# Intro to Python for Computer Science and Data Science

Jay Prakash Singh Patel

September 15, 2025

# *CSE 3651: Python for Computer Science and Data Science 1*

| | | | |
|---|---|---|---|
| | **ITER, SIKSHA 'O' ANUSANDHAN**<br>**(Deemed to be University)** | | **LESSON PLAN** |
| Programme | **B.Tech.** | Academic Year | **2025-2026** |
| Department | **CSE** | Semester | **5th** |
| Credit | **3** | Grading Pattern | **2** |
| Subject Code | **CSE 3651** | | |
| Subject Name | **Python for Computer Science and Data Science 1** | | |
| Weekly Course Format | **2L - 2P** | | |
| Subject Coordinator (s) | **Dr. Amarjit Haty** | | |

## Text Books(s):

(1) Intro to Python for Computer Science and Data Science by Paul Deitel and Harvey Deitel, Pearson India.[PH[1]]

(2) The Python Workbook by Ben Stephenson, Springer Verlag.[PH[2]]

# Intro to Python®

for Computer Science and Data Science

Texts in Computer Science

Ben Stephenson

# The Python Workbook

A Brief Introduction with Exercises and Solutions

*Second Edition*

Springer

# Program Educational Objectives (PEOs) of BTech(CSE)

1. Our graduates will have successful professional careers in industry, government, academia, or non-profit organizations.
2. Our graduates will effectively lead, work, and communicate in multidisciplinary teams and apply sound engineering principles and design methodology to solve societal problems.
3. Our graduates will engage in lifelong learning in their chosen fields through higher study,through organizational participation and through participation in professional developmental activities.

# Program Outcomes (PO's)

PO1: An ability to apply knowledge of computations and mathematics appropriate to the program's student outcomes and to the discipline.

PO2: An ability to analyze a problem, and identify and define the computing requirements appropriate to its solution.

PO3: An ability to design, implement, and evaluate a computer-based system, process, component, or program to meet desired needs.

PO4: An ability to function effectively on teams to accomplish a common goal.

PO5: An understanding of professional, ethical, legal, security and social issues, and responsibilities.

PO6: An ability to communicate effectively with a range of audiences.

PO7: An ability to analyze the local and global impact of computing on individuals, organizations, and society.

PO8: Recognition of the need for and an ability to engage in continuing professional development.

PO9: An ability to use current techniques, skill, and tools necessary for computing practice.

PO10: An ability to apply mathematical foundations, algorithmic principles, and computer science theory in the modeling and design of computer-based systems in a way that demonstrates comprehension of the trade-offs involved in design choices.

PO11: An ability to apply design and development principles in the construction of software systems of varying complexity.

# Course Outcomes

Students will be able to

- CO1: Understand the fundamental principles of computer science, including hardware, software, data hierarchies, and the basic syntax and semantics of Python programming.
- CO2: Develop Python applications using control structures, functions, and string operations to solve practical problems.
- CO3: Analyze and apply Python data structures such as lists, tuples, dictionaries, and sets to manage and manipulate data efficiently.
- CO4: Utilize NumPy and Pandas libraries for data analysis, including managing arrays, Series, and DataFrames for scientific and data science applications.
- CO5: Apply string manipulation techniques and analyze patterns using regular expressions in Python to process and handle text data.
- CO6: Solve real-life problems using modular programming approaches, effective file handling, and robust exception handling techniques.

# CO-PO Articulation Matrix:

| CO | PO1 | PO2 | PO3 | PO4 | PO5 | PO6 | PO7 | PO8 | PO9 | PO10 | PO11 | PEO1 | PEO2 | PEO3 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| CO1 | 3 | 1 | 1 | | | | | | | | | 1 | | |
| CO2 | 3 | 2 | 2 | | | | | | | | | 1 | 1 | |
| CO3 | | 3 | 3 | 2 | | | | | | | | 2 | 1 | |
| CO4 | | 2 | 3 | 3 | 1 | | | | | 1 | 1 | 2 | 2 | |
| CO5 | | | 2 | 2 | 1 | | | | | 1 | 1 | 2 | 2 | |
| CO6 | | | | 2 | 2 | 1 | 1 | | | | | 2 | 3 | |
| Average Strength | 3.0 | 2.0 | 2.2 | 2.3 | 1.3 | 1.0 | 1.0 | 0 | 0 | 1.0 | 1.0 | 1.7 | 1.8 | 0 |

# Syllabus

1. Introduction to Computers and Python
2. Introduction to Python Programming
3. Control Statements and Program Development
4. Functions
5. Sequences: Lists and Tuples
6. Dictionaries and Sets
7. Array-Oriented Programming with NumPy
8. Strings: A Deeper Look
9. Files and Exceptions.

# CH-1: Introducton to Computers and Python
## 1.1 Introduction

- Python is one of the most widely used computer programming language and, according to PYPL Index, the world's most popular.
- **Software** (that is the Python instructions you write, which are also called **code** controls **hardware** (that is, computers and related devices).

# 1.2 Hardware and Software

- Computer process data under the control of sequences of instructions called **computer programs** ( or simply **programs**).
- These software programs guide the computer through ordered actions specified by the people called **programmers**.
- A computer consists of various physical devices referred to as **hardware** (such as keyboard, screen, mouse, solid-state disks, hard disk, memory, DVD drivers and processing units).

# 1.2.1 Moore's Law

- Unlike most products, hardware costs in computer and communication fields have fallen rapidly.
- Computer capacities (processing power, memory, storage) have roughly doubled every 1–2 years.
- This trend is known as **Moore's Law**, proposed by Gordon Moore in the 1960s.
- It mainly applies to:
  - Memory size for programs
  - Secondary storage for data and programs
  - Processor speed

# 1.2.2 Computer Organization

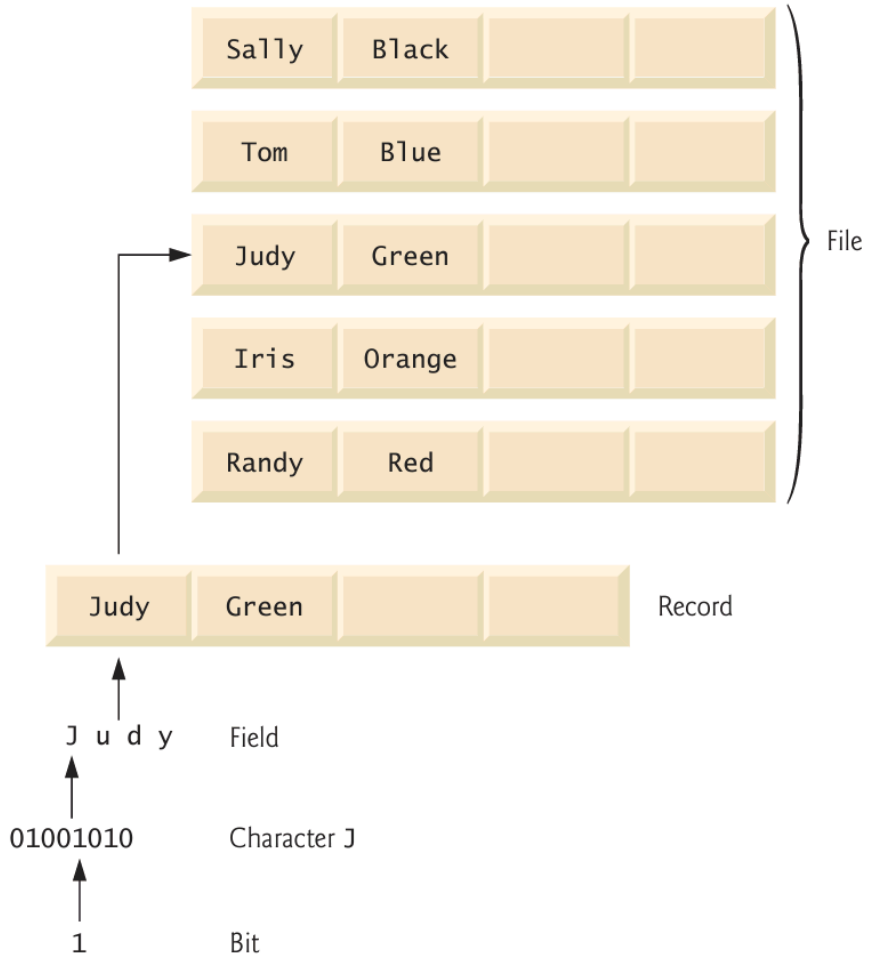
It's about how the computer hardware is structured internally and how it executes instructions.

**Basic Units:**

- **Input Unit** → Devices used to enter data (keyboard, mouse, scanner).
- **Output Unit** → Devices used to display results (monitor, printer).
- **Memory Unit** → Stores data and instructions temporarily (RAM, cache).
- **Arithmetic and Logic Unit (ALU)** → Performs calculations and logical operations.
- **Central Processing Unit (CPU)** → Controls overall operations; includes ALU + Control Unit.
- **Secondary Storage Unit** → Long-term storage of data and programs (hard disk, SSD, DVD).

# 1.3 Data Hierarchy

Data items processed by computers form a **data hierarchy** that becomes larger and more complex in structure as we progress the simplest data items (called bits) to richer ones, such as characters and fields.



## Bits

- A **bit** is the smallest data item in a computer.
- It can have value 0 or 1.
- Remarkably, the impressive functions performed by computers involve only the simplest manipulations of 0s and 1s–examining a bit's value, setting a bit's value and reversing a bit's value (from 1 to 0 or from 0 to 1).

## Characters

- Digits, letters and special symbols are known as **characters**.
- The computer's **character set** contains the characters used to write the programs and represent data items.
- Computer process only 1s and 0s, so a computer's character set represent every character as a pattern of 1s and 0s.
- Python uses **Unicode** character that are composed of one, two, three or four bytes (8, 16, 24 or 32 bits, respectively)-known as **UTF-8 encoding.**

## Fields

- A field is a group of characters or bytes that conveys meaning.
- For example, a field consisting of uppercase and lowercase letters can be used to represent a person's name, and a field consisting of decimal digits could represent a person's age.

## Records

- Several fields can be used to compose a **record**.
- In a payroll system, for example, the record for an employee might consist of following fields
  - Employee identification number (a whole number)
  - name (a string of characters)
  - address (a string of characters)
  - Hourly pay rate (a number with a decimal point)
  - Year-to-date earning (a number with decimal point)
  - Amount of taxes withheld (a number with a decimal point).
- Thus, a record is a group of related fields. All the fields listed above belong to the same employee.

## Files

- A file is a group of related records.
- In general, a file contains arbitrary data in arbitrary formats.

## Databases

- A database is a collection of data organized for easy access and manipulation.
- The most popular model is the relational database, in which data is stored in simple tables.
- A table includes records and fields.
- For example, a table of students might include first name, last name, major, year, student ID number and grade-point-average fields.
- The data for each student is a record, and the individual pieces of information in each record are the fields.
- You can search, sort and otherwise manipulate the data, based on its relationship to multiple tables or databases.

# Big Data

| Unit | Bytes | Which is approximately |
|---|---|---|
| 1 kilobyte (KB) | 1024 bytes | $10^3$ (1024) bytes exactly |
| 1 megabyte (MB) | 1024 kilobytes | $10^6$ (1,000,000) bytes |
| 1 gigabyte (GB) | 1024 megabytes | $10^9$ (1,000,000,000) bytes |
| 1 terabyte (TB) | 1024 gigabytes | $10^{12}$ (1,000,000,000,000) bytes |
| 1 petabyte (PB) | 1024 terabytes | $10^{15}$ (1,000,000,000,000,000) bytes |
| 1 exabyte (EB) | 1024 petabytes | $10^{18}$ (1,000,000,000,000,000,000) bytes |
| 1 zettabyte (ZB) | 1024 exabytes | $10^{21}$ (1,000,000,000,000,000,000,000) bytes |

- The amount of data being produced worldwide is enormous and its growth is accelerating.
- Big data applications deal with massive amounts of data.
- This field is growing quickly, creating lots of opportunity for software developers.

# 1.4 Machine Languages, Assembly Languages and High-Level Languages

- Programmers write instructions in various programming languages, some directly understandable by computers and others requiring intermediate translation step.
- Hundreds of such languages are in use today. These may be divided into three types:
  1. Machine languages
  2. Assembly languages
  3. High-level languages.

## Machine Languages

- Any computer can understand only its own machine language, defined by its hardware design.
- Machine languages generally consist of strings of numbers (ultimately reduced to 1s and 0s) that instruct computers to perform their most elementary operations one at a time.
- Machine languages are machine dependent (a particular machine language can be used on only one type of computer).

## Assembly Languages and Assemblers

- Programming in machine language was simply too slow and tedious for most programmers.
- Instead of using the strings of numbers that computers could directly understand, programmers began using English-like abbreviations to represent operations.
- These abbreviations form the basis of the assembly languages.

- Translator programs called assemblers were developed to convert assembly-language programs to machine language at computer speeds.

## High-Level Languages and Compilers

- With the advent of assembly languages, computer usage increased rapidly, but programmers still had to use numerous instructions to accomplish even the simplest task.
- To speed the programming process, high-level languages were developed in which single statement could be written to accomplish substantial tasks.
- A typical high-level language program contains many statements, known as program's source code.
- Translator programs called compilers convert high-level-language source code into machine language.
- High-level languages allow you to write instructions that look almost like everyday English and contain commonly used mathematical notations.

## Execution Process: C/C++ vs Java vs Python

- **C/C++**:
  - Source code → Compiler → Machine code (Executable).
  - Runs directly on hardware.
  - Fast execution, but platform-dependent.
- **Java**:
  - Source code (*.java*) → Compiler (javac) → Bytecode (*.class*).
  - Bytecode → Java Virtual Machine (JVM) → Execution.
  - Platform-independent (Write once, run anywhere).
- **Python**:
  - Source code (*.py*) → Python Compiler → Bytecode (*.pyc*).
  - Bytecode → Python Virtual Machine (PVM) → Execution line by line.
  - Easier debugging, but slower execution.

# 1.5 Introduction to Object Technology

- As demands for new and more powerful software are soaring, building software quickly, correctly and economically is important.
- Objects, or more precisely, the classes object come from, are essentially reusable software components.
- There are date objects, time objects, audio objects, video objects, automobile objects, people objects, etc.
- Almost any noun can be reasonably represented as a software object in terms of attributes (e.g., name, color and size) and behaviours (e.g., calculating, moving and communicating).
- Software development groups can use a modular, object-oriented design-and-implementation approach to be much more productive than with earlier popular techniques like **structured programming**.

## Automobile as an Object

## Methods and Classes

- Performing a task in a program requires a **method**.
- The method houses the program statements that perform its tasks.
- The method hides these statements from its user.
- In Python, a program unit called a class houses the set of methods that perform the class's tasks.

## Instantiation

- Just as someone has to build a car from its engineering drawings before you can drive a car, you must build an object of a class before a program can perform the task that the class's method define.
- The process of doing this is called instantiation.
- An object is then referred to as an instance of its class.

## Reuse

- Just like a car's engineering can be reused many times to build many cars, you can reuse class many times to build many objects.
- Reuse of existing classes when building new classes and programs save time and effort.

## Messages and Method Calls

- When you drive a car, pressing its gas pedal sends a message to the car to perform a task that is, to go faster. Similarly, you send messages to an object.
- Each message is implemented as a method call that tells a method of the object to perform its task.
- For example, a program might call a bank-account object's deposit method to increase the account's balance.

## Attributes and Instance Variables

- An object has attributes that it carries along as it's used in a program. These attributes are specified as part of the object's class.
- For example, a bank-account object has a balance attribute that represents the amount of money in the account. Each bank account object knows the balance in the account it represents, but not the balances of the other accounts in the bank.
- Attributes are specified by the class's instance variables.
- A class's (and its object's) attributes and methods are intimately related, so classes wrap together their attributes and methods.

## Inheritance

- A new class of objects can be created conveniently by inheritance—the new class (called the subclass) starts with the characteristics of an existing class (called the superclass), possibly customizing them and adding unique characteristics of its own.

## Object-Oriented Analysis and Design (OOAD)

- Suppose you were asked to create a software system to control thousands of automated teller machines for a major bank.
- Or, Suppose you were asked to work on a team of 1,000 software developers building the next generation of the U.S. air traffic control system.
- For projects so large and complex, you should not simply sit down and start writing programs.

- To create the best solutions, you should follow a detailed analysis process for determining your project's requirements, then develop a design that satisfies them.
- Ideally, you'd go through this process and carefully review the design (and have your design reviewed by other software professionals) before writing any code.
- If this process involves analyzing and designing your system from an object-oriented point of view, it's called an object-oriented analysis-and-design (OOAD) process.
- Languages like Python are object-oriented.
- Programming in such a language, called object-oriented programming (OOP), allows you to implement an object-oriented design as a working system.

# 1.6 Operating Systems

- Operating systems are software systems that make using computers more convenient for users, application developers and system administrators.
- They provide services that allow each application to execute safely, efficiently and concurrently with other applications.
- The software that contains the core components of the operating system is called the kernel.
- Linux, Windows and macOS are popular desktop computer operating systems.
- The most popular mobile operating systems used in smartphones and tablets are Google's Android and Apple's iOS.

# 1.7 Python

- Python is an object-oriented scripting language that was released publicly in 1991.
- It was developed by Guido van Rossum of the National Research Institute for Mathematics and Computer Science in Amsterdam.
- Python has rapidly become one of the world's most popular programming languages.
- It's now particularly popular for educational and scientific computing, and it recently surpassed the programming language R as the most popular data-science programming language.

Here are some reasons why Python is popular and everyone should consider learning it:

- It's open source, free and widely available with a massive open-source community.
- It's easier to learn than languages like C, C++, C# and Java, enabling novices and professional developers to get up to speed quickly.
- It's easier to read than many other popular programming languages.
- It's widely used in education.
- It enhances developer productivity with extensive standard libraries and thousands of third-party open-source libraries, so programmers can write code faster and perform complex tasks with minimal code. applications.

- There are massive numbers of free open-source Python
- It's popular in web development (e.g., Django, Flask).
- It supports popular programming paradigms—procedural, functional, object-oriented and reflective.
- There are lots of capabilities for enhancing Python performance.
- It's used to build anything from simple scripts to complex apps with massive numbers of users, such as Dropbox, YouTube, Reddit, Instagram and Quora.
- It's popular in artificial intelligence, which is enjoying explosive growth, in part because of its special relationship with data science.
- It's widely used in the financial community.
- There's an extensive job market for Python programmers across many disciplines, especially in data-science-oriented positions, and Python jobs are among the highest paid of all programming jobs.

## Anaconda Python Distribution

- We use the Anaconda Python distribution because it's easy to install on Windows, macOS and Linux and supports the latest versions of Python, the IPython interpreter and Jupyter Notebooks.
- Anaconda also includes other software packages and libraries commonly used in Python programming and data science, allowing students to focus on learning Python, computer science and data science, rather than software installation issues.
- The IPython interpreter has features that help students and professionals explore, discover and experiment with Python, the Python Standard Library and the extensive set of third party libraries.

## Zen of Python

- We adhere to Tim Peters' The Zen of Python, which summarizes Python creator Guido van Rossum's design principles for the language.
- This list can be viewed in IPython with the command import this.
- The Zen of Python is defined in Python Enhancement Proposal (PEP) 20. "A PEP is a design document providing information to the Python community, or describing a new feature for Python or its processes or environment.

# 1.8 It's the Libraries!

- The **Python Standard Library** provides ready-made modules that support many tasks:
  - Text and binary data processing (JSON, XML).
  - Mathematics and functional programming.
  - File handling, storage, compression.
  - Cryptography and security tools.
  - Operating system services.
  - Concurrency and multiprocessing.
  - Networking and Internet protocols.
  - Multimedia (basic image/audio).
  - Internationalization (language support).
  - GUI development.
  - Debugging and performance profiling.

## Some of the Python Standard Library modules we use

- collections: Additional data structures beyond lists, tuples, dictionaries and sets.
- csv: Processing comma-separated value files.
- datetime, date: Date and time manipulations.
- decimal: Fixed-point and floating-point arithmetic, including monetary calculations.
- math: Common math constants and operations.
- os: Interacting with the operating system.
- random: Pseudorandom numbers.
- re: Regular expressions for pattern matching.
- statistics: Mathematical statistics functions like mean, median, mode and variance.
- string: String processing.
- sys: Command-line argument processing; standard input, standard output and standard error streams.

## Popular Python libraries used in data science

- Scientific Computing and Statistics
  - **NumPy (Numerical Python):** Python does not have a built-in array data structure. It uses lists, which are convenient but relatively slow. NumPy provides the more efficient ndarray data structure to represent lists and matrices, and it also provides routines for processing such data structures.
  - **SciPy (Scientific Python):** Built on NumPy, SciPy adds routines for scientific processing, such as integrals, differential equations, additional matrix processing and more. scipy.org controls SciPy and NumPy.
  - **StatsModels:** Provides support for estimations of statistical models, statistical tests and statistical data exploration.

- Data Manipulation and Analysis
  - **Pandas:** An extremely popular library for data manipulations. Pandas makes abundant use of NumPy's ndarray. Its two key data structures are Series (one dimensional) and DataFrames (two dimensional).

  Visualization
  - **Matplotlib:** A highly customizable visualization and plotting library. Supported plots include regular, scatter, bar, contour, pie, quiver, grid, polar axis, 3D and text.
  - **Seaborn:** A higher-level visualization library built on Matplotlib. Seaborn adds a nicer look-and feel, additional visualizations and enables you to create visualizations with less code.

- Machine Learning, Deep Learning and Reinforcement Learning
  - **scikit-learn:** Top machine-learning library. Machine learning is a subset of AI. Deep learning is a subset of machine learning that focuses on neural networks.
  - **Keras:** One of the easiest to use deep-learning libraries. Keras runs on top of TensorFlow (Google), CNTK (Microsoft's cognitive toolkit for deep learning) or Theano (Université de Montréal).
  - **TensorFlow:** From Google, this is the most widely used deep learning library. TensorFlow works with GPUs (graphics processing units) or Google's custom TPUs (Tensor processing units) for performance. TensorFlow is important in AI and big data analytics—where processing demands are enormous. You'll use the version of Keras that's built into TensorFlow.

- **OpenAI Gym:** A library and environment for developing, testing and comparing reinforcement learning algorithms.

- Natural Language Processing (NLP)
  - **NLTK (Natural Language Toolkit):** Used for natural language processing (NLP) tasks.

# Test-Drives: Using IPython and Jupyter Notebooks

In this section, you will test drive the IPython interpreter in two modes:

- In interective mode, you'll enter small bits of Python code called snippets and immediately see their results.
- In script mode, you'll execute code loaded from a file that has the .py extension (short for Python). Such files are called scripts or programs, and they're generally longer than the code snippets you'll do in interactive mode.

## Using IPython Interactive Mode as a Calculator

### Entering IPython in interactive Mode

- On macOS, open a Terminal from the Applications folder's Utilities subfolder.
- On Windows, open the Anaconda Command Prompt from the start menu.
- On Linux, open your system's **Terminal** or **shell** (this varies by Linux distribution).

In the command-line window, type **ipython**, then press *Enter*.

```
C:\Users\jps15>ipython
Python 3.11.8 | packaged by Anaconda, Inc. | (main, Feb 26 2024, 21:34:05) [MSC v.1916 64 bit (AMD64)]
Type 'copyright', 'credits' or 'license' for more information
IPython 8.20.0 -- An enhanced Interactive Python. Type '?' for help.

In [1]:
```

## Using IPython Interactive Mode as a Calculator

### Entering IPython in interactive Mode

- The text "`In [ ]:`" is a prompt, indicating that IPython is waiting for your input.
- "`In [ ]:`" is input prompt, and "`Out [ ]:`" is output prompt.
- In interactive mode, you can **evaluate expressions**.
- To leave interactive mode, you can:
  - Type the `exit` command at the current `In [ ]:` prompt and press *Enter* to exit immediately.
  - Type `<Ctrl> + d (or <control> + d)` twice (macOS and Linux only).

# Self Check

1. (**Fill in**) In IPython interactive mode, you'll enter small bits of Python code called _____ and immediately see their results.
2. In IPython _____ mode, you'll execute Python code loaded from a file that has the .py extension (short for Python).
3. (**IPython Session)** Evaluate the expression $5 * (3 + 4)$ both with and without the parentheses. Do you get the same result? Why or why not?

- Executing a Python Program Using the IPython Interpreter
- Writing and Executing Code in a Jupyter Notebook

# 1.11 Internet and World Wide Web

Internet: It's a giant network of smaller networks all over the world, using TCP/IP rules to communicate.

ARPANET

- **ARPANET (1960s):** Built by ARPA (U.S. DoD) to connect university computers at 50 kbps (vs. 110 bps via telephone).
- People started using it for **email**, which was fast and reliable.
- **TCP (Transmission Control Protocol):** Ensures data is split into packets, delivered intact, in correct order.
- **IP (Internet Protocol):** Assigns each device a unique **IP address** to locate devices on the Internet.

World Wide Web (WWW)

- A service that works on the Internet, letting people access websites, pages, and files (text, images, videos, music) through links.
- 1989: **Tim Berners-Lee (CERN)** created HTML(language for making web pages) and HTTP(the rules for sharing them) $\rightarrow$ basis of the Web.
- 1994: He also started the World Wide Web Consortium (W3C) to make sure websites follow common standards, so the Web works the same for everyone.

## The Cloud

- Computing resources (servers, storage, databases) delivered over the Internet.
- Accessed via **web services** (APIs). Examples:
  - Twitter API (via `Tweepy`) $\rightarrow$ lets you fetch tweets.
  - Google Translate (via `TextBlob`) $\rightarrow$ translate text easily.
  - IBM Watson $\rightarrow$ speech recognition, language translation.

## Internet of Things (IoT)

- Network of physical objects ("things") with IP addresses that send/receive data online.
- Examples: cars with toll transponders, heart monitors, smart meters, smart thermostats, sensors.
- Over 23 billion IoT devices today, expected >75 billion by 2025 (Statista).

# 1.12 Software Technologies

Software Technologies is a broad term that refers to the tools, methods, and approaches used to design, build, test, and maintain software systems. It covers:

- **Programming Practices** – ways of writing and improving code.
  *Example:* Refactoring (cleaning up code).
- **Software Design Approaches** – structures/blueprints for building systems.
  *Example:* Design Patterns (reusable solutions).
- **Infrastructure & Services** – platforms that provide computing resources.
  *Example:* Cloud Computing (services via the Internet).
- **Developer Tools** – toolkits that make development easier.
  *Example:* SDKs (Software Development Kits).

# 1.13 How Big Is Big Data?

- **Big Data:** massive, fast, varied, and valuable digital information that's reshaping business, science, and daily life — but it demands huge computing power and energy
- **Daily creation:** ~2.5 quintillion bytes (2.5 exabytes) of data created every day.
- **Future scale:** Global data expected to reach 221 zettabytes annually by 2026.
- 90% of world's data created in just the last two years.

## Data Size Examples:

- **MB (Megabyte):**
  - 1 MP3 min = 1–2.4 MB
  - Photo = 8–10 MB
  - 1080p video = 130 MB/min
  - 4K video = 350 MB/min
- **GB (Gigabyte):** 1 GB $\approx$ 1000 MB
  - DVD = 8.5 GB ($\approx$ 141 hrs MP3, 1000 photos, 7.7 min 1080p video)
- **TB (Terabyte):** 1 TB $\approx$ 1000 GB
  - 15 TB drive = 28 years of MP3, 1.68M photos, 226 hrs 1080p, 84 hrs 4K
- **PB/EB/ZB:**
  - 1 PB = 1000 TB
  - 1 EB = 1000 PB
  - 1 ZB = 1000 EB

## Big Data Stats

- 1.7 MB data/sec/person $\rightarrow$ 13 PB/sec, 1.123 ZB/day globally.
- YouTube: 24,000 hrs uploaded/hr, 1B hrs watched/day.
- Internet activity/sec: $\sim$52 TB traffic, 7.9k tweets, 64k Google searches, 72k YouTube views.
- Facebook: 800M likes/day, 60M emojis/day.
- Planet satellites: 7 TB/day for Earth monitoring.

## Computing Power

- FLOPS = Floating-point operations/sec.
- 1990s: gigaflops $\rightarrow$ teraflops $\rightarrow$ petaflops.
- IBM Summit: 122.3 petaflops.
- Next: exaflops ($10^{18}$); quantum computing much faster.

## Energy & Challenges

- Data centers may consume 20% of world power by 2025.
- Bitcoin transaction = energy of a home for a week.
- Cryptocurrencies consume more energy annually than some countries.

## Big Data Analytics (Four Vs)

- **Volume:** enormous amounts of data.
- **Velocity:** rapid speed of generation & processing.
- **Variety:** structured + unstructured (text, images, video, IoT).
- **Veracity:** trustworthiness & accuracy.

## Applications / Use Cases

- **Healthcare:** disease diagnosis, precision medicine, outbreak prediction.
- **Business:** customer retention, marketing analytics, fraud detection.
- **Tech:** AI assistants, recommendation systems, self-driving cars.
- **Security:** crime prediction, terrorism prevention, cybersecurity.
- **Environment:** crop yield, deforestation tracking, pollution reduction.
- **Everyday life:** smart cities, IoT devices, voice recognition, streaming.

# 1.14 Case Study — Big Data Mobile App: Waze

- **Waze:** GPS app with **90M+ monthly users**, uses real-time crowd data.
- **Features:**
  - Dynamic rerouting
  - Traffic, obstacles, police alerts
  - Gas price reports
- **Technologies:**
  - Open-source tools, JSON for data exchange
  - Speech recognition + synthesis, NLP for commands
  - Dynamic maps and alerts
  - IoT: each phone acts as GPS sensor
  - Cloud clusters for big data storage + parallel processing
  - AI/ML/DL for route prediction, computer vision for self-driving
  - Graph databases (Neo4j) for shortest path

# 1.15 Intro to Data Science — Artificial Intelligence

- **AI:** Learning from data $\rightarrow$ enables games, self-driving cars, chatbots, healthcare, translation.
- Artificial General Intelligence (human-like intelligence).
- **Milestones:**
  - 1997: IBM DeepBlue beat world chess champion (200M moves/sec).
  - 2011: IBM Watson won Jeopardy! using NLP + ML.
  - 2015: Google DeepMind's AlphaGo beat Go champion.
  - 2017: AlphaZero taught itself chess/Go via reinforcement learning.
- **Methods:** Machine learning, deep learning, reinforcement learning.
- Thousands of deep-learning projects (e.g., Google); fast-growing field.