Python Day-1

Introduction to computer systems, Algorithms and flowcharts, History and application areas of Python, Features of Python, Setting up Python and IDEs, Understanding the Python programming cycle, Simple Programming Problems.

Introduction to Computer Systems

This is a basic outline and notes on the fundamental concepts of computer systems designed for first-year undergraduate students.

1. What is a Computer System?

- A **computer system** is a combination of hardware and software that processes data to perform specific tasks.
- **Components of a Computer System:**
    1. **Hardware**: The physical components of a computer (CPU, memory, storage, input/output devices).
    2. **Software**: Programs and operating systems that instruct the hardware on what tasks to perform.
    3. **User**: The individual who interacts with the computer system.

---

2. Components of a Computer System

1. **Hardware**
    - **Central Processing Unit (CPU)**: Executes instructions and processes data.
        - **Control Unit (CU)**: Directs operations within the computer.
        - **Arithmetic Logic Unit (ALU)**: Performs arithmetic and logical operations.
    - **Memory**:
        - **Primary Memory** (RAM, ROM): Stores data and instructions temporarily.
        - **Secondary Memory** (Hard drives, SSDs): Permanent storage of data.
    - **Input Devices**: Keyboard, mouse, microphone, etc.
    - **Output Devices**: Monitor, printer, speakers, etc.
    - **Storage Devices**: USB drives, optical disks, etc.
2. **Software**
    - **System Software**: Operating systems (Windows, macOS, Linux), utilities.
    - **Application Software**: Word processors, web browsers, games.
    - **Programming Software**: Tools like compilers, interpreters, and debuggers.

3. **Data**
    - Raw facts and figures processed into meaningful information.
    - Represented in binary (0s and 1s).
4. **Users**
    - **End-users** interact with the system to perform tasks.
    - Developers and system administrators maintain and enhance systems.

---

3. Types of Computer Systems

1. **Personal Computers (PCs)**: Used for general-purpose tasks (home, office).
2. **Servers**: Provide services to other computers or devices in a network.
3. **Embedded Systems**: Special-purpose computers embedded in devices (e.g., washing machines, ATMs).
4. **Mainframes and Supercomputers**: Handle large-scale data processing and complex computations.

---

4. Working of a Computer System

- **Input**: Data is entered into the system via input devices.
- **Processing**: The CPU processes the data based on instructions from the software.
- **Storage**: Data is stored temporarily in RAM or permanently in secondary storage.
- **Output**: The processed data is presented to the user via output devices.

---

5. Representation of Data in Computers

- **Binary System**: Computers use the binary number system (0s and 1s).
- **Bits and Bytes**:
    - 1 bit = smallest unit of data (0 or 1).
    - 1 byte = 8 bits (e.g., 10101100).
- **Data Units**: Kilobyte (KB), Megabyte (MB), Gigabyte (GB), Terabyte (TB).

---

6. Operating Systems (OS)

- A critical system software that manages hardware, software, and user interaction.
- Functions:
    1. **Process Management**: Scheduling and execution of processes.
    2. **Memory Management**: Allocating and freeing memory.
    3. **File System Management**: Organizing and accessing files.

4. **Device Management**: Controlling input/output devices.
5. **Security and Access Control**: Protecting data and resources.

- Examples: Windows, Linux, macOS, Android.

---

7. Programming and Software Development

- **Programming**: Writing instructions (code) for the computer to execute tasks.
- **Programming Languages**:
  - **Low-level languages**: Assembly, Machine Language.
  - **High-level languages**: Python, C++, Java.
- **Compiler vs. Interpreter**:
  - **Compiler**: Translates the entire program into machine code before execution.
  - **Interpreter**: Translates code line-by-line during execution.

---

8. Computer Networks

- **Definition**: A group of interconnected computers that share resources.
- **Types of Networks**:
  1. **LAN (Local Area Network)**: Small geographic area (e.g., office).
  2. **WAN (Wide Area Network)**: Large geographic area (e.g., Internet).
  3. **MAN (Metropolitan Area Network)**: Covers a city.
- **Internet**: The largest WAN connecting millions of networks globally.

---

9. Evolution of Computers

- **Generations of Computers**:
  1. **First Generation** (1940-1956): Vacuum tubes, large and slow.
  2. **Second Generation** (1956-1963): Transistors replaced vacuum tubes.
  3. **Third Generation** (1964-1971): Integrated Circuits (ICs).
  4. **Fourth Generation** (1971-Present): Microprocessors and personal computers.
  5. **Fifth Generation** (Present and Beyond): AI and quantum computing.

---

10. Applications of Computer Systems

1. **Education**: E-learning platforms, research tools.
2. **Healthcare**: Diagnostic systems, medical records.
3. **Finance**: Online banking, stock trading.
4. **Entertainment**: Gaming, streaming services.

5. **Business**: Data analytics, ERP systems.

ALGORITHM

- Algorithm refers to the logic of the program. It is a step by step description of how to arrive at the solution of the problem.
- An algorithm is a complete, detailed and precise step by step method for solving a problem independently of the hardware and software.

Characteristics of a good algorithm are:

- **Input** – the algorithm receives input.
- **Output** – the algorithm produces output.
- **Finiteness** – the algorithm stops after a finite number of instructions are executed.
- **Precision** – the steps are precisely stated (defined).
- **Uniqueness** – results of each step are uniquely defined and only depend on the input and the result of the preceding steps.
- **Generality** – the algorithm applies to a set of inputs.

Example: Algorithm to find sum of two numbers:

1. Begin
2. Input the value of A and B
3. SUM= A+B
4. Display SUM
5. End.

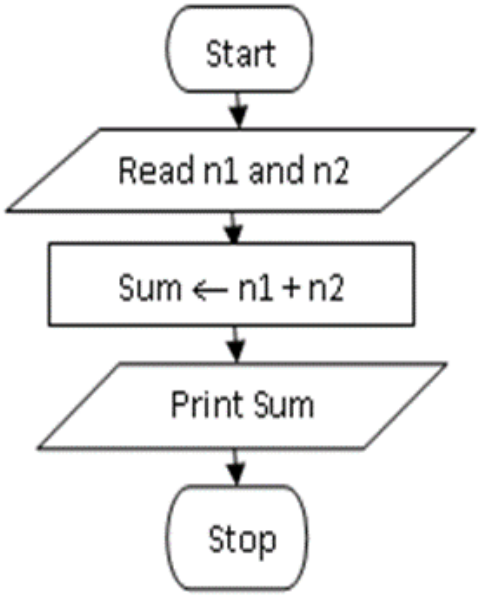
FLOWCHART

- A flowchart is a pictorial representation of an algorithm or process.

Symbols used:

Symbol	Name	Function
	Start/end	An oval represents a start or end point
	Arrows	A line is a connector that shows relationships between the representative shapes
	Input/Output	A parallelogram represents input or output
	Process	A rectagle represents a process
	Decision	A diamond indicates a decision

Example: Flow-Chart to find sum of two numbers



Questions: Write an algorithms and design flow charts for the following:

1. Find average of three numbers.
2. Swap two numbers using: i. Using third variable ii. Without using third variable
3. Find area and perimeter of a circle.
4. Check if a number is divisible by 7 or not.
5. Among three numbers: a,b and c, find the greatest number.
6. Check if a number is an Armstrong number or not.
7. Find Factorial of a number.

History of Python

- **Developed By**: Guido van Rossum.
- **First Released**: In 1991.
- **Key Philosophy**: Python was designed with readability and simplicity in mind.
- **Name Origin**: Python is named after the British comedy group "Monty Python," not the snake.
- **Key Milestones**:
    - **Python 2.0 (2000)**: Introduced list comprehensions, garbage collection via reference counting.
    - **Python 3.0 (2008)**: Major revision that addressed design flaws and enhanced features (not backward-compatible with Python 2).

Application Areas of Python

1. **Web Development**:
    - Frameworks: Django, Flask, FastAPI.
    - Building dynamic websites and APIs.
2. **Data Science and Machine Learning**:
    - Libraries: NumPy, pandas, scikit-learn, TensorFlow, PyTorch.
    - Used for data analysis, visualization, and predictive modeling.
3. **Automation/Scripting**:
    - Automating repetitive tasks and workflows.
4. **Game Development**:
    - Libraries: Pygame.
    - Building simple games.
5. **Scientific Computing**:
    - Libraries: SciPy, SymPy, Matplotlib.
    - Used for mathematical and scientific computations.
6. **Embedded Systems and IoT**:
    - Lightweight frameworks for controlling hardware.
7. **Desktop GUI Applications**:
    - Libraries: Tkinter, PyQt, Kivy.
8. **Cybersecurity and Ethical Hacking**:
    - Libraries: Scapy, Nmap.
    - Used for penetration testing and network analysis.
9. **Blockchain Development**:
    - Libraries: web3.py for working with Ethereum.
10. **Cloud Computing**:
    - Automating cloud environments (e.g., AWS with Boto3).

---

2. Features of Python

1. **Easy to Learn and Use**:
    - Python has a simple syntax, making it beginner-friendly.
2. **Open Source**:
    - Freely available and maintained by a large community.
3. **Interpreted Language**:
    - Executes code line-by-line, making debugging easier.
4. **Dynamic Typing**:
    - Variable types are inferred at runtime.
5. **Platform-Independent**:
    - Python code can run on multiple operating systems without modification.
6. **Extensive Libraries**:
    - Libraries for almost every domain (e.g., NumPy, Flask, Matplotlib).
7. **Object-Oriented**:
    - Supports object-oriented programming concepts (classes, inheritance).
8. **Scalable**:
    - Suitable for small scripts and large-scale applications.
9. **Readable Syntax**:
    - Encourages the use of proper indentation and clean code.
10. **Integrated Development Environments (IDEs)**:
    - Python supports many IDEs like PyCharm, VS Code, and Jupyter Notebook.

Setting Up Python and IDEs

Installing Python

1. **Download Python**:
    - Go to the official website: https://www.python.org/.
    - Download the latest version for your operating system.
2. **Install Python**:
    - Follow the installation wizard and ensure the "Add Python to PATH" option is selected.
3. **Verify Installation**:

    Open a terminal or command prompt and type:

    ```
    python --version
    ```

or

```
python3 --version
```

Popular IDEs for Python

1. **IDLE**:
   - Comes pre-installed with Python.
   - Lightweight, suitable for beginners.
2. **PyCharm**:
   - Advanced IDE with features like debugging, testing, and version control.
3. **Visual Studio Code (VS Code)**:
   - Lightweight and extensible editor with Python plugins.
4. **Jupyter Notebook**:
   - Ideal for data science and interactive computing.
5. **Spyder**:
   - Popular among scientific computing users.
6. **Thonny**:
   - Beginner-friendly IDE with simple debugging tools.

---

4. Understanding the Python Programming Cycle

Python Development Workflow

1. **Writing the Code**:
   - Create a `.py` file using an editor or IDE.

     Example:
     ```
     print("Hello, World!")
     ```
2. **Saving the File**:
   - Save the file with a `.py` extension (e.g., `hello.py`).
3. **Running the Code**:
   - Use a terminal or IDE to execute the code:

     In a terminal:
     ```
     python hello.py
     ```
   - In an IDE: Use the "Run" button.
4. **Debugging**:
   - Use tools or manual debugging to fix errors and exceptions.

Example of a common error:

```
print("Hello, World!)  # Missing closing quote
```

Fix:

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

5. **Testing**:
   - Test your code with various inputs to ensure reliability.

Example:

```
def add(a, b):
    return a + b
print(add(2, 3))  # Output: 5
```

6. **Iterating**:
   - Modify and refine the code based on requirements or errors.

Interpreted Nature of Python

- Python does not require compilation.
- Code is executed line-by-line by the Python interpreter.

Example:

```
a = 5
b = 10
print(a + b)  # Outputs: 15
```

Different ways to use the `print()` statement in Python.

Printing string	print("Hello, World!")
Printing Variables	name = "Alice" age = 25 print(name) print(age)
Printing Multiple Values	name = "Alice" age = 25 print("Name:", name, "Age:", age)

Using f-strings (Formatted String Literals) (Python 3.6 and above)	name = "Alice" age = 25 print(f"Name: {name}, Age: {age}")
Using `.format()` Method	name = "Alice" age = 25 print("Name: {}, Age: {}".format(name, age))
Printing Formatted Numbers	number = 1234567.89123 print(f"Formatted Number: {number:.2f}")
Using String Concatenation	name = "Alice" print("Hello, " + name + "!")
Using `sep` Parameter	print("Apple", "Banana", "Cherry", sep=", ") print("NIET", "Greater", "Noida", sep="-")
Using `end` Parameter	print("Hello", end=" ") print("World!")
Printing Escape Sequences	print("Line1\nLine2") # Newline
Printing Raw Strings Use an **r** or **R** prefix to print raw strings without interpreting escape sequences.	print(r"Path: C:\Users\Alice\Documents")

Basic Programs:

1. Program to Print "Hello, World!"

```
# we write comments here..
print("Hello, World!")
```

2. Program to Add Two Numbers

```
# Taking input from the user
num1 = float(input("Enter the first number: "))
num2 = float(input("Enter the second number: "))
```

```
# Performing operations
addition = num1 + num2

# Displaying results
print("Addition:", addition)
```

3. Program to Swap Two Numbers Using a Temporary Variable

```
# Taking input from the user
a = int(input("Enter the first number: "))
b = int(input("Enter the second number: "))

# Swapping using a temporary variable
temp = a
a = b
b = temp

# Displaying swapped values
print("After swapping:")
print("First number:", a)
print("Second number:", b)
```

4. Program to swap two numbers without using a third variable

```
# Taking input from the user
a = int(input("Enter the first number: "))
b = int(input("Enter the second number: "))

# Swapping without using a third variable
a = a + b
b = a - b
a = a - b

# Displaying swapped values
print("After swapping:")
print("First number:", a)
print("Second number:", b)
```

5. Program to calculate the perimeter and area of a triangle

Explanation

Perimeter:
The perimeter is the sum of all three sides of the triangle:
Perimeter=a+b+c

Area (using Heron's formula):

a. First, calculate the semi-perimeter: s=Perimeter/2
b. Then calculate the area using: Area= $\sqrt{s \cdot (s - a) \cdot (s - b) \cdot (s - c)}$

```
import math

# Taking input for the three sides of the triangle
a = float(input("Enter the first side of the triangle: "))
b = float(input("Enter the second side of the triangle: "))
c = float(input("Enter the third side of the triangle: "))

# Calculating the perimeter
perimeter = a + b + c
print("Perimeter of the triangle:", perimeter)

# Using Heron's formula to calculate the area
# Semi-perimeter (s)
s = perimeter / 2

# Area calculation
area = math.sqrt(s * (s - a) * (s - b) * (s - c))

print("Area of the triangle:", area)
```

6. Program to convert Fahrenheit to Celsius

The formula to convert Fahrenheit to Celsius is:
c = (f - 32) * 5 / 9

```
# Taking temperature input in Fahrenheit
f= float(input("Enter temperature in Fahrenheit: "))

# Conversion formula
c= (f - 32) * 5 / 9
```

```
# Displaying the result
print(f"{f}°F is equal to {c:.2f}°C")
```

7. Program to Calculate Simple Interest

```
principal = float(input("Enter the principal amount: "))
rate = float(input("Enter the rate of interest (in %): "))
time = float(input("Enter the time (in years): "))

simple_interest = (principal * rate * time) / 100

print("The simple interest is:", simple_interest)
```

8. Program to Calculate the Surface Area of a Cylinder

```
import math

r= float(input("Enter the radius of the cylinder: "))
h= float(input("Enter the height of the cylinder: "))

surface_area = 2 * math.pi * r * (r + h)

print("The surface area of the cylinder is:", surface_area)
```

9. Program to Calculate the Hypotenuse of a Right-Angled Triangle

- hypotenuse = $\sqrt{side1 \ ** \ 2 \ + \ side2 \ ** \ 2}$ for exponent(power) we use **

```
import math

side1 = float(input("Enter the first side of the triangle: "))
side2 = float(input("Enter the second side of the triangle: "))

h= math.sqrt(side1**2 + side2**2)

print("The hypotenuse of the triangle is:", h)
```

10. Program to Calculate Compound Interest

```
principal = float(input("Enter the principal amount: "))
rate = float(input("Enter the annual rate of interest (in %): "))
time = float(input("Enter the time (in years): "))

compound_interest = principal * (1 + rate / 100)**time - principal

print("The compound interest is:", compound_interest)
```

List of programs to practice (Home work)

11. Write a program to Find the Average of Three Numbers.
12. Write a program to Calculate Sum of 5 Subjects and Find Percentage.
13. Write a program to find gross salary.
14. Write a program to Calculate Area of Circle.
15. Write a program to Calculate Area of Rectangle.
16. Write a program to Calculate Area of Square.
17. Write a program to Calculate Area and Circumference of Circle.
18. Write a program to Calculate Area of Scalene Triangle.
19. Write a program to Calculate Area of Right angle Triangle.
20. Write a program to find the area of trapezium.