

Lecture 1

Artificial Intelligence Programming

Artificial Intelligence Programming
Department: Artificial Intelligence, College of Computer Science and Mathematics,
University of Mosul
Level: 2nd Year

General Introduction

The world has witnessed rapid advancements in computer science over the past few decades, not only in processor speed and memory capacity, but also in the very way programming thinks.

Defining Artificial Intelligence:

Artificial Intelligence (AI) stands as a pivotal branch of computer science dedicated to endowing machines and computer systems with the capacity to undertake tasks traditionally demanding human-like intelligence. These tasks span a wide spectrum, including learning, problem-solving, decision-making, pattern recognition, and even creativity. At its heart, AI seeks to emulate human cognitive processes, enabling systems to interpret their surroundings, reason logically, and act autonomously. This encompasses everything from basic rule-based automation to sophisticated systems like self-driving cars and intelligent chatbots.

AI is not a singular technology but rather a broad, encompassing term that includes various subfields.

The AI tasks include:

- **Learning:** Acquiring knowledge and skills from data or experience.
- **Reasoning:** Using logic to solve problems and draw conclusions.
- **Problem-Solving:** Developing strategies to achieve specific goals.
- **Perception:** Interpreting the world through sensory data (vision, sound, etc.).
- **Language Understanding:** Comprehending and generating human language.

One of the most significant of these shifts is the transition from traditional programming to artificial intelligence programming.

Modern computer systems no longer simply execute fixed, predefined instructions written by the programmer. They are now capable of:

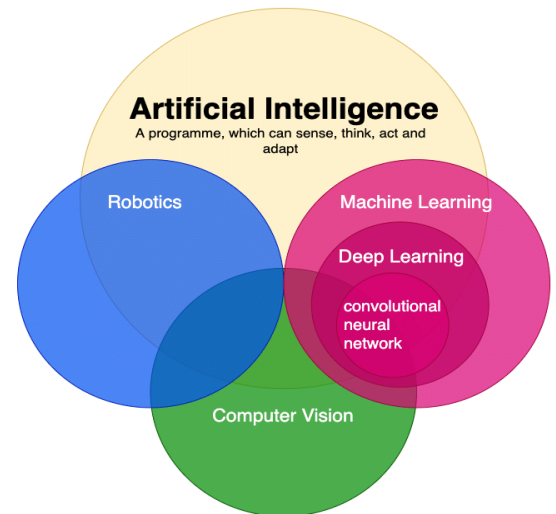
- **Learning from data**
- **Adapting to different conditions**
- **Making approximate decisions that closely resemble human decisions.**

This shift forms the scientific foundation for broad fields such as:

Machine Learning

- **Deep Learning**
- **Natural Language Processing**
- **Computer Vision**
- **Robotics.**

These fields are not isolated; they are deeply interconnected. A self-driving car, for instance, uses **Computer Vision** to see the road, **Deep Learning** to make sense of the visual data, **Machine Learning** to predict the behavior of other drivers, and **Robotics** to physically control the vehicle. Our journey in this course is to learn how to program these capabilities



Subfield	Core Idea	A Simple Analogy
Machine Learning (ML)	Algorithms that learn patterns from data without being explicitly programmed.	A child learning to identify a cat by seeing many pictures of cats.
Deep Learning (DL)	A subset of ML using complex, multi-layered neural networks to learn from vast amounts of data.	The human brain's network of neurons learning to recognize complex patterns.
Natural Language Processing (NLP)	Enabling computers to understand, interpret, and generate human language.	A universal translator that understands both text and spoken context.
Computer Vision (CV)	Enabling computers to "see" and interpret visual information from the world.	The visual cortex of the brain processing light to identify objects.
Robotics	Integrating AI with mechanical systems to perform physical tasks in the real world.	The human body's ability to combine thought (brain) with action (limbs).

Traditional Programming

Traditional programming is an approach in which the programmer explicitly defines all rules, conditions, and steps that the computer must follow to solve a problem.

The system does not learn or adapt; it simply executes predefined instructions. The output is fully determined by the logic written by the programmer.

Characteristics of Traditional Programming

- Rule-based and deterministic
- No learning capability
- Requires manual modification to handle new cases
- Best suited for well-defined and structured problems

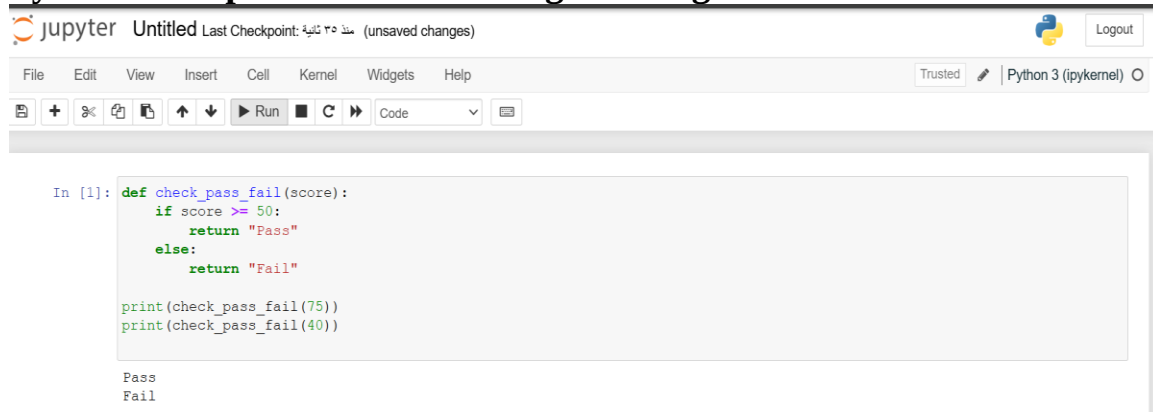
Strengths of Traditional Programming:

- **Predictability and Reliability:** Outputs are consistent and expected for given inputs, making debugging straightforward.
- **Efficiency for Rule-Based Tasks:** Ideal for well-defined, repetitive tasks with clear logic, such as calculating sums or sorting lists.
- **Interpretability:** The logic is transparent, making it easier to understand and verify.
- **Lower Resource Needs:** Generally requires less computational power and data compared to AI.

Weaknesses of Traditional Programming:

- **Rigidity:** Struggles with ambiguity and cannot adapt to new scenarios without explicit code modifications.
- **Scalability Issues:** For complex problems, coding all possible edge cases becomes unmanageable and time-consuming.
- **Poor at Pattern Recognition:** Inefficient for tasks involving unstructured data like images, speech, or complex human language.

Python Example: Traditional Programming



The screenshot shows a Jupyter Notebook interface. At the top, the title bar says "jupyter Untitled Last Checkpoint: منذ ٣٥ ثانية (unsaved changes)". Below the title bar is a menu bar with "File", "Edit", "View", "Insert", "Cell", "Kernel", "Widgets", and "Help". To the right of the menu bar are "Trusted" and "Python 3 (ipykernel)" buttons. Below the menu bar is a toolbar with icons for file operations, running, and code execution. The main area of the notebook contains a code cell with the following Python code:

```
In [1]: def check_pass_fail(score):  
        if score >= 50:  
            return "Pass"  
        else:  
            return "Fail"  
  
        print(check_pass_fail(75))  
        print(check_pass_fail(40))
```

Below the code cell, the output of the code is displayed:

```
Pass  
Fail
```

```
1
2 def check_pass_fail(score):
3     if score >= 50:
4         return "Pass"
5     else:
6         return "Fail"
7
8 print(check_pass_fail(75))
9 print(check_pass_fail(40))
10
```

Explanation:

The decision rule ($\text{score} \geq 50$) is explicitly written by the programmer. The program cannot improve or adapt unless the rule itself is changed.

Artificial Intelligence Programming

Artificial Intelligence programming focuses on creating systems that can learn patterns from data, make predictions, and improve performance over time.

Instead of writing explicit rules, the programmer provides data and learning algorithms, allowing the system to infer decision logic automatically.

Characteristics of Artificial Intelligence Programming

- Data-driven rather than rule-driven
- Capable of learning and adaptation
- Handles uncertainty and complex patterns
- Performance improves with more data

Strengths of AI Programming:

- **Adaptability and Scalability:** Continuously improves with more data and can generalize to new, unforeseen scenarios.
- **Handles Complexity:** Excels in tasks involving unstructured data, such as image recognition, natural language processing, and predictive analytics.

- **Innovation Driver:** Enables automation in dynamic and uncertain environments, leading to groundbreaking applications like self-driving cars.

Weaknesses of AI Programming:

- **Data Dependency:** Requires vast amounts of high-quality data; biased data can lead to flawed and unfair models.
- **"Black Box" Nature:** Many AI models, particularly deep learning models, can be opaque, making their decision-making process difficult to interpret or explain.
- **High Computational Cost:** Training complex AI models often demands significant computational resources, including powerful GPUs or TPUs.
- **Debugging Challenges:** Identifying and rectifying errors in AI models can be complex due to their iterative and data-driven nature

Python Example: AI-Based Programming (Machine Learning)

```
1
2  from sklearn.linear_model import LogisticRegression
3
4  X = [
5      [2, 60],
6      [4, 70],
7      [6, 80],
8      [8, 90]
9  ]
10
11  y = [0, 0, 1, 1]
12
13  model = LogisticRegression()
14  model.fit(X, y)
15
16  result = model.predict([[2, 75]])
17
18  if result[0] == 1:
19      print("Pass")
20  else:
21      print("Fail")
22
```

Explanation:

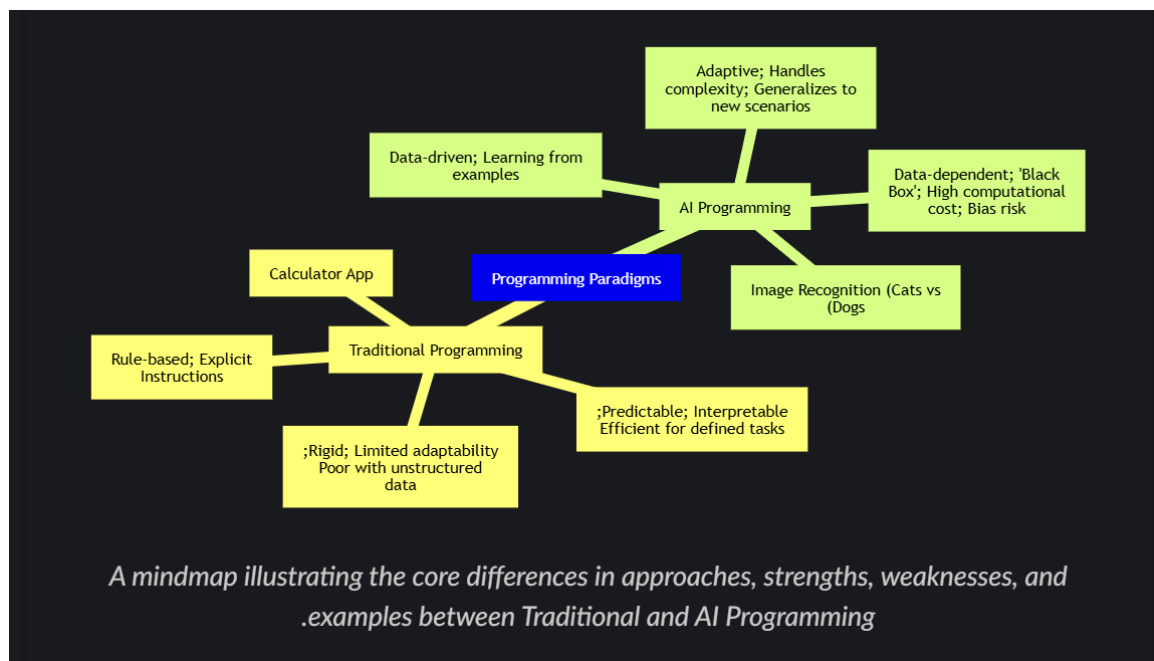
The model learns the relationship between study hours, attendance, and success from data—without explicitly coding pass/fail rules.

Key Differences Between Traditional Programming and AI Programming

Aspect	Traditional Programming	AI Programming
Decision logic	Written explicitly by programmer	Learned from data
Learning capability	None	Yes
Flexibility	Low	High
Handling complexity	Limited	Strong
Adaptation to change	Requires reprogramming	Automatic through retraining
Output nature	Deterministic	Probabilistic

Traditional programming is ideal for simple, rule-based problems with clear logic.

AI programming is more suitable for complex, data-intensive problems where rules are difficult to define explicitly.



```

1  from sklearn.linear_model import LogisticRegression
2
3  # بيانات التدريب : [ساعات الدراسة , الدرجة]
4  X = [
5      [1, 50],
6      [2, 55],
7      [3, 60],
8      [4, 65],
9      [5, 70],
10     [6, 75],
11     [7, 80],
12     [8, 85],
13     [9, 90],
14     [10, 95]
15 ]
16
17 # النتائج : Fail, 1 = Pass = 0
18 y = [0, 0, 0, 0, 0, 1, 1, 1, 1, 1]
19
20 model = LogisticRegression()
21 model.fit(X, y)
22
23 # اختبار طالب جديد
24 result = model.predict([[6, 72]])
25
26 if result[0] == 1:
27     print("Pass")
28 else:
29     print("Fail")

```

Explanation:

The model learns the relationship between study hours, attendance, and success from data—without explicitly coding pass/fail rules.

Programming Languages for Artificial Intelligence

1. Introduction

Artificial Intelligence (AI) is not bound to a single programming language. Contrary to a common misconception, **there is no dedicated or exclusive language designed solely for artificial intelligence**. Instead, AI systems are developed using a variety of programming languages, each selected according to the nature of the problem, performance requirements, and application domain.

The effectiveness of an AI system depends primarily on the **algorithms, data, and computational models**, rather than on the programming language itself. However, over time, certain languages have become **preferred in AI development** due to their flexibility, performance, availability of libraries, and strong community support. These languages facilitate the implementation of AI techniques such as machine learning, deep learning, natural language processing, and computer vision.

2. Python

Python is currently the **most widely used programming language in artificial intelligence** and has become the de facto standard in both academic research and industrial applications.

Key Features and Advantages:

- Simple and readable syntax, which enhances productivity and reduces development time
- Extensive ecosystem of AI and data science libraries
- Strong support for rapid prototyping and experimentation
- Large and active global community

Major AI Libraries:

- NumPy and Pandas for numerical computation and data processing
- Scikit-learn for machine learning algorithms
- TensorFlow and PyTorch for deep learning
- OpenCV for computer vision
- NLTK and spaCy for natural language processing

Python is particularly suitable for **machine learning, deep learning, data analysis, and research-oriented AI development.**

3. Java

Java is a widely used general-purpose programming language known for its **platform independence, scalability, and robustness**. It plays an important role in large-scale AI systems and enterprise-level applications.

Key Features and Advantages:

- Platform-independent execution through the Java Virtual Machine (JVM)
- Strong memory management and security mechanisms
- High performance in large, distributed systems
- Suitable for long-term, maintainable software projects

AI Applications in Java:

- Intelligent enterprise systems
- Distributed AI platforms
- Robotics and autonomous systems

Common AI Libraries:

- Deeplearning4j
 - Weka
 - MOA (Massive Online Analysis)
-

4. C and C++

C and C++ are low-level programming languages that provide **direct control over system resources and hardware**, making them essential for performance-critical AI applications.

Key Features and Advantages:

- High execution speed and efficiency
- Direct memory management
- Suitable for real-time and embedded systems
- Excellent support for hardware acceleration

AI Application Areas:

- Robotics and autonomous vehicles
- Computer vision systems
- Embedded and real-time AI applications

C++ is often used alongside Python, where **performance-intensive components** are implemented in C++ and integrated with high-level Python code.

5. R

R is a programming language and environment specifically designed for **statistical computing and data analysis**. It is widely used in academic research and data-driven AI applications.

Key Features and Advantages:

- Strong statistical and mathematical foundations
- Rich set of data visualization tools

- Effective for exploratory data analysis
- Ideal for research and experimental modeling

AI Application Areas:

- Statistical machine learning
- Data mining
- Predictive analytics

R is especially valuable in applications where **statistical interpretation and analysis** are critical.

6. MATLAB

MATLAB is a high-level programming environment extensively used for **numerical computation, modeling, and simulation**. It is particularly popular in engineering and academic contexts.

Key Features and Advantages:

- Powerful mathematical and matrix-based computation
- Built-in toolboxes for AI, machine learning, and deep learning
- Excellent visualization and simulation capabilities
- Suitable for algorithm development and testing

AI Application Areas:

- Computer vision
 - Robotics and control systems
 - Signal and image processing
-

7. Lisp and Prolog (Historical Perspective)

Lisp and Prolog are among the earliest programming languages used in artificial intelligence research. They played a significant role in the development of **symbolic AI and expert systems**.

Key Features and Advantages:

- Strong support for symbolic reasoning
- Rule-based logic representation

- Useful for knowledge-based systems

Although their industrial use has declined, they remain **important from an academic and historical perspective.**

In conclusion, artificial intelligence does not rely on a single programming language. Instead, **different languages are selected based on the application requirements, performance constraints, and development goals.** Python dominates modern AI development due to its simplicity and rich ecosystem, while Java, C/C++, R, and MATLAB serve specialized and complementary roles.

For students of artificial intelligence, it is essential to understand that **mastery of AI concepts, algorithms, and data handling is far more important than proficiency in any single programming language.** Languages are tools, whereas intelligence lies in the models and methods implemented using those tools.