AI AND DATA SCIENCE

Session 1

Date: 3-12-2025

Instructor: Ahmed Diab

Outlines

- Introduction
- Python Basics

Outcomes

·Build strong foundation in applied math - Statistics/probability.

·experience python language and libraries (NumPy, Pandas, Matplotlib) .

·Able To write EDA to prepare the data with loading - visualization -exploring - analysis - fixing the data

·Flexible with using Power BI for analysis

·Learning Machine learning (Supervised Learning → Regression – Classification, Unsupervised Learning → Clustering - Dimensionality Reduction, Neural Network) .

·Write End to End Deployment ML project

·overview about Deep Learning, CV, NLP problems/solutions

Phases

Phase 1: Data Analysis

- Python Language, OOP, Jupyter
- Git & GitHub, Venv
- Probability & descriptive statistics
- Python libraries (NumPy - SciPy - Pandas - Matplotlib)
- Power Bi & EDA

Phases

Phase 2: Machine Learning

- Linear Regression, Gradient Descent, Polynomial Regression
- Regularization (Ridge, Lasso, Cross Validation)
- Binary/Multi Classification, Evaluation Metrices
- Gradient & boosting techniques

Phases

Phase 3: Unsupervised ML & NN && End to End

- Unsupervised ml (KMeans) & KNN & PCA
- Neural Network, forward & backpropagation
- End to End ML Deployment

Overview about deep learning (nlp, cv) projects and trends

Projects:

Hospital Project (OOP, Python)

Write Python & OOP project for managing and handling hospital & patients \

Medical Cost Personal (EDA)

Write EDA for Medical Cost Personal Datasets project (Loading Data - Exploring - Analysis - Feature Engineering)

Superstore (EDA)

Write EDA for Store Sales (Loading Data - Exploring - Analysis - Feature Engineering)

- After these project you will be write structure python project, write EDA for data with analysis and extract insights and report Power BI dashboard

Projects:

Car Price Prediction Multiple (Regression)

Regression Project that aims to predict the price of cars

Credit Card Fraud Detection (Classification)

Classification Project that predicts fraud transactions with high imbalanced data

- After these 2 project you will be able to write full ml project start with loading data, write EDA, do feature engineering, modeling to deployment the model

INTRO

Evaluation - Follow Up

- Weekly progress tracking and attendance
- Weekly quizzes/assignments before sessions & HWs
- HR led feedback forms and evaluations
- Absence is not permitted without Apology email.
- Maximum 3 apology emails
- If you don't send an email, you will receive a warning email.
- 2 warning emails will get you a fire
- End-of-season performance review with certificates and recognition

Weekly content

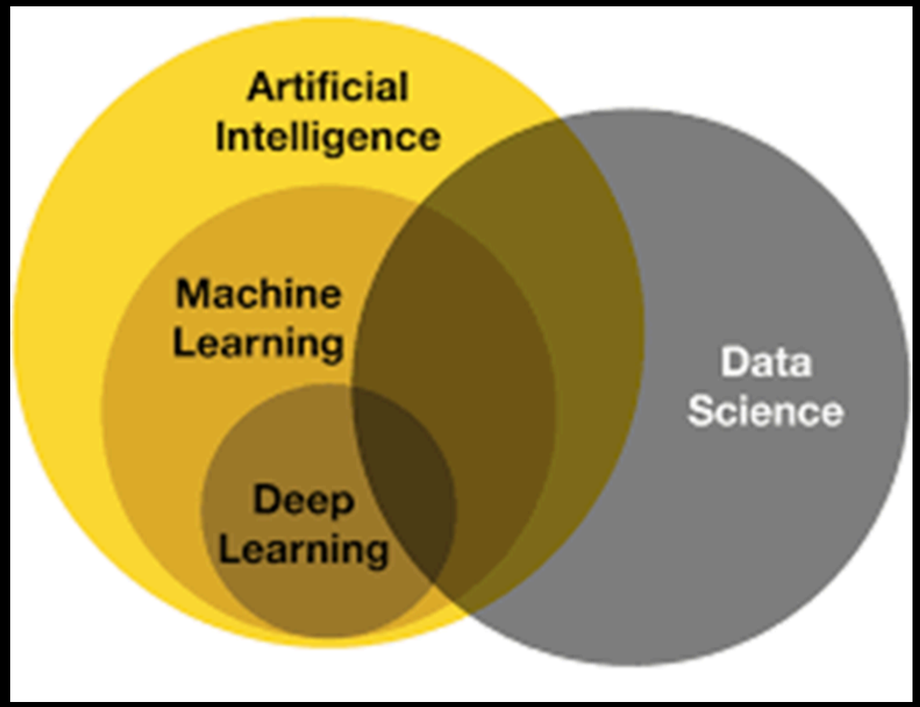
Week 1	Introduction, Python pasics	Week 12	fast-api, deployment, project 5
Week 2	Python oop, git/github, venv, project 1	Week 13	binary Classification
Week 3	Numpy, Pandas - Matplotlib	Week 14	binary Classification 2
Week 4	EDA practice, intro Descriptive Statistics, project 2	Week 15	Evaluation Metrices, intro to multi-class
Week 5	Descriptive Statistics/Probability	Week 16	Multi-class classification
Week 6	power-bi , project 3	Week 17	Clustring (Kmeans)
Week 7	Normal Equation / Calculas , Intro LR	Week 18	KNN - PCA
Week 8	Linear Regression, Gradient Descent, project 4	Week 19	Graph Theory, Neural Network
Week 9	LR , polynomial regression	Week 20	nn - BackPropagation
Week 10	Regularization Ridge/Lasso, Cross Validation	Week 21	Info about DL/CV/NLP
Week 11	bagging-boosting techniques		

ML/Data Science/AI

• AI (Artificial Intelligence): The broad field of making machines think and act intelligently.

• ML (Machine Learning): A subset of AI where machines learn patterns from data automatically.

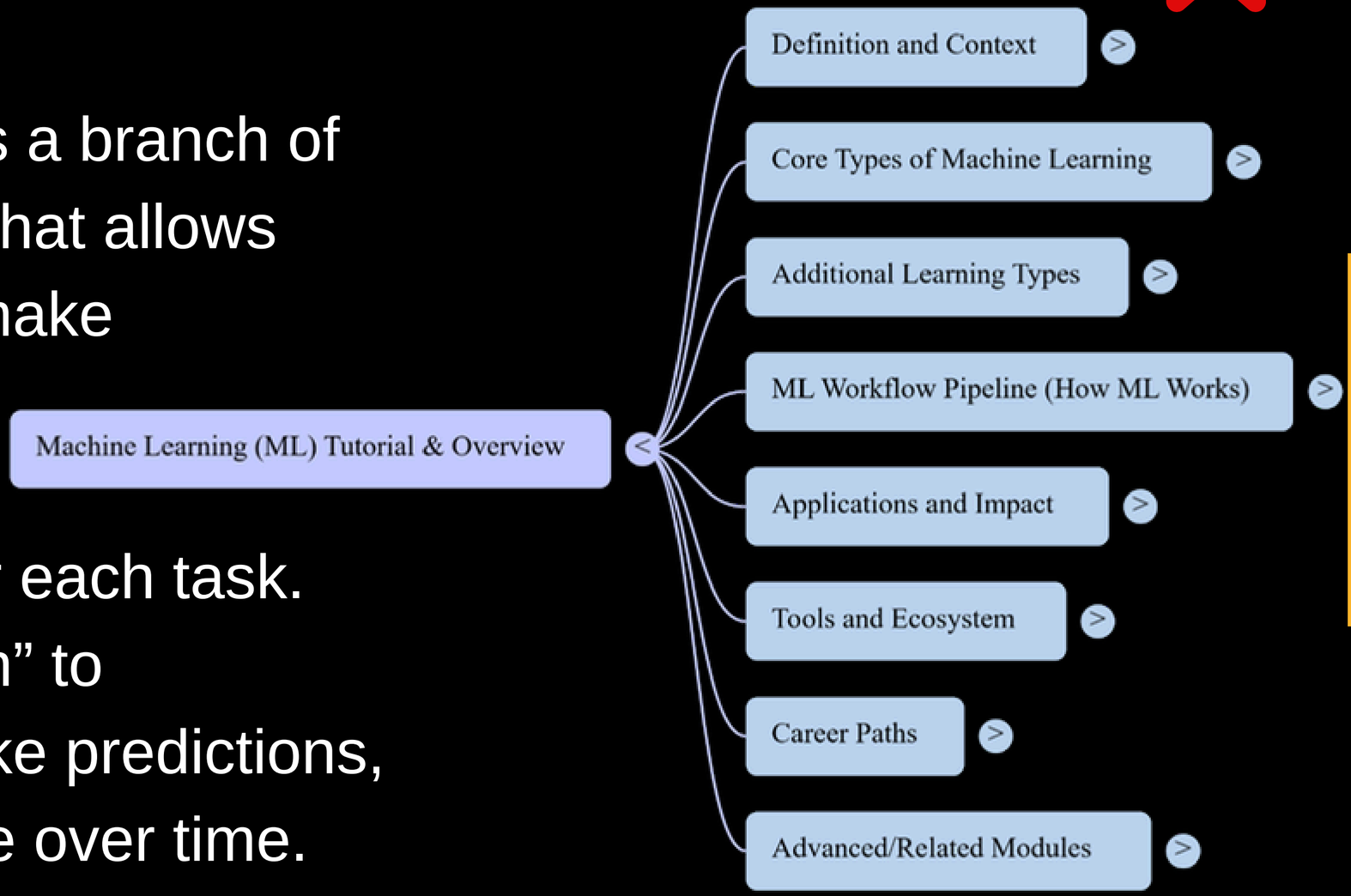
• Data Science: The process of analyzing and interpreting data to gain insights, often using ML.



ML INTRO

What is ML:

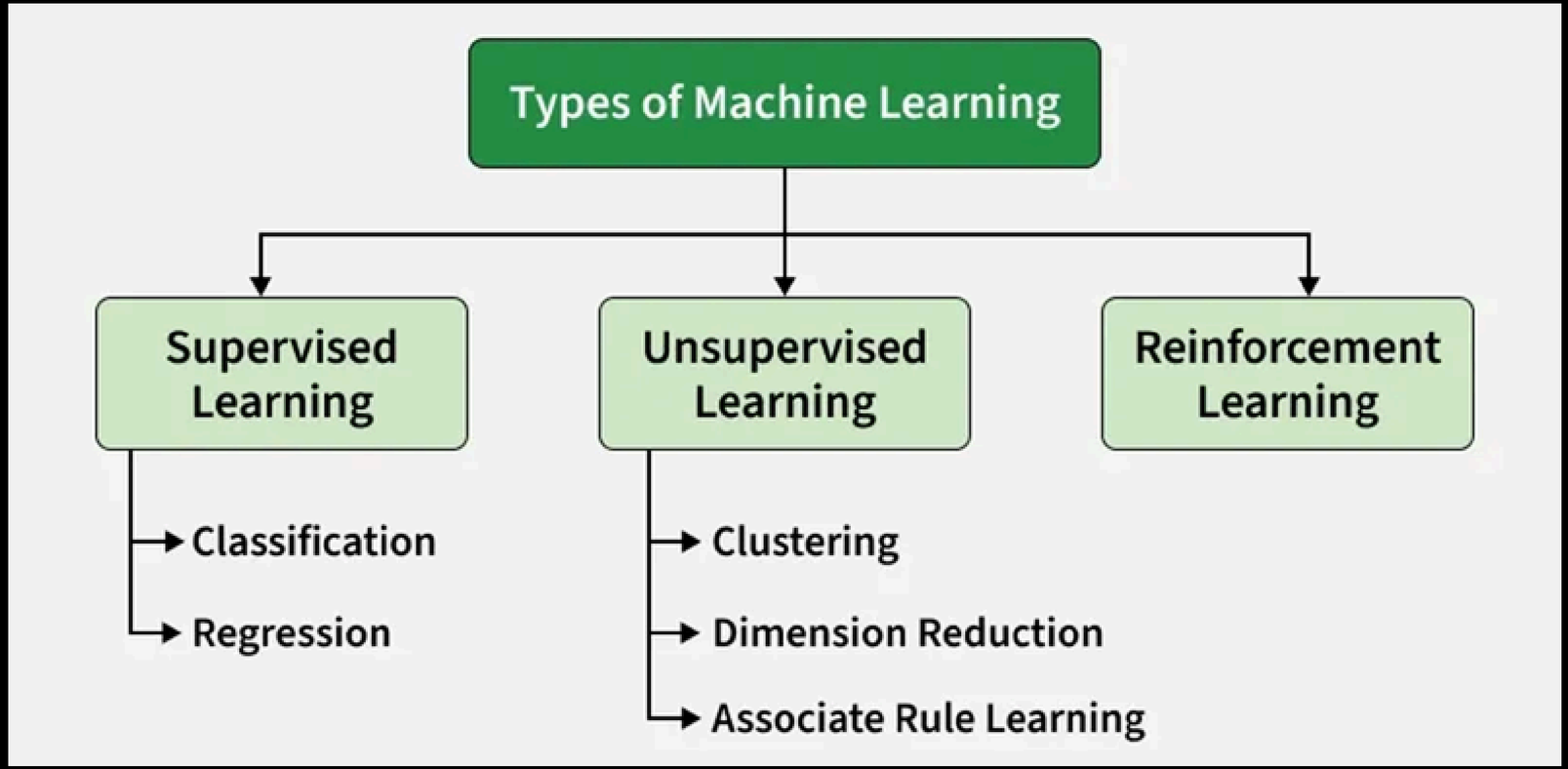
Machine Learning (ML) is a branch of artificial intelligence (AI) that allows computers to learn and make predictionsor decisions without being explicitly programmed for each task. Instead, use data to “train” to understand patterns, make predictions, and improve performance over time.



ML INTRO

Types of ML:

1. Supervised Learning
2. Unsupervised Learning
3. Reinforcement Learning



ML INTRO

Supervised Learning

In Supervised Learning, the model is trained on labeled data, meaning each input (the data) has a corresponding output (the label). The model learns the relationship between inputs and outputs and can then predict outputs for new, unseen data.

Unsupervised Learning

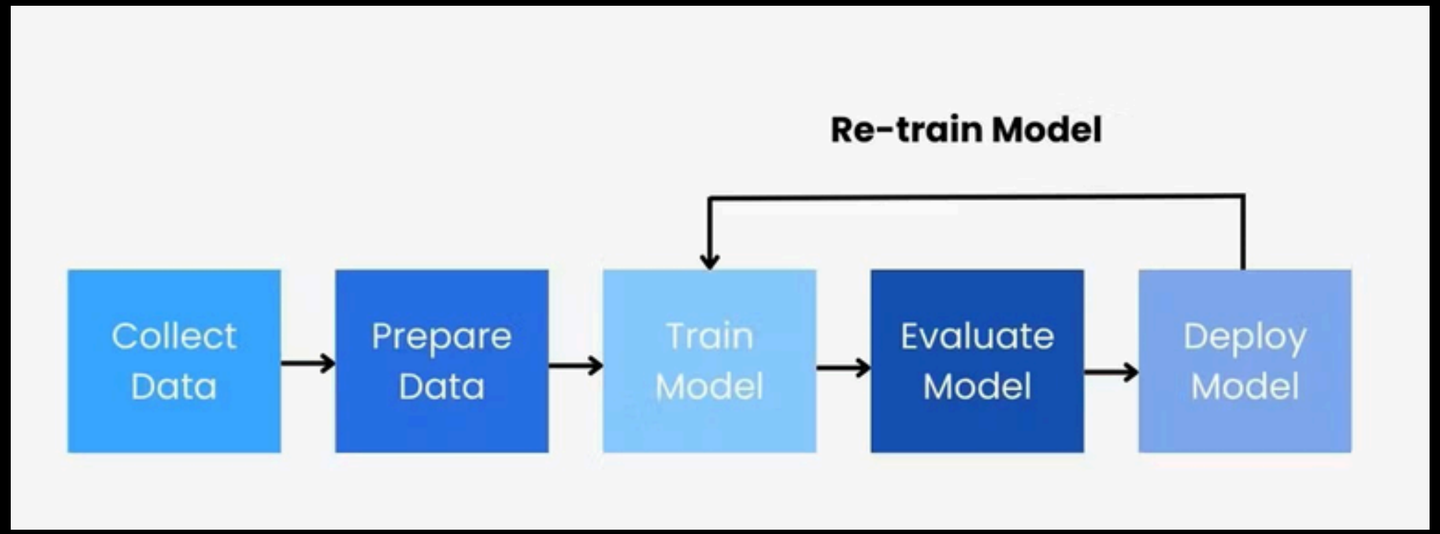
In Unsupervised Learning, the data does not have labels. The model is given only the inputs and must find patterns or relationships between them. It often groups similar data points together.

Reinforcement Learning

Reinforcement Learning (RL) involves an agent (a model) that learns through trial and error by interacting with an environment. The agent receives rewards for good actions and penalties for bad actions and adjusts its behavior to maximize rewards over time.

ML INTRO

ML Workflow



Data Collection

Data Collection phase involves systematic collection of datasets that can be used as raw data
to train model. The quality and variety of data directly affect the model’s performance

ML INTRO

Data Cleaning and Preprocessing

Data preprocessing is the first step in any data analysis or machine learning pipeline. It involves cleaning, transforming and organizing raw data to ensure it is accurate, consistent and ready for modeling

Model Training

The process of applying the machine learning algorithm on training data to train an ML model. It also includes feature engineering and the hyperparameter tuning for the model training activity.

Model Evaluation

Validating the trained model to ensure it meets original codified objectives before serving the ML model in production to the end-user.

ML INTRO

Model Deployment

Once we trained a machine learning model, we need to deploy it as part of a business application such as a mobile or desktop application. The ML models require various data points (feature vector) to produce predictions. The final stage of the ML workflow is the integration of the previously engineered ML model into existing software

Tools & Prog languages

Programming Language: Python

Writing code: Jupiter for experiment / VSCode for deployment projects

Libraries/package : NumPy, SciPy, Matplotlib, Pandas, SKLearn for ML

Deployment: fast Api , StreemLit

Intro to Python

Prepare the environment:

Download languages/packages will be used throughout the season:

1. Download Python installer (add python to PATH)

   (3.10) https://www.python.org/ftp/python/3.10.10/python-3.10.10-amd64.exe

   (3.13) https://www.python.org/ftp/python/3.13.9/python-3.13.9-amd64.exe

2. download jupyter notebook

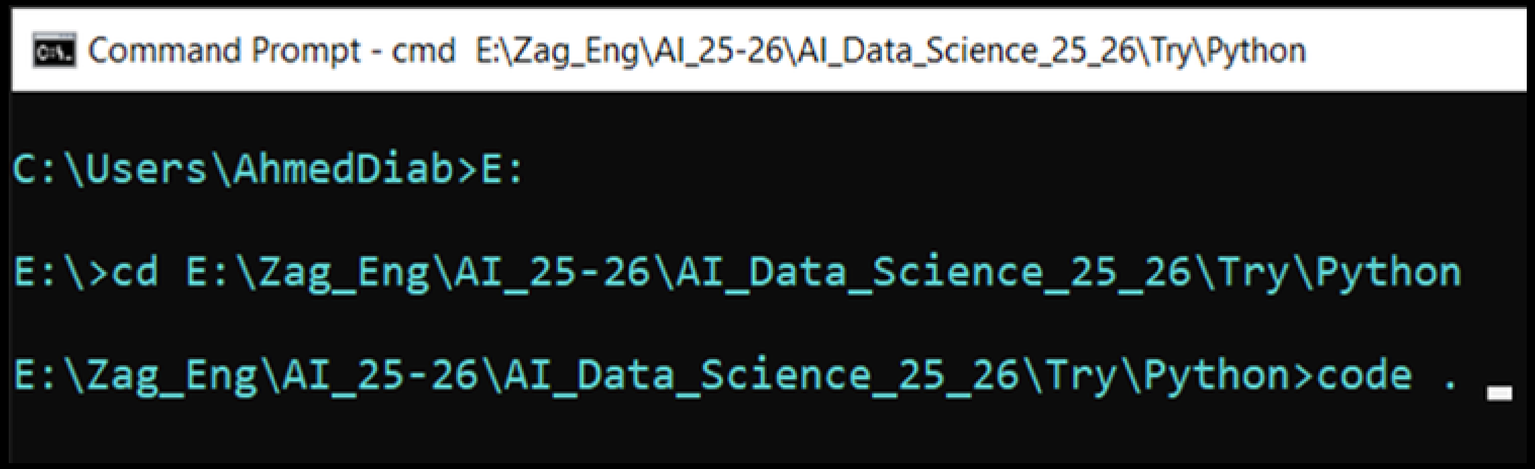
- Open cmd (Windows) or Terminal (Mac/Linux)
- Write pip install jupyter
- Verify installed jupyter –version

Launch Jupyter: open cmd and write jupyter notebook

3. download vscode

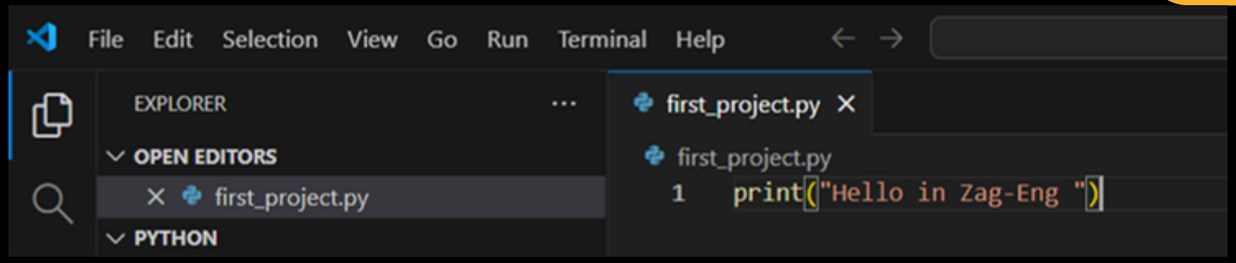
- Go to https://code.visualstudio.com/Download and choose your os
- Choose download (choose Add to PATH)

Open cmd and Write

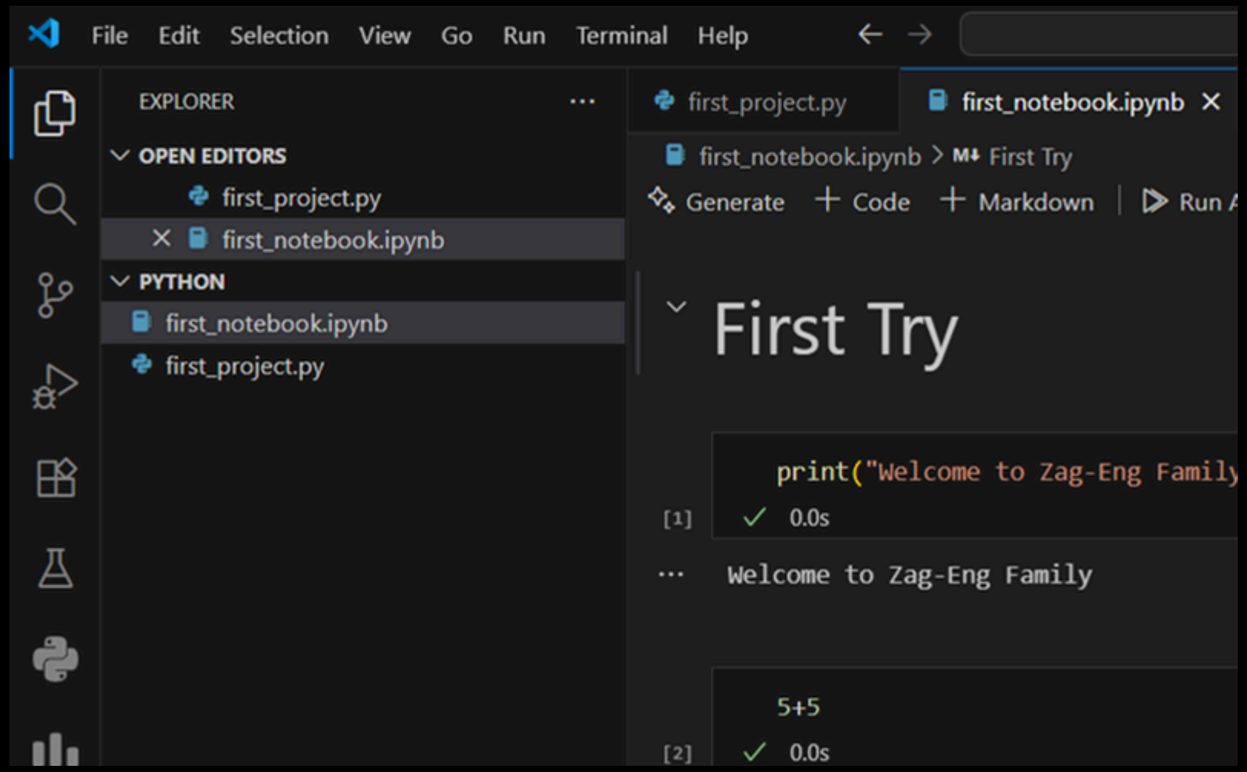


Choose new file and write the file name “first_project.py”

ML INTRO



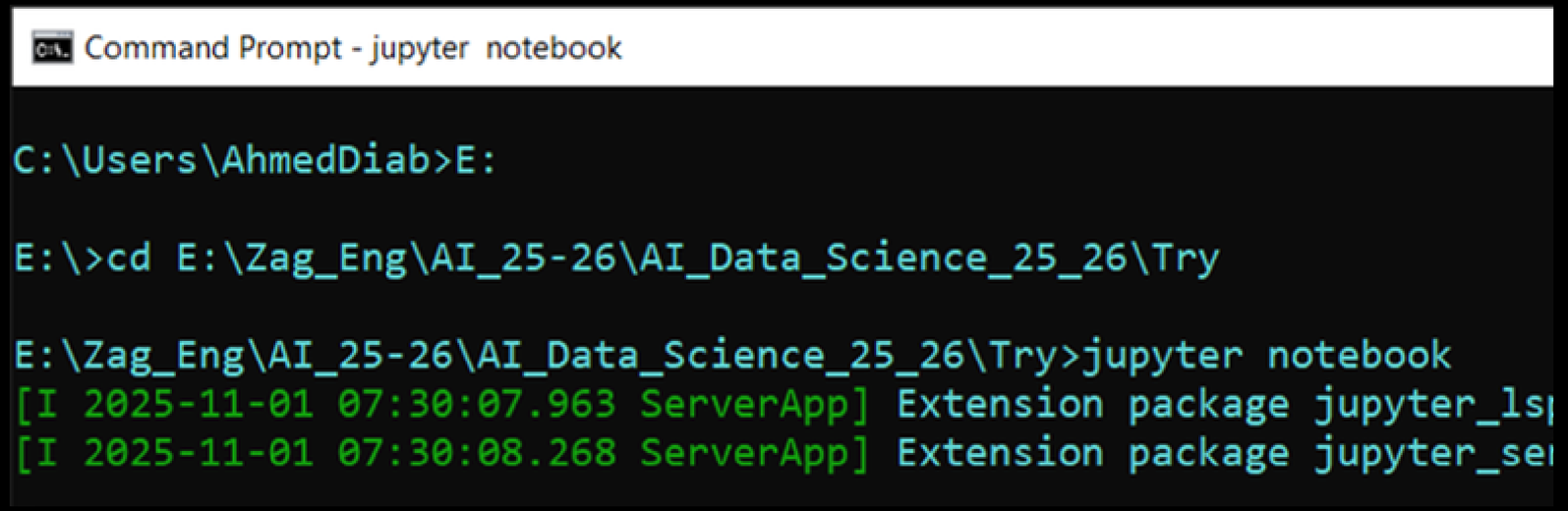
Add another file, with name “first_notebook.ipynb”



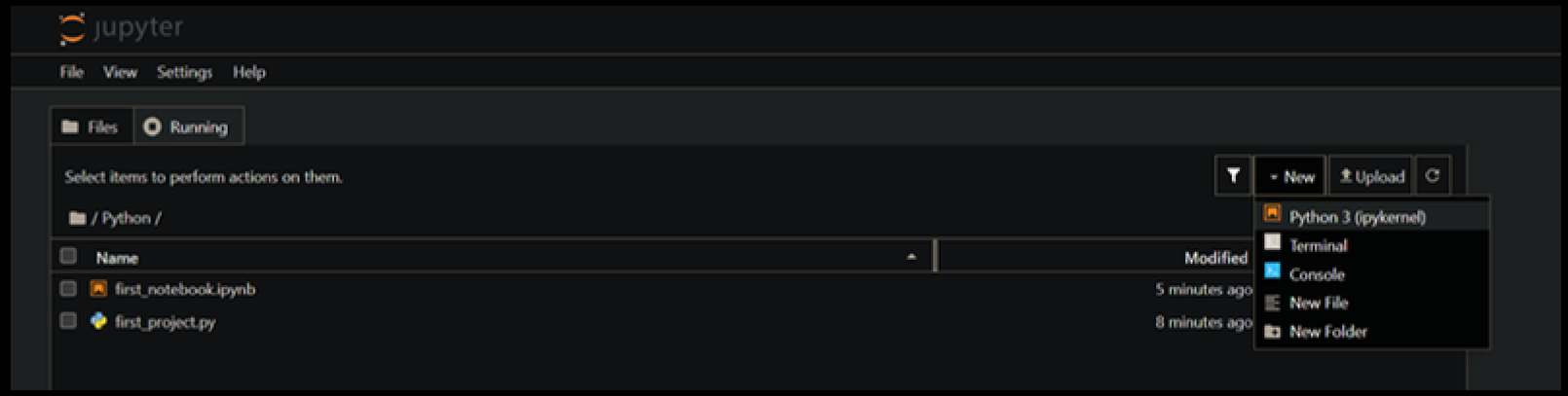
ML INTRO

If you want to write notebook in jupyter

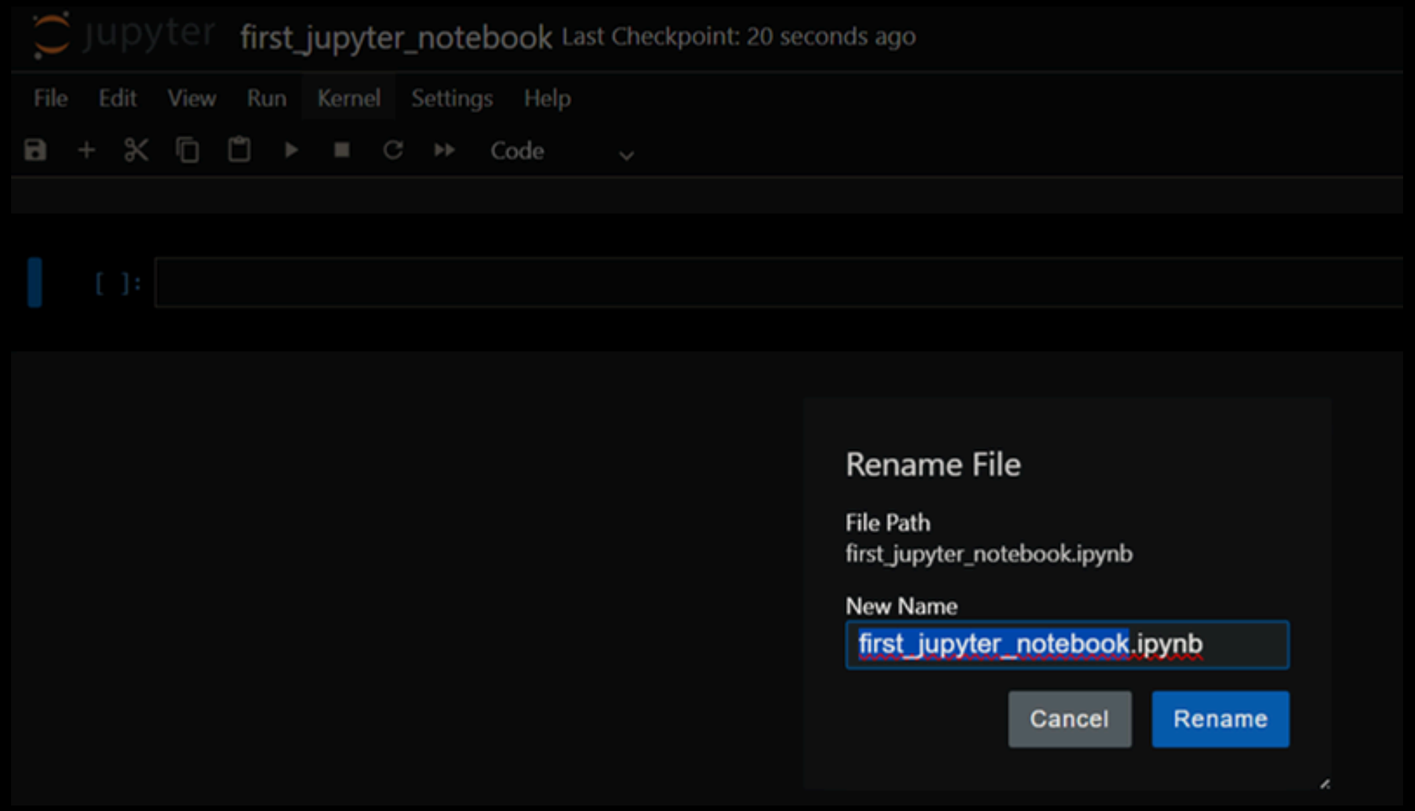
1) open cmd and write:



2)Then choose new – python 3



Rename the file name



And write kernel code with the same way

ML INTRO

Kaggle https://www.kaggle.com/
Kaggle is the data source that will use, support Free GPU.
Also, you can write notebook, join challenges, and projects.
Register and start code

Colab: https://colab.google/
Colab is a google service support also write notebooks,
Free GPU.

INTRO TO PYTHON

Python is a popular programming language. It was created by Guido van Rossum, and released in 1991.

It is used for:

- web development (server-side),
- software development,
- Mathematics,
- Machine Learning / Data Science

INTRO TO PYTHON

Why Python:

- Python works on different platforms (Windows, Mac, Linux, Raspberry Pi, etc).
- Python is more Readable and Easier to Learn/Debug
- Python runs on an interpreter system, meaning that code can be executed as soon as it is written. This means that prototyping can be very quick.
- Very Popular in AI (ML, Data Science) Field - Huge Community.

let’s start with the python basics

- input & output
- data types & variables
- if else condition
- files (read, write, handle)
- error handling (try&except)
- comments & dictionary
- string & string format
- loops for & while
- data structures (list, tuple, map, set)

INTRO TO PYTHON

Input and Output in Python

- Input means taking data from the user during program execution.
- The input() function always returns a string, even if you enter numbers.

Output in Python

- Output means displaying data or results to the user.
- It is used to show text, numbers, or variable values.

```
name = input("Enter your name: ")
print("Hello", name)
```

```
Hello Ahmed
```

```
print("Welcome to Zag-Eng!")
```

```
Hello World 123
```

```
x = 10
print("Value of x is", x)
```

```
Value of x is 10
```

```
num = input("Enter a number: ")
print(type(num))
```

```
<class 'str'>
```

```
age = int(input("Enter your age: "))
print("Next year, you'll be", age + 1)
```

```
Next year, you'll be 31
```

Comments in Python

INTRO TO PYTHON

What are Comments?

- Comments are notes written inside your code that Python ignores when running the program.
- They help you and others understand what code does.
- Think of comments as your “code explanation
- notes.”

Types of Comments

Single-line comment

- Starts with a # (hash) symbol.
- Everything after # on that line is ignored by Python.

Multi-line comment

- Used when you want to explain multiple lines of code together.
- We use triple quotes (''' or """) as a convention.

```
# ingle-line comment
print("Hello Zag Eng!")  # This prints a greeting
```

```
Hello Zag Eng!
```

```
'''
This is a multi-line comment.
It helps describe long code blocks.
Useful for documentation or explanations.
'''
print("Welcome Zag Eng Family!")
```

```
Welcome Zag Eng Family!
```

INTRO TO PYTHON

Variables in Python

A variable is a name that stores a value in your computer's memory.
It acts like a container or label for data.

- The box name = variable name
- The content inside = value stored

- You don't need to declare a variable type before using it.
- Python automatically detects the type of value you assign.

```
name = "Ahmed"
age = 21
track = "AI / Data Science"
```

```
name = "ahmed"
print(name)
print(type(name))
```

```
ahmed
<class 'str'>
```

Rules for Variable Names

- Allowed -> Letters (a-z, A-Z) and Digits (0-9) and Underscore _ .
- **Not** Allowed -> special characters like @, $, % and Variable starting with a number and Spaces in names.

```
"""
valid :
name, age_1, total_marks, _city

Invalid :
1name, total-marks, my city

"""
```

Strings & String Formatting

INTRO TO PYTHON

- A string is a sequence of characters (letters, numbers, symbols, spaces) enclosed inside quotes.
- It’s one of Python’s most common and powerful data types.
- Think of a string as a piece of text stored inside quotes.

You can use:

- Single quotes ‘ ‘
- Double quotes " "
- Triple quotes ''' ''' or """ """ → for multi-line strings.

```
text = "Python"
print(text[0])
print(text[1])
print(text[-1])
```

```
[32]
...  P
     y
     n
```

```
name = "Ahmed"
track = 'AI / Data Science'
greeting = """Welcome to Zag Eng Family!"""
```

String Formatting in Python

INTRO TO PYTHON

- Formatting helps you insert variables or expressions inside a string in a clean way.

Method 1: Using Commas (Basic Print)

```
name = "Ahmed"
age = 21
print("My name is", name, "and I am", age, "years old.")
```

```
My name is Ahmed and I am 21 years old.
```

Method 2: Using the + Operator

```
name = "Sara"
track = "Data Science"
print("Hello " + name + "! Welcome to " + track + " track.")
```

```
Hello Sara! Welcome to Data Science track.
```

Method 3: Using format() Method

```
name = "Omar"
track = "AI"
print("Hello {}, welcome to the {} track!".format(name, track))
```

```
Hello Omar, welcome to the AI track!
```

Method 4: Using f-Strings
> Modern & Recommended

```
name = "Ahmed"
age = 21
track = "AI"
print(f"My name is {name}, I'm {age} years old, and I'm learning {track}.")
```

```
My name is Ahmed, I'm 21 years old, and I'm learning AI.
```

INTRO TO PYTHON

Python Conditions — if, elif, else

- Conditional statements let your program make decisions based on certain conditions.

- Example : If it’s raining → take an umbrella
  Else → enjoy the sun

```
age = 20

if age >= 18:
    print("You can join the Zag Eng Family!")
```

[43]

```
You can join the Zag Eng Family!
```

```
age = 16

if age >= 18:
    print("You can join Zag Eng Family.")
else:
    print("Sorry, you must be 18 or older.")
```

[44]

```
Sorry, you must be 18 or older.
```

For small simple decisions, you can write if...else in one line:

```
age = 20
message = "Adult" if age >= 18 else "Minor"
print(message)
```

[50]

```
Adult
```

FOR LOOP

used when you **know how many times** to repeat.

- **For Loop with condition**

```
for name in members:
    if name == "Ahmed":
        print(name, "is the Leader ")
    else:
        print(name, "is a Member ")
```

```
Ahmed is the Leader
Sara is a Member
Omar is a Member
Nour is a Member
```

```
print("Welcome Zag Eng Family!")
print("Welcome Zag Eng Family!")
print("Welcome Zag Eng Family!")
```

```
Welcome Zag Eng Family!
Welcome Zag Eng Family!
Welcome Zag Eng Family!
```

For Loop

```
for i in range(10):
    print("Welcome Zag Eng Family!")
```

```
Welcome Zag Eng Family!
Welcome Zag Eng Family!
Welcome Zag Eng Family!
Welcome Zag Eng Family!
Welcome Zag Eng Family!
Welcome Zag Eng Family!
Welcome Zag Eng Family!
Welcome Zag Eng Family!
Welcome Zag Eng Family!
Welcome Zag Eng Family!
```

INTRO TO PYTHON

- Using continue

continue

```
for name in members:
    if name == "Sara":
        continue
    print(name)
```

```
Ahmed
Omar
Nour
```

- Using break

break

```
for name in members:
    if name == "Omar":
        break
    print(name)
```

```
Ahmed
Sara
```

- **Using else**

else with loops

```
for i in range(3):
    print("Step:", i)
else:
    print("Loop finished successfully!")
```

6]

```
Step: 0
Step: 1
Step: 2
Loop finished successfully!
```

WHILE LOOP

- used when you don’t know how many times but repeat until a condition is false.

```
i = 1
while i <= 5:
    print("Zag Eng Loop", i)
    i += 1
```

```
Zag Eng Loop 1
Zag Eng Loop 2
Zag Eng Loop 3
Zag Eng Loop 4
Zag Eng Loop 5
```

```
command = ""
while command != "exit":
    command = input("Type 'exit' to stop: ")
```

Files & File Handling

INTRO TO PYTHON

- Working with files is one of the most important skills for every developer and data scientist.
- In real-world projects whether in AI, Data Science, or Web Development
- you'll often need to read, write, and manage files & folders.

File handling means working with files on your computer using Python code.

- Create new files
- Read data from existing files
- Write or add new data
- Delete or rename files and folders

INTRO TO PYTHON

- Opening Files in Python
- Opening and Reading a File
- Reading Line by Line
- Writing to Files
- appends new data
- with Statement (Best Practice)

```
file = open("members.txt", "r")
content = file.read()
print(content)
file.close()
```

```
Ahmed
Sara
```

Writing

```
file = open("leaders.txt", "w")
file.write("Ahmed - Leader\n")
file.write("Sara - Vice Leader\n")
file.close()
```

Append

```
file = open("leaders.txt", "a")
file.write("Omar - Member\n")
file.close()
```

Files & File Handling

INTRO TO PYTHON

- Working with files is one of the most important skills for every developer and data scientist.
- In real-world projects whether in AI, Data Science, or Web Development
- you'll often need to read, write, and manage files & folders.

File handling means working with files on your computer using Python code.

- Create new files
- Read data from existing files
- Write or add new data
- Delete or rename files and folders

INTRO TO PYTHON

Working with Folders

- Check Current Directory
- Create a New Folder
- Create Nested Folders
- List Files and Folders

```
import os
```

```
print(os.getcwd())
```

```
e:\Zag_Eng\AI_25-26\Content\Week1_intro_python
```

create new folder

```
os.mkdir("ZagEngData")
```

create nested folders

```
os.makedirs("ZagEng/Projects/Data")
```

INTRO TO PYTHON

- Rename file
- Delete file
- Delete folder

```
[4]  os.rename("leaders.txt", "team_leaders.txt")

[5]  os.remove("team_leaders.txt")

[6]  os.rmdir("ZagEngData")
```

Python Lists

- A list is a collection of ordered, changeable items (elements) in Python.
- It can store different data typesintegers, strings, floats, even other lists!
- Lists are indexed, starting from 0
- Lists are mutable (you can change items after creation)
- Extract parts of a list using [start:end:step]
- Avoid using = because it links both lists to the same memory.

```
[89]  members = ["Ahmed", "Sara", "Omar", "Nour"]
      print(members)

...   ['Ahmed', 'Sara', 'Omar', 'Nour']
```

```
[90]  print(members[0])
      print(members[-1])

...   Ahmed
      Nour
```

```
[91]  members[1] = "Salma"
      print(members)

...   ['Ahmed', 'Salma', 'Omar', 'Nour']
```

LIST COMPREHENSION

INTRO TO PYTHON

List comprehension = a short and powerful way to create lists using one line of code.

```
new_list = [expression for item in iterable if condition]
```

```
[101]
numbers = [1, 2, 3, 4, 5]
squares = [n**2 for n in numbers]
print(squares)
```

```
[1, 4, 9, 16, 25]
```

```
[102]
even = [n for n in range(10) if n % 2 == 0]
print(even)
```

```
[0, 2, 4, 6, 8]
```

Tuple in Python

INTRO TO PYTHON

- A tuple in Python is a collection data type that can store multiple items in a single variable just like a list, but it cannot be changed (it’s immutable)
- Accessing tuple items:

```
my_tuple = (item1, item2, item3)
```

```
student = ("Ahmed", 21, "Engineering")
print(student)
```

- Tuples are Immutable, u cannot modify it :

```
x = (1, 2, 3)
x[0] = 5
```

```
[107]
...
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[107], line 2
      1 x = (1, 2, 3)
----> 2 x[0] = 5

TypeError: 'tuple' object does not support item assignment
```

INTRO TO PYTHON

- **Tuple Packing:**
  You can assign multiple values into a tuple

- **Tuple Unpacking:**
  You can unpack tuple values into separate variables:

- **Nested Tuples**

```
[109]
student = ("Omar", 22, "CS")
```

```
[110]
(name, age, dept) = student
print(name)
print(age)
print(dept)
```

```
Omar
22
CS
```

```
[111]
info = (("Ahmed", 21), ("Omar", 22), ("Sara", 23))
print(info[1][0])
```

```
Omar
```

```
[112]
position = (10.5, 20.8)

(x, y) = position

print(f"The bird is at position X={x}, Y={y}")
```

```
The bird is at position X=10.5, Y=20.8
```

Dictionary

- A dictionary is a data structure in Python used to store data in key–value pairs.
- Each key is unique, and it maps to a specific value

- Accessing Data

- Add or Update items

- Loop Through Dictionary

```
student = {"name": "Omar", "age": 22, "track": "AI"}

print(student["name"])
print(student.get("age"))
```

```
Omar
22
```

```
student["city"] = "Zagazig"
student["age"] = 23
print(student)
```

```
{'name': 'Omar', 'age': 23, 'track': 'AI', 'city': 'Zagazig'}
```

```
student = {"name": "Sara", "age": 20, "dept": "CS"}

for key in student:
    print(key, "→", student[key])
```

```
name → Sara
age → 20
dept → CS
```

Set

- A set is a collection of unique, unordered elements.
- It's used when you want to:
  > Store only unique values (no duplicates)
  > Perform mathematical operations like union, intersection, etc.

- Add or Remove elements

- Check Membership
- Empty Set

```
a = {} -> empty dictionary
b = set() -> empty set
```

```
fruits = {"apple", "banana"}

fruits.add("cherry")
fruits.update(["mango", "kiwi"])
fruits.remove("banana")
fruits.discard("apple")
```

```
for fruit in fruits:
    print(fruit)
```

```
kiwi
mango
cherry
```

Error Handling

- When you write code, sometimes errors (exceptions) happen like dividing by zero, using a wrong variable name, or reading a file that doesn't exist.

- If these errors aren't handled, the program crashes.

- Python gives us tools to handle errors safely using try, except, else, and finally so your program keeps running smoothly.

```
try:
    num = int(input("Enter a number: "))
    print(10 / num)
except:
    print("Something went wrong!")
```

[1]

```
Something went wrong!
```

```
try:
    a = int(input("Enter number A: "))
    b = int(input("Enter number B: "))
    result = a / b
    print(result)

except ZeroDivisionError:
    print("You can't divide by zero!")

except ValueError:
    print("Please enter numbers only!")

except Exception as e:
    print("Unknown error:", e)
```

```
Please enter numbers only!
```

Why Use try/except?

- To prevent crashes and keep the program running
- To show friendly messages to users
- To handle different error types in a professional way

- Using else

- Using finally

else

```
try:
    number = int(input("Enter a number: "))
except ValueError:
    print("Invalid input!")
else:
    print("You entered:", number)
```

[3]

```
Invalid input!
```

finally

```
try:
    file = open("data.txt", "r")
    data = file.read()
    print(data)
except FileNotFoundError:
    print("File not found!")
finally:
    file.close()
    print("File closed.")
```

[4]

```
File not found!
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[4], line 8
      6     print("File not found!")
      7 finally:
----> 8     file.close()
      9     print("File closed.")

NameError: name 'file' is not defined
```

Function

A function is a block of code that performs a specific task and can be called multiple times.

- Reusable code
- Easier to debug
- Makes programs organized
- Helps modular programming

```
def greet():
    print("Hello Zag-Eng Family!")

greet()
```

[1] ✓ 0.0s

```
Hello Zag-Eng Family!
```

```
def greet(name):
    print(f"Hello, {name}!")

greet("Ahmed")
greet("Sara")
```

[2] ✓ 0.0s

```
Hello, Ahmed!
Hello, Sara!
```

Variable-Length Arguments

- *args → for multipl positional arguments
- **kwargs → for multiple keyword arguments

- Nested Functions

```
def show_members(*members):
    print("Members:", members)

show_members("Ahmed", "Sara", "Omar")
```

[7] ✓ 0.0s

```
... Members: ('Ahmed', 'Sara', 'Omar')
```

Nested Functions

```
def outer():
    def inner():
        print("Hello from inner function!")
    inner()

outer()
```

[8]

Lambda

A lambda function is a small anonymous function that can take any number of arguments but has only one expression.

- Use map with lambda

```
square = lambda x: x ** 2
print(square(5))
```

✓ 0.0s

```
25
```

```
multiply = lambda a, b: a * b
print(multiply(4, 5))
```

✓ 0.0s

```
20
```

```
numbers = [1, 2, 3, 4]
squares = list(map(lambda x: x**2, numbers))
print(squares)
```

✓ 0.0s

```
[1, 4, 9, 16]
```

References

-Ml introduction Hesham Asem - https://www.youtube.com/playlist?list=PL6-3IRz2XF5Vf1RAHyBo4tRzT8lEavPhR .

Big data Arabic python part 1 - https://www.youtube.com/watch?v=XKQaCF_Om8o , part2- https://www.youtube.com/watch?v=mlbe7Vxr7yA to 6:20

https://www.geeksforgeeks.org/machine-learning/machine-learning/

https://www.geeksforgeeks.org/machine-learning/introduction-machine-learning/

https://www.datacamp.com/blog/the-difference-between-ai-and-machine-learning

Practical Time

Problems

https://www.hackerrank.com/challenges/py-if-else/problem?isFullScreen=true

https://www.hackerrank.com/challenges/python-arithmetic-operators/problem?isFullScreen=true

https://www.hackerrank.com/challenges/python-division/problem?isFullScreen=true

https://www.hackerrank.com/challenges/find-a-string/problem?isFullScreen=true

https://www.hackerrank.com/challenges/text-alignment/problem?isFullScreen=true

دمتم سالمين