**Software Development Lifecycle (SDLC)**

**What is SDLC?**

The Software Development Lifecycle (SDLC) is a structured process that guides a development team in creating high-quality software. This team can include:

* **Client-facing personnel** (liaison between clients and developers)
* **Developers ("Code monkeys")** (write and maintain code)
* **Project management** (oversees development progress and timelines)

**Two Main Approaches:**

1. **Waterfall Model** – A structured, sequential approach
2. **Agile Approach** – A more flexible, iterative approach

**Waterfall Model (Sequential Process)**

Each phase must be completed before moving to the next.

**1. Define Requirements to be met.**

* Conduct interviews with clients
* Create requirements:
  + Project timeline and cost estimates
  + Technical requirements (e.g., Windows 11, macOS compatibility)
  + Functional requirements

**2. Design Software**

* **User Interface (UI) Design** – Visual aspects of the software
* **Logic & Algorithms** – Internal workings of the software
* **Flowcharting** – Visual representation of software processes

**3. Development (Programming)**

* Use **version control** (e.g., Git)
* Select the most suitable programming language
* Team-based coding approach

**4. Testing**

* Detect and fix bugs before deployment

**5. Deployment (Releasing Software)**

* Decide on release strategy:
  + **Phased Release** – Rolling out in stages (e.g., country by country)

**6. Maintenance**

* Ensure compatibility across different platforms
* Adapt to changing requirements (new features, bug fixes)

**Pros & Cons of Waterfall**

✅ **Pros:**

* Well-defined project phases
* Clear documentation
* Easier to estimate costs and timelines
* Reduces scope creep (clients continuously requesting additional features)

❌ **Cons:**

* Requires extensive upfront planning
* Difficult to make changes once a phase is completed
* Higher risk due to limited flexibility

💡 **Common Users of Waterfall:**

* Government agencies
* Large organizations (e.g., banks) that require highly secure and complete software

**Agile Approach (Iterative Process)**

**How Agile Works**

Agile development follows an iterative process where teams work in short cycles called **sprints**. At the end of each sprint, the team:

* Reviews progress
* Shows the latest version to the client
* Assesses feedback
* Starts the next sprint

**Key Features of Agile:**

* **Sprints** – Short, focused development cycles (typically weekly)
* **Daily Check-ins** – Brief meetings to update progress and address challenges
* **Prototyping** – Frequent iterative versions to refine the software
* **Specifications & Requirements** – Flexible targets that evolve with client needs
* **Continuous Testing** – Ongoing testing throughout development to quickly fix issues

**Full Stack Development**

A **Full Stack Developer** works on both the front end (user interface) and backend (server, database, logic).

**1. Frontend Development**

* Focuses on **UI (User Interface) & UX (User Experience)**
* Technologies: HTML, CSS, JavaScript

**2. Backend Development**

* Handles the logic and database interactions
* Technologies: Python, JavaScript (Node.js), PHP

**3. Database Management**

* Uses SQL (Structured Query Language) for data storage and retrieval
* Supports dynamic applications that react to user inputs

**Reminder for Using ChatGPT to Order Notes:**

✅ **Be Specific** – Clearly outline what you need formatted or explained

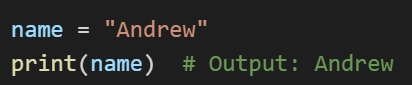
**Data Types in Programming**

**What Are Data Types?**

Data types define how information is stored in a computer's memory. They determine the type of data a variable can hold and how it can be used in a program.

Variables can store different data types and can be retrieved by referencing their variable names.

**Example:**



**Common Data Types and Their Uses**

**1. Integer (int)**

* Stores whole numbers (positive or negative)
* Used for counting and calculations
* **Example:**

A black background with blue text

AI-generated content may be incorrect.

**2. Floating Point (float)**

* Stores decimal numbers (positive or negative)
* Used for precise calculations (e.g., financial transactions)
* **Example:**

A black background with blue and white text

AI-generated content may be incorrect.

**3. String (str)**

* Stores text data
* Enclosed in either single (') or double (") quotes
* **Example:**

A black background with orange and blue text

AI-generated content may be incorrect.

**String Indexing**

* Strings are stored as an **array of characters**, where each character has a specific index.
* **Example:**

A screenshot of a computer program

AI-generated content may be incorrect.

**4. Boolean (bool)**

* **Used for logical operations, often in if statements**
* **Represents True or False (binary: 1 for True, 0 for False)**
* **Example:**

**A black background with blue text

AI-generated content may be incorrect.**

**Summary**

* **Integers are whole numbers.**
* **Floats are decimal numbers.**
* **Strings store text and are indexed character by character.**
* **Booleans are used for logical operations (True/False).**
* **Default Data Types as Classes**
* **In Python, default data types (e.g., str, int, float) are seen as classes.**
* **Each data type has methods attached that allow interactions.**
* **Developers can create their own custom data types using classes, allowing customized interactions.**
* **Arithmetic Operators in Python**
* **A screenshot of a computer

  AI-generated content may be incorrect.Arithmetic operators perform mathematical operations on numeric data types.**
* **Default data types are classes and can be extended using custom classes.**
* **Arithmetic operators allow mathematical operations on numbers.**

**Control Structures**

Control structures dictate the flow of execution in a program. The three main types are:

**1. Sequences**

* Instructions are executed in a linear, step-by-step manner.

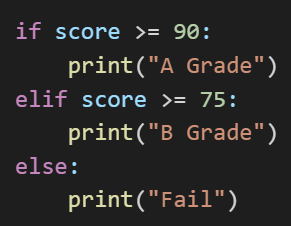
**2. Conditionals (Decision Making)**

* Used to execute specific blocks of code based on conditions.
* **If-Else Statements:** Execute one block of code if a condition is true; another if it is false.

A screenshot of a black background with white text

AI-generated content may be incorrect.

* **Elif (Else-If) Statements:** Allow multiple conditions to be checked sequentially.



**3. Iteration (Loops)**

* Repeats a block of code multiple times.
* **For Loops:** Used for counted iteration.

A screen shot of a computer code

AI-generated content may be incorrect.

* **While Loops:** Executes as long as a condition is true (pre-test loop).

A black background with colorful text

AI-generated content may be incorrect.

**Repeat Until Loops (Post-Test Loop)**

* + Runs at least once and repeats until the condition is met.
  + Not available in Python but present in some other languages.

A black background with blue text

AI-generated content may be incorrect.

**Summary - Control Structures** manage the flow of a program: Sequences, Conditionals, and Loops.

**Pseudocode**

Pseudocode is an informal way of describing an algorithm using simple, human-readable statements that resemble programming concepts but do not follow any strict syntax.

* Used for planning algorithms before coding.
* Able to be translated into other code languages
* Helps in understanding logic without focusing on syntax errors.

Example: (pseudocode)

A screenshot of a computer screen

AI-generated content may be incorrect.

**Summary - Pseudocode** is a simplified, structured way to represent algorithms.

**Flowcharts**

* A **visual representation** of an algorithm’s logic and process flow.
* Uses standard symbols:
  + **Oval**: Start/End
  + **Parallelogram**: Input/Output
  + **Rectangle**: Process
  + **Diamond**: Decision

Flowcharts help in better understanding and debugging of complex logic.

A blue rectangle with black text

Description automatically generated

**Summary** - **Flowcharts** visually represent logic and processes.

**Data Structures**

**What is a Data Structure?**

A **data structure** is a format for organizing, storing, and processing data efficiently in a computer system. Data structures allow operations such as searching, sorting, finding maximum and minimum values, and more.

**1. Arrays**

* An **array** is a collection of elements, all of the same data type, stored in contiguous memory locations.
* In most programming languages, arrays are zero-indexed, meaning the first element is at index 0.

**Example:**

**Array Declaration in Different Languages**

**Python:**



**Finding the Maximum Value in an Array**

A simple algorithm to find the largest number in an array:

**A computer screen with white text

AI-generated content may be incorrect.**

**How It Works (Step-by-Step Execution)**

**A screenshot of a black table

AI-generated content may be incorrect.**

**2. Two-Dimensional Arrays**

**A 2D array is an extension of a one-dimensional array where data is stored in a grid format (rows and columns).**

**A screenshot of a black and white grid

AI-generated content may be incorrect.**

**Referencing Values in a Grid**

* Nums[0,2] → 8
* Nums[2,2] → 3

**Writing a 2D Array in Code**

Pseudocode representation

****

Python representation

A black background with white text and numbers

AI-generated content may be incorrect.

**Looping Through a 2D Array**

**Displaying an Entire Row**

FOR row = 0 TO 2 DO

FOR col = 0 TO 2 DO

DISPLAY nums[row, col]

ENDFOR

ENDFOR

**S**