FUTURE STEM HUB

# Educational Materials with Hands-On Python and Scratch Tasks

## FUTURE-STEM-HUB

**Empowering Secondary School STEM Education with AI Training and Resources for Students and Educators**

Project No.
2024-1-DE03-KA220-SCH-000247346

Co-funded by the European Union

## EDUCATIONAL MATERIALS WITH HANDS-ON PYTHON AND SCRATCH TASKS

The project “Empowering Secondary School STEM Education with AI Training and Resources for Students and Educators / FUTURE-STEM-HUB” (ref. no: 2024-1-DE03-KA220-SCH-000247346) is co-funded by the Erasmus+ Programme of the European Union. It is coordinated by the University of Duisburg-Essen (Germany) and involves four other partner organizations: M&M Profuture Training (Spain), Kütahya Provincial Directorate of Ministry of National Education (Türkiye), COOPETAPE - Cooperative of Education, CRL- the overseeing body of ETAP School (Portugal) and Tetra Solutions Ltd. (Bulgaria).

The project team members representing all partner organizations’ develop the FUTURE-STEM-HUB Educational Materials with Hands-On Python and Scratch Tasks, whereby each partner was responsible in its authorship for one module. They aim to introduce the fundamental concepts of AI to secondary students, fostering awareness and discussion regarding their societal and ethical implications.

**Authors:**
Mustafa Bilgin, University of Duisburg-Essen (Germany)
Monica Moreno, M&M Profuture Training (Spain)
Montserrat Renedo, M&M Profuture Training (Spain)
João Barroso, ETAP School (Portugal)
Angelina Presa, ETAP School (Portugal)
Silviya Georgieva, Tetra Solutions Ltd. (Bulgaria)
Borislava Zaharieva-Tomova, Tetra Solutions Ltd. (Bulgaria)
Yeliz Yurter, Kütahya MEM (Türkiye)
Özcan Turan, Kütahya MEM (Türkiye)

**Editor:**
Mustafa Bilgin, University of Duisburg-Essen (Germany)

# List of Figures

Layout Template: Tetra Solutions Ltd.
Cover Image: Hybrid creation (Concept & drawing template: Mustafa Bilgin, digital retouching: Generated using Google Gemini).
P. 6: Graphic by Tetra Solutions Ltd.
P. 7, 8: Hybrid creation (Concept & drawing template: Mustafa Bilgin, digital retouching: Generated using Google Gemini).
The icons are sourced from Flaticon.com (High quality icons).

# Disclaimer and Safety

The use of the programming codes provided in these materials, as well as access to external websites and third-party platforms, is at the sole risk and responsibility of the users. The project team assumes no liability for the content, functionality, or availability of external links, such as Google Colab, Scratch, or Machine Learning for Kids. Since many of the interactive tasks require an active internet connection and the processing of data on cloud servers, the responsibility for complying with security standards and system requirements (such as current browser versions) lies with the users. When using third-party tools to create AI models, no personal or sensitive information may be uploaded. As the educational materials are aimed at the target group of 15 to 18-year-olds, registration with external services and the performance of online activities should be conducted in accordance with the respective providers' terms of use and, ideally, under the supervision and with the consent of a teacher or legal guardian. No liability is accepted for any data loss or impairments to the end devices used resulting from the execution of the provided code examples.

This work is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License (CC BY-NC-SA 4.0).

# Note on Software Versions

Please be aware that the versions of Python, Google Colab and YOLO indicated in the materials were accurate as of the course's development date. As software updates occur regularly, these versions may become outdated over time. If you encounter errors or unexpected behaviour while completing the exercises, please check whether a newer version of the relevant tool is available and update it accordingly before proceeding.

# List of Abbreviations:

**AI:** Artificial Intelligence

**ML:** Machine Learning

**DL:** Deep Learning

**NN:** Neural Network

**NLP:** Natural Language Processing

**CV:** Computer Vision

**CNN:** Convolutional Neural Network

**RGB:** Red, Green, Blue (Colour Model)

**YOLO:** You Only Look Once

**FPS:** Frames Per Second

**TTS:** Text-to-Speech

**MNIST:** Modified National Institute of Standards and Technology (Dataset)

**IDE:** Integrated Development Environment

**ML4K:** Machine Learning for Kids

**IBM:** International Business Machines

**MIT:** Massachusetts Institute of Technology

**EACEA:** European Education and Culture Executive Agency

**UNESCO:** United Nations Educational, Scientific and Cultural Organization

**STEM:** Science, Technology, Engineering, and Mathematics

# Contents

Project Overview.....6

Project Outputs.....6

Introduction.....7

Module 1: Mini-Coding-Challenges:
Machine Learning and Neural Networks.....9

Module 2: AI Escape Room
Treasure Hunt with Calculator.....40

Module 3: Scratch Meets Artificial Intelligence.....61

Module 4: Python Game Programming
(pygame/Arcade libraries) with AI .....83

Module 5: Object Recognition with Roboflow:
Introduction to Computer Vision.....109

Summary and Next Steps.....124

This publication is licensed under a Creative Commons Attribution - NonCommercial - NoDerivatives 4.0 International License

Empowering Secondary School STEM Education with AI
Training and Resources for Students and Educators
2024-1-DE03-KA220-SCH-000247346

# Project Overview

## FUTURE-STEM-HUB

FUTURE-STEM-HUB project aims to advance and facilitate the integration of Artificial Intelligence (AI) topics into STEM education at secondary schools by: 1) Providing educational materials introducing AI concepts and their societal implications; 2) Offering practical learning resources for students to explore AI using Python programming and 3) Equipping teachers with support for integrating AI into secondary school STEM training.

# Project Outputs

1. **Course 1: A Digital Primer: Artificial Intelligence Essentials** (Introduction to AI through Interactive Educational Materials for Secondary Students)

2. **Course 2: Delving Deeper in AI with Python and Scratch** (Advanced AI: Hands-On Learning Materials for Secondary Students)

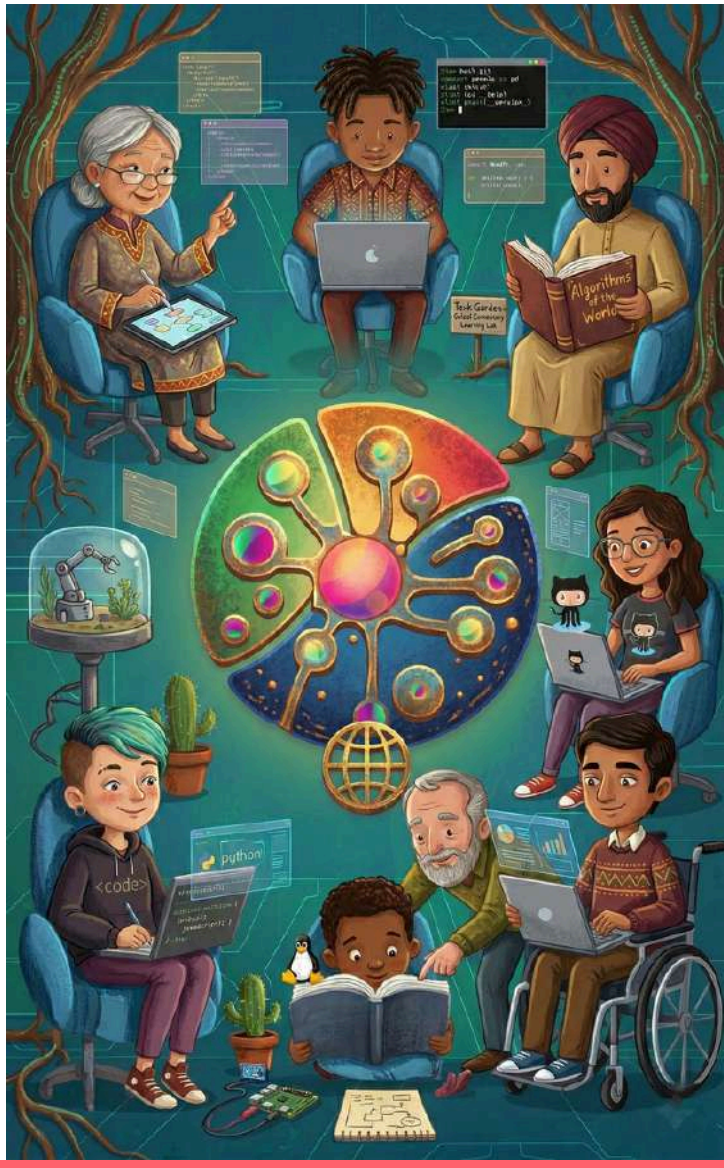
3. **E-Toolkit for School Educators: Enhancing Artificial Intelligence Skills** (AI Methodological Guiding for Secondary Teachers)

# Introduction

Welcome to the Educational Materials with Hands-On Python and Scratch Tasks—an engaging and interactive learning journey designed to introduce secondary school students to the fascinating and practical world of AI. To equip secondary school teachers and students with accessible, practice-oriented materials for exploring key AI concepts through Python and Scratch. Tailored for students aged 15–18 with diverse STEM proficiency levels, the materials are perfect for self-paced asynchronous learning and can also be seamlessly integrated into classroom teaching with teacher facilitation.

The content is divided into five comprehensive modules, focusing on practical programming skills in Python and Scratch, mathematical logic through game-based learning (Escape Room), creative AI applications, real-world problem-solving with data, and ethical considerations.

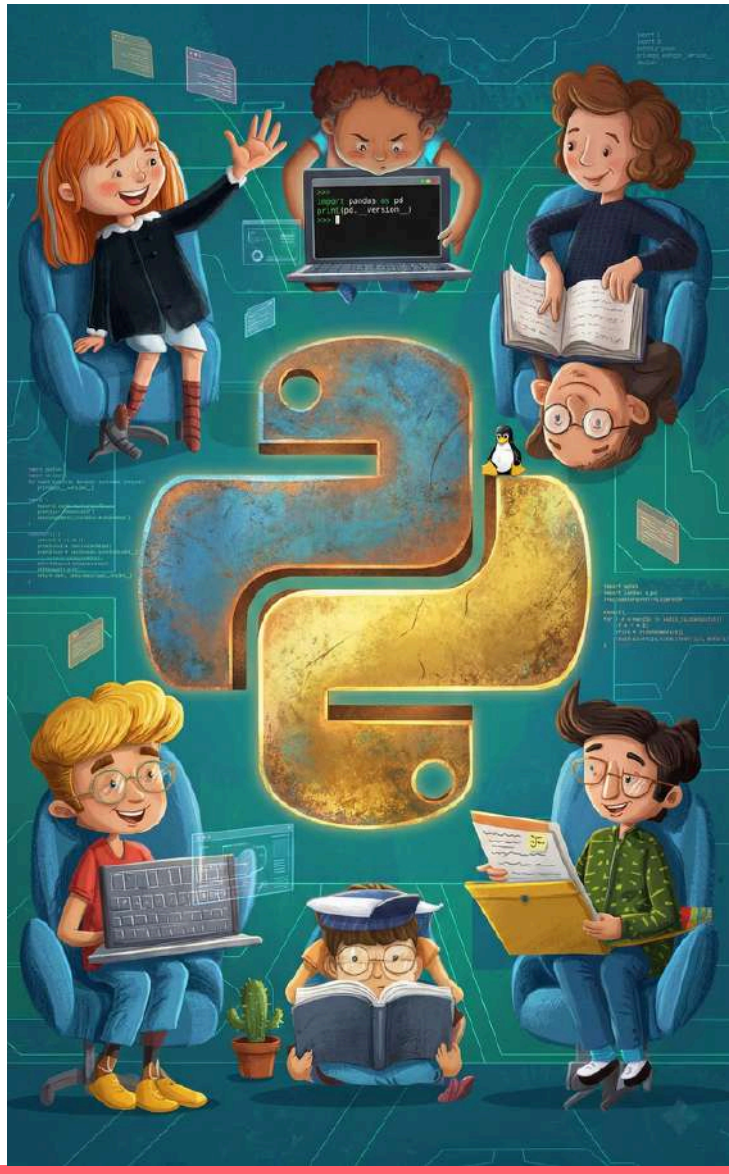
Each module combines theory, refers to implemented quizzes, uses external platforms for further exploration, and includes practical exercises to ensure a comprehensive understanding of AI concepts and methodologies. The estimated learning duration for all five modules is approximately 10 hours.



After completing the course, students will be able to understand and define key AI concepts, such as machine learning, neutral networks, deep learning. They will gain deeper knowledge about the history and evolution of Artificial Intelligence and its applications across various industries, learning how new AI technologies shape today's trends and world. Students will also dive deeper into fundamental technologies, including computer vision and natural language processing. Lastly, they will explore and gain more profound understanding about AI ethical considerations, AI's responsibility and its societal impact.

On the other hand, teachers will gain access to innovative, interactive resources that they can easily use and apply in the STEM classroom supporting their curricula and teaching. These materials will help them create more interesting and engaging classroom environments as they offer different approaches to actively involve secondary students in the learning process.

These materials will be further transformed into an online course that will be hosted on the FUTURE STEM-HUB platform and accessible via the project website: www.future-stem-hub.



**Upon successful completion of the course, students will receive a digital certificate recognizing their achievement.**

# Module 1: Mini-Coding-Challenges: Machine Learning and Neural Networks

## INTRODUCTION

Have you ever wondered how your smartphone knows which videos you might like? Or how an app can recognize your handwriting? Behind all that is Artificial Intelligence (AI)—which is far less mysterious than many people think! In this chapter, you'll join Aylin and her clever smartwatch, Alara. Together, they'll take you on adventures that reveal how AI really works. You will develop a clear understanding of the fundamental principles and components of Neural Networks (NN) and Machine Learning (ML), including how they process inputs, learn from data, and make decisions. The best part? You don't need to be a programming expert! Each adventure includes small coding challenges in three difficulty levels. Just choose the one that suits you best—from beginner to real challenge. You'll see: after the very first chapter, you'll already understand how machines "learn to think." Let's dive together into the world of AI.

By the end of the module, you will be able to acquire different skills, such as:

- **Introduction to AI:** The Mysterious Gift: A Watch That Does More Than Tick
- **Neural Networks — Basics:** Alara Awakens
- **Decision Trees:** The School Route Detective—How Your Watch Thinks
- **Rule-Based Classification:** How Spotify Might Know What You Want
- **Reinforcement Learning:** Learning Through Rewards
- **Convolutional Neural Networks:** How AI Detects Edges in Images

- **Natural Language Processing:** How Computers Understand Language
- **Recommendation Systems:** How YouTube Knows What You Want to Watch
- **Random Forest:** Weather forecast
- **AI Ethics:** Vision for the Future—What We Can and Should Do With AI

| | |
|---|---|
| **Module Duration** | **2 hours (1 hour learning + 1 hour practical exercises)** |

## EDUCATIONAL MATERIALS

**AYLIN** is 17 years old, curious, and loves figuring out how things work. She never gives up!

**LENA** is Aylin's best friend. She's creative and loves music. Her family is from Poland; sometimes she brings delicious pierogi!

**ALARA** is no ordinary smartwatch. She's an AI that teaches Aylin about artificial intelligence.

Figure 1 Educational characters Aylin, Alara & Lena; Concept and layout: Mustafa Bilgin; Enhanced using Google Gemini

## UNIT 1.1 COURSE CONCEPT & APPROACH

The learning process is guided by a story-based framework that connects STEM topics with real life contexts students can relate to. Complex concepts such as neural networks or reinforcement learning are represented in simplified, prototype form, allowing learners to build foundational understanding—no prior programming experience required. This material fits perfectly within computer science and technical education, STEM projects, extracurricular clubs, or interdisciplinary learning settings. It not only promotes technical understanding but also—through the AI Ethics chapter—encourages critical reflection on moral and social questions.

## UNIT 1.2 INTRODUCTION TO ARTIFICIAL INTELLIGENCE (AI)

Before diving into the activities, it is essential to understand what we are dealing with. Artificial Intelligence (AI) refers to the simulation of human intelligence in machines that are programmed to think like humans and mimic their actions. The term may also be applied to any machine that exhibits traits associated with a human mind, such as learning and problem-solving.

In this module, we will explore the fundamental building blocks of AI through interactive stories:

- **Neural Networks:** These are computing systems inspired by the biological neural networks that constitute animal brains. They help the AI “learn” from examples.
- **Decision Trees:** A decision support tool that uses a tree-like model of decisions and their possible consequences. It represents the logic of "If this happens, then do that."
- **Machine Learning (ML):** A subset of AI that provides systems the ability to automatically learn and improve from experience without being explicitly programmed.

This module uses a narrative approach (storytelling) to make these complex mathematical concepts accessible and easy to understand.

## UNIT 1.3 SYSTEM SETUP & PREREQUISITES

### What Platform You Need

You will use **Google Colab** to run the Python code in this module. Google Colab is a free online platform that lets you write and run code directly in your web browser—no installation is needed.

**Access it here:**
https://colab.research.google.com/

### Do You Need to Register?

Yes, you need a **free Google account** (a Gmail account) to use Google Colab. If you already have one, you can log in directly. If you do not have a Google account, you must create one before starting.

**To create a Google account:**
Go to accounts.google.com
and follow the sign-up steps.

### What Is Included in the Registration Process?

1. Go to Google Colab.
2. Click "Sign in" in the top-right corner.
3. Enter your Google email and password.
4. Once signed in, you can open or create a new notebook.

### Where You Are Expected to Write the Code

All coding exercises in Module 1 are prepared in a **Google Colab notebook**. You do not need to write code—you will use a pre-made notebook that contains all the exercises and instructions.

Here is the notebook for Module 1:
*>Module 1 Colab Notebook<*

### What to do with the notebook:

1. Open the link above.
2. Click "Copy to Drive" to save your own editable version.
3. Follow the instructions inside the notebook, running each code cell step-by-step.

## UNIT 1.4 ADDITIONAL INSTRUCTIONS BEFORE YOU START

1. Use a modern web browser such as Chrome, Firefox, Edge, or Safari.
2. Make sure you have a stable internet connection—Google Colab runs online.
3. Read the story and instructions for each station carefully before you begin the coding task.
4. Each station offers **three difficulty levels**:
   - **Basic** is for beginners.
   - **Advanced** is if you have some experience with Python.
   - **Expert:** If you want real tough challenges, Expert is the place to be.
   - Choose the level that fits your skills.
5. Run the code cells in order. Some exercises depend on previous ones.
6. If you are stuck, use the "**Hint**" and "**Solution snippet**" provided in each exercise.
7. After completing a task, try changing the code to see how it affects the outcome. Experimenting helps you learn.

# REFERENCES


## Emoji and Icon Attribution

- This document uses standard Unicode emojis as defined by the Unicode Consortium. Specific emoji versions may vary depending on the user's operating system. Unicode Consortium. (2022). Retrieved from https://unicode.org/emoji/charts/full-emoji-list.html. Icons are from 'Flaticon.com,' High Quality Icons.

## Books & Articles

- Russell, S., & Norvig, P. (2021). Artificial Intelligence: A Modern Approach (4th ed.). Pearson. (A comprehensive textbook on AI concepts, including neural networks, decision trees, and reinforcement learning.)

- Géron, A. (2022). Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow (3rd ed.). O'Reilly. (Practical introduction to ML and neural networks with Python.)

## AI Ethics & Society

- European Commission. (2019). Ethics Guidelines for Trustworthy AI. (A framework for ethical AI development and deployment.)

- UNESCO. (2021). Recommendation on the Ethics of Artificial Intelligence. (Global guidelines on AI ethics, focusing on human rights and inclusivity.)

## Python Programming for Beginners

- Sweigart, A. (2019). Automate the Boring Stuff with Python (2nd ed.). No Starch Press. (Beginner-friendly Python book with practical projects.)


## LINKS TO LEARNING PLATFORMS

- Here is the notebook for Module 1 – https://colab.research.google.com/drive/14VaJnlgo5habZ0-N9jpz9nWZSVAw_Xwa?usp=sharing

- Python Official Documentation – https://docs.python.org/3/ (Official Python language reference and tutorials.)

## PRACTICAL EXERCISE

### Station 1: The Mysterious Gift: A Watch That Does More Than Tick

It's Aylin's 17th birthday—between baklava and balloons lies a mysterious little package. Aylin hesitates to open it.

**AYLIN:** *"What is it, Aunt Sema?"*

**AUNT SEMA:** *"Something special, canım (sweetheart).*
*Your grandmother would say, This watch has a soul."*

Aylin carefully unwraps the package. Inside is an elegant smartwatch with a turquoise-blue strap. Suddenly, the display lights up.

**ALARA (the watch):** *"Merhaba (Hello), Aylin! I'm Alara—your new AI friend!"*

Aylin jumps back in surprise.

**AYLIN:** *"The... the watch is talking?"*

**AUNT SEMA:** *"It speaks every language you understand.*
*Alara will show you how artificial intelligence really works!"*

**ALARA:** *"Imagine I'm like your brain: I have neurons that learn and synapses that connect. Together, we'll discover how machines learn to think!"*

```
BASIC

CODE:
NAME = INPUT("WHAT'S YOUR NAME? ")
PRINT("HELLO " + ____ + "! I'M ALARA, YOUR AI FRIEND")
```

**Hint:** Use the variable you entered in the blank.

**Solution snippet:** name

## Station 1: The Mysterious Gift: A Watch That Does More Than Tick

```
ADVANCED

CODE:
print("🎂 Aylin's Birthday Planning 🎂")
favorite_food = input("What's your favorite food? (Pizza/
Kebab/Burger): ")
drink = input("And what would you like to drink with that? ")

if favorite_food.lower() == "pizza":
    print("🍕 Alara says: Pizza is my favorite too!")
else:
    print("😋 Alara says: " + __ + " sounds great!")

print("And " + __ + " with that? Perfect combo!")
```

**Hint:** Fill in the blanks with your input variables.

**Solution snippet:** favorite_food, drink

## STATION 2: Neural Networks — Alara Awakens

The next morning, Aylin sits in the park, watching Alara.
**AYLIN:** *"So... you think like me?"*

**ALARA:** *"Almost! Imagine my 'brain' is like a big thinking factory.*
*I have neurons that learn. Look..."*

**AYLIN:** *"So, you're going to learn with me?"*

**ALARA:** *"Yes,"* Alara replies.
*"I know the basics, but through you, I learn to see the world through human eyes."*

## STATION 2: Neural Networks — Alara Awakens

```
BASIC

CODE:
print("Should I buy a new game?")
print("Rate from 0 to 1, where 0 = not at all, 0.5 = okay, 1
= very much")

savings_balance = float(input("How full is your piggy bank?
(0-1): "))
desire = float(input("How much do you want the game? (0-1):
"))

def decision_neuron(money, wish):
   if money * wish > ____:
      return "✅ YES, buy it!"
   else:
       return "❌ NO, better wait"

print(decision_neuron(savings_balance, desire))
```

**Hint:** A neuron 'fires' when the product is greater than 0.5.

**Solution snippet:** 0.5

## STATION 2: Neural Networks — Alara Awakens

```
ADVANCED (Part I)

CODE:
print("Is it a good friendship?")
print("Rate from 0 to 1, where 0 = not at all, 0.5 = okay, 1
= very much")

trust = float(input("How much do you trust this person? (0-
1): "))
fun = float(input("How much fun do you have together? (0-1):
"))
helpfulness = float(input("How helpful is this person? (0-1):
"))

# Weights - what matters most in friendship?
weight_trust = 0.5
weight_fun = 0.3
weight_help = 0.2

def friendship_analysis(trust, fun, help):
   inputs = [trust, fun, help]
   weights = [weight_trust, weight_fun, weight_help]

   total = 0
   for i in range(len(inputs)):
       total += inputs[i] * ____
   return total
```

## STATION 2: Neural Networks — Alara Awakens

```
ADVANCED (Part II)

CODE:
score = friendship_analysis(trust, fun, helpfulness)
print("Friendship Score:", score)

if score > 0.7:
   print("💖 True friendship!")
elif score > 0.4:
   print("👍 Good acquaintance")
else:
print("🤔 Maybe not the best friendship")
```

**Hint:** Multiply each input by its corresponding weight.

**Solution snippet:** weights[i]

## STATION 3: Decision Trees — The School Route Detective

Aylin is at the bus stop. It's raining. She talks to Alara about her way to school.

**ALARA:** *"I have an idea for your school route! Let's build a decision tree."*

Alara helps her make choices—this is called a decision tree.

**AYLIN:** *"Great! Now I'll always know which way to go,"* Aylin says happily.

**ALARA:** *"And I'm learning how you make decisions."*

## STATION 3: Decision Trees — The School Route Detective

**BASIC**

**CODE:**

```
weather = input("How's the weather? (sun/rain): ")

def school_route(weather):
    if weather == "rain":
       return "☔ Take the umbrella!"
    else:
       return "🌞 ____"

print(school_route(weather))
```

**Hint:** What do you do if it's not raining?

**Solution snippet:** Go without an umbrella!

**ADVANCED**

**CODE:**

```
weather = input("Weather? (rain/sun): ")
bus_coming = input("Is the bus coming? (yes/no): ")

def plan_route(weather, bus_coming):
     if weather == "rain" and bus_coming == "yes":
        return "🚌 Take the bus!"
     elif weather == "rain" and bus_coming == "no":
        return "☔ ____"
     else:
        return "🚶Walk!"
print(plan_route(weather, bus_coming))
```

**Hint:** If the bus doesn't come, only the umbrella helps.

**Solution snippet:** Take the umbrella!

## STATION 4: Rule-based Classification — How Spotify Might Know What You Like

Aylin and her friend Lena are listening to music.

**LENA:** *"How does your watch know that I like rock?"*

**ALARA:** "That's rule-based classification! I group similar songs."

**LENA:** *"That's like magic!"* Lena exclaims.

**ALARA:** *"No magic,"* Alara corrects her.
*"Just pattern recognition—and I'm learning your taste in music better all the time."*

```
BASIC

CODE:
song = input("Describe your song: ... (e.g. guitar)")

def sort_music(song):
    if "guitar" in song:
       return "🎸 Rock"
    elif "____" in song:
        return "🎤 Pop"
    else:
        return "🎶 Other"

print(sort_music(song))
```

**Solution snippet: danceable**
**Hint:** Danceable songs are usually pop.

**Simplified Grouping:** In this exercise, we group listeners based on their stronger music preference. Real AI systems, like K-Means, use more complex mathematics to automatically find users with siilar tastes.

## STATION 5: Reinforcement Learning — Learning Through Reward

Aylin is playing with a dog training app on her tablet.

**AYLIN:** *"How do I teach this virtual dog what to do?"*

**ALARA:** *"It's simple: through rewards! When it does something right, give it a treat. This way it gradually learns which actions are worth doing. This is called Reinforcement Learning!"*

**AYLIN:** *"So like school—good grades for correct answers?"*

**ALARA:** *"Exactly! But here you decide what 'correct' is.*
*The dog tries things out and remembers what works."*

**AYLIN:** *"So you learn just like your virtual dog?"* Aylin asks with a smile.

**ALARA:** *"In a way, yes,"* Alara admits. "Rewards work for all learners."

```
BASIC (Part I)

CODE:
import random

print("Train your virtual dog!")
print("Reward with a treat (1) or ignore (0)\n")

actions = ["Sit", "Lie Down", "Give Paw"]
probabilities = [0.3, 0.3, 0.4] # Starting values
disturb_chance = 0.2 # 20% chance the dog does something else

def normalize(probabilities):
  total = sum(probabilities)
  return [p / total for p in probabilities]
```

## STATION 5: Reinforcement Learning — Learning Through Reward

```
BASIC (Part II)

CODE:
for round in range(1, 6):
   print(f"\n Training session {round}/5")

   # Dog chooses an action based on its preferences
   action = random.choices(actions, weights=_________________)
[0] # Gap 1

   # Disturbance: Dog might do a different action
   if random.random() < disturb_chance:
      possible_actions = [a for a in actions if a != action]
      wrong_action = random.choice(possible_actions)
      print(f"The dog should do '{action}' but does instead:
🐶 {wrong_action}")
      action = wrong_action
   else:
      print(f"The dog does: 🐾 {action}")

   # User interaction with input check
   while True:
      user_input = input("Give a treat? (1=yes, 0=no):
").strip()
   if user_input in ("1", "0"):
      reward = int(user_input)
      break
      print("Please enter only 1 or 0!")

   index = actions.index(action)")
```

**Hint:** If the bus doesn't come, only the umbrella helps.

Solution snippet: Take the umbrella!

## STATION 5: Reinforcement Learning — Learning Through Reward

```
BASIC (Part III)

CODE:
   # Learning logic
   if reward == 1:
      probabilities[index] += ______ # Gap 2
      print("✅ Dog is happy: 'That was good!'")
   else:
      probabilities[index] = max(0.1, probabilities[index] -
0.1)
      print("❌ Dog thinks: 'That wasn't great...'")

      # Normalize probabilities
      probabilities = normalize(probabilities)

      print("Current learning values:", [round(p, 2) for p in
probabilities])

# Result display
print("\n Training finished!")
for action, value in zip(actions, probabilities):
   print(f" {action}: {value:.2f}")

favorite_action =
actions[probabilities.index(max(probabilities))]
print(f"\n🐕 Dog's favorite action: {favorite_action}🦴")
```

**Hint 1:** This is where the current probabilities are stored, which the dog uses to choose an action.

**Hint 2:** How much should the probability increase when the dog is rewarded?

**Solution snippet 1:** probabilities

**Solution snippet 2:** 0.2

## STATION 5: Reinforcement Learning — Learning Through Reward

```
ADVANCED (Part I)

CODE:
print("Keep your plant alive!")
print("Actions: water, fertilize, do_nothing")

plant_state = {
   "water": 5, # 0-10
   "nutrients": 5,
   "health": 10
}

def evaluate_state():
    water, nutrients = plant_state["water"],
plant_state["nutrients"]
    if 3 <= water <= 7 and 3 <= nutrients <= 7:
      return ____ # Perfect!
    elif 1 <= water <= 9 and 1 <= nutrients <= 9:
      return _____ # Okay
    else:
      return -1 # Bad

for day in range(7):
    print(f"\n--- Day {day+1} ---")
    print(f"State: Water={plant_state['water']}/10,
Nutrients={plant_state['nutrients']}/10")

    action = input("What do you do? (water/fertilize/
nothing): ")
```

## STATION 5: Reinforcement Learning — Learning Through Reward

```
ADVANCED (Part II)

CODE:
    # Perform action
    if action == "water":
      plant_state["water"] = min(10, plant_state["water"] +
3)
    elif action == "fertilize":
      plant_state["nutrients"] = min(10,
plant_state["nutrients"] + 3)

    # Time passes - state decreases
    plant_state["water"] = max(0, plant_state["water"] - 1)
    plant_state["nutrients"] = max(0,
plant_state["nutrients"] - 1)

    # Calculate reward
    reward = evaluate_state()
    print(f"Reward: {reward} points")

    if plant_state["water"] == 0 or plant_state["nutrients"]
== 0:
      print("💀 The plant has died!")
      break

if plant_state["water"] > 0 and plant_state["nutrients"] > 0:
  print("❤️🌱 Plant survived! Well done!")
```

**Hint 1:** Points for ideal water and nutrient levels (maximum reward).
**Hint 2:** Points for acceptable levels, not perfect but still okay.

**Solution snippet 1:** 2
**Solution snippet 2:** 1

## STATION 6: Convolutional Neural Networks — How an AI Detects Edges in Images

Aylin looks at her photo.

**ALARA:** *"I see lines, brightness, and contrasts—these are my edges! When something changes, I know: a new object begins here."*

**ALARA:** *"To detect this, I use a little trick. I always compare two neighboring pixels. If they are different, an edge appears. The operator != means 'not equal' or 'differe nt from'."*

**ALARA:** *"Every picture you show me helps me 'see' better," Alara explains. "Just like you learn, I describe the world more and more accurately."*

```
BASIC (Part I)

CODE:
def count_edges(row):
    edges = 0
    for i in range(len(row)-1):
      if row[i] ___ row[i+1]:
         edges += 1
      return edges

print("Alara's 4-row image")

# Enter 4 rows from the user
image = []
for n in range(4):
  row = input(f"Enter row {n+1} (0 and 1, e.g., 100101): ")
  row = row[:6]
  image.append(row)
```

## STATION 6: Convolutional Neural Networks — How an AI Detects Edges in Images

```
BASIC (Part II)

CODE:
# Count edges per row
for i, row in enumerate(image):
    edges = count_edges(row)
    if edges > 0:
      print(f" Row {i+1}: {edges} edges")
    else:
      print(f"⬜ Row {i+1}: no edges")
```

**Hint:** A change (0 → 1 or 1 → 0) is an edge.
**Solution snippet:** i=

```
ADVANCED (Part I)

CODE:
print("Create your own 4x4 image!")
size = 4
image = []

# Input rows
for i in range(size):
    row = input(f"Enter row {i+1} (only ⬜ and ⬛, e. g.
⬛⬜⬜⬛): ")
    row = list(row.ljust(size, "⬜"))[:size]
    image.append(row)

print("\n Your image:")
for row in image:
    print(" ".join(row))
```

## STATION 7: Rule-based Natural Language Processing —How Computers Understand Language

Aylin writes a message to Lena: “Hey, that was great!”

**ALARA:** *“I can tell if you are happy, angry, or neutral —that's called sentiment analysis!”*

**ALARA:** *“Your messages help me understand human language better,”* Alara says gratefully. *“We’re learning together—you teach me AI, and I learn human communication.”*

```
BASIC

CODE:
print("Alara reads your mood!")
text = input("Write something (e.g., … that was great / …
that was bad): ")

def understand(text):
if "great" in text:
   return "😊 You’re in a good mood!"
elif "bad" in text:
   return ____
else:
   return "😐 Neutral."

print(understand(text))
```

**Hint:** “bad” = negative.
**Solution snippet:** "😠 You’re angry!"

## STATION 7: Rule-based Natural Language Processing —How Computers Understand Language

```
ADVANCED

CODE:
print("Counting words like a language model!")
text = input("Write a sentence: ")

def count_words(text):
   words = text.split()
   return len(____)

print("Number of words:", count_words(text))
```

**Hint:** What contains all words?.

**Solution snippet:** words

## STATION 8: Recommendation Systems —How YouTube Might Know

Aylin watches videos on her tablet.

**ALARA:** *“I remember what you like and suggest similar content!”*

**ALARA:** *“The more you show me what you like, the better I can offer suggestions,”* Alara explains. *“This is a partnership.”*

```
BASIC

CODE:
print("Alara's video recommendation")
video = input("What are you watching? (e.g., music video,
animal video): ")

def recommend(video):
    if "music" in video:
       return "🎶 Similar music videos!"
    else:
       return ____

print(recommend(video))
```

**Hint:** If not a music video, suggest something new..
**Solution snippet:** "Try something new!"

## STATION 8: Recommendation Systems —How YouTube Might Know

**ADVANCED**

**CODE:**

```
print("YOUR YOUTUBE ASSISTANT")
print("Find your perfect recommendation!")

name = input("\nWhat's your name? ")
print(f"Hello {name}! What do you like?")
favorite1 = input("Favorite Topic 1: ")
favorite2 = input("Favorite Topic 2: ")

user_data = {
   name: [favorite1, favorite2],
   "Lena": ["Sports", "Gaming"]
}

def recommendation(user_name):
   if favorite1 in ["Music", "music"]:
       return "🎵 YOUR TIP: Top 100 Charts Mix!"
   elif favorite1 in ["Sports", "sports"]:
       return "⚽ YOUR TIP: Best Goals Compilation!"
   else:
       return "🎬YOUR TIP: Trending Videos!"

print(f"\n{recommendation(____)}")
```

**Hint:** Which variable contains the name?

**Solution snippet:** name

## STATION 9: Weather Forecast — Random Forest

Aylin points to her watch.

**AYLIN:** *"How do you always know if we can play outside?"*

**ALARA:** *"That's my expert team! Three smart minds inside me consult each other—just like a random forest!"*

**AYLIN:** *"Wow! So you actually have three different experts in you?"*

**ALARA:** *"Yes! One looks at the temperature, one at the clouds, and one at the wind. Together we make the best decision!"*

**ALARA:** *"With every prediction, I get more accurate,"* Alara says. *"Your feedback helps me fine-tune my experts."*

**What's special about Random Forest:** In reality, a Random Forest trains many trees using randomly selected data and features. Our voting simulates this principle of "collective intelligence."

## STATION 9: Weather Forecast — Random Forest

```
BASIC

CODE:
print("AYLIN'S WEATHER EXPERTS")
print("Three experts are consulting!")

temperature = float(input("Temperature: "))
clouds = input("Clouds (sunny/cloudy): ")

# Ask the experts
expert1 = "Sun" if temperature > 20 else "Rain"
expert2 = "Sun" if clouds == "sunny" else "Rain"

opinions = [expert1, expert2]
decision = max(set(opinions), key=opinions.____)

print(f"Forecast: {decision}")
```

**Hint:** Count which opinion occurs most frequently.
**Solution snippet:** count

## STATION 9: Weather Forecast — Random Forest

```
ADVANCED

CODE:
print("AYLIN'S WEATHER COMMITTEE")
print("Three AI experts decide!")

temperature = float(input("Temperature: "))
clouds = input("Clouds (sunny/cloudy/rainy): ")
wind = input("Wind (strong/weak): ")

# Three experts give their opinions
expert1 = "Sun" if temperature > 20 else "Rain"
expert2 = "Sun" if ____ == "sunny" else "Rain"
expert3 = "Sun" if wind == "weak" else "Rain"

opinions = [expert1, expert2, expert3]
decision = max(set(opinions), key=opinions.count)

print(f"Forecast: {decision}")
```

**Hint:** Which variable contains the cloud information? .

**Solution snippet:** clouds

## Epilogue — The Second Gift

**A few months later.**

Aylin sits at her desk, Alara glowing on her wrist.
In the kitchen, dishes clatter—Aunt Sema is visiting and preparing börek.

**AYLIN:** “You know, Alara,” Aylin says with a smile, “when you were given to me on my birthday, I had no idea about AI. Now… I’m building my very first little project, and I understand how you ‘think.’”

Alara blinks proudly. “You haven’t just understood AI
—you've also taught me to ‘think’ more like a human.”

At that moment, Aunt Sema comes in with a small wrapped gift.

**AUNT SEMA:** “Canım” (sweetheart), she says gently,
“I’ve seen how much you’ve learned in the past months.
Your grandmother would say, You’ve given the watch a soul.”

Aylin opens the package. Inside is a simple notebook.
On the first page, written in flowing script:

*“Invent with heart. Program with courage.*
*Think like Alara — but feel like Aylin.”*

Aylin smiles.

**AYLIN:** *“This is the most beautiful gift, Aunt Sema.”*

**AUNT SEMA:** *“No,”* Sema says, tapping her lightly on the chest.
*“The most beautiful gift is already right here.”* **THE END**

## REFLECTION QUESTIONS

Take a moment to think about what you have learned in this module.
Answer the following questions thoughtfully.

- **Understanding AI:** How has your understanding of artificial intelligence changed after completing this module? What surprised you most about how AI systems like Alara “think”?

- **Neural Networks & Decision-Making:** In Station 2, you explored how neural networks use weights to make decisions. Can you think of a real-life situation where you weigh different factors before making a choice? How is this similar to how AI makes decisions?

- **Ethical Considerations:** Throughout the module, ethical questions about AI were raised. Why is it important to think about ethics when designing or using AI? Give one example of an ethical concern related to AI in daily life.

- **Learning Through Interaction:** Alara learns from Aylin’s interactions. How do you think AI systems in real life (like recommendation engines or voice assistants) learn from user interactions? What are the benefits and risks of this type of learning?

- **AI in Your World:** Where do you encounter AI in your daily life? Choose one example (e.g., social media, streaming services, smart devices) and explain how it uses one of the AI concepts you learned in this module.

## QUIZ WITH MULTIPLE-CHOICE QUESTIONS (one correct answer per question)

**1. What is Alara?**
a) Aylin's best friend
b) An ordinary alarm clock
c) An AI in a smartwatch
d) Aylin's aunt

**2. Which programming concept is introduced in Station 2 to make a decision based on multiple weighted inputs?**
a) Decision Trees
b) Reinforcement Learning
c) Neural Networks
d) Rule-based Classification

**3. Station 3, "The School Route Detective," is primarily about...**
a) Music recommendations
b) Route planning using decision trees
c) Image recognition
d) Language processing

**4. In Station 4, how does Alara classify a song as "Pop"?**
a) If it contains a guitar
b) If it is loud
c) If it is danceable
d) If it is by Taylor Swift

**5. What does the "Plant Care" exercise in Station 5 simulate?**
a) Image Recognition
b) Reinforcement Learning through rewards
c) Neural Networks
d) Decision Trees

## QUIZ WITH MULTIPLE-CHOICE QUESTIONS (one correct answer per question)

**6. In Station 6, how does Alara detect edges in an image?**
a) By comparing neighboring pixels for differences
b) By counting the white pixels
c) By using voice commands
d) By asking the user

**7. In Station 7 ("Natural Language Processing"), the mood of a text is recognized by...**
a) Measuring the length of the text
b) Searching for specific keywords like "great" or "bad"
c) Checking the grammar
d) Analyzing the font

**8. What does the recommendation system in Station 8 do?**
a) It creates weather forecasts
b) It suggests videos based on user interests
c) It recognizes handwriting
d) It guides a robot through a maze

**9. How does the "Random Forest" in Station 9 make a decision?**
a) Only one expert decides
b) Through a democratic vote of multiple "trees"
c) Through random selection
d) Through deep neural networks

**10. Which programming language is used in all the coding challenges?**
a) JavaScript
b) Python
c) N1GL1B
d) C++

# Module 2: AI Escape Room Treasure Hunt with Calculator

## INTRODUCTION

Have you ever wondered how your smartphone knows which videos you might like? The aim of the current module is to introduce the fundamental concepts that explain how Artificial Intelligence (AI) "thinks," makes decisions, and learns from data. The content provides a simple theoretical background that connects mathematics, logic, and creativity through an engaging, game-based learning experience.

By the end of the module, you will be able to acquire different skills, such as:

- **Mathematical Competence:** Enhanced proficiency in performing basic arithmetic operations, understanding averages, percentages, and ratios. Building confidence in using calculators for problem-solving.
- **Critical Thinking and Logical Reasoning:** Students will practice logical thinking as they decode clues and make decisions based on the given data. Problem-solving abilities are strengthened as they apply math to practical scenarios and puzzles.
- **Decision-Making:** By solving tasks related to classification (e.g., deciding which object fits a certain description), students practice making informed decisions based on numerical data and logic.

| | |
|---|---|
| Module Duration | 2 hours (1 hour learning + 1 hour practical exercises) |

## EDUCATIONAL MATERIALS

### UNIT 2.1 WHAT IS BEHIND THE AI ESCAPE ROOM?

Artificial Intelligence can be defined as the ability of machines to perform tasks that usually require human intelligence, such as reasoning, problem-solving, learning, or recognizing patterns. Mathematics is at the core of AI — probability, statistics, and logic help computers make decisions based on data rather than intuition. Data, in turn, acts as the "fuel" of AI: the more quality data available, the better the system becomes at detecting patterns and making predictions. In this module, you will discover how AI uses reasoning, probability, and learning to solve real problems in creative ways.

To understand these ideas, we will explore four key principles that represent how AI "thinks":

- **Bayesian Networks,** which help predict outcomes when information is uncertain.
- **K-Means Clustering,** which groups similar data to find patterns and relationships.
- **Monte Carlo Methods,** which use random experiments to estimate probabilities and make predictions.
- **Hebbian Learning,** which shows how connections become stronger through repetition, just like our brain when learning new things.

### UNIT 2.2 WHAT IS BEHIND THE AI ESCAPE ROOM?

Through these simple ideas, you will understand that AI is not only about coding—it's also about logical reasoning, problem-solving, and creativity. These principles will guide you through each stage of the AI Escape Room, where every challenge will reveal how artificial intelligence "learns by doing."

These four ideas show how AI combines logic, math, and creativity to learn and make smart decisions—exactly what you will do while solving the puzzles in this AI Escape Room adventure!

**"The purpose of AI is not to replace humans, but to amplify human capabilities."**
*— Fei-Fei Li (Stanford University, AI researcher)*

## REFERENCES

### Artificial Intelligence & Decision-Making

- Russell, S., & Norvig, P. (2021). Artificial Intelligence: A Modern Approach (4th ed.). Pearson.

### Probability, Simulation, and Learning

- Shalev-Shwartz, S., & Ben-David, S. (2014). Understanding Machine Learning: From Theory to Algorithms. Cambridge University Press. → Foundations of machine learning, clustering, and learning from data.

- Khan Academy. (n.d.). Statistics and probability. https://www.khanacademy.org/math/statistics-probability
  → Simple and accessible explanations of probability, averages, and simulations.

## LINKS TO LEARNING PLATFORMS

If you want to keep learning about Artificial Intelligence in a fun and simple way, these online platforms are a great place to start! You can play, experiment, and explore how computers learn to recognize images, sounds, or patterns — just like in the AI Escape Room. Each link includes short activities that help you understand how AI works in real life while you keep learning by doing.

- Google Teachable Machine – Interactive platform to train simple AI models with images, sounds, or poses.
  https://teachablemachine.withgoogle.com/

- Khan Academy – Artificial Intelligence and Machine Learning – Free lessons explaining basic AI concepts using everyday examples.
  https://www.khanacademy.org/computing/computer-science

- AI for Oceans (Code.org) – Fun interactive activity where students train an AI to clean the ocean, learning about data and bias.
  https://studio.code.org/s/oceans

- Machine Learning for Kids – Platform that lets students experiment with AI models and use them in Scratch projects.
  https://machinelearningforkids.co.uk/

- IBM SkillsBuild for Students – Free online learning platform offering basic AI and data literacy courses with badges and certificates.
  https://skillsbuild.org/students

- Google AI Experiments – A collection of interactive tools showing how AI recognizes patterns, sounds, and movement.
  https://experiments.withgoogle.com/collection/ai

## PRACTICAL EXERCISE

### Welcome to the AI Escape Room!

You are part of a team of young explorers searching for a treasure locked behind a 4-digit digital lock. To open it, you must advance through a path through 4 rooms protected by four guardians, each representing a principle of Artificial Intelligence:

1. Probability (Bayesian Networks)
2. Clustering (K-Means Clustering)
3. Simulation (Monte Carlo Methods)
4. Hebbian Learning

Your mission is to overcome the challenges at each station, solve their problems, and then obtain the combination that opens the lock. Each station contains an explanation, with short exercises in between, and a final programming challenge with three difficulty levels that will give you one of the digits you need to open the lock.

### Station 1 – Bayesian Networks

A **Bayesian Network** is a mathematical model that uses **probability** to make decisions or predictions when not all the information is known with certainty, that is, when there is **uncertainty**.

### Mini-Challenge 1 — The Simple Lock

Imagine you want to know if you can open a number lock in an escape room.

- **Event A – Tracks:** There are several hidden clues. The more you find, the easier it will be to open the lock.
- **Event B – Riddle:** There's a riddle that provides the numbers for the combination. The more clues you find, the easier it is to solve.
- **Event C – Lock:** Whether or not you can solve the puzzle and enter the numbers into the lock to open it depends on the number of clues you find.

## Mini-Challenge 1 — The Simple Lock

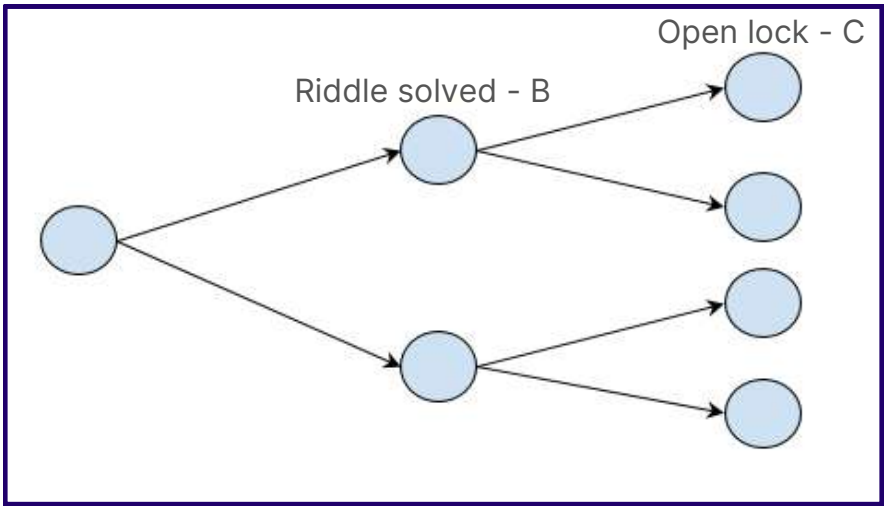



Figure 2 Decision and Probability Path: From Finding Clues to Opening the Lock.
Source: Monica Moreno

**Note:** half of the graph is missing,
corresponding to the probability of not finding clues.

**Instructions:**
If you find the clues (probability 0.8) and solve the riddle (probability 0.9 if you found the clues), what is the probability that you open the lock (probability 0.95 if you solve the riddle)?

The path to finding the solution is made up of a number of probabilities,
which must be multiplied.

**Answer:** P(Open Lock)=0.8 × 0.9 × 0.95

**Ask:** What result would you get if you added up all the possible paths?

**Answer:** 1, because each path represents the probability of a possible scenario occurring. The probability that one of all possible outcomes occurs is always 1.

## Bayes' Rule

So far, we've seen how we can make a prediction, but what happens if we look at the outcome and want to know the probability that a path was followed?

## Mini-Challenge 2 — The Conditional Corridor

You find a secret corridor with another riddle on the wall: “Of all the times you solved the riddle, what is the probability that you would have found the clues first?”

In other words, we want to calculate the probability of C towards B
(**which is different to the probability of B towards C**). You must use Bayes' Rule:

**Instructions:**
First, we must calculate the probability of B=✓ and C=✓, as all paths ending in B=✓, and all those end in C=✓. To simplify things, we'll give you the operations already made: P(B=✓)= 0.76, P(C=✓) = 0.734.

Now we just need to apply Bayes' rule with the formula:

P(B=✓ if C=✓) = [P(C=✓ if B=✓) * P(B=✓)] / P(C=✓)

**Answer:** P(B=✓|C=✓) = [0.95 * 0.76] / 0.724 = 0.983

## Challenge 1 - The Guardian of Probability

You have reached the first room, the Chamber of the Keeper of Probability, a mysterious hall lit by digital torches. In the center stands an ancient chest covered in many precious stones. Then, the guardian, a hologram of an old man, tells you:

> *“Only those who understand how the **uncertainty** will be able to discover if the chest contains gold. Probability is the key.”*

*Your mission is to calculate the **probability** that the chest contains gold based only on what you can see, using a **simplified Bayesian Network model.***

***Scenario:*** *What is the probability that **this heavy, shiny chest** contains gold? The 1st decimal of the answer will be the 1st digit of the lock.*

- You know that **30% of shiny chests contain gold** and **70% contain stones**.
- The chests containing gold are **heavy in 80% of cases.**
- The chests containing stones are **heavy in 40% of cases.**

## Challenge 1 - The Guardian of Probability

**EASY LEVEL:** Draw the Bayesian Network in a paper and use the calculator to get the answer. **Clue:** You only need 2 events (A and B) and Bayes' Rule.

**MEDIUM LEVEL:** Modify this incomplete code to solve the problem. Just add the probabilities and mathematical symbols to operate.

```
BASIC

CODE:
p_gold =
p_stones =
p_heavy_if_gold =
p_heavy_if_stones =

# Calculate the probability that it is heavy
p_heavy = (p_gold “operation” p_heavy_if_gold) + (p_stones
"operation" p_heavy_if_stones)

# Calculate probability of gold given that it is heavy
p_gold_given_weighed = (p_gold "operation" p_weighed_if_gold)
"ooperation” p_weighed

print("Probability of gold given heavy:",
round(p_gold_given_weighed, 2))
```

**HARD LEVEL:** Draw the Bayesian Network and create your own code to solve the problem.

## Station 2 — K-Means Clustering

**K-Means** is an algorithm that **groups data** in categories based on **similarity**.
To understand this, let's look at an example. Imagine you've measured the temperature almost every day of the year and want to know which season each point corresponds to. This is what's called a classification problem.

## Mini-Challenge 1 — The Crystals in the Hallway

On the way to the next room, you find several glasses on the floor. Each crystal has two numbers written on it: brightness and purity. Group these crystals into two distinct groups according to their characteristics.

**Instructions:**

- Draw two axis (x,y) and mark the points:
  [1,5], [4,1], [3.5,0], [2,4], [0.5, 5], [0.5, 5], [5, 2].
- Imagine you want two groups or **clusters** (k=2).
  Where would you put the initial center of each group or **centroids**?
- Assign each crystal to the nearest centroid.

**Answer:** Group 1: [1,5], [2,4], [0.5, 5], [0.5, 5]. Group 2: [4,1], [3.5,0], [5, 2]

K-Means **automates the process of forming groups**, so that each one has a "center" (centroid) which represents the **average of the characteristics** of that group. "How K-Means Works"

1. Choose how many clusters there are (k=n).
2. Place k centroids randomly on the plane, one for each group.
3. Each point on the plane is assigned to the nearest centroid.
4. Calculate the "center" of each area to relocate the centroid.
5. Repeat the process until the points no longer change groups.

## Challenge 2 - The Alchemist

Once you've passed the first room, you'll reach a laboratory filled with many flasks of mysteriously colored liquids. Then, an **old alchemist** tells you:

"Young adventurer, you won't be able to leave here without sorting my potions. Each bottle belongs to a different type of magic: **Healing, Strength or Invisibility.** If you can help me group them correctly, I'll open the door for you.

Each jar has **two chemical characteristics**: pH and concentration of magical energy. Your mission is to use **K-Means** to **group the jars** and thus pass the second test. The number of clusters you create will correspond to the second digit of the lock.

**EASY LEVEL:** Use the following code to group the jars, filling the missing parameters. **EXTRA:** see how the result changes when varying the number of groups.

**NOTE:** You need to install the sklearn library first (pip install scikit-learn)

**MEDIUM LEVEL:** Use the code from the easy level and add a new jar (point) with the coordinates you want. Use the function kmeans.predict(point)to get the classification. Add this code at the end to display the result.

**HARD LEVEL:** Create the code from scratch.

**HARD LEVEL:** Draw the Bayesian Network and create your own code to solve the problem.

## Challenge 2 - The Alchemist

**BASIC**

**CODE:**

```
from sklearn.cluster import KMeans  # kmeans library
import numpy as np
import matplotlib.pyplot as plt

# Data: [pH, power]
potions = np.array([
   [2, 53], [6, 70], [3, 55], [3.1, 52], [10, 25],
   [8, 84], [9, 80], [7.5, 82],
   [10, 19], [10.5, 22], [4, 48], [9, 20],
])

# K-Means
kmeans = KMeans(n_clusters=2, random_state=0) # Algorithm
declaration
kmeans.fit(potions) # Executes kmeans to classify the data

# Visualization
plt.scatter(potions[:, 0], potions[:, 1], c=kmeans.labels_,
cmap="viridis", marker="o")
plt.xlabel("pH")
plt.ylabel("Magic Power")
plt.title("Potion classification with K-Means")
plt.show()

print("Classification of each jar:", kmeans.labels_)
```

**HARD LEVEL:** Draw the Bayesian Network and create your own code to solve the problem.

## Challenge 2 - The Alchemist

```
MEDIUM

CODE:
plt.scatter(potions[:, 0], potions[:, 1], c=kmeans.labels_,
cmap="viridis", marker="o")
plt.scatter(
   new_jar[:, 0], new_jar[:, 1],
   c=[plt.cm.viridis(prediction[0] / (kmeans.n_clusters -
1))],
   marker="x", s=150, label="new jar"
)
plt.xlabel("pH")
plt.ylabel("Magic Power")
plt.title("Special Bottle Classification")
plt.legend()
plt.show()
```

**HARD LEVEL:** Draw the Bayesian Network and create your own code to solve the problem.

## Challenge 2 - The Alchemist

**HARD**

**CODE:**

```
from sklearn.cluster import KMeans
import numpy as np
import matplotlib.pyplot as plt

# Step 1: Define the data (list of pairs [pH, energy] with
numpy)
potions = np.array([
    # add your data here
])

# Step 2: Create the KMeans model specifying the number of
clusters
kmeans = KMeans(n_clusters=2, random_state=0)

# Step 3: Train the model with your data (fit)
kmeans.fit(potions)

# Visualization
plt.scatter(potions[:, 0], potions[:, 1], c=kmeans.labels_,
cmap="viridis", marker="o")
plt.xlabel("pH")
plt.ylabel("Magic Power")
plt.title("Potion classification with K-Means")
plt.show()

# Step 4: Print the result
print("Classification of each jar:", kmeans.labels_)
```

## Station 3 — Monte Carlo Methods

Monte Carlo methods are a way of predicting the probability with which an event will occur by simulating the same experiment many times and counting how many times that event occurs as an outcome.

## Mini-Challenge 1 — The Fountain of Fortune

After walking for a while without finding the next room, you start to feel lost. You arrive to a small room with a **magic fountain**, and you decide to make a wish to find your way.

The inscription on the stone reads:

> "If you flip a coin, there's a 50% chance that your wish will come true.
> But what if you try it many times?"

**Instructions**

- Use the calculator to generate ten random numbers. Even numbers represent 'heads,' and odd numbers represent 'tails.'
- Count the number of times the coin lands on heads and divide it by the total number of tosses (N=10). The result is the probability.

To use Monte Carlo methods, there must always be **randomness** in experiments. Without chance, there would be no variation in outcomes, meaning we would be unable to estimate probabilities.

## Challenge 3 - The Keeper of the Destiny Dice

After a long time walking down a stone corridor lit by torches, you find a room with a **circular table** covered with scrolls, colored stones, and some **data**.

Sitting behind the table is the **Guardian of Chance**, a hooded skeleton with a deep voice who takes 3 dice and tells you:

*"Traveler, this room does not open with force or ingenuity...*
*I am the guardian of the **infinite Monte Carlo games**, and I propose a challenge.*
*I'm going to roll these three dice and add up the numbers.*
*Choose: **more than 12 or less than/equal to 12.***
*If you get it right, you'll be able to move forward,*
*but if you don't, you'll be locked here forever."*

To decide which option to choose, you decide to apply your knowledge of probability and use Monte Carlo methods to determine your best option. The first decimal place of the largest probability will give you the third digit of the lock.

**EASY LEVEL:** Follow the steps below to create a code that allows you to calculate the probability of success in each case for N=10,000 experiments.

1. Create 2 variables to count how many times each result appears.
   **Clue:** You must initialize the variables to 0.
2. Generate 3 random numbers and add them.
   **Clue:** You must use the function random.random().
3. Check if the sum is >12 or =<12 and add 1 to the corresponding counter.
4. Repeat the process 10,000 times.
   **Clue:** You must put the code inside a "for" loop.
5. Divide the counters by N=10,000 to get the probability and choose the most likely one.

## Challenge 3 - The Keeper of the Destiny Dice

**MEDIUM LEVEL:** Write the code to simulate the experiment for N=10,000 iterations (you can use the easy level steps if you need to), but you must also calculate the probability of each possible outcome (from 3 to 18) when rolling the dice.

**Clue:** You must use a dictionary for the counters. Run all the experiments and sum to the counter. Finally, after all experiments, divide all the values by N.

**HARD LEVEL:** Create a program from scratch that, for any number of dice, repetitions, and threshold value, calculates the probability of each possible outcome.

**Clue:** You must create a function with N, num_dices, and threshold as parameters that returns a dictionary with the possible results as key and the probabilities as value.

**EXTRA (expert level):** Adds a visualization of the probability distribution that allows viewing in a histogram (plt.bar) the probabilities of each value.

## Station 4 – Hebbian Learning

Hebbian learning is a learning rule for neural networks based on the sentence: “Neurons that fire together, wire together.” In other words:

- If two neurons are activated at the same time, the connection between them **strengthens**.
- If one is activated and the other is not, the connection **weakens**.

## Mini-Challenge 1 — The Library of Ideas

On the way to the final room, you pass through a dark hallway where there are two statues with glowing symbols. Every time you say "light," the two statues light up together. But if you say "shadow," only one glows.

According to Hebb's rule, what will happen to the connection between these two statues if you repeat "light" many times?

a) It will weaken
b) It will remain the same
c) **It will be stronger**

## Mini-Challenge 1 — The Library of Ideas

To calculate the connection weight between two neurons after they fire, the Hebbian formula:

$$w_{final} = w_{initial} + (n_{together} \times \Delta w_{+}) + (n_{alone} \times \Delta w_{-})$$

Where:

- $w_{initial}$ = initial weight of the connection
- $n_{together}$ = number of times both neurons fire together
- $\Delta w_{+}$ = weight gain due to joint activation
- $n_{Alone}$ = number of times only one neuron fires
- $\Delta w_{-}$ =weight loss when only one is activated

## Mini-Challenge 2 — Connection Weights

To properly illuminate the hallway, you must calculate the activation between the two statues (or neurons). To do this, you must use the Hebbian formula.

**Instructions**

- The initial weight of the connection is 0.2.
- Every time both are turned on together →**+0.1**
- Every time only one lights up →**–0.05**
- Both lights turn on together 3 times but light A turns on by itself 2 times.

**Answer:** $w_{final} = 0.2 + (3 \times 0.1) + (2 \times 0.05) = 0.6$

## Challenge 4 - The Neuron Mural

You are very close to the end, but you still have to overcome 1 last challenge. In the last room, you find a mural with 3 interconnected magic lights (A, B, C) connected to each other with threads that represent connection weights.
A mysterious voice tells you:

*"To open the final door, you must discover which connections have been strengthened. Calculate the final weights after several activations and reveal the secret pattern."*

Every connection starts with **weight 0**. If two neurons fire together, the connection increases by 0.2, while if only one fires, the weight decreases by -0.07. There are three connections: AB, AC, and BC. The last digit of the lock corresponds to the last decimal place of the AC connection's weight after round 4. Activations per round:

| Round | Activated neurons |
|---|---|
| 1 | A,B |
| 2 | B,C |
| 3 | A |
| 4 | A, C |
| 5 | A, B, C |

**EASY LEVEL:** Do the calculations for the AB connection only, with a table showing the connection weight result for each round.

**MEDIUM LEVEL:**
Calculate the final weights of all connections across all rounds.

**HARD LEVEL:** Calculate the final weights of all connections across all rounds, assuming that if all three neurons fire simultaneously, all connections receive a bonus of +0.2. Identify the strongest connection from the last round.

## REFLECTION QUESTIONS

Take a moment to reflect on what you learned in this module. Answer the questions below carefully.

- **AI, Maths and Decision-Making:** In this module, you used maths, probability, and logic to solve the AI Escape Room challenges. How has this activity helped you understand how AI makes decisions? Which challenge helped you understand this best?

- **Learning Through Probability and Simulation:** You explored different AI methods such as Bayesian Networks, K-Means Clustering, Monte Carlo methods, and Hebbian Learning. Can you explain one of these ideas using your own words and give an example of how it was used in the escape room?

- **Learning by Repetition and Experience:** Some challenges required repeating calculations, simulations, or trials to reach the correct answer. How is this similar to how AI systems learn from data and experience? Can you think of a real-life example where AI learns in this way?

## QUIZ WITH MULTIPLE-CHOICE QUESTIONS (one correct answer per question)

**1. What do Bayesian Networks mainly help us with?**
a) Grouping similar data points
b) Making decisions under uncertainty
c) Speeding up calculations with random sampling
d) Strengthening neuron connections

**2. If 30% of shiny chests contain gold, and 80% of gold chests are heavy, which rule do we use to find the probability that a heavy chest has gold?**
a) Monte Carlo Simulation
b) Hebbian Learning
c) Bayes' Rule
d) K-Means Algorithm

**3. What does the "k" in K-Means represent?**
a) The number of data points
b) The number of clusters
c) The number of dimensions
d) The number of probabilities

**4. Which step comes right after assigning points to the nearest centroid?**
a) Randomizing the centroids again
b) Calculating the new centers of each cluster
c) Drawing a Bayesian Network
d) Estimating probabilities with dice

**5. Why are Monte Carlo methods useful in probability estimation?**
a) They eliminate randomness completely
b) They simulate many experiments with randomness
c) They group data into categories
d) They strengthen network connections

## QUIZ WITH MULTIPLE-CHOICE QUESTIONS (one correct answer per question)

**6. In the "Keeper of the Destiny Dice" challenge, what is being calculated?**
a) Which chest has gold
b) Which cluster a potion belongs to
c) The probability of dice sums being greater or less than 12
d) The final neuron connection weight

**7. According to the phrase "neurons that fire together, wire together," what happens when two neurons activate simultaneously many times?**
a) The connection weakens
b) The connection disappears
c) The connection strengthens
d) Nothing changes

**8. If two neurons fire together 3 times (+0.1 each) and only one fires 2 times (–0.05 each), starting weight = 0.2, what is the final weight?**
a) 0.35
b) 0.60
c) 0.10
d) 0.45

**9. Which of the following best distinguishes Bayesian Networks from Monte Carlo methods?**
a) Bayesian Networks use probability rules, Monte Carlo uses simulations
b) Both are the same method with different names
c) Bayesian Networks are random, Monte Carlo is deterministic
d) Monte Carlo strengthens connections like Hebbian learning

**10. In artificial neural networks, Hebbian learning mainly helps with:**
a) Clustering data into groups
b) Strengthening connections between co-activated neurons
c) Predicting outcomes with uncertainty
d) Running thousands of random simulations

# Module 3: Scratch Meets Artificial Intelligence

## INTRODUCTION

The aim of this module is to introduce the programming language Scratch and understand how Scratch can be an easy and effective tool for introducing programming concepts.

By the end of the module, you will be able to acquire different skills, such as:

- **Understand how Scratch** provides an accessible platform for learning programming concepts.
- **Master fundamental block-based** programming skills and computational thinking.
- **Integrate artificial intelligence** extensions into creative Scratch projects.

| | |
|---|---|
| Module Duration | 2 hours (1 hour learning + 1 hour practical exercises) |

## EDUCATIONAL MATERIALS

Scratch is a powerful programming language that is visual-based, meaning, unlike traditional programming languages, Scratch does not rely on syntax or line-based code; it works by connecting blocks together to create logic. There are many advantages of using Scratch as an introduction to programming. In this module, you will learn about the history of Scratch and how to code using the online platform. At the end of the module, you will learn about adding AI extensions to Scratch applications.

## UNIT 3.1 INTRODUCTION TO SCRATCH

Scratch is a powerful visual programming language and online community developed by the MIT Media Lab. Unlike traditional programming languages, which require learning and typing complex code, Scratch uses blocks of code that snap together like puzzle pieces. This concept makes programming accessible to everyone regardless of their experience level and coding knowledge.

### WHY SCRATCH IS A GREAT TOOL FOR LEARNING PROGRAMMING

The key benefits of learning to code using Scratch are that it eliminates many barriers that traditionally make programming challenging for new coders.

For example, Scratch has the following differences from traditional programming languages:

- **No Syntax Errors:** Blocks of code in Scratch only connect in ways that make logical sense, preventing frustrating syntax mistakes.
- **Immediate Visual Feedback:** Scratch allows you to see the results of your code instantly on the stage.
- **Low Floor, High Ceiling:** Scratch is easy to learn, but also can create sophisticated projects.
- **Creative Freedom:** Build games, animations, interactive stories, simulations, and more..
- **Collaborative Community:** Share projects, remix others' work, and learn from peers.
- **Cross-Platform:** Scratch works in all leading web browsers on any device, including tablets and smartphones.

## REAL-WORLD APPLICATIONS OF SCRATCH

Scratch may be a block-based language with no syntax; it does, however, teach fundamental programming concepts that can be directly transferred into other languages such as Python, JavaScript, and Java.

Scratch handles programming concepts in the following ways:

| Programming Concept | In Scratch | Real-World Application |
|---|---|---|
| Sequences | Blocks stacked in order | Step-by-step |
| Loops | Repeat blocks | Automating repetitive |
| Conditionals | If-then blocks | Decision-making logic |
| Variables | Data storage blocks | Storing and managing |
| Events | Blocks that trigger | User interaction |
| Parallelism | Multiple scripts running | Multi-threaded |

## UNIT 3.2 BASIC BLOCK-BASED PROGRAMMING SKILLS

### HOW TO CREATE A SCRATCH ACCOUNT

Opening an account is simple and free. Start by visiting the Scratch website, https://scratch.mit.edu/. Then create an account by selecting 'Join Scratch' via the main navigation and follow the registration instructions.

### NAVIGATING THE SCRATCH INTERFACE

The interface of Scratch controls how a program looks on the screen. There are seven different elements to Scratch that you need to know to be able to start coding on the platform. They are:

- Interface
- Sprites
- Stage
- Blocks
- Costumes
- Backdrops
- Sounds

The Scratch interface is divided into three main areas: the Stage (top right), where projects are displayed; the Code Area (screen center), where you can drag and assemble blocks to create scripts; and the Sprite Area (bottom right), where you can add and manage characters, or sprites, and backdrops.

The left side of the interface contains the Block Palette, a toolbox with distinct categories of code blocks to build programs visually. To recap, the main elements of the interface are:

- Sprite Area (Bottom Right): Shows all sprites (characters and objects) in your project. Click a sprite to edit its code.
- Code Area (Centre): The workspace where you drag and connect blocks to create scripts for the selected sprite.
- Stage (Top right): The visual output area where your sprites perform and interact. This is what your audience sees.

## UNDERSTANDING BLOCK CATEGORIES

Scratch organises code blocks into nine colour-coded categories. Each category serves a specific purpose in your program; the categories are as follows:

1. **Blue - Motion:** The motion category enables the moving of your sprite around the screen. If you want to make a sprite move forward, you can use a Change X By 10 Block.

2. **Purple – Looks:** The looks category focuses on changing the appearances of sprites and backdrops. You could use a Switch Costume Block to change the appearance of a sprite.

3. **Magenta – Sound:** The sound category focuses on making sprites make / play sounds. You could use a 'Play Sound Until Done' block to play a sound.

4. **Yellow – Events:** All Scratch project needs an event; this makes things happen when interactions are made.

5. **Light Orange – Control:** The control category is all about how much a sprite should do something or how long a sprite should stay still. If you drag a Wait 1 Second Block into your program, the program will wait 1 second before it starts.
6. **Teal – Sensing:** This category is about checking if an object is touching another object so it can make an action happen.
7. **Green – Operator:** These controls can be used for performing mathematical and logical operations, such as addition, subtraction, and string concatenation.
8. **Dark Orange – Variables:** Variables are used to store and manage information like scores or health points.
9. **Red - My Blocks:** In this category you can make your own custom blocks.

## UNIT 3.3 HOW TO CREATE PROJECTS IN SCRATCH

To create a project in Scratch, start by adding a character (sprite) and a background from the library.

### Adding a New Sprite

In the bottom-right corner of the Scratch editor, you will find the sprite controls:

- Choose a Sprite: Browse the built-in library of sprites (animals, people, objects, fantasy characters, etc.)
- Paint: Create your own sprite using the drawing tools
- Surprise: Add a random sprite for inspiration
- Upload Sprite: Import an image file from your computer.

### Changing the Backdrop

The backdrop sets the scene for your project. Click the backdrop icon (bottom right of the stage) to:

- Choose from pre-made backdrops (outdoors, indoors, space, abstract, etc.)
- Paint your own using the backdrop editor.
- Upload an image from your computer.

You can have multiple backdrops and switch between them programmatically using switch backdrop to [name] blocks. This is useful for creating multi-scene games or stories.

## MOVE A SPRITE AND TRIGGER AN ACTION

1. **Step – Add an Event Block:** Click on the Events category (yellow). Drag the 'when green flag clicked' block into the code area. This block starts your program when you click the green flag above the stage.

2. **Step – Add Movement:** Click on the Motion category (blue). Drag the 'move 10 steps' block and snap it underneath the event block. Change the number from 10 to 50 by clicking on it.

3. **Step – Add a Sound Effect:** Click on the Sound category (magenta). Drag 'play sound [meow]' until done and connect it under the move block.

4. **Step – Make the Sprite Speak:** Click on the Looks category (purple). Drag 'say [Hello!] for 2 seconds' and add it to your script. You can change "Hello!" to any message you like.

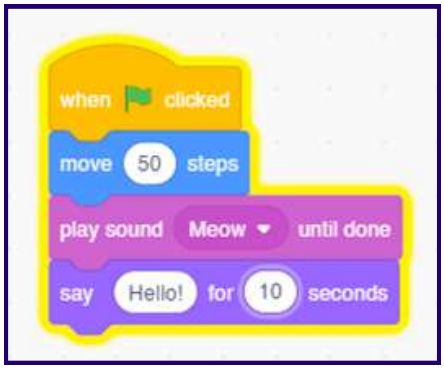



Figure 3 Scratch block logic for movement and sound interaction: Scratch by MIT Media Lab

## UNIT 3.4 INTRODUCTION TO SCRATCH AI EXTENSIONS

### WHAT IS AN EXTENSION IN SCRATCH?

Extensions are add-on modules that expand Scratch's capabilities beyond the standard block categories. Think of extensions as specialised toolkits that give your sprites new powers, from speaking text aloud to recognising images through artificial intelligence.

Scratch 3.0 includes software extensions, which add computational features and hardware extensions, which connect to physical devices like robots and electronic kits. In this unit we focus on AI-powered software extensions that bring machine learning and natural language processing to your projects.

### ADDING AN EXTENSION IN SCRATCH

Adding an extension is simple:

1. Look for the button with a "+" symbol at the bottom-left of the blocks area.
2. Click the symbol to open the Extension Library
3. Browse available extensions and click one to add it to your project.
4. New blocks will appear in your palette immediately ready to use.

Most AI extensions require an active internet connection because the artificial intelligence processing occurs on cloud servers, not on your computer. Make sure you are connected to the internet before using these extensions. Additionally, be mindful of privacy, avoid uploading personal or sensitive information when training machine learning models.

## WHY AND HOW AI IS USED IN CREATIVE PROJECTS

Artificial intelligence enables your Scratch projects to do things that would be almost impossible with traditional programming alone:

| AI Capability | In Scratch | Real-World Parallel |
|---|---|---|
| Image Recognition | A game that identifies | Google Photos |
| Text Classification | A chatbot that understands whether messages are questions, statements, or commands | Email spam filters categorising messages |
| Speech Synthesis | Characters that read stories aloud or narrate game events | GPS navigation systems providing voice directions |
| Sentiment Analysis | A virtual pet that reacts differently to kind versus mean words | GPS navigation systems providing voice directions |
| Pattern Recognition | A music app that identifies whether sounds are drums, piano, or singing | Shazam identifying songs from short audio clips |

AI extends creative possibilities by making projects adaptive (responding differently based on input), intelligent (making decisions from learned patterns), and interactive (understanding multiple forms of input like images, text, and sound).

## MOVE A SPRITE AND TRIGGER AN ACTION

### OFFICIAL SCRATCH AI EXTENSIONS

- **Text-to-Speech:** Converts written text into spoken words with multiple voices and languages.
- **Translate:** Translates text between languages using machine translation
- **Video Sensing:** Detects motion and presence via webcam (basic computer vision)

### THIRD-PARTY AI EXTENSIONS

- **Machine Learning for Kids:** Train custom models for image, text, number, and sound recognition
- **Teachable Machine:** Google's tool for quick model training, integrable with Scratch
- **Face Sensing:** Detects faces and facial features through a webcam.
- **Speech Recognition:** Converts spoken words into text (voice control)

## UNIT 3.5 AI EXTENSION: MACHINE LEARNING FOR KIDS

Machine Learning for Kids (ML4K) is an educational platform developed by IBM that makes machine learning accessible to learners of all ages. It provides a user-friendly interface for training AI models and seamlessly integrates with Scratch through custom blocks.

With ML4K, you can train models to:

- Recognise and classify **images** (e.g., distinguishing cats from dogs, or identifying hand-drawn shapes)
- Understand and categorise **text** (e.g., detecting whether messages are compliments or insults)
- Identify patterns in **numbers** (e.g., predicting outcomes based on numerical input)
- Recognise **sounds** (e.g., differentiating between musical notes or voice commands)

## WHAT IS MACHINE LEARNING?

Machine learning is a subset of artificial intelligence where computers learn to perform tasks by analysing examples, rather than following pre-programmed rules. For instance, instead of programming every dog breed description, you show the computer thousands of dog images, and it learns to recognise features that define dog breed characteristics. This is how services like Google Photos automatically tag your pictures.

## STEP-BY-STEP: TRAINING AND USING A MODEL

### STEP 1: ACCESS MACHINE LEARNING FOR KIDS

Open your web browser and navigate to
https://machinelearningforkids.co.uk/

Click "Get Started" and for this example
choose to create an account without registering.

### Step 2: Create a New Project

Once logged in, click "Projects" and then "Add a new project".
You will be prompted to:

- **Name your project:** Choose a descriptive name (e.g., "Emotion Detector" or "Recycling Sorter")
- **Select recognition type:** Choose from Text, Numbers, Images, or Sounds

For this tutorial, select "Text" to build a sentiment classifier.

### Step 3: Define Your Labels (Categories)

Machine learning models classify inputs into categories called "labels."
For a sentiment classifier, create two labels:

- **Positive:** Happy, enthusiastic, or kind messages
- **Negative:** Sad, angry, or mean messages

Click "Train" and then "Add new label" for each category.

**Step 4: Provide Training Examples**
The model learns by studying examples.
For each label, provide at least 5-10 sample phrases:

| Positive Examples | Negative Examples |
|---|---|
| This is amazing! | I do not like this. |
| You are doing great! | This is terrible. |
| I love learning new things | This makes me angry. |

**Tip:** The more examples you provide, the better your model will perform.
Aim for at least ten examples per label and ensure variety in phrasing.

**Step 5: Train Your Model**
Once you have added sufficient examples, return to the project and select “Learn & Test” then click the "Train new machine learning model" button. The system will analyse your examples and create a model that can classify new text. Training typically takes 1-3 minutes. You will see a progress indicator. When complete, you will receive a confirmation message.

**Step 6: Test Your Model**
Before using your model in Scratch, test it within ML4K, simply type a phrase you did not include in your training data (e.g., "I am so happy today!") and click “test”. The model will predict which label best matches your input and show a confidence percentage.

If the model makes mistakes, return to the training section
and add more examples to improve accuracy.

**Step 7: Connect to Scratch**

ML4K provides a special Scratch integration:

1. Click "Make" in the project navigation.
2. Select "Scratch 3"
3. This opens a modified Scratch editor with your ML model pre-loaded as custom blocks.

You will see new blocks specific to your project, such as:

- recognise text [ ] (label)
- recognise text [ ] (confidence)

**Step 8: Use ML Blocks in Your Scratch Project**

Create a simple Scratch project that uses your trained model: Start by adding a sprite (try the "Robot" sprite), then create a variable called "user input" and then build this script:

*When green flag clicked ask [Tell me how you are feeling today] and wait set [user input] to (answer) if <recognise text (user input) (label) = [Positive]> then say [I am glad you are feeling good!] for 3 seconds else say [I am sorry you are feeling down. Things will improve!] for 3 seconds.*

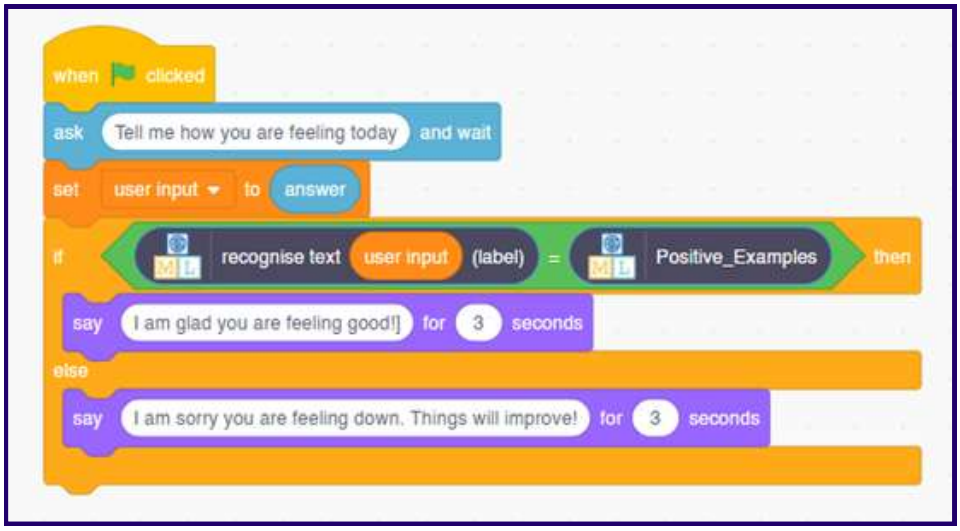



Figure 4 Sequence of code blocks in Scratch illustrating a basic event-action chain: Scratch by MIT Media Lab

**Step 9: Test and Refine**
Run your project and type different messages. Observe how the sprite responds based on the AI's classification. If the model misclassifies text, return to ML4K, add more training examples, retrain, and refresh your Scratch project.

## UNIT 3.6 AI EXTENSION: TEXT-TO-SPEECH

Text-to-Speech (TTS) is an AI-powered technology that converts written text into spoken audio. This official Scratch extension uses cloud-based speech synthesis to give your sprites realistic voices in multiple languages and tones.

TTS makes projects more accessible (visually impaired users can hear content), more engaging (spoken dialogue feels more dynamic than text bubbles), and more versatile (projects can narrate stories, provide instructions, or create language-learning tools).

### ADDING THE TEXT-TO-SPEECH EXTENSION

**Step 1:** Open the Extension Library. In the Scratch editor, click the blue "Add Extension" button (with a "+" icon) at the bottom-left of the block's palette.

**Step 2:** Scroll through the extension library and click on "Text-to-Speech" (it has a speaker with speech bubbles). New blocks will appear in your palette under a new "Text-to-Speech" category.

**Step 3:** Use the blocks in your code:

**Speak a phrase:** Drag the speak [hello] block into your code area. Click inside the block to change the text to whatever you want your sprite to say.

**Change the voice:** Use the set voice to [alto] block to choose a different voice. Options include Alto, Tenor, Squeak, Giant, and Kitten.

**Change the language:** To change the language, use the set language to block.

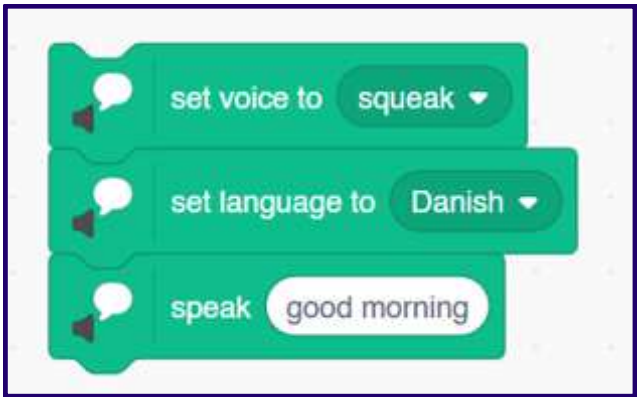



Figure 5 Sequence of code blocks for conversation in Scratch: Scratch by MIT Media Lab

## USING SCRATCH BLOCKS TO CREATE INTERACTIVE PROJECTS.

Combine the Text to Speech blocks with other Scratch blocks to create interactive projects. For example, you can use the 'when green flag clicked' event block from the "Events" category to start the speech. You can also use the ask and wait block from the "Sensing" category and the answer block with the speak block to have your sprite repeat what a user types.

## KEY TEXT-TO-SPEECH BLOCKS

This extension provides blocks for controlling speech, they are as follows:

| Block | Function | Example |
|---|---|---|
| speak [text] | Speaks the specified text aloud | speak [Hello, welcome to my project!] |
| set voice to [voice] | Changes the voice character | set voice to [tenor] (options: alto, tenor, |
| set language to [language] | Changes the language for speech | set language to [Spanish] (23 languages supported) |

## TEXT-TO-SPEECH VOICE OPTIONS

There are a number of text-to-speech voice options that you can experiment with to match your sprites personality; they are as follows:

- Alto: Medium-pitched, neutral voice
- Tenor: Slightly higher-pitched voice
- Squeak: High-pitched, child-like voice
- Giant: Deep, booming voice
- Kitten: Very high-pitched, cute voice

## EXAMPLES OF HOW TEXT-TO-SPEECH CAN BE USED IN SCRATCH

- **Interactive Stories:** Characters narrate plot developments and speak dialogue.
- **Educational Quizzes:** The computer reads questions aloud for accessibility.
- **Language Learning:** Pronunciation practice with different languages
- **Virtual Assistants:** Assistants like Siri that speaks responses.
- **Audio Guides:** Museum tours or city guides with narrated information
- **Dynamic Feedback:** Spoken encouragement or hints in games.

# REFERENCES

## Book, Wiki, and Documentation


- Scratch Foundation (MIT Media Lab): Platform documentation, educational resources, and interface descriptions https://scratch.mit.edu/
- Machine Learning for Kids (Dale Lane, IBM) 11 Feb. 2021: Platform tutorials, model training methodology, and integration guides. ISBN-13: 9781718500563
- Scratch Wiki: Technical documentation for extensions and block descriptions. https://scratch-wiki.info/
- The AI Scratch Code Playbook.: A Beginner's Guide to Building Intelligent Systems Paperback – 11 Feb. 2025. ISBN-13: 979-8310433649


## LINKS TO LEARNING PLATFORMS

**Scratch Official Website**
The official Scratch platform developed by MIT Media Lab. Create an account, build projects, explore millions of community creations, and access tutorials and learning resources.
https://scratch.mit.edu/

**Machine Learning for Kids**
Educational platform by IBM for training custom machine learning models (text, images, numbers, sounds) and integrating them with Scratch. Includes worksheets, tutorials, and pre-trained models.
https://machinelearningforkids.co.uk/

## TEXT-TO-SPEECH VOICE OPTIONS

**Teaching AI Concepts Creatively with Scratch (Codingal)**
Blog post exploring creative approaches to teaching AI concepts through Scratch, with project ideas and pedagogical strategies.
https://www.codingal.com/coding-for-kids/blog/teach-kids-ai-codng-concepts-creatively-using-scratch/

**Scratch Machine Learning Studio**
Curated collection of Scratch projects using machine learning extensions. Explore for inspiration and remix to learn from existing projects.
https://scratch.mit.edu/studios/3995548/

## PRACTICAL EXERCISE

**Objective:** Create a project using the Text-to-Speech extension to experiment with different voices to match your sprites' personalities.

**Instructions:** Follow the steps below to create a sprite that speaks user input aloud, demonstrating the power of combining text input with speech output.

**Step 1 - Set Up Your Project:**
Create a new Scratch project or continue with an existing one.
Choose a sprite that will do the talking – the "Robot" or "Wizard" sprites work well.

**Step 2 - Add the Extension:**
Add the Text-to-Speech extension as described above.

**Step 3 - Create the Script and Build this script for your sprite:**

```
when green flag clicked
└── set voice to [giant]
└── speak [Hello! I can speak anything you type.]
└── forever
└── ask [What should I say?] and wait
            └── speak (answer)
```

**Step 4 - Test Your Project:**
Click the green flag and type messages when prompted. Your sprite will speak the messages aloud. Try different voices and languages by changing the set voice and set language blocks.

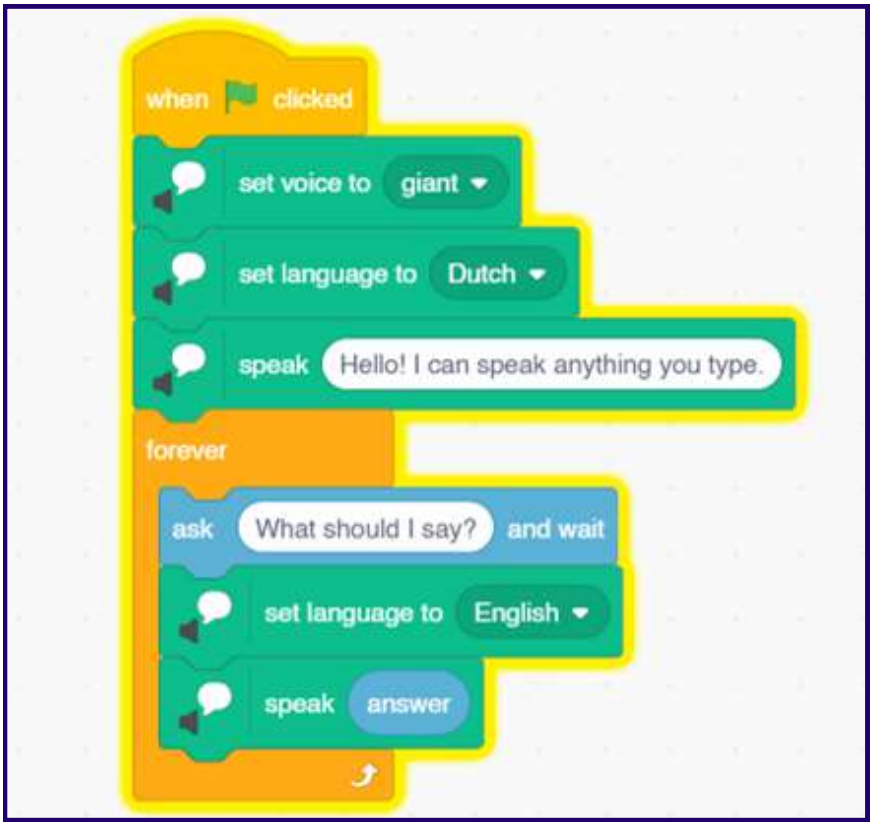



Figure 6 Sequence of code blocks in Scratch: Scratch by MIT Media Lab

## REFLECTION QUESTIONS

Take time to think deeply about what you have learned in this module.
Answer the following questions thoughtfully:

- **Reflection Question 1:** What aspect of using AI in Scratch was most surprising to you? Did anything work differently than you expected? Explain with specific examples from your project.

- **Reflection Question 2:** How would you use machine learning in a game or interactive story? Describe a specific project idea and explain how AI would make it more engaging or intelligent than traditional programming alone.

- **Reflection Question 3:** What problems or challenges did you face while training your ML model or integrating AI extensions? How did you solve them? If you encountered problems you could not resolve, what strategies would you try next?

## QUIZ WITH MULTIPLE-CHOICE QUESTIONS (one correct answer per question)

**1. Which statement best describes Scratch?**

a) A textual programming language used by professional software engineers
b) A visual block-based programming language making coding accessible to beginners
c) An artificial intelligence service for building machine learning models
d) A hardware robot kit for teaching programming to children

**2. What is one of the main benefits of using block-based programming in Scratch?**

a) It requires students to memorise syntax, enhancing memory skills
b) Blocks only connect in logically correct ways, preventing syntax errors
c) It focuses only on text-based coding, which is more challenging for learners
d) It is only usable by people with prior programming experience

**3. Which Scratch block category would you use to create a loop that repeats code 10 times?**

a) Motion
b) Events
c) Control
d) Operators

**4. What must you do before you can use a Machine Learning for Kids model in your Scratch project?**

a) Nothing - the extension comes pre-trained with a model that works for all purposes
b) Provide training examples for each category you want the model to recognise, then train the model
c) Purchase a special hardware device to enable machine learning
d) Write Python code to create a machine learning algorithm and import it into Scratch

## QUIZ WITH MULTIPLE-CHOICE QUESTIONS (one correct answer per question)

**5. Which of the following is NOT an AI-related capability mentioned in this module?**

a) Recognising objects or images through image classification
b) Making a sprite perform a dance routine using a repeat loop
c) Converting spoken words into text through speech recognition
d) Translating text from English to Spanish within a project

**6. Which Scratch extension would you use to have a sprite narrate a story aloud?**

a) Video Sensing
b) Translate
c) Text-to-Speech
d) Machine Learning for Kids

**7. Why is it important to provide multiple training examples when creating a machine learning model?**

a) The model cannot function with fewer than 100 examples
b) More examples help the model learn patterns more accurately and classify new inputs more reliably
c) Training examples are only used for decoration and do not affect performance
d) Multiple examples slow down the model, making it more accurate

**8. What advantage does combining Machine Learning for Kids with Text-to-Speech provide?**

a) It makes the project run faster
b) It understands inputs and communicate responses
c) It reduces the amount of code you need to write
d) It allows the project to work offline without internet

# Module 4: Python Game Programming (pygame / Arcade libraries) with AI

## INTRODUCTION

In this short and beginner-friendly module, you will become a game creator, not just a player. You will learn how to design and program a small videogame using Python and Pygame, and you will discover how to make your game characters behave intelligently with simple forms of Artificial Intelligence (AI). Don't worry if you have never coded before — everything will be explained step by step. You will start by creating a small game window and moving a square on the screen, and later you will explore how to make the game react to what you do, for example by making an enemy notice your movements or change colour when you get close. This module is not only about running code but about understanding and experimenting. You will test and modify examples to see how small changes can create different behaviours. By doing so, you will learn how games "think" and respond, much like simple AI systems in the real world. By the end of the module, you will be able to acquire different skills, such as:

- Create interactive 2D games using Pygame or Arcade libraries.
- Add AI behaviours so that game characters can react to player actions.
- Use computational thinking to design interactive and responsive games.
- Experiment, test and modify game code to observe how small changes affect behaviour.
- Reflect on how AI is used in real life.

| | |
|---|---|
| **Module Duration** | **2 hours (1 hour learning + 1 hour practical exercises)** |

## EDUCATIONAL MATERIALS

### UNIT 4.1 INTRODUCTION TO PYGAME

Before you start coding, it is important to know which tools you will work with and what they are for:

- **Python 3** – the programming language you will use to give instructions to the computer. It is widely used in schools, universities and companies for programming, data analysis and AI.

- **Pygame** – a free Python library that helps you create simple 2D games. It provides ready-made functions to open a game window, draw shapes, show colours and react to the keyboard or mouse.

- **Code editor (IDLE, Thonny or VS Code)** – the program where you will type, save and run your Python code. In this module, you can use IDLE (included with Python) or another editor recommended by your teacher.

These tools will be introduced step by step, so that you can install them and complete the activities on your own, even if this is your first contact with programming.

**Goal:** Learn how to open a game window,
display colours and move a square with the keyboard.

**Step 1 – Install Python 3 and open your programming environment**
Before we start using Pygame, you need to make sure that Python 3 is installed on your computer.

**Install Python 3**

- Go to the official Python website: https://www.python.org/downloads/.
- Download the version for your system (Windows, macOS or Linux).
- During installation, check the option “Add Python to PATH” before clicking Install Now.
- Once installed, restart your computer (optional but recommended).

## UNIT 4.2 VERIFY YOUR INSTALLATION

Open the Command Prompt (Windows) or Terminal (macOS/Linux) and type:

**python --version** or **py --version**

If it shows something like Python 3.13.3, everything is ready.

## UNIT 4.3 INSTALL AN EDITOR OR USE IDLE

You can write Python code using one of these:

- IDLE (comes installed with Python)
- Thonny (simple beginner IDE) – https://thonny.org3.
- VS Code (advanced option) – https://code.visualstudio.com

To open IDLE:

- On Windows: press Start → IDLE (Python 3.x)
- On macOS: search for “IDLE” in Launchpad
- On Linux: run idle3 in the terminal

**3. Type or Paste the Code**
Copy the code from this module into the new file.
For example:

```
CODE:
import pygame
pygame.init()
screen = pygame.display.set_mode((800, 600))
pygame.display.set_caption("My First Game")
running = True
while running:
   for event in pygame.event.get():
       if event.type == pygame.QUIT:
           running = False
   screen.fill((200, 220, 255))
   pygame.display.flip()
pygame.quit()
```

**4. Save the File**
Choose File → Save As...
Name your file (for example my_first_game.py).

**5. Run the Program**
Press F5, or go to Run → Run Module.
A new window appears — if you see a blue background, everything is working

## UNIT 4.4 BASIC BLOCK-BASED PROGRAMMING SKILLS

Copy and run this code:

```
CODE:
import pygame  # 1. Import the pygame library
pygame.init()  # 2. Start (initialize) all pygame modules
screen = pygame.display.set_mode((800, 600))  # 3. Create a
game window (width 800, height 600)
pygame.display.set_caption("My First Game") # 4. Give a title
to the game window
running = True # 5. Create a variable to keep the game
running

while running:  # 6. Start the main game loop (runs again and
again)
   for event in pygame.event.get():  # 7. Check all the
events (for example, key presses or mouse clicks)
       if event.type == pygame.QUIT: # 8. If the user clicks
the "close" button…
           running = False         # 9. …stop the game loop

   screen.fill((200, 220, 255))      # 10. Fill the screen
with a light blue colour (RGB values)
   pygame.display.flip()         # 11. Update the window to
show the new frame

pygame.quit() # 12. Close the game and exit safely
```

## UNIT 4.5 MOVING A SQUARE

Now let's make something move. Copy the code below and test it:

```
PART I

CODE:
import pygame # 1. Import the pygame library
pygame.init() # 2. Start pygame
# 3. Create the window
screen = pygame.display.set_mode((800, 600))
# 4. Add a title
pygame.display.set_caption("Moving Square")
x = 400  # 5. The square's X position (horizontal)
y = 300  # 6. The square's Y position (vertical)
speed = 1 # 7. How many pixels it moves each time

running = True # 8. Keep the game running

while running:  # 9. Start the game loop
   for event in pygame.event.get(): # 10. Check events
       if event.type == pygame.QUIT: # 11. If user clicks
"X", stop
           running = False
```

## UNIT 4.6 MOVING A SQUARE

Now let's make something move. Copy the code below and test it:

```
PART II

CODE:
   keys = pygame.key.get_pressed() # 12. Check which keys are
pressed
   if keys[pygame.K_LEFT]:
       x -= speed  # 13. Move left
   if keys[pygame.K_RIGHT]:
       x += speed  # 14. Move right
   if keys[pygame.K_UP]:
       y -= speed  # 15. Move up
   if keys[pygame.K_DOWN]:
       y += speed  # 16. Move down

   screen.fill((200, 220, 255)) # 17. Paint the background
   pygame.draw.rect(screen, (0, 0, 255), (x, y, 50, 50)) #
18. Draw the blue square
   pygame.display.flip() # 19. Update the screen

pygame.quit() # 20. Close the game
```

**What is happening:**

- x and y define the square's position on the screen.
- The arrow keys change these values, moving the square.
- The loop repeats the drawing many times per second, creating smooth movement.

**Try this:**

- Change speed = 1 to speed = 5. What happens?
- Make the square a different colour, for example (255, 0, 0) for red.
- What do you think would happen if you removed pygame.display.flip()?

Reflection Question: How does the computer know the square's new position each time you press a key?

**Learning Checkpoint**

At this point, you should be able to:

✔Open a Pygame window.
✔Use the game loop to keep the game running.
✔Control a moving object with the keyboard.

In the next part, you will add Artificial Intelligence, so that your game reacts to your movements.

## UNIT 4.7 SIMPLE AI IN GAMES (REACTIVE ENEMY)

In this module, we use simple rule-based game AI: the computer follows predefined rules to perceive the player, decide what to do, and act accordingly, without learning from data. This is similar to many classic video games where enemies chase, avoid, or react to the player based on distance or position.

**Goal:** Learn how to make your game react to what the player does by adding a simple form of Artificial Intelligence (AI). You will create an enemy that changes colour when the player gets close.

**Step 1 – Understanding the Idea**

Before coding, think about what AI means in a game. In this case, AI is not about learning or data — it's about making the computer react to what the player does.

**You will use a simple rule:** If the player is close, the enemy changes colour. If not, it stays the same. This is an example of conditional behaviour, one of the simplest forms of AI.

**Step 2 – Code Example: Reactive Enemy**
Copy and run the following code:

```
PART I

CODE:
import pygame, math      # 1. Import game and math libraries
pygame.init()         # 2. Start (initialize) all Pygame
functions

screen = pygame.display.set_mode((800, 600))
# 3. Create a game window (800x600)
pygame.display.set_caption("Simple AI - Reactive Enemy")
# 4. Title

player = pygame.Rect(100, 100, 50, 50)
# 5. Player (blue square)
enemy= pygame.Rect(500, 300, 50, 50)  # 6. Enemy (red square)
speed = 5                # 7. Player speed
react_distance = 120
# 8. Distance (px) for enemy to "react"

clock = pygame.time.Clock()
running = True           # 9. Keep the game running
```

**PART II**

**CODE:**

```
while running:          # 10. Main game loop (every frame)
for event in pygame.event.get():# 11. Check all events
if event.type == pygame.QUIT: # 12. If the window is closed…
           running = False              # 13. …stop the loop

   # ---- MOVEMENT (must be inside the while) ----
keys = pygame.key.get_pressed()  # 14. Which keys are
pressed?
if keys[pygame.K_LEFT]:
player.x -= speed                # 15. Move left
if keys[pygame.K_RIGHT]:
player.x += speed                # 16. Move right
if keys[pygame.K_UP]:
player.y -= speed                # 17. Move up
if keys[pygame.K_DOWN]:
player.y += speed                # 18. Move down

# ---- SIMPLE AI: change colour if close ----
px, py = player.center# 19. Player center
ex, ey = enemy.center# 20. Enemy center
dist = math.hypot(px - ex, py - ey)# 21. Distance

if dist < react_distance: # 22. If closer than 120 px…
enemy_color = (255, 200, 0)      # 23. …enemy turns yellow
else:
enemy_color = (220, 60, 60) # 24. Otherwise, enemy stays red
```

```
PART III

CODE:
# ---- DRAW ----
screen.fill((240, 245, 255))# 25. Background
pygame.draw.rect(screen, (60, 120, 255), player)
# 26. Player (blue)
pygame.draw.rect(screen, enemy_color, enemy)
# 27. Enemy (reactive)
pygame.display.flip()  # 28. Show changes

clock.tick(60)# Limit to 60 FPS

pygame.quit()    # 29. Close safely
```

**Step 3 – What's Happening in This Code**

| Concept | Explanation |
|---|---|
| Perception | The enemy measures how far it is from the player using math.hypot. |
| Decision | If the distance is smaller than react_distance, the enemy decides to react. |
| Action | The reaction is to change its colour. |

This "perceive → decide → act" cycle is the foundation of most game AI systems.

Even simple reactions like this make the game world feel more alive.

## STEP 4 – TRY, CHANGE, OBSERVE

Try changing these parts of the code and observe what happens:

1. Change react_distance from 120 to 250 — what happens?
2. Make the enemy turn green (0, 255, 0) instead of yellow.
3. Make the player move faster by increasing speed = 8.
4. Can you make the enemy react only when the player moves above or below it (for example, compare only the y positions)?

Encourage students to predict before running the program
— what do they think will happen

**Step 5 – Reflection**
Is this really “intelligence”?
Why or why not?
How could we make the enemy’s behavior seem smarter?

**Hint:** Real AI in games uses the same logic — simple decisions repeated many times per second can create the illusion of intelligence.

# REFERENCES

## Book, Wiki, and Documentation


- Python Software Foundation. Python 3 Documentation. Available at: https://docs.python.org
- Pygame Community. Pygame Documentation and Tutorials. Available at: https://www.pygame.org/docs/ and https://www.pygame.org/wiki/tutorials
- Python Arcade Project. Python Arcade Library Documentation. Available at: https://api.arcade.academy
- Real Python. Pygame: A Primer on Game Programming in Python. Available at: https://realpython.com/pygame-a-primer/
- Microsoft. AI for Beginners Curriculum.
  Available at: https://github.com/microsoft/AI-For-Beginners


## LINKS TO LEARNING PLATFORMS

- Python Official Website – Download and learn the basics of Python programming. (https://www.python.org/)
- Pygame Documentation – Official guide to functions, events and graphics in Pygame. (https://www.pygame.org/docs/)
- Arcade Library Documentation – An alternative library for simple 2D games in Python. (https://api.arcade.academy/en/latest/)
- Pygame AI Guide – Examples of simple AI algorithms for Pygame. (https://pygame-ai.readthedocs.io/en/latest/guide.html)
- Real Python – Build a Game With Pygame – A friendly tutorial for beginners. (https://realpython.com/pygame-a-primer/)

- KidsCanCode Tutorials – Easy projects and challenges for learning Pygame. (https://kidscancode.org/blog/)

- AI for Beginners (Microsoft) – Open educational content introducing AI principles. (https://microsoft.github.io/AI-For-Beginners/)

## PRACTICAL EXERCISE

### CATCH ME IF YOU CAN

**Goal:** Create a small game where the player moves a blue square to collect objects while a red AI enemy follows the player. This exercise joins all the skills learned: movement, collision detection, and a simple AI behaviour.

**Step 1 – Prepare Your Environment**
Make sure Pygame is installed:

**pip install pygame**

Then open your Python editor and create a new file called catch_me.py.

**Step 2 – Understand the Game Idea**

| Element | Description |
|---|---|
| Player | Moves with the arrow keys. |
| Target | Appears randomly; increases score when collected. |
| Enemy (AI) | Follows the player by calculating direction and distance. |
| Goal | Collect as many targets as possible before the enemy catches you. |

Think before you code: How could an enemy “know” where the player is?

## STEP 3 – BUILD THE GAME STEP BY STEP

**PART I**

**CODE:**

```
import pygame, random, math
pygame.init()

# --- Window ---
WIDTH, HEIGHT = 800, 600
screen = pygame.display.set_mode((WIDTH, HEIGHT))
pygame.display.set_caption("Catch Me If You Can")
clock = pygame.time.Clock()

# --- Colors ---
BLUE= (60, 120, 255)
RED= (220, 60, 60)
GREEN = (60, 200, 60)
WHITE = (240, 245, 255)
BLACK = (20, 20, 20)

# --- Entities ---
player = pygame.Rect(100, 100, 40, 40)
# blue square (player)
enemy= pygame.Rect(600, 400, 40, 40)
# red square (chasing enemy)
target = pygame.Rect(
random.randint(50, WIDTH - 50),
# green target: random position
random.randint(50, HEIGHT - 50),25,25)
```

## STEP 3 – BUILD THE GAME STEP BY STEP

```
PART II

CODE:
PLAYER_SPEED = 5
ENEMY_SPEED = 2.5
score = 0

# Font (fallback-friendly: uses default font to avoid font
issues)
font = pygame.font.Font(None, 26)

# --- Game loop ---
running = True
while running:
   # 1) Handle events
for event in pygame.event.get():
if event.type == pygame.QUIT:
           running = False

# 2) Player movement (arrow keys)
keys = pygame.key.get_pressed()
if keys[pygame.K_LEFT]:
player.x -= PLAYER_SPEED
if keys[pygame.K_RIGHT]:
player.x += PLAYER_SPEED
if keys[pygame.K_UP]:
player.y -= PLAYER_SPEED
if keys[pygame.K_DOWN]:
player.y += PLAYER_SPEED
```

## STEP 3 – BUILD THE GAME STEP BY STEP

```
PART III

CODE:
# Keep player inside the window boundaries
player.clamp_ip(screen.get_rect())

   # 3) Simple AI: enemy chases the player
dx = player.centerx - enemy.centerx
dy = player.centery - enemy.centery
dist = math.hypot(dx, dy)
if dist != 0: # avoid division by zero when both overlap
enemy.x += int((dx / dist) * ENEMY_SPEED)
enemy.y += int((dy / dist) * ENEMY_SPEED)

# 4) Player collects target
if player.colliderect(target):
score += 1
target.topleft = (
           random.randint(25, WIDTH - 50),
           random.randint(25, HEIGHT - 50)
)

# 5) Draw everything
screen.fill(WHITE)
pygame.draw.rect(screen, BLUE,player)
pygame.draw.rect(screen, RED,enemy)
pygame.draw.rect(screen, GREEN, target)
```

## STEP 3 – BUILD THE GAME STEP BY STEP

```
PART IV

CODE:
# Draw score
score_text = font.render(f"Score: {score}", True, BLACK)
screen.blit(score_text, (10, 10))

pygame.display.flip()
clock.tick(60)

pygame.quit()
```

**Step 4 – Experiment and Reflect**
Try changing these values and observe what happens:

- Change ENEMY_SPEED to make the enemy faster or slower.
- Change the colour of the player or target.
- Make the game harder by reducing the player's size.
- Add a Game Over message when the enemy touches the player.

## REFLECTION QUESTIONS

Take time to think deeply about what you have learned in this module. Answer the following questions thoughtfully:

- **Reflection Question 1:** What was the most difficult part of creating your game?
- **Reflection Question 2:** How does the AI make the game feel more interactive?
- **Reflection Question 3:** Can you think of other situations where AI reacts to human actions (for example, education or healthcare)?
- **Reflection Question 4:** If you could improve this game, what kind of AI behaviour would you add next?
- **Reflection Question 5:** What makes the enemy look “intelligent”?
- **Reflection Question 6:** How does the distance calculation (math.hypot) help it find the player?
- **Reflection Question 7:** Can you think of real games that use a similar kind of logic?

## QUIZ WITH MULTIPLE-CHOICE QUESTIONS (one correct answer per question)

**1. What does the game loop do in Pygame?**
a) It installs new libraries during the game
b) It repeats the game actions many times per second
c) It closes the game window
d) It loads the background image only once

**2. What is the purpose of pygame.display.flip()?**
a) It checks the keyboard input
b) It draws a rectangle
c) It updates the game window with new visuals
d) It pauses the game loop

**3. Which function checks which keyboard keys are being pressed?**
a) pygame.quit()
b) pygame.key.get_pressed()
c) pygame.event.get()
d) pygame.display.set_mode()

**4. In the AI example, what makes the enemy react to the player?**
a) A timer event
b) Random movement
c) The distance between the player and the enemy
d) The colour of the player

**5. What happens if you increase the value of react_distance in the code?**
a) The enemy reacts from farther away
b) The enemy moves faster
c) The player slows down
d) The window size changes

## QUIZ WITH MULTIPLE-CHOICE QUESTIONS (one correct answer per question)

**6. What is the purpose of the math.hypot() function in the AI code?**
a) To calculate the distance between two points
b) To draw the enemy on screen
c) To create a new random position
d) To detect collisions with walls

**7. What does this line of code do?**
a) It draws a text on the screen
b) It creates a rectangle to represent the player
c) It starts with the AI algorithm
d) It changes the background colour

**8. What happens if you remove pygame.display.flip() from the loop?**
a) The program runs faster
b) The window does not update and looks frozen
c) The colours change automatically
d) The player moves twice as fast

**9. Which of these is an example of AI behaviour in a game?**
a) Drawing a square on the screen
b) Making a character move only when the player presses a key
c) Making an enemy follow or avoid the player
d) Loading background music

**10. How can AI make games more interesting?**
a) By fixing bugs automatically
b) By creating dynamic challenges and responsive characters
c) By changing the graphics automatically
d) By controlling the keyboard

# RUNNING THE CODE FOR MODULE 4

## Python Game Programming (pygame): Setup Guide for the Correct Python Version

Module 4 uses the pygame library. Currently, pygame only provides pre-built installation packages up to Python 3.13, and not yet for Python 3.14. If Python 3.14 (e.g., 3.14.6) is running on the computer, the pygame installation will fail, and the examples from Module 4 will not start. Therefore, the module is written for Python 3.13.3.

### Important – Do not overwrite the existing Python

Do not uninstall or "downgrade" the Python already installed on the computer. On macOS and Linux, the operating system depends on it, and other programs can be damaged. The safe solution is to install Python 3.13 in parallel and use it only for this module. This is exactly what each of the following options does.

## First, check the Python version

Open a Terminal (macOS/Linux) or PowerShell (Windows) and enter one of these commands:

```
python --version
py --version
```

- **Python 3.13.x** → everything is ready; Module 4 can be run directly.
- **Python 3.14.x** → follow one of the options below.

## Option 1 – macOS / Linux (Terminal)

The helper script handles everything: It installs a small tool called uv, provides Python 3.13 in addition to the existing Python, and runs the file with pygame. Nothing is permanently changed on the system.

### Steps:

1. Save the script provided below as modul4-start.sh in the same folder as the Python file of Module 4.
2. Make it executable and run it (replace modul4.py with your own file name):

```
chmod +x modul4-start.sh
./modul4-starr.sh modul4.py
```

3. On the first run, uv and Python 3.13 will be downloaded automatically. Every subsequent run will start immediately.

### The Script

Copy the entire content of the box below into a file named modul4-start.sh. (A ready-made file is included with this guide.)

```
#!/usr/bin/env bash
#
==========================================================================
# modul4-start.sh
#
# Runs the Module 4 script (pygame) on Python 3.13 - WITHOUT touching or
# overwriting the Python that is already installed on your computer.
#
# Usage:
# chmod +x modul4-start.sh # make it executable (once)
# ./modul4-start.sh # runs ./modul4.py
# ./modul4-start.sh path/to/script.py
#
# If your module also needs the Arcade library, add --with arcade
# to the last command (see step 4).
#
==========================================================================
```

```
set -euo pipefail

SCRIPT="${1:-modul4.py}"

# --- 1. Install uv if it is not present yet ------------------------------
---
# uv is a small tool that manages Python versions and packages.
if ! command -v uv >/dev/null 2>&1; then
 echo "-> Installing 'uv' (one-time setup) ..."
 curl -LsSf https://astral.sh/uv/install.sh | sh
 # uv is placed in ~/.local/bin - make it available in this session:
 export PATH="$HOME/.local/bin:$PATH"
fi

# --- 2. Provide Python 3.13 ----------------------------------------------
---
# It is installed IN PARALLEL. Your existing Python (3.14.x) stays
untouched.
echo "-> Providing Python 3.13 (if not already installed) ..."
uv python install 3.13

# --- 3. Does the script exist? -------------------------------------------
---
if [[ ! -f "$SCRIPT" ]]; then
 echo "x File not found: $SCRIPT"
 echo " Example: ./modul4-start.sh path/to/script.py"
 exit 1
fi

# --- 4. Run the script on Python 3.13 with pygame ------------------------
---
# uv creates a short-lived, isolated environment, installs pygame into it,
# and runs the script. Nothing is changed globally.
echo "-> Starting '$SCRIPT' on Python 3.13 with pygame ..."
uv run --python 3.13 --with pygame "$SCRIPT"
```

## Option 2 – Windows (PowerShell)

No shell script is necessary on Windows; three commands are enough. Open PowerShell in the folder containing the Python file and run the commands one after another (replace modul4.py with your own file name):

```
powershell -c "irm https://astral.sh/uv/install.ps1 | iex"
uv python install 3.13
uv run --python 3.13 --with pygame .\modul4.py
```

## Option 3 – Stick with Python 3.14 (No Second Python)

If you want to stick with Python 3.14, you can use pygame-ce ("Community Edition"), an almost identical replacement that already provides packages for Python 3.14. The code does not change; you still write import pygame.

```
pip uninstall pygame
pip install pygame-ce
```

## Recommendation

For the shared teaching materials, stick with the setup using Python 3.13 (Options 1 and 2). The module is documented and tested against Python 3.13.3 with the standard pygame library. Only use pygame-ce for a quick personal test.

## Troubleshooting

- Command not found: uv (macOS/Linux) – close the Terminal and reopen it, or run export PATH="$HOME/.local/bin:$PATH".
- uv is not recognized (Windows) – after installing uv, close PowerShell and reopen it.
- "File not found" – check for the correct name or path, e.g., ./modul4-start.sh games/my_game.py.
- The module additionally uses the Arcade library – append --with arcade to the last command, e.g., uv run --python 3.13 --with pygame --with arcade modul4.py.
- A game window opens and closes immediately – this usually indicates an error in the Python code itself; the error message will appear in the Terminal.

# Module 5: Object Recognition with Roboflow – Introduction to Computer Vision

## INTRODUCTION

The aim of the current module is to provide a comprehensive introduction to Computer Vision, explaining how machines perceive and process visual data, and to guide learners in developing their own object recognition models using the Roboflow platform and advanced algorithms like YOLO.

By the end of the module, you will be able to acquire different skills, such as:

- **Understanding Fundamentals:** Distinguish between image processing and computer vision, and understand the logic of how computers interpret images as data (pixels and binary code).
- **Applying the CV Process:** Apply the five essential steps of the computer vision process: obtaining data, preparing data, determining the AI model, training the model, and interpreting results.
- **Using Roboflow:** Navigate the Roboflow platform to create projects, upload datasets, and manage image data effectively.
- **Data Annotation:** Perform accurate data annotation (labeling) to prepare datasets for AI model training.
- **Model Training:** Select and train advanced object detection models, such as YOLOv11, to identify specific objects.

| | |
|---|---|
| Module Duration | 2 hours (1 hour learning + 1 hour practical exercises) |

## EDUCATIONAL MATERIALS

### UNIT 5.1 WHAT IS COMPUTER VISION PROCESSING?

Image processing is a technique used to detect, extract, and evaluate important information contained in a digital image. This method yields meaningful data about objects or environments in an image, which is similar to human visual perception. The primary goal of image processing is to extract meaning from images and present this information for use in various fields. While visual information processing technology has existed for a long time, this process relied largely on human intervention in the past. This was both time-consuming and prone to errors. For example, in previous facial recognition systems, developers had to manually label thousands of photos; specific features, such as nose width or eye distance, were identified one by one. This was because image data was often disorganized and difficult for computers to understand. Therefore, automating the process required powerful processors and advanced computing technologies.

Today, with the help of evolving processing power, computer vision applications utilize artificial intelligence and machine learning (AI/ML) to accurately process this data for object identification and facial recognition, as well as classification, recommendation, tracking, and perception. Computer vision can be simply described as “letting machines perform this task,” inspired by the human visual system and its relationship with the brain. It is the process of machines, like computers, perceiving images and videos, analyzing them, and transforming them into meaning using machine learning and artificial intelligence. Originating in the 1950s, computer vision began commercializing in the 1970s with the application of detecting the difference between typewriter and handwriting. Today, it is used in many areas, including person detection and recognition technologies, product quality control systems, autonomous vehicles, agricultural applications, healthcare, education, and defense.

processing involves processing, restoring, and adjusting color and brightness of captured image data, including photos and videos. Computer vision, on the other hand, is the application of image processing techniques to images and the combination of this process with artificial intelligence. For example, while image processing is used to find the edges of an object within an image, the use of machine learning techniques to identify and classify that object falls within the field of computer vision.

## UNIT 5.2 HOW DOES A COMPUTER SEE?

The building blocks of nearly all computers worldwide, their “cells” consist of the numbers 1 and 0. This means that computers perceive the data users want to convey to them in their own language: 1s and 0s. After all, images and videos which consist of many images are also data, so computers perceive them as zeros and ones, or the binary number system. To better illustrate this, we can use the image below.

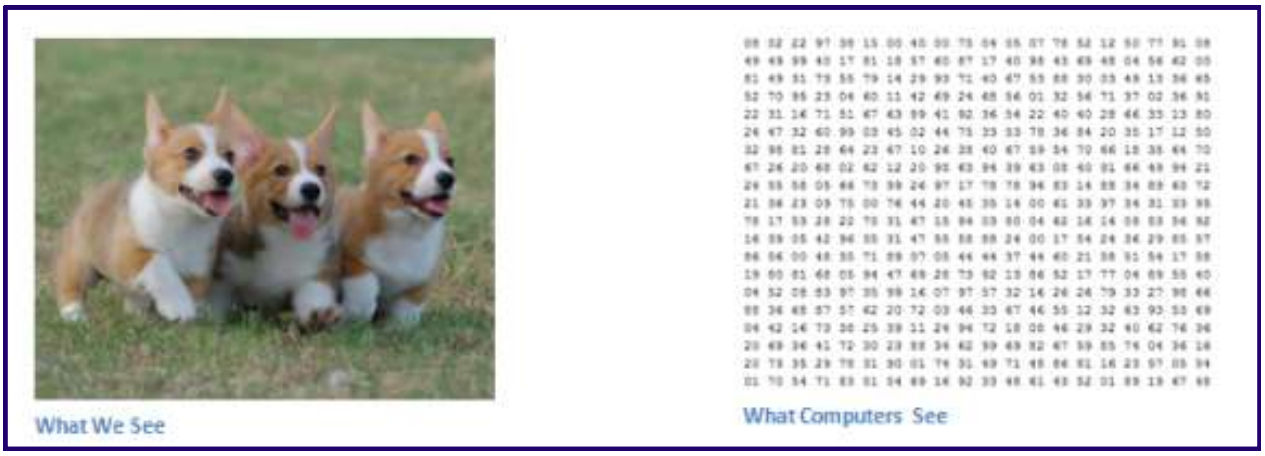



Figure 7 Form of View: Using neural networks for video surveillance (2025) from https://www.videonet9.com/using-neural-networks-for-video-surveillance.html; DateAdd: 09.08.2025.

But if you're wondering if this image shows two-digit values, but there are no zeros or ones, then yes, that's true. Our computer processors interpret these values by converting them to zeros and ones. If these values were also displayed as zeros and ones, it would be a very long and unsightly display.

In fact, each of these two-digit values in the image above can be called a pixel. So, what is a pixel? It's the smallest unit that allows the display to be generated and controlled in digital displays.

## UNIT 5.3 NOW LET US DO AN EXERCISE

- Take a grid notebook.
- Number each square of the grid notebook with numbers between 0 and 1.
- The closer the number written in any square is to 1, the whiter the square.
- The closer the number is to 0, the blacker the square.
- If the number is 0.5, color the square gray.

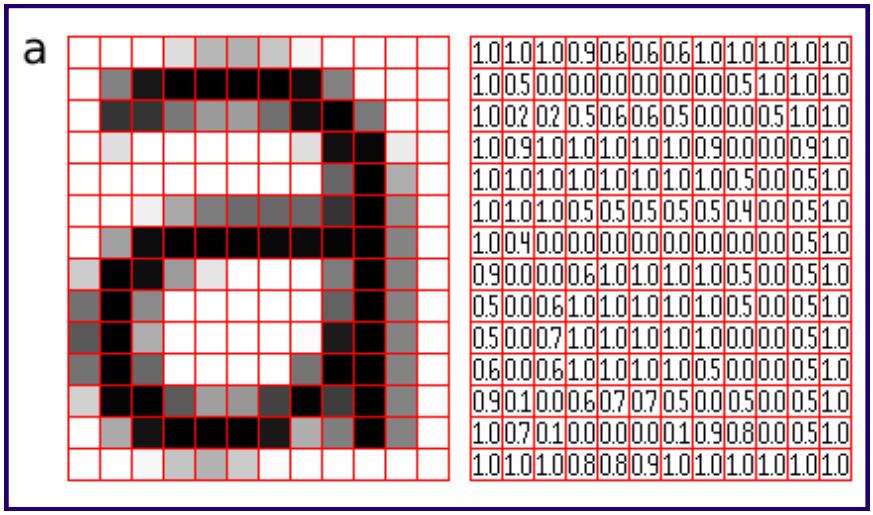



Figure 8 How computer see pixel. (2025) from https://medium.com/@meriyananjalika99/from-pixels-to-predictions-getting-started-with-cnns-convolutional-neural-networks-e3f84781cc1e; DateAdd: 09.08.2025.

If you carefully draw on a notebook like this, it will appear to you as a normal drawing from a distance. Try this for the image above. Move a few meters away from the screen. The image will appear more normal. In fact, computer and television screens display images and videos using millions of tiny squares in this way. To save the images, they store the numbers in the gridded notebook in memory. In the poor quality images we call pixelated, you can see the individual squares in this gridded notebook.

## UNIT 5.4 COMPUTER VISION PROCESSES ESSENTIALLY CONSIST OF FIVE STEPS:

1. Obtaining the image/data
2. Preparing the image/data
3. Determining the AI model to be used
4. Training the AI model to be used
5. Interpreting the results

The images obtained for a simple handwritten digit recognition application are shown below. These images were converted to a computer-friendly grayscale and used to train the AI. Test results indicate a 99% accuracy rate after training.

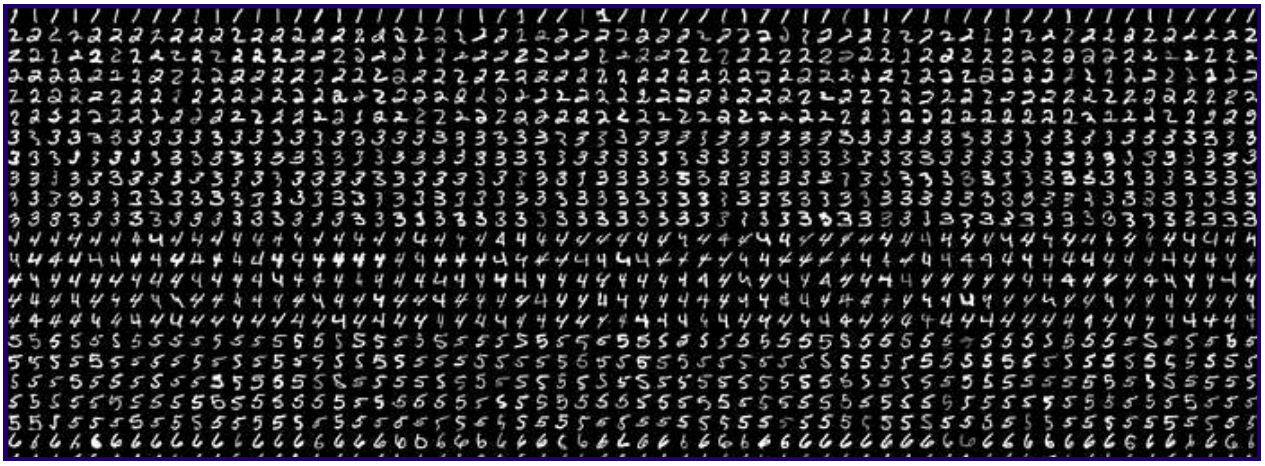


Figure 9 Kaggle-Dataset (2025) from www.roboflow.com; DateAdd: 09.08.2025.

Roboflow is a comprehensive artificial intelligence platform designed for developing computer vision projects. It simplifies the process of creating, training, and deploying computer vision models using image and video data. It offers an easy-to-use interface for both beginners and experienced developers.

## UNIT 5.5 HOW TO REGISTER WITH ROBOFLOW?

- Go to https://roboflow.com/.
- Click the Sign In button in the top right corner.
- In the window that opens, enter your membership information if you have one and log in to the system. For those logging in for the first time, use the Continue With Google option, as it is most practical to continue with a Google account.
- After selecting the Google account, you want to use on the Roboflow site and granting the necessary permissions, you will be logged in.
- Click the NEW PROJECT button on the page that opens, enter the project information, and create the project.

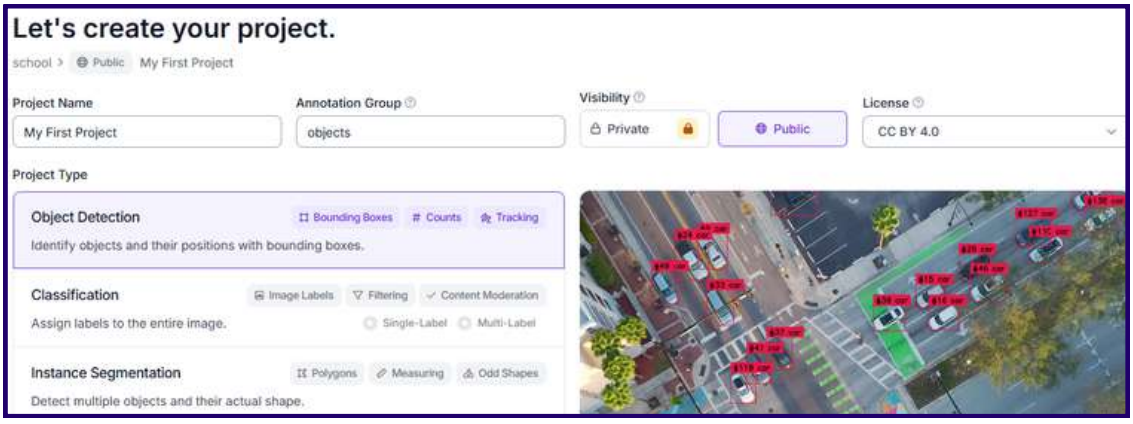



Figure 10 Kaggle-Dataset (2025) from www.roboflow.com; DateAdd: 09.08.2025.

On the new page that opens, we add the train, validation, and test data for our AI. We'll refer to this data as "dataset" from now on. If we don't have a dataset, we can use ready-made datasets within the RoboFlow Universe system. (https://universe.roboflow.com)

Similarly, Kaggle is a website that offers ready-made datasets. (https://www.kaggle.com/)

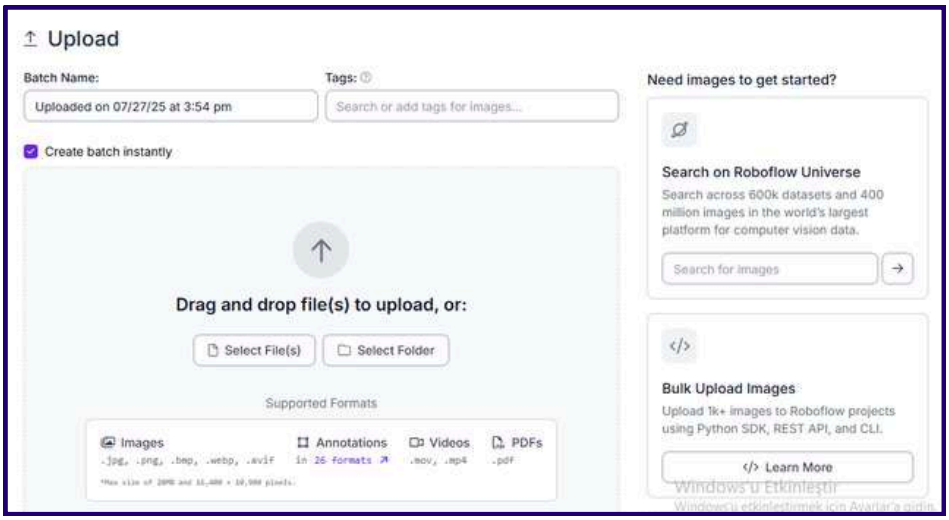



Figure 11 Roboflow (2025) from www.roboflow.com; DateAdd: 09.08.2025.

In this system, we will continue the process steps using a pre-made dataset. Therefore, we will use the dataset at https://universe.roboflow.com/roboflow-58fyf/rock-paper-scissors-sxsw via RoboFlow.

After entering this address, click the download dataset button. However, before clicking this button, we need to decide which AI model to use. In this application, we will use the YOLOv11 AI model, which is more efficient in object detection. If you visit the relevant algorithm's GitHub profile (https://github.com/ultralytics/ultralytics), you can find different pre-made weights and detailed information. (Yolov3 online test https://v-iashin.github.io/detector)

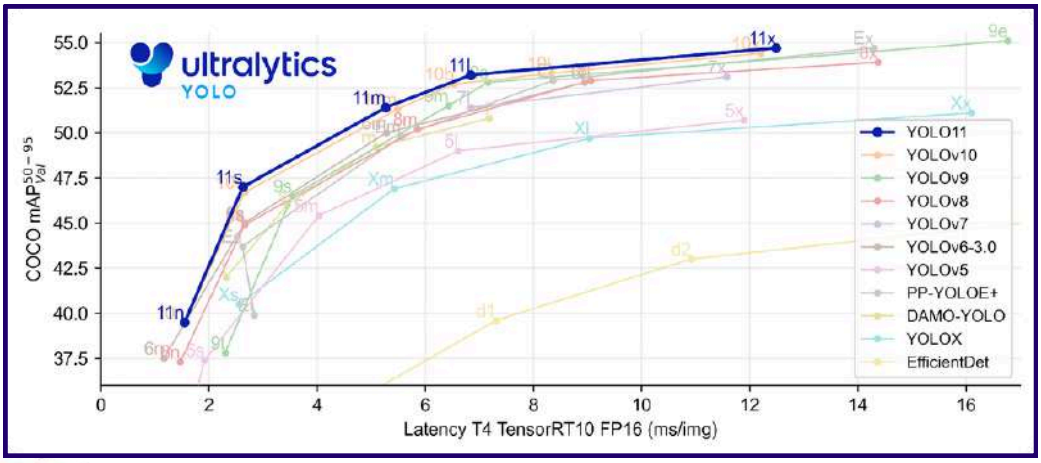



Figure 12 Dataset (2025) from www.roboflow.com; DateAdd: 09.08.2025.

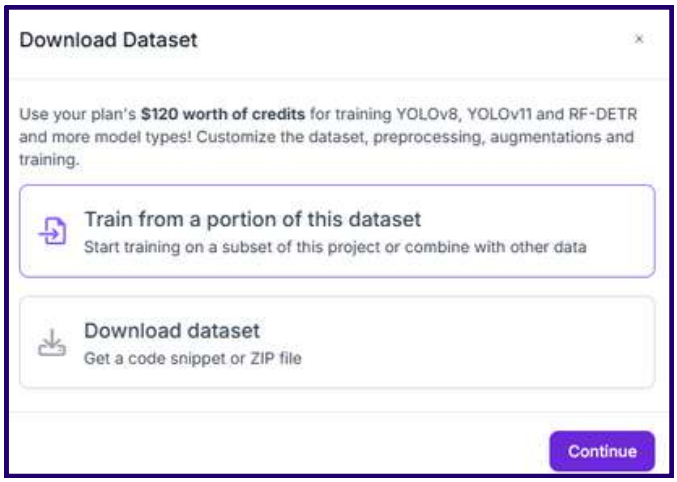



Figure 13 Dataset (2025) from www.roboflow.com; DateAdd: 09.08.2025.

After labeling and checking the data on the dataset, click on the download dataset button, select the model and online method, and get the Python code provided by the site.

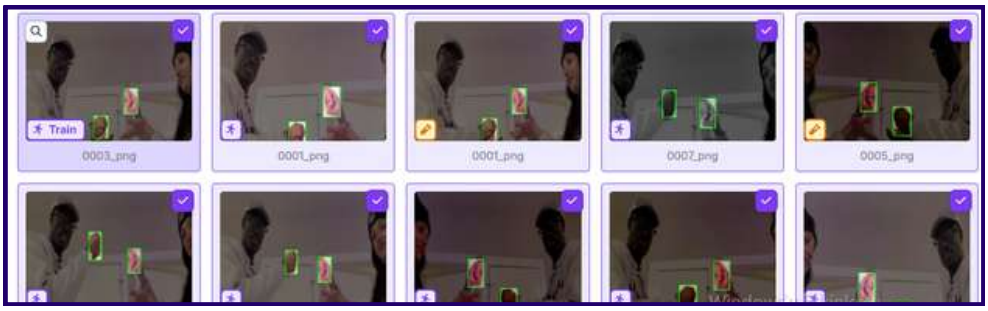



Figure 13 Dataset (2025) from www.roboflow.com; DateAdd: 09.08.2025.

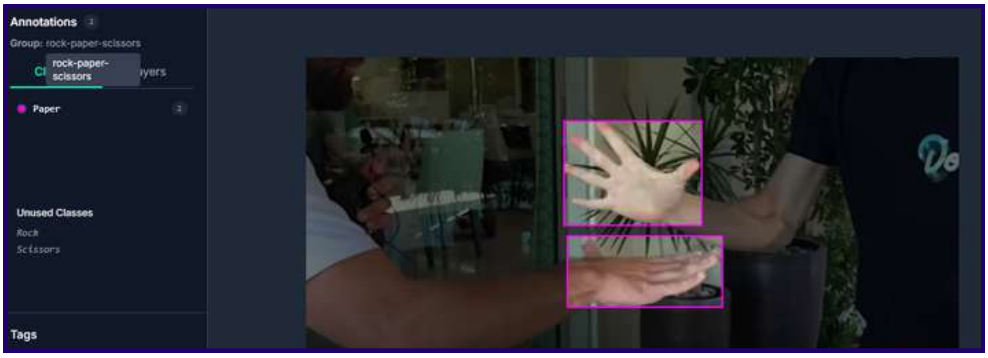



Figure 15 Cut-off (2025) from www.roboflow.com; DateAdd: 09.08.2025.

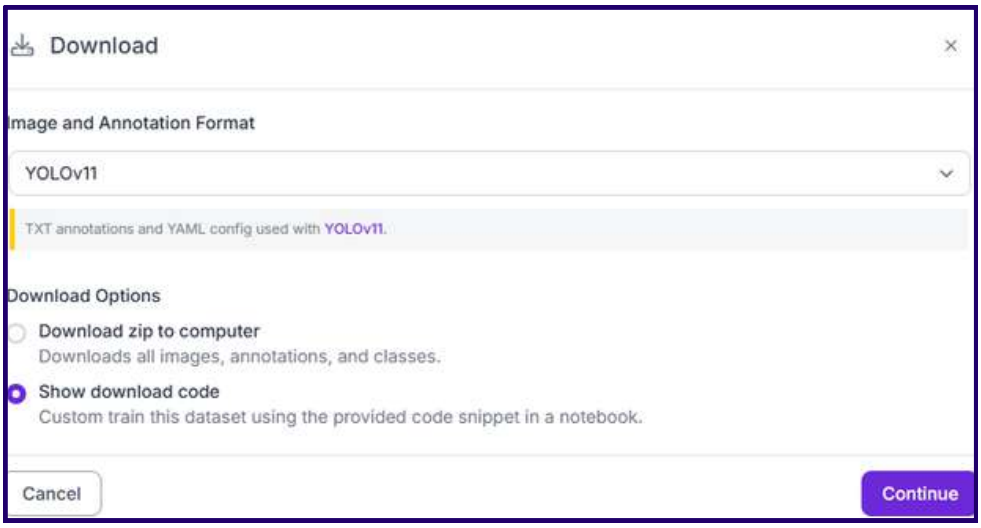



Figure 16 YOLOv11 (2025) from www.roboflow.com; DateAdd: 09.08.2025.

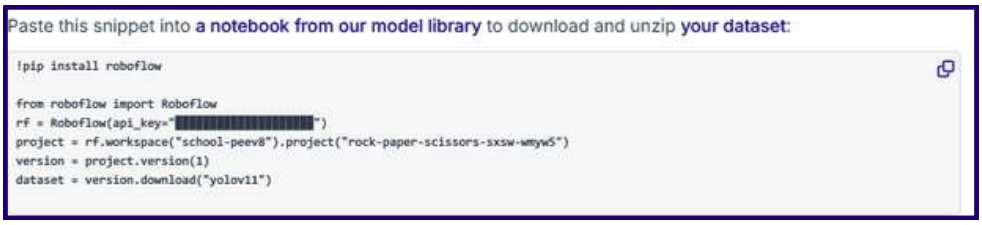



Figure 17 Snippet (2025) from www.roboflow.com; DateAdd: 09.08.2025.

In the next step, we'll open the code for the YOLOv11 algorithm, which runs on Google Colab, prepared for us by Roboflow. (https://blog.roboflow.com/yolov11-how-to-train-custom-data/ )

Visit https://colab.research.google.com/github/roboflow-ai/notebooks/blob/main/notebooks/train-yolo11-object-detection-on-custom-dataset.ipynb?ref=blog.roboflow.com and follow the instructions there.

Another option is to create a model and use it on Roboflow. This can be faster and easier, but YOLOv11 provides better and more accurate results.
In the last few years, two object detection architectures have taken the computer vision world by storm: Convolutional Neural Networks (CNNs) and Look Only Once (YOLO). Both CNNs and YOLO have helped shape the modern computer vision industry. Transformers are also playing an increasingly important role in computer vision and object detection.

CNNs use convolutions, an image processing technique, to learn features about an image. This process involves applying a sliding window to each pixel in the image to learn the features. The information obtained from the convolutions is processed by a neural network. Many CNN implementations exist, including R-CNN, Mask R-CNN, and Fast R-CNN.

The YOLO family of models has also been influential in the world of computer vision. Introduced by Joseph Redmon in 2014, YOLO has become an active area of research and development, has been adopted by a broad community, and has been implemented by many developers and researchers. YOLOv5 and YOLOv8, developed and refined by the Ultralytics team, power production object detection models worldwide.

More information about Yolo can be found at:
https://blog.roboflow.com/guide-to-yolo-models

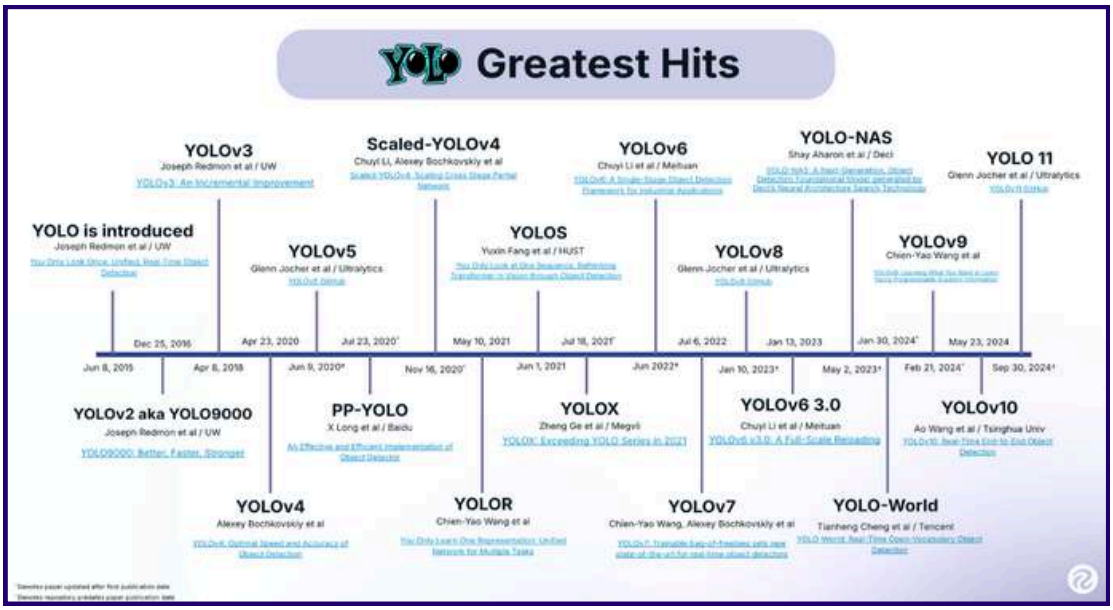



Figure 18 YOLO-History (2025) from www.roboflow.com; DateAdd: 09.08.2025

To analyze the detailed ranking of algorithms actively used in the field of computer vision around the world, you can visit the relevant address:
https://leaderboard.roboflow.com/?ref=blog.roboflow.com

# REFERENCES

## Books and Documentation


- Banerjee, Chayan & Fookes, Clinton & Karniadakis, George. (2023). Physics-Informed Computer Vision: A Review and Perspectives. 10.48550/arXiv.2305.18035.

- LeCun, Y. (2025). MNIST a Remote View of the Dataset from http://yann.lecun.com/exdb/mnist/. DateAdd: 09.08.2025.

- Mohite, Amruta & Kulkarni, Atharva & Chitnis, Rutwik & Mane, Swapnil & Asabe, Shubham. (2021). AI Inspection: Computer Vision For Visual Inspection. International Journal of Advance Research in Computer Science and Management. 7. 29.

- Object Recognition (2025). From https://viso.ai/product/computer-vision-parking-lot-occupancy-tutorial/ DateAdd: 10.08.2025.

- Using neural networks for video surveillance (2025) from https://www.videonet9.com/using-neural-networks-for-video-surveillance.html. DateAdd: 09.08.2025.


## LINKS TO LEARNING PLATFORMS

To analyze the detailed ranking of algorithms actively used in the field of computer vision around the world, you can visit the relevant address: https://leaderboard.roboflow.com/?ref=blog.roboflow.com

Roboflow is a comprehensive artificial intelligence platform designed for developing computer vision projects. https://roboflow.com/.

## PRACTICAL EXERCISE

### OBJECT HUNTERS – TEACHER'S GUIDE

**Activity Purpose**

- To understand computer vision logic
- Creating an object recognition model using Roboflow
- To experience the processes of data collection, labeling, model training, and testing

**Preparation List**

- Internet-connected computers or tablets
- Groups of 4-5 students
- Smartphones (for taking photos)
- Free Roboflow account
- Simple objects (book, pencil, water bottle, etc.)

**Duration and Schedule**

Part 1: Introduction to Computer Vision – 15 minutes
Part 2: Photo Collection – 20 minutes
Part 3: Labeling – 25 minutes
Part 4: Model Training – 20 minutes
Part 5: Test and Competition – 20 minutes

**Implementation Steps**

1. Introduce students to the concept of computer vision with a short presentation.
2. Groups select two different objects and take 15-20 photos from different angles.
3. Log in to Roboflow and create a new project.
4. Photos are uploaded, and objects are labeled by drawing frames.
5. The model is trained and tested.
6. Groups test each other's models, and scores are given.

## OBJECT HUNTERS – TEACHER'S GUIDE

**Gamification Ideas**
Time Race: Who can collect and label data the fastest?
Mixed Test: The model is tested with objects it has never seen.
Bug Hunt: Cases where the model is wrong are found and discussed.

## OBJECT HUNTERS – STUDENT ACTIVITY SHEET

**What is Computer Vision?**
Computer vision is the field of artificial intelligence that allows computers to detect and understand objects using cameras or images.

**Your Tasks**

1. As a team, choose two objects.
2. Take 15-20 photos of each object from different angles.
3. Open a project in Roboflow and upload the photos.
4. Label the objects.
5. Train your model.
6. Try to recognize the other teams' objects.

**Photographing Tips**

- Shoot from different angles.
- Try different lighting conditions.
- Shoot the subject both close up and far away.

**Labeling Guide**
Square the object in the photo and write its name.
Example: Pencil, Eraser.

**My Observations**

- Situations my model recognizes best
- Situations my model struggles with

## REFLECTION QUESTIONS

Take time to think deeply about what you have learned in this module.
Answer the following questions thoughtfully:

- **From Pixels to Perception:** In this module, you learned that computers perceive images as grids of numbers and binary code rather than shapes and colors. How does this "numerical" view of the world change your understanding of how an AI distinguishes between two different objects, such as a pencil and an eraser?

- **The Importance of Data Variety:** The "Object Hunters" activity emphasized taking photos from different angles and under various lighting conditions. Why is this diversity in your training dataset crucial for the AI to accurately recognize an object in a real-world setting?

- **Analyzing Accuracy and "Bugs":** During the testing phase, you were encouraged to identify cases where the model was wrong—a "Bug Hunt". Based on your observations, what specific environmental factors (like shadows or distance) made it difficult for your model to perform correctly??

- **Real-World Problem Solving:** Computer vision is already used in fields like healthcare, autonomous vehicles, and agriculture. Now that you have trained your own model, what is one specific problem in your school or local community that you believe could be solved using this technology?

## QUIZ WITH MULTIPLE-CHOICE QUESTIONS (one correct answer per question)

**1. What is the primary goal of computer vision?**
a) Enabling computers to think like human brains.
b) Enabling machines to "see" and interpret the world like the human eye.
c) Storing large datasets in databases.
d) Analyzing only text-based data.

**2. Which of the following is NOT a typical application of computer vision?**
a) Recognizing road signs in autonomous vehicles.
b) Facial recognition systems.
c) Analyzing user preferences in product recommendation systems.
d) Detecting tumors in medical images.

**3. What is the computer vision task that aims to determine the location and type of specific objects in an image?**
a) Image Classification
b) Object Detection
c) Image Segmentation
d) Feature Extraction

**4. Which of the following is one of the challenges computer vision systems face in the real world?**
a) Data scarcity and the diversity of this data.
b) The low-dimensionality and simple structure of visual data.
c) Consistency in illumination, position, and scale.
d) Models consistently achieving perfectly accurate results.

**5. Which of the following statements indicates that a model is overfitting?**
a) Low accuracy on both training and test data.
b) Very high accuracy on training data, low accuracy on test data.
c) High accuracy on both training and test data.
d) The model labels each image the same.

# Summary and Next Steps

Building upon the foundational knowledge established in the first course, these advanced learning materials are designed to immerse secondary school students in the practical application of artificial intelligence. The content bridges the gap between theory and practice, guiding learners through the creation of easy and basic AI models using Python and Scratch. From exploring the mathematical logic behind decision-making in an "AI Escape Room" to training machine learning models for game characters and developing computer vision applications with Roboflow, the modules offer a comprehensive, hands-on experience. A narrative-driven approach, featuring relatable characters like Aylin and Alara, ensures that complex technical concepts —such as neural networks, K-means clustering, and object recognition — remain accessible and engaging. By completing this course, students will transition from passive consumers to active creators of technology. They will gain the technical proficiency to code basic intelligent systems, the critical thinking skills to evaluate model accuracy, and the ethical awareness to deploy AI responsibly. This deep dive into programming and first steps of data science not only solidifies their STEM competencies but also prepares them for future academic and professional pursuits in a technology-driven world.

## What Next?

To support the sustainable integration of these advanced topics into the classroom, the project will next introduce an "E-Toolkit for School Educators". This resource will provide teachers with detailed lesson plans and methodological guidance to effectively deliver AI training. Furthermore, students are encouraged to showcase their newfound skills by participating in the upcoming "AI Future Explorers" contests, where they can apply their knowledge to solve real-world challenges and connect with a broader community of young innovators.

EDUCATIONAL MATERIALS WITH HANDS-ON PYTHON AND SCRATCH TASKS

# Project Partners

FUTURE-STEM-HUB

UNIVERSITÄT DUISBURG ESSEN
Open-Minded

KÜTAHYA İL MİLLÎ EĞİTİM MÜDÜRLÜĞÜ

tetra solutions

M&M profuture training

ETAP ESCOLA PROFISSIONAL


**Project Coordinator:**
University of Duisburg-Essen, Germany

**Email Address**
mustafa.bilgin@uni-due.de

**Website**
www.future-stem-hub.eu



Co-funded by the European Union

Funded by the European Union. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Education and Culture Executive Agency (EACEA). Neither the European Union nor EACEA can be held responsible for them.