; 2) A scenario-based email. Students will exchange drafts for peer review.
- **Assessment:** Quality of search queries and justification of source credibility during group presentation, adherence to professional email formatting and content guidelines in drafted emails.

1.5. S5: Cybersecurity Basics

Learning Objectives

- **Define** common cybersecurity threats, including malware (viruses, ransomware), phishing, and social engineering.
- **Formulate** strong, unique passwords adhering to current best practices (e.g., length, complexity, randomness, avoidance of personal information).
- **Identify** key characteristics and red flags commonly found in phishing attempts (e.g., suspicious links, urgent/threatening language, grammatical errors).
- **Explain** the function and importance of Two-Factor Authentication (2FA) in enhancing account security.
- **Develop** an awareness of personal data privacy and the importance of secure online habits.

Materials

- **Hardware:** Computers.
- **Software:** Web browser, password strength checker tool (online), simulated phishing email examples.
- **Teaching Aids:** Presentation slides, infographics on password best practices, examples of real and fake phishing emails, checklist for identifying phishing.

Lecture (1 hour)

- **Introduction to Cybersecurity:** Definition of cybersecurity and its relevance in daily life. Overview of the "CIA Triad" (Confidentiality, Integrity, Availability) as core principles.
- **Common Cyber Threats:** Malware (viruses, worms, trojans, ransomware), Phishing (tactics, targets), Social Engineering (manipulation tactics).
- **Strong Passwords:** Dispelling myths, password entropy, recommended length, character mix, unique passwords, password managers.
- **Two-Factor Authentication (2FA):** Explanation of "something you know" + "something you have/are." Demonstration of how 2FA works.
- **Personal Data Privacy:** What personal data is, why it's valuable, strategies for protecting it online.

Interactive Session (2 hours)

- **Typing Practice Integration (30 minutes):** Dedicated time for students to continue practicing their touch-typing skills using the online platform.

- **Password Creation Workshop (30 minutes):** Students use an online password strength checker to test strategies and collaboratively develop strong passwords. Discussion on remembering unique passwords.
- **Phishing Email Analysis (45 minutes):** Students analyze sample emails (legitimate/phishing) to identify red flags and categorize them. Class discussion on "phishing literacy."
- **2FA Simulation & Discussion (15 minutes):** Walkthrough of 2FA setup on a common platform. Discussion of scenarios where 2FA prevents breaches.
- **Assessment:** Participation in password workshop, ability to formulate strong passwords, accurate identification of phishing attempts.

1.6. S6: Productivity & Cloud Collaboration

Learning Objectives

- **Create and format** basic text documents using a word processing application (Google Docs, Microsoft Word), incorporating elements like headings, lists, and basic text styling.
- **Develop and manipulate** simple spreadsheets (Google Sheets, Microsoft Excel), including data entry, basic arithmetic calculations (sum, average), and chart creation.
- **Utilize** cloud-based tools for real-time collaborative document editing, commenting, and version control.
- **Demonstrate** effective teamwork and communication within a shared digital workspace.

Materials

- **Hardware:** Computers with internet access.
- **Software:** Access to Google Workspace (Docs, Sheets) or Microsoft 365 (Word, Excel) online.
- **Teaching Aids:** Presentation slides, sample documents and spreadsheets for collaborative tasks, collaboration checklist.

Lecture (1 hour)

- **Introduction to Productivity Suites:** Overview of integrated software suites for office tasks.
- **Word Processors - Core Functions:** Creating, opening, saving documents. Basic formatting (font size/style, bold, italics, underline), paragraph alignment, lists, headings, inserting simple images.
- **Spreadsheets - Core Functions:** Cells, rows, columns. Data entry, basic data types. Simple formulas: `SUM()`, `AVERAGE()`, cell referencing. Creating simple charts.
- **Cloud Collaboration Principles:** Why cloud tools are essential for teamwork. Features like real-time co-editing, commenting, suggesting edits, version history, sharing permissions.
- **Best Practices for Collaboration:** Communication within collaborative documents, assigning tasks, managing conflicts.

Interactive Session (2 hours)

- **Typing Practice Integration (30 minutes):** Dedicated time for students to continue practicing their touch-typing skills using the online platform.
- **Collaborative Document Creation (60 minutes):** Students work in pairs/groups on a shared document. Task: Research a topic, collaboratively draft a short report using formatting, commenting, and suggested edits.
- **Shared Spreadsheet & Data Analysis (60 minutes):** Using a shared spreadsheet, students enter mock data, apply basic formulas (`SUM`, `AVERAGE`), and collaboratively create a simple chart.
- **Assessment:** Instructor observation of collaborative workflow, proper use of document/spreadsheet features, quality of formatted documents/calculated spreadsheets.

2. CM101 - Communication Skills (Writing & Speaking)

Module Overview

This module equips students with essential fundamental speaking and writing skills, enabling them to communicate effectively in various contexts. It focuses on building confidence in self-expression, engaging in meaningful conversations, structuring ideas for both oral presentations and written texts, and refining their work through revision and editing.

2.1. S1: Speaking Basic Introductions & Everyday Conversation

Learning Objectives

- **Confidently introduce** themselves and others using basic English phrases.
- **Engage** in simple everyday conversations (e.g., asking about name, origin, simple preferences).
- **Apply** basic politeness expressions in conversation.

Materials

- **Teaching Aids:** Whiteboard/markers, projector, flashcards with common greetings and questions, audio clips of basic conversations, handouts with dialogue examples.

Lecture (1 hour)

- **Basics of Greetings and Introductions:** Formal and informal greetings, introducing oneself (name, age, origin/background) and others.
- **Simple Questions and Answers for Daily Interactions:** Covering topics like "How are you?", "What do you do?", "Where are you from?".
- **Pronunciation Focus on Key Phrases:** Emphasizing clarity and common stress patterns in conversational English.
- **Politeness and Common Expressions:** Incorporating "please," "thank you," "excuse me," "I'm sorry," and "you're welcome."

Interactive Session (2 hours)

- **Pair Practice (60 minutes):** Role-playing self-introductions and greetings in various scenarios (e.g., meeting a new classmate, introducing a friend).
- **Listening Exercises (20 minutes):** Students listen to audio clips of basic conversations and answer comprehension questions, then practice repeating key phrases.
- **Group Activities (10 minutes):** "Find someone who. . . " game using simple questions (e.g., "Find someone who likes to read," "Find someone who has a pet").
- **Instructor Feedback (30 minutes):** Individualized feedback on pronunciation, intonation, and overall conversational fluency.
- **Assessment:** Active participation in role-plays, ability to form complete sentences in introductions and answers, observed confidence.

2.2. S2: Speaking Topic-Based Conversations & Fluency

Learning Objectives

- **Discuss** familiar topics (e.g., family, school, hobbies) using simple sentences with increased clarity and fluency.
- **Utilize** techniques to improve speaking fluency, such as linking words and appropriate pacing.
- **Express** basic opinions and preferences politely.

Materials

- **Teaching Aids:** Topic cards (e.g., "My Family," "My School Day," "My Hobbies," "Future Dreams"), pronunciation drill worksheets, audio recordings of sample dialogues.

Lecture (1 hour)

- **Vocabulary and Sentence Structures for Everyday Topics:** Expanding vocabulary related to common themes and practicing forming descriptive sentences.
- **Fluency Techniques:** Introduction to linking words (e.g., "I wanna go," "gonna"), pausing for effect, and maintaining a natural conversational pace.
- **Pronunciation Drills:** Focusing on common problem sounds for the students' native language background, intonation patterns for questions and statements.
- **Introduction to Expressing Opinions Politely:** Phrases like "I think...", "In my opinion...", "I agree/disagree because...".

Interactive Session (2 hours)

- **Small Group Discussions (60 minutes):** Students pick topic cards and engage in guided discussions, using the vocabulary and sentence structures introduced.
- **Fluency Games and Tongue Twisters (30 minutes):** Activities designed to improve articulation and speaking rhythm, such as repeating challenging phrases or simple tongue twisters.
- **Roleplay Scenarios (20 minutes):** Practicing asking and answering questions about

opinions and preferences (e.g., "What's your favorite type of music and why?").

- **Peer and Instructor Feedback (10 minutes):** Feedback focused on clarity of expression, use of new vocabulary, and improvements in fluency.
- **Assessment:** Participation in discussions, observed improvement in fluency and pronunciation drills, ability to express simple opinions.

2.3. S3: Speaking Short Presentations & Group Discussions

Learning Objectives

- **Prepare and deliver** a short, organized presentation on a familiar topic.
- **Participate** actively and respectfully in group discussions, demonstrating turn-taking skills.
- **Apply** foundational public speaking skills (eye contact, voice control, body language) in a presentation setting.

Materials

- **Teaching Aids:** Note cards for presentations, presentation checklist handouts, video examples of short speeches, timers for practice.

Lecture (1 hour)

- **Structure of a Short Presentation:** Introduction (hook, topic, roadmap), Main Points (supported by simple details), Conclusion (summary, takeaway).
- **Tips for Eye Contact, Voice Control, and Body Language:** Practical advice and exercises for maintaining engagement, varying voice tone/volume, and using gestures effectively.
- **Basics of Group Discussion Etiquette:** Active listening in a group, turn-taking signals, agreeing/disagreeing politely, staying on topic.

Interactive Session (2 hours)

- **Presentation Preparation (60 minutes):** Students prepare a 2-3 minute speech on a familiar topic (e.g., "My Family," "My Favorite Place," "My Dream Job") using note cards and the provided checklist.
- **Deliver Presentations (45 minutes):** Students deliver their presentations to small groups (3-4 students). Each presentation is followed by a brief Q&A from the group.
- **Group Discussion Exercises (15 minutes):** Facilitated discussions on guided questions where students practice active listening and respectful turn-taking.
- **Constructive Feedback (20 minutes):** Peers and the instructor provide specific, actionable feedback on presentation structure, delivery elements (eye contact, voice), and participation in discussion.
- **Assessment:** Delivery of a structured short presentation, demonstrated ability to participate constructively in group discussions, application of basic public speaking techniques.

2.4. S4: Writing Sentence Structure and Basic Paragraphs

Learning Objectives

- **Identify** the components of a simple sentence (subject, verb, object).
- **Construct** grammatically correct simple and compound sentences.
- **Understand and apply** the basic structure of a paragraph (topic sentence, supporting details, concluding sentence).
- **Utilize** common punctuation rules (periods, commas, capitalization).

Materials

- **Teaching Aids:** Sentence structure charts, paragraph writing worksheets, sample sentences and paragraphs for analysis, simple grammar rule handouts.

Lecture (1 hour)

- **Parts of Speech Review:** Focus on nouns, verbs, and adjectives and their roles in sentences.
- **Constructing Simple and Compound Sentences:** Explanation of subject-verb agreement. Introduction to coordinating conjunctions (FANBOYS: For, And, Nor, But, Or, Yet, So) for compound sentences.
- **Paragraph Structure:** Introduction to the concept of a main idea (topic sentence) and how supporting details develop that idea. Discussion of concluding sentences.
- **Common Punctuation Rules:** Emphasis on periods for sentence endings, commas in lists and compound sentences, and capitalization for proper nouns and sentence beginnings.

Interactive Session (2 hours)

- **Sentence Building Exercises (45 minutes):** Students work with word cards (or digital equivalents) to build correct simple and compound sentences. Exercises to identify subjects, verbs, and objects in given sentences.
- **Writing Simple Sentences and Paragraphs (35 minutes):** Students practice writing a series of simple sentences on a given topic, then combine them into a coherent short paragraph (5-7 sentences) with a clear topic sentence.
- **Peer Review and Instructor Guidance (25 minutes):** Students exchange paragraphs for peer review, focusing on sentence clarity, grammar, punctuation, and paragraph structure. Instructor provides targeted feedback.
- **Writing Prompts (35 minutes):** Quick, guided writing prompts for immediate paragraph practice (e.g., "Describe your favorite animal," "What did you do last weekend?").
- **Assessment:** Correct formation of simple/compound sentences, ability to write a structured paragraph, demonstrated understanding of basic punctuation.

2.5. S5: Writing Writing for Real Life & Basic Essay Structure

Learning Objectives

- **Compose** short, practical texts for real-life situations, such as informal emails and friendly letters.
- **Outline** the fundamental structure of a simple essay (introduction, body paragraphs, conclusion).
- **Employ** basic linking words and transition phrases to improve text cohesion.
- **Identify** and avoid common grammatical pitfalls in extended writing.

Materials

- **Teaching Aids:** Sample informal emails and friendly letters, simple essay outline templates, writing prompt handouts focusing on opinion-based topics.

Lecture (1 hour)

- **Format and Language for Informal Emails and Letters:** Key components (salutation, body, closing), appropriate tone, common abbreviations, and expressions.
- **Introduction to Essay Structure:** Detailed breakdown of the three main parts of a basic essay:
  - **Introduction:** Hook, background, thesis statement.
  - **Body Paragraphs:** Topic sentence, supporting details/examples, concluding sentence.
  - **Conclusion:** Restate thesis, summarize main points, final thought.
- **Linking Words and Transition Phrases:** Introduction to words and phrases that connect ideas and paragraphs (e.g., "first," "also," "however," "in conclusion").
- **Common Grammar Pitfalls in Writing:** Review of frequent errors like run-on sentences, sentence fragments, and subject-verb agreement in longer texts.

Interactive Session (2 hours)

- **Practical Writing Practice (40 minutes):** Students draft a short informal email to a friend or family member, or a friendly letter inviting someone to an event. They will then peer-edit each other's drafts for clarity, tone, and appropriate language.
- **Essay Brainstorming and Outlining (45 minutes):** Students choose from a list of familiar topics (e.g., "The importance of hobbies," "My favorite season," "Should students have homework?"). They will brainstorm ideas and create a detailed outline for a simple 3-paragraph essay (intro, one body, conclusion) using the provided templates. **Group Sharing of Essay Ideas and Outlines (25 minutes):** Students share their essay outlines and topic sentences with the class for initial feedback on organization and clarity.
- **Instructor Feedback (20 minutes):** Feedback on the structure of drafted practical texts and the logical flow of essay outlines.
- **Assessment:** Completion of a well-structured informal email/letter, logical and complete essay outline, demonstrated use of linking words.

2.6. S6: Revision, Editing & Creative Writing

Learning Objectives

- **Apply** systematic revision strategies to improve the content, organization, and clarity of their written work.
- **Utilize** editing checklists to identify and correct common grammatical, spelling, and punctuation errors.
- **Experiment** with basic creative writing techniques (e.g., descriptive language, imagery) in short tasks.
- **Reflect** on their writing process and identify areas for personal improvement.

Materials

- **Teaching Aids:** Editing checklists (e.g., for grammar, spelling, punctuation, clarity), sample writing with intentional errors for correction, creative writing prompts (e.g., picture prompts, opening lines), vocabulary building lists for descriptive writing.

Lecture (1 hour)

- **Importance of Revision and Self-Editing:** Why writing is a process, not a single event. Distinction between revision (content, organization) and editing (grammar, mechanics).
- **Common Writing Errors and How to Fix Them:** Review of persistent errors identified in previous sessions (e.g., fragments, run-ons, comma splices, common spelling mistakes). Strategies for identifying and correcting these.
- **Techniques for Creative Expression in Writing:** Show, don't tell; using sensory details; figurative language (similes, metaphors - simplified).
- **Vocabulary Building for Descriptive Writing:** Introduction to stronger verbs and more precise adjectives to enhance descriptive quality.

Interactive Session (2 hours)

- **Editing Practice (45 minutes):** Students work individually or in pairs to identify and correct errors in provided sample texts that contain common grammatical, spelling, and punctuation mistakes.
- **Revising and Improving Student Work (60 minutes):** Students apply revision and editing checklists to their own previously written paragraphs or essay outlines. They will focus on strengthening topic sentences, adding more supporting details, improving transitions, and correcting errors.
- **Sharing and Feedback (10 minutes):** Students share excerpts from their revised or creative writing in small groups, receiving feedback on clarity, impact, and areas for further refinement.
- **Assessment:** Demonstrated ability to revise and edit texts using checklists, application of creative writing techniques, thoughtful self-reflection on writing process.

3. PY101 - Introduction to Computer Programming with Python

Module Overview

This module introduces students to the fundamental concepts of computer programming using the Python language, covering basic syntax and culminating in a mentored mini-project. The focus is on exploring the core principles of programming and demonstrating what students can achieve by developing coding skills. Through hands-on exercises and problem-solving activities, students will gain an introductory understanding of programming logic and its real-world applications, setting a foundation for future learning.

Total Sessions: 9

Session Structure

Each session maintains the 1-hour lecture + 2-hour interactive format:

- **Lectures** introduce programming fundamentals through conceptual explanations, Python syntax demonstrations, and exploratory discussions of programming possibilities.
- **Interactive Sessions** feature guided coding explorations, collaborative concept-mapping exercises, and scaffolded problem-solving activities. Students discover programming applications through carefully structured challenges of progressive complexity. Project-based sessions (S8, S9) will operate as discovery workshops where instructors facilitate conceptual understanding, demonstrate solution pathways, and highlight creative potential through mentor-guided experimentation.

3.1. S1: Intro to Python

Learning Objectives

- **Recognize** the core components of a Python program.
- **Observe** program execution through simple output examples.
- **Identify** fundamental data representations (integers, floats, strings, booleans).
- **Understand** how variables act as containers for data.

Materials

- **Hardware:** Computers.
- **Software:** Python 3 interpreter, IDE (e.g., VS Code) or online interpreter.
- **Teaching Aids:** Conceptual diagrams, exploratory code snippets, "Why Programming?" discussion prompts.

Lecture (1 hour)

- **The Landscape of Programming:** How algorithms solve real-world problems. Python's role in data science, automation, and creative coding.
- **Environment as a Discovery Tool:** Brief environment setup emphasizing its purpose for exploration.

- **"Hello World!" as Concept Demonstration:** Using `print()` to reveal program flow.
- **Data Representation Concepts:** How computers model information through types. Real-world analogies (e.g., numbers as measurements, text as messages).
- **Variables as Labels:** Metaphors for assignment operations. Basic arithmetic as transformation examples.

Interactive Session (2 hours)

- **Environment Exploration (30 minutes):** Guided discovery of coding interfaces. Observe execution of provided "Hello World!" examples.
- **Data Type Investigation (45 minutes):**
  - Modify pre-written variable examples to observe type behaviors
  - Experiment with `type()` to discover data categories
- **Variable Relationships (45 minutes):**
  - Adjust given arithmetic templates to observe computational outcomes
  - Mapping exercise: Connect variables to real-world representations (e.g., `temperature = 28.5`)
- **Concept Synthesis (30 minutes):**
  - Group discussion: "How might these concepts enable creating a simple calculator?"
  - Reflection: "Where have you encountered similar data representations?"
- **Assessment:** Participation in explorations, accurate observations of type behaviors, conceptual connections during synthesis.

3.2. S2: Operators & Input

Learning Objectives

- **Recognize** different operator categories and their purposes in expressions.
- **Observe** how user interaction works through the ‘input()‘ function.
- **Understand** the concept of data type conversion in program interactions.
- **Explore** how simple programs can respond to user-provided information.

Materials

- **Hardware:** Computers.
- **Software:** Python 3 interpreter, IDE/online interpreter.
- **Teaching Aids:** Concept diagrams showing operator functions, real-world analogies for input/output, sample conversational programs.

Lecture (1 hour)

- **Operators as Building Blocks:** Conceptual overview of arithmetic, comparison, and logical operators as tools for expressing relationships.
- **Programs as Conversations:** Metaphor of ‘input()‘ as "asking questions" and ‘print()‘

as "giving answers".
- **Data Types in Dialog:** Why input needs conversion (text vs. numbers) demonstrated through relatable examples.
- **Simple Decision Concepts:** Introduction to boolean results as yes/no outcomes using everyday comparisons.

Interactive Session (2 hours)

- **Operator Exploration (45 minutes):**
  - Experiment with provided operator templates to observe outcomes
  - Predict-and-check exercises with comparison chains (e.g., ‘5 > 3 and 2 == 2‘)
- **Input Discovery (45 minutes):**
  - Run pre-built programs that use ‘input()‘ for simple questionnaires
  - Modify conversion functions (‘int()‘, ‘float()‘) to observe type changes
- **Concept Connection (30 minutes):**
  - Guided creation of a number-guessing hint generator (e.g., "Is your number > 10?")
  - Discussion: "How could these concepts enable a quiz program?"
- **Assessment:** Participation in explorations, accurate predictions of operator behaviors, understanding demonstrated in concept discussions.

3.3. S3: Conditionals

Learning Objectives

- **Recognize** the concept of decision-making and branching logic in programs.
- **Observe** the flow of program execution based on conditions using `if`, `elif`, and `else` statements.
- **Understand** how boolean expressions control different program paths.
- **Explore** how to build programs that respond differently to varying inputs and scenarios.

Materials

- **Hardware:** Computers.
- **Software:** Python 3 interpreter, IDE/online interpreter.
- **Teaching Aids:** Flowcharts illustrating conditional logic, scenario cards for decision-making programs (e.g., a "choose your own adventure" game snippet), debugging examples with common logical errors.

Lecture (1 hour)

- **Programs as Decision-Makers:** Introduce the idea that programs need to make choices, mimicking human decision processes. Analogies: traffic lights responding to cars, choosing clothes based on weather conditions.
- **The `if` Statement:** Basic syntax and the concept of executing a block of code *only if* a specified condition evaluates to `True`.

- **`else` for Alternatives:** Demonstrating how the `else` block provides a fallback path when the initial `if` condition evaluates to `False`.
- **`elif` for Multiple Choices:** Handling scenarios with more than two possible outcomes, explaining the order of evaluation for `elif` statements.
- **Boolean Expressions Revisited:** Deepen understanding of how comparison and logical operators combine to form the complex conditions that `if` statements evaluate, leading to `True` or `False` outcomes.

Interactive Session (2 hours)

- **Basic `if` Statement Exploration (45 minutes):**
  - Run and modify provided simple code snippets containing `if` statements (e.g., checking if a number is positive, if a string is empty).
  - Predict program outcomes for different input values, focusing on whether the conditional block executes.
- **`if-else` and `elif` Challenges (60 minutes):**
  - Develop programs that use `if-else` for binary choices (e.g., checking voting eligibility based on age, determining pass/fail).
  - Create programs incorporating `elif` for multiple choices (e.g., a simple grading system assigning A, B, C, D, or F based on a score range).
  - Introduce and explore nested conditionals for more complex decision trees (e.g., checking both age and if a special ID is present for entry).
- **Debugging Conditional Logic (30 minutes):**
  - Students will be presented with programs containing subtle logic errors within conditionals (e.g., incorrect operator, wrong order of `elif` clauses, off-by-one errors in ranges).
  - Guided practice in using print statements or a debugger to identify and rectify these logical flaws.
- **Concept Synthesis (15 minutes):**
  - Group discussion: "How can conditionals make programs appear 'intelligent' or more 'responsive' to user input?"
  - Brainstorming real-world scenarios where conditionals are indispensable (e.g., simple games, login systems, automated responses).
- **Assessment:** Successful completion of coding challenges involving `if`, `elif`, and `else`, demonstrated ability to identify and fix conditional logic errors, and active participation in concept discussions.

3.4. S4: Loops

Learning Objectives

- **Recognize** the need for repetition in programming.
- **Understand** the concept of iteration using `for` and `while` loops.
- **Observe** how loops automate repetitive tasks.
- **Explore** how to control loop execution (e.g., `break`, `continue`).

Materials

- **Hardware:** Computers.
- **Software:** Python 3 interpreter, IDE/online interpreter.
- **Teaching Aids:** Diagrams illustrating loop flow (flowcharts), examples of repetitive real-world tasks, debugging examples for infinite loops.

Lecture (1 hour)

- **The Power of Repetition:** Introduce the concept of loops as a way to avoid writing repetitive code. Analogies: a recipe instructing to stir until smooth, a fitness routine repeating exercises.
- **`for` loops (Iterating over Sequences):** Basic syntax of `for` loops. Explanation of iterating over strings, lists (briefly introduce lists conceptually as sequences), and `range()` function.
- **`while` loops (Conditional Repetition):** Syntax of `while` loops. Concept of a loop continuing as long as a condition is `True`. Importance of loop termination to avoid infinite loops.
- **Controlling Loop Flow:** `break` statement (exiting a loop early). `continue` statement (skipping current iteration).

Interactive Session (2 hours)

- **`for` Loop Exploration (50 minutes):**
  - Run and modify simple `for` loops to print sequences of numbers or characters.
  - Practice iterating over strings to count specific characters.
  - Create a program that prints a multiplication table for a given number using a `for` loop and `range()`.
- **`while` Loop Challenges (50 minutes):**
  - Develop a program using a `while` loop to count down from a number to zero.
  - Create a simple interactive program that repeatedly asks for user input until a specific keyword is entered (e.g., "quit").
  - Introduce a simple guessing game where the user guesses a number until correct, using a `while` loop.
- **Loop Control Statements Practice (30 minutes):**
  - Implement `break` in a loop that searches for a specific item in a list and stops once found.
  - Implement `continue` in a loop that prints numbers but skips multiples of 3.

  - Debugging: Introduce an infinite `while` loop and guide students to identify and fix the termination condition.
- **Concept Synthesis (10 minutes):**
  - Group discussion: "When would you choose a `for` loop versus a `while` loop?"
  - Brainstorming real-world applications of loops (e.g., processing data, game animations, repetitive calculations).
- **Assessment:** Successful completion of coding challenges using both `for` and `while` loops, correct application of `break` and `continue`, demonstrated ability to prevent/debug infinite loops, and active participation in concept discussions.

3.5. S5: Functions

Learning Objectives

- **Recognize** the concept of code reusability and modularity through functions.
- **Understand** how to define and call basic functions in Python.
- **Observe** the role of parameters and return values in functions.
- **Explore** how functions help organize code and solve larger problems incrementally.

Materials

- **Hardware:** Computers.
- **Software:** Python 3 interpreter, IDE/online interpreter.
- **Teaching Aids:** Diagrams illustrating function calls and data flow, analogy of functions as "mini-programs" or "recipes," code examples demonstrating function definition and calls.

Lecture (1 hour)

- **The Need for Functions:** Introduce the idea of avoiding "copy-pasting" repetitive code. Analogy: a machine that performs a specific task.
- **Defining Functions (`def`):** Basic syntax for defining a function using the `def` keyword, function name, parentheses, and colon. Importance of indentation for the function body.
- **Calling Functions:** How to execute a defined function.
- **Parameters (Inputs to Functions):** Explain how parameters act as placeholders for values passed into a function, allowing it to be flexible.
- **Return Values (Outputs from Functions):** The `return` keyword and its role in sending a result back from a function. Distinction between printing and returning.
- **Scope (Brief Introduction):** Briefly touch upon local variables within functions vs. global variables outside, without deep dive.

Interactive Session (2 hours)

- **Basic Function Definition and Calling (50 minutes):**
  - Write simple functions that greet a user or perform a basic calculation without parameters or return values initially.

- Practice calling these functions multiple times to observe code reuse.
- Modify a previous loop or conditional program to use a simple function to perform a repeated action.

- **Functions with Parameters and Return Values (50 minutes):**
  - Create functions that take one or more parameters (e.g., a function that calculates the area of a rectangle, taking length and width).
  - Implement functions that return a value (e.g., a function that returns the result of adding two numbers, or a function that returns a formatted string).
  - Combine concepts: create a function that takes input, performs a calculation using parameters, and returns the result.
- **Modular Problem Solving (30 minutes):**
  - Take a slightly larger problem (e.g., a simple text-based "calculator" that adds, subtracts, etc.) and break it down into smaller functions.
  - Discuss how functions improve readability and make debugging easier.
  - Debugging: Introduce functions with incorrect parameter passing or missing return statements, and guide students to fix them.
- **Concept Synthesis (10 minutes):**
  - Group discussion: "How do functions help us build more complex programs?"
  - Brainstorming real-world parallels for functions (e.g., a vending machine, a button on a remote control).
- **Assessment:** Successful definition and calling of functions with and without parameters/return values, demonstrated understanding of modularity, and active participation in discussions on problem decomposition.

3.6. S6 & S7: Data Structures

3.6.1. S6: Data Structures 1 (Lists, Tuples)

Learning Objectives

- **Recognize** the concept of collections for storing multiple pieces of data.
- **Understand** the characteristics and common uses of Python Lists.
- **Observe** how to create, access, modify, and iterate through Lists.
- **Explore** the immutability and applications of Python Tuples.

Materials

- **Hardware:** Computers.
- **Software:** Python 3 interpreter, IDE/online interpreter.
- **Teaching Aids:** Visual diagrams of lists/tuples, analogies (e.g., shopping list, fixed recipe ingredients), common list/tuple methods cheatsheet.

Lecture (1 hour)

- **Beyond Single Variables:** Introduce the idea of needing to store collections of related data. Analogy: a single box vs. a shelving unit.
- **Lists (`[]`):**
  - Definition: Ordered, mutable collections.
  - Creation: empty lists, lists with initial elements.
  - Accessing Elements: indexing (positive and negative), slicing.
  - Modifying Lists: adding (append, insert, extend), removing (remove, pop, del), changing elements.
  - Common List Methods: `len()`, `sort()`, `count()`, `index()`.
- **Tuples (`()`):**
  - Definition: Ordered, immutable collections.
  - Creation: tuple literals, single-element tuples.
  - Accessing Elements: indexing, slicing (similar to lists).
  - Immutability: demonstrating what cannot be changed in a tuple.
  - Use Cases: When data should not change (e.g., coordinates, record of events).
- **Iterating with Loops:** Briefly revisit `for` loops to iterate over elements in lists and tuples.

Interactive Session (2 hours)

- **List Manipulation Exercises (60 minutes):**
  - Create lists of different data types.
  - Practice adding/removing elements (e.g., manage a to-do list, update student names).
  - Use indexing and slicing to retrieve specific parts of a list.
  - Write a program to find the largest/smallest number in a list.
- **Tuple Exploration (45 minutes):**
  - Create tuples and attempt to modify them to observe immutability errors.
  - Use tuples to store fixed data (e.g., a person's birthdate, RGB color values).
  - Convert between lists and tuples.
- **Combining Concepts (25 minutes):**
  - Write a function that takes a list of numbers and returns their average.
  - Use a loop to process elements from a list of strings.
- **Assessment:** Successful creation and manipulation of lists, demonstrated understanding of tuple immutability, correct use of list/tuple methods in coding exercises.

3.6.2. S7: Data Structures 2 (Dictionaries, Sets)

Learning Objectives

- **Understand** the concept of key-value pairs and their application in Dictionaries.
- **Recognize** the characteristics and common uses of Python Dictionaries.
- **Observe** how to create, access, modify, and iterate through Dictionaries.

- **Explore** the concept of unique elements and their application in Python Sets.

Materials

- **Hardware:** Computers.
- **Software:** Python 3 interpreter, IDE/online interpreter.
- **Teaching Aids:** Analogies for dictionaries (e.g., phonebook, glossary), analogies for sets (e.g., unique collection of items), common dictionary/set methods cheatsheet.

Lecture (1 hour)

- **Dictionaries ({}):**
  - Definition: Unordered, mutable collections of key-value pairs. Keys must be unique and immutable.
  - Creation: Empty dictionaries, dictionaries with initial pairs.
  - Accessing Elements: using keys. Handling missing keys.
  - Modifying Dictionaries: adding new pairs, updating values, removing pairs (`del`, `pop`).
  - Common Dictionary Methods: `keys()`, `values()`, `items()`, `get()`.
  - Use Cases: Storing structured data (e.g., student records, configuration settings).
- **Sets ({}):**
  - Definition: Unordered collections of unique elements. Useful for membership testing and eliminating duplicates.
  - Creation: from lists, set literals.
  - Basic Set Operations: adding, removing elements.
  - Mathematical Set Operations: union, intersection, difference, symmetric difference.
  - Use Cases: Checking for unique items, finding common elements between collections.
- **Choosing the Right Data Structure:** Brief discussion on when to use lists, tuples, dictionaries, or sets based on problem requirements.

Interactive Session (2 hours)

- **Dictionary Manipulation Exercises (60 minutes):**
  - Create a dictionary to store contact information (name, phone, email).
  - Practice adding new contacts, updating phone numbers, and deleting contacts.
  - Write a program to search for a key or value in a dictionary.
  - Iterate through dictionary keys, values, and items using loops.
- **Set Operations Practice (45 minutes):**
  - Create sets from lists to remove duplicate elements.
  - Perform union, intersection, and difference operations between two sets (e.g., finding common hobbies between two people).
  - Use sets for fast membership checking.
- **Data Structure Application Challenge (25 minutes):**

  - A mini-challenge combining multiple data structures (e.g., using a list of dictionaries to store multiple student records, then using sets to find unique courses taken).
  - Discuss problem-solving approaches for data storage.
- **Assessment:** Successful creation and manipulation of dictionaries, correct application of set operations, ability to choose appropriate data structures for given problems.

3.7. S8: Project Session Simple Calculator

Learning Objectives

- **Apply** all learned Python concepts (input, variables, operators, conditionals, functions) to build a cohesive application.
- **Design** a user-friendly command-line interface for interaction.
- **Implement** error handling for invalid user inputs (e.g., non-numeric input, division by zero).
- **Practice** modular programming by breaking down the calculator's functionality into distinct functions.

Materials

- **Hardware:** Computers.
- **Software:** Python 3 interpreter, IDE/online interpreter.
- **Teaching Aids:** Flowchart examples for calculator logic.

Lecture (1 hour)

- **Project Introduction & Scope (15 minutes):** Define the simple calculator project. Discuss its components (getting numbers, choosing operation, performing calculation, displaying result).
- **Modular Design for Calculators (20 minutes):** Review how to break down the calculator into functions: `add(a, b)`, `subtract(a, b)`, `multiply(a, b)`, `divide(a, b)`. Emphasize main loop for user interaction.
- **Error Handling Fundamentals (15 minutes):** Introduction to common errors (e.g., `ValueError` for non-numeric input, `ZeroDivisionError`). Simple `try-except` blocks to gracefully handle these.
- **User Experience Considerations (10 minutes):** Tips for clear prompts, informative messages, and a continuous loop until the user decides to quit.

Interactive Session (2 hours)

- **Core Calculator Logic (60 minutes):**
  - Students start by implementing the four basic arithmetic functions (`add`, `subtract`, `multiply`, `divide`).
  - Guided practice in getting numeric input from the user and converting it.
  - Implement conditional logic to select the correct operation based on user input (e.g.,

"+", "-", "*", "/").

- **User Interface and Loop (50 minutes):**
  - Wrap the core logic in a `while` loop to allow multiple calculations without restarting the program.
  - Implement a "quit" option for the user.
  - Focus on clear `print()` statements for instructions and results.
- **Error Handling Implementation (30 minutes, as a challenge):**
  - Add `try-except` blocks around input conversion to handle non-numeric input.
  - Implement specific handling for `ZeroDivisionError` in the division function.
  - Test the calculator with various invalid inputs to ensure robust behavior.
- **Assessment:** A functional simple calculator program, clear and concise code, proper use of functions and conditionals, basic error handling implemented.

3.8. S9: Project Session Number Guessing Game

Learning Objectives

- **Utilize** random number generation to create unpredictable game elements.
- **Implement** a game loop that continues until a winning condition is met.
- **Apply** conditional logic to provide user feedback (e.g., "too high," "too low").
- **Track** and display the number of attempts the user takes to guess correctly.

Materials

- **Hardware:** Computers.
- **Software:** Python 3 interpreter, IDE/online interpreter.
- **Teaching Aids:** Diagram illustrating the game flow, ‘random‘ module documentation (simplified), examples of game feedback messages.

Lecture (1 hour)

- **Game Introduction Scope (15 minutes):** Define the number guessing game. Outline its rules (computer picks, user guesses, hints provided).
- **Random Number Generation (`random` module) (20 minutes):** Introduce the `random` module, specifically `random.randint()` for picking a secret number within a range.
- **The Game Loop (`while` loop) (15 minutes):** Explain how a `while` loop is perfect for this game, continuing as long as the guess is incorrect.
- **Conditional Feedback (10 minutes):** Discuss how `if-elif-else` statements will be used to tell the user if their guess is too high, too low, or correct.
- **Counting Attempts (5 minutes):** Show how a simple counter variable can track the number of guesses.

Interactive Session (2 hours)

- **Setting Up the Game (60 minutes):**
  - Import the `random` module and generate a random secret number (e.g., between 1 and 100).
  - Initialize a counter for guesses.
  - Create the basic game loop using `while False` or a placeholder condition that will be updated.
  - Get the first user guess.
- **Implementing Game Logic (50 minutes):**
  - Inside the loop, use `if-elif-else` to compare the user's guess with the secret number.
  - Provide appropriate feedback: "Too high!", "Too low!", "Correct!".
  - Update the counter with each guess.
  - Adjust the `while` loop condition so it terminates when the correct number is guessed.
- **Refinement and Polish (30 minutes):**
  - Add a message displaying the number of attempts when the user wins.
  - Implement error handling for non-numeric input (reusing skills from S8).
  - Optional challenge: Allow the user to choose the range of the secret number.
  - Test the game thoroughly with various guesses and edge cases.
- **Assessment:** A fully functional number guessing game, correct use of random numbers, effective game loop, appropriate conditional feedback, and accurate tracking of attempts.

4. EL100 - Basics of Electronics

Module Overview

This module introduces students to the fundamental concepts of electronics through hands-on circuit building, from basic principles to microcontroller programming. The goal is to provide a foundational understanding of how electronic components work together to create functional devices.

Total Sessions: 9

Session Structure

Each session adheres to the 1-hour lecture and 2-hour interactive format.

- **Lectures** will cover theoretical principles (e.g., Ohm's Law, Kirchhoff's Circuit Laws), introduce specific electronic components (resistors, capacitors, LEDs, sensors), and delve into microcontroller programming concepts using platforms like Arduino.
- **Interactive Sessions** will be almost entirely practical, taking place at individual workstations equipped with breadboards, various electronic components, multimeters, and microcontrollers. Students will build, test, and troubleshoot circuits from schematics, measure electrical properties, and program their microcontrollers to interact with physical inputs and outputs. Project-based sessions (S8, S9) will serve as intensive workshops where students integrate their learned coding and circuit-building skills to design, assemble, and test a functional electronic device or interactive system.

4.1. S1: Intro to Electricity

Learning Objectives

- **Define** fundamental electrical concepts: voltage, current, resistance.
- **Explain** Ohm's Law and its application in simple circuits.
- **Identify** common circuit components (resistors, LEDs, wires, power source).
- **Differentiate** between open and closed circuits.

Materials

- **Hardware:** Breadboards, jumper wires, 9V batteries and clips, resistors (various values), LEDs (various colors), multimeters.
- **Teaching Aids:** Presentation slides, circuit diagrams, breadboard layout diagrams, safety guidelines for handling electricity.

Lecture (1 hour)

- **What is Electricity?** Introduction to electrons, charge, and basic concepts of current flow.
- **Voltage, Current, Resistance:** Analogies (water pressure, water flow, pipe friction) to explain these concepts. Units of measurement (Volts, Amperes, Ohms).
- **Ohm's Law ($V = IR$):** Detailed explanation and derivation. Simple calculation examples.
- **Basic Circuit Components:** Resistors (function, color code introduction), LEDs (polarity, current limiting), wires, power sources.
- **Open vs. Closed Circuits:** Concept of a complete path for current.
- **Safety First:** Emphasize electrical safety procedures.

Interactive Session (2 hours)

- **Multimeter Introduction (30 minutes):** Hands-on practice using multimeters to measure voltage of a battery, continuity of a wire, and resistance of various resistors.
- **Simple LED Circuit (60 minutes):** Guided build on breadboard:

  - Connect a 9V battery, a resistor (e.g., 220 Ohm), and an LED in series.
  - Ensure LED lights up. Experiment with different resistor values to see brightness change.
  - Introduce concept of current limiting resistor for LED.
- **Ohm's Law Practice (30 minutes):** Using their built circuit, students will measure voltage across resistor and LED, and current in the circuit. They will then calculate expected values using Ohm's Law and compare with measured values.
- **Troubleshooting Basics (15 minutes):** Intentional introduction of common faults (e.g., reversed LED, open circuit) and guide students to troubleshoot.
- **Assessment:** Successful construction of simple LED circuit, accurate multimeter readings, basic Ohm's Law calculations.

4.2. S2: Breadboarding

Learning Objectives

- **Understand** the internal structure and connections of a breadboard.
- **Properly place** components and connect them on a breadboard to create a working circuit.
- **Interpret** basic circuit schematics and translate them to a physical breadboard layout.
- **Practice** neat and organized breadboarding techniques.

Materials

- **Hardware:** Breadboards (more than one size if possible), jumper wires (assorted lengths/colors), resistors, LEDs, pushbuttons, capacitors.
- **Teaching Aids:** Large diagrams of breadboard internal connections, simple schematics, laminated "cheatsheets" for breadboard best practices.

Lecture (1 hour)

- **Anatomy of a Breadboard:** Detailed explanation of rows (horizontal, power rails) and columns (vertical, component connections). Show internal metal clips.
- **Power Rails:** Explain how power and ground rails work.
- **Component Placement:** Best practices for neatly inserting components without short circuits.
- **Translating Schematics to Breadboard:** Step-by-step method for converting a 2D schematic into a 3D breadboard layout. Emphasize planning wire routes.
- **Common Breadboarding Mistakes:** Discuss issues like short circuits, loose connections, incorrect polarity.

Interactive Session (2 hours)

- **Breadboard Exploration (30 minutes):** Students use a multimeter to check continuity on different parts of the breadboard to confirm internal connections.
- **Switch-Controlled LED Circuit (75 minutes):** Guided build of a circuit where a

pushbutton controls an LED (e.g., simple push-to-light circuit).

- Introduce pushbuttons and their operation.
- Students follow a schematic to build the circuit from scratch.
- Instructor checks for proper connections and provides troubleshooting tips.

- **Neat Wiring Challenge (15 minutes):** Encourage students to rebuild a previous circuit with tidier wiring, using shorter wires and organizing components.
- **Assessment:** Successful construction of the switch-controlled LED circuit, demonstration of understanding breadboard connections, neatness of wiring.

4.3. S3: Series/Parallel Circuits

Learning Objectives

- **Differentiate** between series and parallel circuit configurations for resistors.
- **Calculate** total resistance in simple series and parallel resistor circuits.
- **Observe** how current and voltage behave differently in series vs. parallel branches.
- **Apply** Kirchhoff's Voltage Law (KVL) to series circuits and Kirchhoff's Current Law (KCL) to parallel circuits qualitatively.

Materials

- **Hardware:** Breadboards, jumper wires, 9V batteries, assorted resistors, multimeters.
- **Teaching Aids:** Presentation slides with clear circuit diagrams for series and parallel configurations, worksheets for calculation practice.

Lecture (1 hour)

- **Introduction to Circuit Configurations:** Why components are connected in different ways. Analogy: a single lane road (series) vs. a multi-lane highway (parallel).
- **Series Circuits:**
  - Definition: Components connected end-to-end, forming a single path for current.
  - Current: Same through all components.
  - Voltage: Divides across components.
  - Total Resistance: Sum of individual resistances ($R_{total} = R_1 + R_2 + \dots$).
  - **Qualitative KVL:** Voltage drops around a closed loop sum to zero (simplified for beginners).
- **Parallel Circuits:**
  - Definition: Components connected across the same two points, providing multiple paths for current.
  - Current: Divides among branches.
  - Voltage: Same across all components.
  - Total Resistance: Reciprocal sum of reciprocals ($\frac{1}{R_{total}} = \frac{1}{R_1} + \frac{1}{R_2} + \dots$, or product/sum for two resistors). Focus on concept, maybe simple two-resistor calculation.
  - **Qualitative KCL:** Current entering a junction equals current leaving it (simplified).

Interactive Session (2 hours)

- **Series Circuit Build Measurement (60 minutes):**
  - Students build a simple series circuit with 2-3 resistors and an LED.
  - Use the multimeter to measure:
    - Total voltage from battery.
    - Voltage drop across each resistor.
    - Current at different points in the series circuit (to confirm it's constant).
  - Compare measured values with calculated expected values using Ohm's Law and series resistance formula.
- **Parallel Circuit Build Measurement (60 minutes):**
  - Students build a simple parallel circuit with 2-3 resistors and LEDs.
  - Use the multimeter to measure:
    - Voltage across each parallel branch (to confirm it's constant).
    - Current in each branch (to observe current division).
    - Total current from the battery.
  - Compare measured values with calculated expected values.
- **Concept Reinforcement (30 minutes):**
  - Discussion: "What happens if one LED breaks in a series circuit vs. a parallel circuit?" (leading to understanding reliability).
  - Quick design challenge: "Design a circuit to power two LEDs at the same brightness, but if one breaks, the other stays on."
- **Assessment:** Correct construction of series and parallel circuits, accurate measurements demonstrating circuit laws, ability to explain differences between series and parallel behavior.

4.4. S4: Capacitors

Learning Objectives

- **Identify** a capacitor and describe its basic function in a circuit.
- **Explain** how a capacitor stores and releases electrical energy.
- **Observe** the charging and discharging behavior of a capacitor using an LED.
- **Understand** the concept of RC time constant qualitatively.

Materials

- **Hardware:** Breadboards, jumper wires, 9V batteries, resistors (e.g., 1k, 10k), LEDs, electrolytic capacitors (e.g., 100μF, 470μF, 1000μF), multimeters.
- **Teaching Aids:** Presentation slides with capacitor symbols and internal structure (simplified), video demonstrations of capacitor charging/discharging.

Lecture (1 hour)

- **Introduction to Capacitors:** What is a capacitor? Analogy: a small water tank that

can store water (charge) and release it.
- **Functionality:** How capacitors store electrical charge in an electric field between two conductive plates separated by a dielectric.
- **Charging and Discharging:** Explain the process of a capacitor charging when connected to a voltage source and discharging when the source is removed or shorted.
- **Capacitor Types and Polarity:** Briefly introduce different types, emphasizing the importance of polarity for electrolytic capacitors.
- **Qualitative RC Time Constant:** Explain that the time it takes for a capacitor to charge/discharge depends on both its capacitance (C) and the resistance (R) in the circuit. No complex formulas, just the concept of "longer R or C means slower change."
- **Applications (Brief):** Mention basic applications like smoothing power, timing circuits (very briefly).

Interactive Session (2 hours)

- **Capacitor Identification Polarity (15 minutes):** Students identify different capacitors, paying attention to markings for capacitance and voltage, and recognizing the polarity of electrolytic capacitors.
- **RC Charging/Discharging with LED (60 minutes):**
  - Students build a simple circuit: 9V battery, push-button (to charge), resistor, capacitor, and LED.
  - Observe the LED gradually dim as the capacitor discharges through it.
  - Experiment with different resistor values (e.g., 1k vs 10k) and capacitor values (e.g., $100\mu F$ vs $1000\mu F$) to see how the dimming time changes.
  - Discussion: "Why does the LED dim slowly instead of turning off instantly?"
- **Measuring Capacitor Voltage (30 minutes):**
  - Use a multimeter to measure the voltage across a charging capacitor over time (students can record readings at intervals to see the voltage curve).
  - Measure the voltage across a discharging capacitor.
- **Concept Reinforcement (15 minutes):**
  - Quick discussion on how this "storage" behavior is useful in electronics.
  - Brainstorming where a capacitor might be used in everyday devices (e.g., camera flash, power adapter).
- **Assessment:** Correct wiring of capacitor circuits, observation and qualitative explanation of charging/discharging behavior, ability to relate R and C values to timing.

4.5. S5: Microcontrollers (Arduino)

Learning Objectives

- **Identify** a microcontroller (e.g., Arduino Uno) and understand its purpose in electronics.
- **Recognize** the basic components of an Arduino board (digital pins, analog pins, power, USB).
- **Upload** a simple "blink" program to an Arduino board.
- **Understand** the basic structure of an Arduino sketch (`setup()`, `loop()`).

Materials

- **Hardware:** Arduino Uno boards (1 per student/pair), USB cables, breadboards, jumper wires, LEDs, 220 Ohm resistors.
- **Software:** Arduino IDE (pre-installed), sample "Blink" sketch.
- **Teaching Aids:** Large diagram of Arduino Uno pinout, simplified Arduino C++ syntax cheatsheet.

Lecture (1 hour)

- **What is a Microcontroller?** Introduction to embedded systems tiny computers designed for specific tasks. Analogy: the "brain" of an electronic device.
- **Introduction to Arduino:** Open-source platform, ease of use for beginners, role in prototyping.
- **Arduino Uno Board Layout:** Explain key components: digital I/O pins (input/output), analog input pins, power pins (5V, GND), USB port for programming and power.
- **The Arduino IDE:** Overview of its interface (sketch editor, verify/upload buttons, serial monitor).
- **Basic Arduino Sketch Structure:**
  - `void setup():` Runs once at the beginning (for initial setup of pins, etc.).
  - `void loop():` Runs repeatedly forever (for continuous operations).
- **First Program: "Blink LED":** Walkthrough of the classic "Blink" sketch (`pinMode()`, `digitalWrite()`, `delay()`).

Interactive Session (2 hours)

- **Arduino IDE Board Familiarization (30 minutes):**
  - Students open Arduino IDE, connect Arduino via USB.
  - Select correct board and port in the IDE.
  - Identify key components on their physical Arduino board.
- **Build and Upload "Blink" Circuit (60 minutes):**
  - Guided build of an LED circuit connected to a digital pin on the Arduino via a resistor.
  - Open the "Blink" example sketch.
  - Students verify and upload the sketch to their Arduino. Observe the LED blinking.
  - Experiment by changing the `delay()` values and re-uploading to see the blink speed change.
- **First Custom Code: "SOS" Blink (30 minutes):**
  - Challenge students to modify the "Blink" sketch to create an "SOS" pattern using different delays (short, long, short).
  - Discuss the concept of sequential execution and timing.
- **Troubleshooting Common Issues (15 minutes):**
  - Discuss common problems: incorrect board/port selection, wiring errors, missing semicolons in code.

    - Guide students through basic troubleshooting steps.
- **Assessment:** Successful upload and execution of the "Blink" sketch, modification to create a custom pattern, ability to identify basic Arduino components and IDE elements.

4.6. S6: Simple Sensors

Learning Objectives

- **Understand** the basic concept of a sensor and its role in converting physical phenomena into electrical signals.
- **Connect** a simple digital sensor (e.g., push-button, PIR motion sensor) to Arduino.
- **Read** digital input from a sensor using Arduino code (`digitalRead()`).
- **Control** an LED or other output based on sensor input.

Materials

- **Hardware:** Arduino Uno boards, USB cables, breadboards, jumper wires, LEDs, 220 Ohm resistors, push-buttons, PIR motion sensors (passive infrared).
- **Software:** Arduino IDE.
- **Teaching Aids:** Datasheets (simplified) for push-button and PIR sensor, circuit diagrams for connecting sensors to Arduino.

Lecture (1 hour)

- **What are Sensors?** Introduction to how electronic devices "sense" the world. Analogy: eyes, ears, touch.
- **Digital vs. Analog Sensors (Brief):** Distinguish between sensors that provide ON/OFF signals (digital) and those that provide a range of values (analog). Focus mainly on digital for this session.
- **Push-button as a Digital Sensor:** How a button changes a circuit's state from HIGH to LOW or vice-versa. Concepts of pull-up/pull-down resistors (simplified, may use internal pull-up).
- **PIR Motion Sensor (Digital):** Explain how it detects infrared radiation from movement, outputting a HIGH or LOW signal.
- **Reading Digital Input (`digitalRead()`):** How Arduino reads the state of a digital pin.
- **Connecting Sensors to Arduino:** Wiring diagrams and best practices for sensor integration.

Interactive Session (2 hours)

- **Push-button Control LED (60 minutes):**
    - Students build a circuit: push-button connected to a digital input pin, LED connected to a digital output pin.
    - Write an Arduino sketch:
        - Use `pinMode()` for input and output.

        - Use `digitalRead()` to check button state.
        - Use `digitalWrite()` to control LED based on button state (e.g., LED turns on when button is pressed).
    - Experiment with internal pull-up resistor if applicable.
- **PIR Motion Sensor Alarm (60 minutes):**
    - Students connect a PIR sensor to an Arduino digital input pin.
    - Write an Arduino sketch:
        - Read the PIR sensor's state.
        - If motion is detected (HIGH signal), turn on an LED "alarm" for a few seconds.
        - Use `Serial.println()` to print messages to the Serial Monitor when motion is detected/cleared (introducing Serial Monitor for debugging).
- **Concept Application (15 minutes):**
    - Discussion: "Where could you use a motion sensor or a button in a real device?" (e.g., automatic lights, doorbell).
    - Brainstorming: "What other digital sensors might exist?"
- **Assessment:** Successful implementation of circuits where LED output is controlled by digital sensor input, basic understanding of ‘digitalRead()‘, effective use of Serial Monitor for debugging.

4.7. S7: More Sensors

Learning Objectives

- **Understand** the basic concept of analog signals and analog-to-digital conversion (ADC).
- **Connect** a simple analog sensor (e.g., potentiometer, photoresistor/LDR) to Arduino.
- **Read** analog input from a sensor using Arduino code (`analogRead()`).
- **Control** an LED's brightness based on analog sensor input using Pulse Width Modulation (PWM).

Materials

- **Hardware:** Arduino Uno boards, USB cables, breadboards, jumper wires, LEDs, 220 Ohm resistors, potentiometers, photoresistors (LDRs), 10k Ohm resistors (for LDR voltage divider).
- **Software:** Arduino IDE.
- **Teaching Aids:** Diagrams of analog input pins, explanation of ADC, simple voltage divider circuit diagram, PWM concept visual.

Lecture (1 hour)

- **Analog Signals:** Explain that the real world is analog (continuous values like temperature, light intensity). How analog sensors provide a varying voltage output.
- **Analog-to-Digital Conversion (ADC):** How Arduino converts a continuous analog voltage into a discrete digital number (0-1023 range for Arduino Uno). Explain ‘analogRead()‘.

- **Potentiometer as an Analog Sensor:** How it works as a variable resistor, providing a variable voltage output.
- **Photoresistor/LDR (Light Dependent Resistor):** How its resistance changes with light. Explain simple voltage divider circuit to get a variable voltage output.
- **Pulse Width Modulation (PWM):** Introduction to how digital pins can simulate analog output (e.g., dimming an LED) by rapidly turning ON/OFF the voltage. Explain ‘analogWrite()‘.
- **Mapping Values (`map()` function):** Briefly introduce the ‘map()‘ function to convert a range of values (e.g., 0-1023 from ADC) to another range (e.g., 0-255 for PWM).

Interactive Session (2 hours)

- **Potentiometer Serial Monitor (60 minutes):**
  - Students connect a potentiometer to an analog input pin on the Arduino.
  - Write an Arduino sketch:
    - Use `analogRead()` to read the potentiometer value.
    - Print the raw analog value to the Serial Monitor.
    - Rotate the potentiometer and observe the changing values in the Serial Monitor.
  - Introduce the `map()` function to convert the 0-1023 range to a more understandable 0-100 percentage.
- **LDR/Photoresistor Light Dimmer (60 minutes):**
  - Students build a voltage divider circuit with an LDR.
  - Connect the LDR circuit to an analog input pin and an LED to a PWM-enabled digital output pin.
  - Write an Arduino sketch:
    - Read the LDR value.
    - Use the `map()` function to convert the LDR reading to an appropriate 0-255 range for LED brightness.
    - Use `analogWrite()` to control the LED brightness.
  - Test the circuit by varying the light intensity on the LDR and observe the LED brightness change.
- **Concept Application (15 minutes):**
  - Discussion: "Where could you use a potentiometer or a light sensor in a real device?" (e.g., volume control, automatic street lights).
  - Brainstorming: "What other analog sensors might exist and what could they measure?"
- **Assessment:** Successful implementation of circuits reading analog sensors, correct use of `analogRead()` and `analogWrite()`, demonstrated understanding of mapping values, ability to control output based on analog input.

4.8. S8: Project Session Automated Night Light

Learning Objectives

- **Integrate** an analog light sensor (LDR) to detect ambient light levels.
- **Program** an Arduino to automatically control an LED based on light intensity.
- **Apply** conditional logic (`if/else`) to switch the LED ON/OFF at a specified light threshold.
- **Calibrate** the light threshold for optimal performance in different environments.

Materials

- **Hardware:** Arduino Uno boards, USB cables, breadboards, jumper wires, LED, 220 Ohm resistor, Photoresistor (LDR), 10k Ohm resistor (for LDR voltage divider).
- **Software:** Arduino IDE.
- **Teaching Aids:** Simple circuit diagram for LDR and LED, Serial Monitor output examples for calibration, troubleshooting guide for common wiring/logic errors.

Lecture (1 hour)

- **Project Overview (15 minutes):** Introduce the Automated Night Light concept. Discuss how it combines a sensor (LDR) with a microcontroller (Arduino) to automate a task.
- **Review: LDR and Analog Input (15 minutes):** Briefly recap how the LDR works and how ‘analogRead()‘ translates light intensity into a numerical value. Emphasize the voltage divider.
- **Thresholding with Conditionals (15 minutes):** Explain how to use an ‘if‘/‘else‘ statement to create a "decision point" based on the analog reading (e.g., if light reading is below X, turn LED on; else, turn off).
- **Calibration Importance (15 minutes):** Discuss the need to find the right numerical threshold for "dark" and "light" conditions, and how to use the Serial Monitor for this calibration.

Interactive Session (2 hours)

- **Circuit Assembly (60 minutes):**
  - Students build the circuit for the LDR voltage divider and connect it to an analog input pin on the Arduino.
  - Connect an LED to a digital output pin on the Arduino (with a current-limiting resistor).
- **Initial Programming and Calibration (60 minutes):**
  - Write an Arduino sketch:
    - In `setup()`, initialize Serial communication (`Serial.begin()`) and set LED pin as output.
    - In `loop()`, read the LDR value using `analogRead()`.

    - Print the LDR value to the Serial Monitor (`Serial.println()`) to observe readings in different light conditions.
    - Experiment with various light levels (e.g., cover LDR, shine light) and note down approximate "dark" and "light" threshold values.
- **Implementing Logic (45 minutes):**
  - Based on their calibration, students add an `if/else` statement to their code:
    - If LDR reading is below the "dark" threshold, use `digitalWrite()` to turn the LED HIGH.
    - Else, turn the LED LOW.
  - Test the automated night light by varying ambient light conditions.
- **Refinement Troubleshooting (15 minutes):**
  - Refine the threshold value for better sensitivity.
  - Debug any wiring or logic errors preventing the light from working as expected.
  - Discuss how to make the light only activate in specific conditions (e.g., only after 5 seconds of darkness).
- **Assessment:** A working automated night light circuit, correctly calibrated light threshold, proper use of analog input and conditional logic for automation.

4.9. S9: Project Session - Simple Motion-Activated Alarm

Learning Objectives

- **Utilize** a digital motion sensor (PIR) to detect presence or movement.
- **Program** an Arduino to trigger an output (LED or buzzer) when motion is detected.
- **Implement** a timed response for the alarm/light to stay active for a set duration.
- **Debug** sensor integration and timing issues.

Materials

- **Hardware:** Arduino Uno boards, USB cables, breadboards, jumper wires, LED, 220 Ohm resistor, PIR Motion Sensor. (Optional: small passive buzzer).
- **Software:** Arduino IDE.
- **Teaching Aids:** PIR sensor pinout diagram, state diagram for alarm logic, troubleshooting checklist for motion sensors.

Lecture (1 hour)

- **Project Overview (15 minutes):** Introduce the motion-activated alarm/light project. Discuss its use in security systems, automatic lighting, etc.
- **Review: PIR Sensor and Digital Input (15 minutes):** Recap how the PIR sensor outputs a digital HIGH when motion is detected.
- **Timed Responses (`delay()`) (15 minutes):** Explain how to make the alarm/light stay on for a specific duration after motion is detected, using the ‘delay()‘ function.
- **Designing the Alarm Logic (15 minutes):** Discuss the sequence: motion detected $\rightarrow$ turn on alarm $\rightarrow$ wait $\rightarrow$ turn off alarm (or wait until motion clears).

Interactive Session (2 hours)

- **Circuit Assembly (60 minutes):**
  - Students connect the PIR motion sensor to a digital input pin on the Arduino. Ensure correct power (`VCC`), ground (`GND`), and data out (`OUT`) pins are used.
  - Connect an LED (or optional buzzer) to a digital output pin on the Arduino (with a current-limiting resistor for LED).
- **Programming Core Logic (60 minutes):**
  - Write an Arduino sketch:
    - In `setup()`, initialize LED/buzzer pin as output and PIR sensor pin as input. Initialize Serial communication.
    - In `loop()`, use `digitalRead()` to check the state of the PIR sensor.
    - If motion is detected (`HIGH`), use `digitalWrite()` to turn on the LED (or buzzer). Print "Motion Detected!" to Serial Monitor.
    - Add a `delay()` to keep the alarm/light on for a few seconds.
    - Use `digitalWrite()` to turn off the LED (or buzzer) after the delay. Print "Motion Cleared."
- **Refinement Troubleshooting (45 minutes):**
  - Adjust the delay time for the alarm/light.
  - Explore the PIR sensor's sensitivity and delay potentiometers (if available on the sensor module).
  - Debug any wiring or logic errors (e.g., LED always on/off, sensor not responding). Use Serial Monitor extensively for debugging.
  - Optional: Add a counter for how many times motion has been detected.
- **Assessment:** A functional motion-activated alarm/light circuit, correct integration of PIR sensor, appropriate use of digital input/output and timing for automated response.

5. Seminar Series

Module Overview

This seminar series is designed to ignite students' passion for STEM by exploring groundbreaking advancements and future possibilities in various fields. Each session will delve into a cutting-edge topic, featuring discussions, multimedia, and critical thinking exercises to inspire innovation and highlight diverse career paths.

Total Sessions: 6

5.1. Seminar 1: James Webb Space Telescope (JWST): Unveiling the Universe

Learning Objectives

- **Understand** the scientific purpose and engineering marvels of the James Webb Space Telescope.
- **Recognize** the significance of infrared astronomy in exploring the early universe.

- **Discuss** the process of scientific discovery through space exploration and data analysis.

Materials

- **Teaching Aids:** High-resolution images and videos from JWST, NASA/ESA educational resources, simplified diagrams of JWST's instruments (e.g., golden mirrors, sunshield).
- **Guest Speaker (Optional):** Astronomer, astrophysicist, or space engineer.

Lecture (1 hour)

- **Introduction to Telescopes and Space Exploration (15 minutes):** Brief history, why we put telescopes in space.
- **The Engineering Feat of JWST (25 minutes):** Its unique design (segmented mirror, sunshield), deployment sequence, and location (L2 point). Focus on the "how it works" and challenges overcome.
- **Scientific Goals and Early Discoveries (20 minutes):** Looking back in time, exoplanet atmospheres, stellar nurseries. Explain infrared light simply.

Interactive Session (2 hours)

- **Image Deep Dive & Discussion (60 minutes):**
  - Students analyze selected JWST images (e.g., Carina Nebula, Stephan's Quintet, exoplanet spectrum).
  - Guided discussion: "What do these images tell us about the universe?" "How does infrared light reveal things visible light can't?"
  - Small group research: Assign different JWST instruments (NIRCam, MIRI, etc.) for groups to quickly research and present their function.
- **Q&A / Future Implications (30 minutes):**
  - If guest speaker: Q&A session.
  - Otherwise: Open discussion on future discoveries, careers in astronomy, and the impact of space exploration on humanity.
- **Creative Challenge (30 minutes):** Imagine and describe a hypothetical exoplanet based on JWST data, or design a mission to a newly discovered object.
- **Assessment:** Active participation in image analysis and discussions, demonstrated understanding of JWST's purpose and key features.

5.2. Seminar 2: SpaceX's Reusable Rockets: Redefining Space Travel

Learning Objectives

- **Understand** the concept and economic impact of reusable rocket technology.
- **Analyze** the engineering challenges and innovations behind SpaceX's Falcon 9 and Starship.
- **Discuss** the future of space exploration, including Mars colonization and space tourism.

Materials

- **Teaching Aids:** Videos of Falcon 9 landings, Starship test flights, infographics comparing traditional vs. reusable launch costs, simulations of rocket launches/landings (if available).

Lecture (1 hour)

- **The Cost of Space Travel (15 minutes):** Historically, why rockets were expensive (disposable stages).
- **The Reusability Revolution (25 minutes):** Introduction to Falcon 9 and its vertical landing. Explain the physics simply (thrust, gravity, aerodynamics for controlled descent). Economic benefits of reusability.
- **Starship: The Next Frontier (10 minutes):** Overview of Starship's ambitious goals (fully reusable, Mars colonization).

Interactive Session (2 hours)

- **Video Analysis & Discussion (60 minutes):**
  - Watch and discuss videos of Falcon 9 landings and Starship test flights (failures and successes).
  - Group discussion: "What makes these landings so difficult?" "How does reusable tech change the game for space?"
  - Debate: "Is Mars colonization a realistic and ethical goal?"
- **Q&A / Career Paths (30 minutes):**
  - If guest speaker: Q&A session.
  - Otherwise: Discussion on careers in aerospace engineering, physics, and space industry entrepreneurship.
- **Assessment:** Active participation in discussions, demonstrated understanding of reusable rocket principles, engagement with future space concepts.

5.3. Seminar 3: OpenAI's Sora: The Future of Text-to-Video AI

Learning Objectives

- **Understand** the capabilities and underlying principles of text-to-video generative AI models like Sora.
- **Discuss** the creative potential and practical applications of AI in media generation.
- **Explore** the ethical implications and societal impact of highly realistic AI-generated content.

Materials

- **Teaching Aids:** Official Sora demonstration videos, examples of text prompts and their video outputs, short articles on generative AI, simple AI explanation diagrams.

Lecture (1 hour)

- **Introduction to Generative AI (15 minutes):** What is AI? What is generative AI? (e.g., DALL-E, Midjourney for images).
- **Sora's Breakthrough (25 minutes):** How it generates video from text. Concepts like diffusion models (simplified explanation). Show compelling demo videos.
- **Applications and Industries (10 minutes):** Filmmaking, advertising, education, content creation.
- **Ethical Considerations (10 minutes):** Deepfakes, misinformation, copyright, job displacement, bias in AI.

Interactive Session (2 hours)

- **Video Analysis & Prompt Guessing (60 minutes):**
  - Watch Sora videos and try to guess the text prompt used. Discuss the details and realism.
  - Small group activity: Provide a scenario and challenge groups to write the most creative and detailed Sora text prompt.
  - Analyze examples of potential misuse or ethical dilemmas related to AI-generated video.
- **Creative Brainstorm & Debate (30 minutes):**
  - Brainstorm new beneficial applications for text-to-video AI.
  - Debate: "Should AI-generated content always be clearly labeled?"
- **Q&A / Future of AI (30 minutes):**
  - If guest speaker: Q&A session.
  - Otherwise: Discussion on future developments in generative AI and careers in AI research, ethics, or content creation.
- **Assessment:** Active participation in discussions, ability to formulate creative prompts, thoughtful consideration of ethical aspects.

5.4. Seminar 4: Boston Dynamics' Atlas Robot: The Dawn of Humanoid Robotics

Learning Objectives

- **Appreciate** the advanced capabilities and challenges in developing highly agile humanoid robots.
- **Understand** the blend of mechanical engineering, control systems, and AI that enables robot movement.
- **Discuss** the potential applications and societal impact of humanoid robots in the future.

Materials

- **Teaching Aids:** Boston Dynamics' Atlas robot videos (running, jumping, dancing, parkour), simplified diagrams of robot joints and sensors, articles on robotics ethics.

- **Guest Speaker (Optional):** Robotics engineer, control systems expert, or industrial automation specialist.

Lecture (1 hour)

- **Introduction to Robotics (10 minutes):** What is a robot? Different types of robots (industrial, service, humanoid).
- **Atlas's Marvels (25 minutes):** Showcase compelling videos of Atlas performing dynamic tasks. Explain the underlying technology: hydraulic actuators, advanced sensors (LiDAR, cameras), sophisticated control algorithms, and AI for perception/navigation.
- **Engineering Challenges (15 minutes):** Balance, locomotion, manipulation, energy efficiency.
- **Future of Humanoid Robots (10 minutes):** Potential roles in dangerous environments, logistics, assistance, ethical considerations (job displacement, decision-making).

Interactive Session (2 hours)

- **Video Analysis & Speculation (60 minutes):**
  - Watch Atlas videos multiple times, focusing on specific movements. Discuss: "How do you think it achieves that movement?" "What sensors are at play?"
  - Group activity: Brainstorm new tasks or environments where Atlas-like robots could be beneficial.
- **Robot Design & Ethics Discussion (30 minutes):**
  - Design a simple robot for a specific purpose (e.g., house cleaning, space exploration), sketching its key features.
  - Debate: "Should robots be designed to look and act like humans?" "What are the ethical concerns of robots taking over jobs?"
- **Q&A / Careers in Robotics (30 minutes):**
  - If guest speaker: Q&A session.
  - Otherwise: Discussion on career paths in mechanical engineering, electrical engineering, computer science (AI/control systems), and robotics.
- **Assessment:** Active participation in video analysis and design challenge, demonstrated understanding of robot capabilities and ethical implications.

5.5. Seminar 5: CRISPR Gene Editing: Reshaping Biology and Medicine

Learning Objectives

- **Understand** the basic principles of CRISPR-Cas9 gene editing technology.
- **Recognize** the revolutionary potential of CRISPR in treating diseases and modifying organisms.
- **Engage** in ethical discussions surrounding human gene editing and its societal implications.

Materials

- **Teaching Aids:** Animated videos explaining CRISPR (e.g., from HHMI BioInteractive, TED-Ed), simplified diagrams of DNA and gene editing process, articles on CRISPR applications and ethics.
- **Guest Speaker (Optional):** Geneticist, biochemist, bioethicist.

Lecture (1 hour)

- **Introduction to DNA and Genes (15 minutes):** Quick recap of basic biology (DNA as instruction manual, genes as recipes).
- **What is CRISPR? (25 minutes):** Explain CRISPR as a bacterial "immune system" adapted for gene editing. Analogy: "molecular scissors" that can cut and paste DNA. Simplified explanation of Cas9 protein and guide RNA.
- **Applications of CRISPR (10 minutes):** Potential to cure genetic diseases (e.g., sickle cell, cystic fibrosis), modify crops, develop new diagnostics.
- **Ethical Considerations (10 minutes):** "Designer babies," off-target edits, access and equity, unintended consequences.

Interactive Session (2 hours)

- **CRISPR Animation & Discussion (60 minutes):**
  - Watch short animated videos explaining CRISPR mechanism.
  - Group discussion: "How is gene editing different from traditional medicine?" "What are the most exciting applications of CRISPR?"
- **Ethical Dilemma Scenarios (45 minutes):**
  - Present hypothetical scenarios involving CRISPR (e.g., editing genes for disease prevention vs. enhancement).
  - Students work in groups to debate the ethical pros and cons of each scenario.
  - Class discussion on the "slippery slope" argument and the role of regulation.
- **Q&A / Future of Biotech (15 minutes):**
  - If guest speaker: Q&A session.
  - Otherwise: Discussion on careers in biotechnology, genetic engineering, medicine, and bioethics.
- **Assessment:** Active participation in discussions, demonstrated understanding of CRISPR's basic function, engagement with ethical dilemmas.

5.6. Seminar 6: Neuralink Brain-Computer Interface: Connecting Minds to Machines

Learning Objectives

- **Understand** the fundamental concept of Brain-Computer Interfaces (BCIs) and Neuralink's approach.
- **Recognize** the potential of BCIs in medical applications (e.g., restoring movement,

communication).

- **Explore** the profound ethical and philosophical implications of directly interfacing the human brain with technology.

Materials

- **Teaching Aids:** Neuralink presentation videos, diagrams of brain anatomy (simplified) and neural activity, articles on BCI research, ethical frameworks for human enhancement.
- **Guest Speaker (Optional):** Neuroscientist, biomedical engineer, neurologist, or philosopher specializing in technology ethics.

Lecture (1 hour)

- **Introduction to the Brain (10 minutes):** Brief overview of neurons, brain signals (electrical impulses).
- **What is a BCI? (15 minutes):** Explain how BCIs work: reading brain signals, translating them into commands, and sending them to a device. Different types (invasive vs. non-invasive).
- **Neuralink's Vision (25 minutes):** Focus on their implantable device, "threads," and the goal of direct neural communication. Showcase their demo videos (e.g., monkeys playing Pong).
- **Medical Applications (5 minutes):** Helping paralysis patients, restoring sight/hearing, treating neurological disorders.
- **Ethical and Philosophical Dilemmas (5 minutes):** Privacy of thought, identity, human augmentation, equitable access.

Interactive Session (2 hours)

- **Neuralink Demo & Discussion (60 minutes):**
  - Watch Neuralink demonstration videos. Discuss the technology shown and its immediate implications.
  - Group discussion: "If you could connect your brain to a device, what would you want it to do?" "What are the biggest risks?"
- **Ethical Debate: Augmentation vs. Therapy (45 minutes):**
  - Present scenarios: BCI for restoring lost function vs. BCI for enhancing cognitive abilities.
  - Students debate the ethical boundaries and societal impact of each.
  - Discuss the concept of "digital immortality" and what it might mean for human identity.
- **Q&A / Future of Humanity (15 minutes):**
  - If guest speaker: Q&A session.
  - Otherwise: Open discussion on the long-term impact of BCIs on human evolution, society, and our understanding of consciousness.
- **Assessment:** Active participation in discussions, demonstrated understanding of BCI concepts, thoughtful engagement with ethical implications.