Modul-1 : Overview of IT Industry - practical

Write a simple "Hello World" program in two different programming languages of your choice. Compare the structure and syntax.

→

1. Python

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

2. C

```
#include <stdio.h>

int main() {
    printf("Hello, World!\n");
    return 0;
}
```

What are the key steps involved in the programming process? Types of Programming Languages.

→ Key Steps in the Programming Process

The **programming process** is like solving a puzzle with a step-by-step plan. Here are the main steps:

1. Understand the Problem

- Clearly define what needs to be done.
- Ask: What is the goal? What input do I have? What should the output be?

2. Plan the Solution (Design)

- Break the problem into smaller steps.
- Use tools like **flowcharts** or **pseudocode** to map out logic.

3. Write the Code (Implementation)

- Use a **programming language** to write the instructions.
- Focus on structure, logic, and readability.

4. Test the Program

- Run the program with different inputs.

- Check if it behaves as expected.

5. Debug (Fix Errors)

- Identify and correct any mistakes or bugs in the code.
- Use error messages, print statements, or debugging tools.

6. Refine and Optimize

- Improve code efficiency and readability.
- Remove unnecessary parts, add comments, or refactor logic.

7. Document and Maintain

- Write documentation for users or future developers.
- Update and fix the code over time as needed.

Types of Programming Languages

Programming languages are tools used to write software. There are several types, categorized based on their features and use cases:

1. Low-Level Languages

- **Machine Language**: Binary code (0s and 1s) that the computer understands directly.
- **Assembly Language**: A step above machine code, with symbolic names for operation

---

2. High-Level Languages

- Easier for humans to read and write.
- Abstract away hardware details.

Examples:

- **Python**
- **Java**
- **C++**
- **JavaScript**
- **Ruby**

Research and create a diagram of how data is transmitted from a client to a server over the internet.

1. Client Initiation & DNS Lookup

- The user enters a URL (e.g., www.example.com) into a browser, which sends a request to a **DNS server** to resolve the domain into an **IP address**.GeeksforGeeksMedium

2. Establishing a TCP Connection

- The browser initiates a **TCP three-way handshake** with the server:
  1. **SYN** from client
  2. **SYN-ACK** from server
  3. **ACK** from client
     This establishes a reliable, ordered communication channel.WikipediaMedium

3. (Optional) TLS/SSL Handshake

- If the request uses **HTTPS**, the client and server perform a **TLS handshake** to establish secure, encrypted communication.DEV CommunityWikipedia

4. Sending the HTTP Request

- The client sends an **HTTP (or HTTPS) request** to the server, asking for a resource (e.g., a webpage).Wikipediadocsallover.com

5. Packetization & Routing

- The data is broken into **packets**, each with headers (source, destination, sequence numbers, etc.). These packets traverse the internet through **routers**, **switches**, and various networks toward the server.GeeksforGeeksblogs.brain-mentors.com
- If a link fails or is busy, the packets are re-routed dynamically via alternate paths.GeeksforGeeks

6. Server Processing

- The server reassembles packets into requests, processes them (e.g., fetches data from the application or database), and constructs an **HTTP response** with the resource data and headers.docsallover.comLifewire

7. Response Transmission

- The server's response packets are sent back to the client following similar routing logic.

8. Client Renders the Response

- The browser reconstructs the response and renders the requested content (such as a webpage).docsallover.com

9. Connection Teardown

- Once the data exchange is complete (especially in HTTP/1.0 or default TCP scenarios), the connection is closed using TCP termination steps

Research and create a diagram of how data is transmitted from a client to a server over the internet.
→

Research and Comparison

Type	Pros	Cons
Broadband (DSL/Cable)	Widely available, good speeds	Can slow during peak hours
Fiber-optic	Very high speed, reliable	Limited availability, costly
Satellite	Available in remote areas	High latency, weather-affected
Mobile (4G/5G)	Portable, fast (5G)	Data caps, signal issues

Research different types of internet connections (e.g., broadband, fiber, satellite)and list their pros and cons.

Broadband vs. Fiber

- **Broadband** uses copper cables (DSL/Cable); it's slower and more prone to interference.
- **Fiber-optic** uses light signals in glass fibers; it’s faster, more stable, and future-proof.

Design a simple HTTP client-server communication in any language.

→ Client-Server Communication

Client-server communication refers to the exchange of data where the **client initiates a request** and the **server processes and responds**. Types include:

- **HTTP** (web pages)
- **FTP** (file transfers)
- **SMTP/IMAP** (emails)
- **WebSocket** (real-time communication)

Identify and explain three common application security vulnerabilities. Suggestpossible solutions.

→ **Identify and Fix 3 Vulnerabilities**

Vulnerability	Description	Solution
SQL Injection	**Attacker inserts SQL code via input**	**Use prepared statements / ORM**
Cross-Site Scripting (XSS)	**Injects malicious script into web page**	**Sanitize user inputs**
Broken Authentication	**Weak login mechanisms**	**Use multi-factor authentication, strong password policy**

Identify and classify 5 applications you use daily as either system software orapplication software.

→ **Classify 5 Applications You Use Daily**

Application	Type
Google Chrome	Application Software
Windows 11	System Software
Microsoft Word	Application Software
Antivirus (e.g., Avast)	Utility Software
File Explorer	System Software

- **Design a basic three-tier software architecture diagram for a web application.**
  →

```
+-------------------------+
|     Presentation Layer  |
| (HTML, CSS, JavaScript) |
+-------------------------+
          ↓
+-------------------------+
|     Business Logic      |
|   (Python, Java, Node)  |
+-------------------------+
          ↓
+-------------------------+
|     Data Access Layer   |
| (SQL, MongoDB, Firebase)|
+-------------------------+
```

- **Create a case study on the functionality of the presentation, business logic, and dataaccess layers of a given software system.**

  - **Case Study Example (E-commerce Website)**

Layer	Function
Presentation Layer	User interface (product pages, shopping cart UI)
Business Logic Layer	Handles checkout, payment processing, discounts
Data Access Layer	Connects to the database to fetch products, orders

- **Explore different types of software environments (development, testing, production).Set up a basic environment in a virtual machine.**
  →

  1. **Development Environment** – Where code is written (IDE, local server).
  2. **Testing Environment** – Used for QA and bug testing (staging server).
  3. **Production Environment** – Live environment used by end users.

- **Write and upload your firstsource code file to Github.**

→

```
# 1. Create project folder

mkdir hello-world

cd hello-world



# 2. Add your source code file

echo 'print("Hello, World!")' > hello.py



# 3. Initialize Git

git init



# 4. Add your file

git add hello.py
```

```
# 5. Commit your changes

git commit -m "Add hello world program"


# 6. Connect to your GitHub repo

git remote add origin https://github.com/yourusername/hello-world.git


# 7. Push the code to GitHub

git push -u origin master
```

- **Create a Github repository and document how to commit and push code changes**.

→

```
# Set up Git (only once)

git config --global user.name "Your Name"

git config --global user.email "your@email.com"


# Project setup

mkdir my-first-repo

cd my-first-repo

git init

echo 'print("Hello, GitHub!")' > hello.py

git add hello.py

git commit -m "Initial commit: Add hello.py"

git remote add origin https://github.com/yourusername/my-first-repo.git

git push -u origin main
```

- **Create a student account on Github and collaborate on a small project with aclassmate.**

→ How Git Improves Collaboration

- Tracks individual contributions and changes.
- Supports **branching**, allowing team members to work independently.
- Simplifies **merging** and **conflict resolution**.
- Maintains a history of all project changes (audit trail).
- Enables **pull requests** and code reviews.

- **Create a list of software you use regularly and classify them into the followingcategories: system, application, and utility software.**

→ 1. System Software

These control and manage the hardware and basic system operations.

Software Name	Description
Windows 11 / macOS / Linux	Operating system – core system manager
Device Drivers	Interface between OS and hardware (e.g., printer, keyboard)
BIOS/UEFI	Firmware used to boot your computer
File System Manager	Manages data storage on drives

---

2. Application Software

These are the programs you use to do specific tasks (like writing, browsing, editing).

Software Name	Purpose
Microsoft Word / Google Docs	Word processing
Chrome / Firefox / Edge	Web browsing

Software Name	Purpose
Spotify / VLC Media Player	Music/video playback
Zoom / Microsoft Teams	Video conferencing
Adobe Photoshop / GIMP	Image editing
VS Code / PyCharm	Programming and code editing
Excel / Google Sheets	Spreadsheets

---

3. Utility Software

These help maintain, analyze, and optimize your system.

Software Name	Function
Windows Defender / Avast	Antivirus and security
CCleaner	Junk file and registry cleaner
Disk Cleanup / Disk Utility	Free up space, manage disks
WinRAR / 7-Zip	File compression and extraction
Backup Software (e.g., Time Machine, Acronis)	Data backup and recovery
Task Manager / Activity Monitor	Monitor system performance

Follow a GIT tutorial to practice cloning, branching, and merging repositories

→

1. Clone a Repository

Cloning means making a local copy of a remote repo.

Steps:

1. Pick a repo to clone. Let's use this example repo:

https://github.com/octocat/Hello-World.git

2. Open your terminal and run:

```
git clone https://github.com/octocat/Hello-World.git
cd Hello-World
```

Now you have the entire project locally.

---

2. Create and Switch to a New Branch

Branches let you work on different features without affecting the main codebase.

```
git checkout -b feature-branch
```

This creates a new branch called `feature-branch` and switches to it.

---

3. Make Changes and Commit

Create or edit a file, for example:

```
echo "This is a test file" > test.txt
```

Add and commit your changes:

```
git add test.txt
git commit -m "Add test.txt with sample content"
```

---

4. Switch Back to Main Branch

```
git checkout main
```

---

5. Merge Your Feature Branch Into Main

```
git merge feature-branch
```

This merges the changes you made in `feature-branch` back into `main`.

---

6. Push Changes to Remote (Optional)

If you have write access to the repo or your own fork:

```
git push origin main
```

Write a report on the various types of application software and how they improveproductivity.

→ Types of Application Software and Their Impact on Productivity

---

Introduction

Application software refers to programs designed to perform specific tasks for users, helping them achieve goals efficiently. Unlike system software, which manages hardware and basic system operations, application software is focused on enhancing user productivity in various domains, from office work to creative design and communication.

This report explores common types of application software and explains how they contribute to improved productivity in both personal and professional contexts.

Types of Application Software

1. Productivity Software

These tools help users perform general tasks that enhance productivity in office and business environments.

- **Examples:** Microsoft Office (Word, Excel, PowerPoint), Google Workspace (Docs, Sheets, Slides)
- **How it improves productivity:**
  - Enables document creation, editing, and sharing with ease.
  - Facilitates data organization and analysis through spreadsheets.
  - Supports professional presentations and collaboration.
  - Cloud-based options allow real-time teamwork and access from anywhere.

---

2. Communication Software

Software that facilitates interaction and information exchange among users.

- **Examples:** Email clients (Outlook, Gmail), Messaging apps (Slack, Microsoft Teams), Video conferencing (Zoom, Google Meet)
- **How it improves productivity:**
  - Enables instant communication, reducing delays.
  - Supports remote and distributed teamwork.
  - Integrates with other productivity tools to streamline workflows.
  - Improves coordination and decision-making through quick feedback.

---

3. Database Management Software

Programs that allow users to create, manage, and manipulate databases.

- **Examples:** MySQL, Oracle Database, Microsoft Access
- **How it improves productivity:**
  - Organizes large volumes of data efficiently.
  - Enables quick retrieval and analysis of information.
  - Supports automation of repetitive data tasks.
  - Enhances data accuracy and integrity.

---

4. Graphic Design and Multimedia Software

Tools used for creating and editing images, videos, animations, and other multimedia content.

- **Examples:** Adobe Photoshop, Illustrator, Premiere Pro, Canva
- **How it improves productivity:**
  - Provides powerful features for professional content creation.
  - Streamlines editing processes with intuitive interfaces.
  - Supports collaboration through file sharing and version control.
  - Enables faster turnaround times for marketing and creative projects.

---

5. Project Management Software

Applications that help plan, execute, and monitor projects.

- **Examples:** Trello, Asana, Microsoft Project, Jira
- **How it improves productivity:**
  - Helps organize tasks and deadlines.
  - Enhances team collaboration and accountability.
  - Tracks progress and resource allocation.
  - Provides insights through reports and analytics for better decision-making.

---

6. Web Browsers

Software for accessing and interacting with the internet.

- **Examples:** Google Chrome, Mozilla Firefox, Safari, Microsoft Edge
- **How it improves productivity:**
  - Provides access to vast online resources and tools.
  - Supports web-based applications essential for modern work.
  - Offers extensions and plugins to customize and automate tasks.
  - Facilitates research, communication, and cloud-based work.

Create a flowchart representing the Software Development Life Cycle (SDLC)

❖ SDLC Flowchart

→

[Requirement] → [Design] → [Development] → [Testing] → [Deployment] → [Maintenance]

Write a requirementspecification for a simple library managementsystem.

→ Sample Library Management System Requirements

- Users: Admin, Librarian, Member
- Features:
  - Add/remove books
  - Borrow/return books
  - Search catalog
  - Track overdue items
  - Generate reports

- **Perform a functional analysis for an online shopping system.**

→ Functional Analysis – Online Shopping System

Function	Description
User Registration	Create account
Product Browsing	Search/filter items
Add to Cart	Select products
Checkout	Process payment
Order Tracking	Monitor delivery status

- **Design a basic system architecture for a food delivery app.**

→ Food Delivery App – System Architecture

```
+--------------------------+
|   User Interface (App)   |
+--------------------------+
           ↓
+--------------------------+
|  Business Logic (APIs)   |
+--------------------------+
           ↓
+--------------------------+
|  Database (Orders, Menu) |
+--------------------------+
```

- **Develop test cases for a simple calculator program.**

→ Test Cases – Calculator Program

Test Case	Input	Expected Output
Addition	**2 + 3**	**5**
Division	**10 ÷ 2**	**5**
Division by zero	**5 ÷ 0**	**Error**
Multiplication	**4 × 2**	**8**

- **Document a real-world case where a software application required criticalmaintenance.**

→ Real-World Case – Windows 10 Critical Patch (2021)

- Microsoft released a critical patch for a vulnerability (PrintNightmare).
- This maintenance update fixed a security flaw that allowed remote code execution.

Create a DFD for a hospital managementsystem.

→

DFD for Hospital Management System (Level 0 - Context Diagram)

```
[Patient] → (Registers) → [Hospital Management System] ← (Schedules/Checks) ←
[Doctor]
                                    |
                                    ↓
                                [Database]
```

Build a simple desktop calculator application using a GUI library.

→ import tkinter as tk

```
# Create the main window

window = tk.Tk()

window.title("Simple Calculator")

window.geometry("300x400")


# Entry widget to display the input/output

entry = tk.Entry(window, font=("Arial", 20), borderwidth=2, relief="solid", justify="right")

entry.grid(row=0, column=0, columnspan=4, ipadx=10, ipady=20, padx=10, pady=10)


# Function to handle button click

def on_click(button_text):

    current_text = entry.get()
```

```
    if button_text == "=":
        try:
            result = eval(current_text)
            entry.delete(0, tk.END)
            entry.insert(tk.END, str(result))
        except:
            entry.delete(0, tk.END)
            entry.insert(tk.END, "Error")
    elif button_text == "C":
        entry.delete(0, tk.END)
    else:
        entry.insert(tk.END, button_text)


# Button layout
buttons = [
    ["7", "8", "9", "/"],
    ["4", "5", "6", "*"],
    ["1", "2", "3", "-"],
    ["0", ".", "=", "+"],
    ["C"]
]


# Create and place buttons
for i, row in enumerate(buttons):
    for j, btn_text in enumerate(row):
```

```
    btn = tk.Button(
        window,
        text=btn_text,
        width=5,
        height=2,
        font=("Arial", 18),
        command=lambda text=btn_text: on_click(text)
    )
    btn.grid(row=i+1, column=j, padx=5, pady=5, sticky="nsew")


# Run the application
window.mainloop()
```

Draw a flowchart representing the logic of a basic online registration system.

→ Flowchart for Online Registration System

```
[Start]
  ↓
[User visits registration page]
  ↓
[Enter details]
  ↓
[Validate input]
  ↓
┌──────────────┐
│Valid inputs? │
└──────┬───────┘
```

↓Yes ↓No

[Store data] [Show error message]

↓

[Send confirmation email]

↓

[End]