Lab 01

Objectives:

1. Learn to use GitHub to upload and manage lab notebooks.
2. Set up Python environment using Anaconda or Google Colab.
3. Practice basic Python programming: variables, loops, conditionals, and functions.

GitHub

What GitHub Actually Is (Very Simply)

- **GitHub** = a website where you store your code and files online
- It keeps **history** of changes
- It allows **sharing** and **collaboration**

Think of it like:

Google Drive + history + teamwork (for code & data)

Important Terms

Term	**Meaning (Simple)**
Repository (Repo)	A folder/project on GitHub
Git	Software that tracks changes
Commit	Saving a change with a message
Push	Upload your changes to GitHub
Clone	Download a GitHub repo to your PC
Fork	Make your own copy of someone else's repo

GitHub Account

1. Go to **github.com**
2. Click **Sign up**
3. with gmail or
4. Use:
    a. Real name
    b. University email (if possible)
5. Open Repository

6. https://github.com/junaidiqbalbaig-cloud/BCS-FA23-ML
7. Navigate to your Section folder:
    a. Section_B
    b. Section_C
    c. Section_D
8. Click **Add file → Upload files**
9. Upload your notebook or text file
10. File name = Roll Number (e.g., FA23-BCS-123.ipynb)
11. Inside the file, write:
12. Scroll down and click **Commit changes**
13. Your submission is now in the repository

Python Environment Setup

Option 1: Using Anaconda (Local Installation)

Step 1: Download Anaconda

1. Go to: https://www.anaconda.com/products/distribution
2. Click **Download → Windows → Python 3.11 version** (or latest)
3. Wait for download

Step 2: Install Anaconda

1. Open downloaded installer
2. Click **Next** → Accept License
3. Choose **Install for "Just Me"** → Next
4. Select installation path (default is fine) → Next
5. Check **Add Anaconda to PATH** → Important for command-line (Optional, but recommended)
6. Click **Install** → Wait for completion
7. Click **Finish**

Step 3: Open Anaconda Navigator

1. Press **Windows Key** → search **Anaconda Navigator** → Open
2. You will see GUI with apps:
    - **Jupyter Notebook**
    - **JupyterLab**
    - **Spyder**
    - **VS Code**

For ML labs, we will mainly use **Jupyter Notebook** or **JupyterLab**.

Step 4: Create a Conda Environment for ML (Optional but Recommended)

Open **Anaconda Prompt** and type:

bash

conda create -n ml_lab python=3.11

- This creates a **separate environment** named ml_lab
- Activate it:

bash

conda activate ml_lab

Step 5: Install Common ML Libraries

Inside activated environment:

bash

pip install numpy pandas matplotlib seaborn scikit-learn jupyter

- These libraries cover **basic ML labs**
- GPU/Deep Learning can be added later

Step 6: Start Jupyter Notebook

1. In **Anaconda Prompt**, type:

   jupyter notebook

2. Browser opens → navigate to your Lab 01 folder
3. Open .ipynb files → run cells

Option 2: Using Google Colab (No Installation)

Perfect for students **without strong PCs** or who don't want to install anything.

Step 1: Open Google Colab

1. Go to: https://colab.research.google.com/
2. Sign in with **Google account**

Step 2:

Open Lab Notebook

1. Click **File → Open notebook → GitHub**
2. Paste **your repository link**:

https://github.com/junaidiqbalbaig-cloud/BCS-FA23-ML

3. Click **Enter** → Select the notebook (e.g., hello_ml.ipynb)

OR

Create a New Notebook in Google Drive

- Click **File → New notebook**
- The new notebook will open in your browser
- Click **Play button** to run each cell
- Students can **edit the notebook directly**
- Changes are saved in their **Google Drive copy**

Optional: Students can **download the notebook** and upload to your GitHub folder for Lab submission.

Python Coding Basics

a) Print and Variables

```
# Print a message to the screen
print("Hello Machine Learning!")


# Variables store information we can use later
name = "Ali Ahmed"        # String variable
roll_no = "FA23-BCS-123"  # String variable
section = "B"             # String variable


# Print variables with labels
print("Name:", name)
print("Roll No:", roll_no)
print("Section:", section)
```

Note:

- print() displays text or variables in the notebook output.
- A **variable** is like a labeled box where you store information.
- Strings are text enclosed in quotes " " or ' '.
- We can combine text and variables in print() using commas.

b) Data Types and Simple Operations

```
# Numbers (integers and floats)
a = 10      # integer
b = 3       # integer
```

```
# Arithmetic operations
print("Addition:", a + b)      # Adds two numbers
print("Subtraction:", a - b)   # Subtracts b from a
print("Multiplication:", a * b)# Multiplies a and b
print("Division:", a / b)      # Divides a by b (result is float)
print("Integer Division:", a // b) # Divides and rounds down
print("Modulus:", a % b)       # Remainder of division
print("Exponent:", a ** b)     # a raised to the power b

# Strings
greeting = "Welcome to ML"
print(greeting.upper())  # Converts string to uppercase
print(greeting.lower())  # Converts string to lowercase
print(len(greeting))     # Number of characters in string
```

Note:

- Numbers: Python supports integers (int) and decimals (float).
- Arithmetic operators: + - * / // % **.
- Strings are sequences of characters; methods like .upper(), .lower(), .len() help manipulate text.

c) Lists and Loops

```
# A list is a collection of items
marks = [80, 75, 90, 85]  # list of numbers

# For loop: repeats actions for each item in the list
for mark in marks:
   print("Mark:", mark)
```

Note:

- **List**: A container that holds multiple items, accessed one by one.
- **For loop**: Lets us repeat actions automatically.

- mark is a temporary variable that takes **each value in the list** one by one.

- Loops avoid writing repetitive code.

Example Modification:

```
# Print only marks greater than 80
for mark in marks:
    if mark > 80:
        print("Good mark:", mark)
```

- Combines **loops** + **conditionals**

d) Conditionals

```
score = 85
# If-elif-else checks conditions and executes the first one that is True
if score >= 90:
    print("Excellent")
elif score >= 75:
    print("Good")
else:
    print("Needs Improvement")
```

Note:

- if checks a condition; if True, executes the code block.
- elif (else if) checks **another condition** if the first was False.
- else executes if **all previous conditions were False**.
- Code blocks are **indented**, usually 4 spaces. Indentation is mandatory in Python.

e) Functions

```
# Function: a reusable block of code
def greet(name):
    """This function takes a name and returns a greeting."""
    return "Hello " + name

# Call the function
```

```
print(greet("Ali"))

print(greet("Sara"))
```

Note:

- def defines a **function** with a name (greet) and input parameters (name).
- return sends a value back to the caller.
- Functions **avoid repeating code**; we can call them with different values.
- Triple quotes """ """ are used for **docstrings** (optional explanation of the function).

f) Taking Input in Python

```
# input() allows the user to type something while the program is running

name = input("Enter your name: ")  # User types their name, it is stored in variable 'name'

age = input("Enter your age: ")    # User types age, stored as a string


print("Hello", name, "! You are", age, "years old.")
```

Note:

- input("Prompt text") shows the message inside the quotes and waits for the user to type something.
- Everything entered by the user is stored as a **string**.
- If you want a **number**, you need to convert it using int() or float().

```
# Example: converting input to integer

age = int(input("Enter your age: "))  # Converts input string to integer

print("Next year you will be", age + 1, "years old.")
```

Tasks

Task 1: Personal Information

- Ask the user to input their **name**, **roll number**, **section**, and **age**.
- Print a formatted message like:
- My name is Ali, Roll No FA23-BCS-123, Section B, Age 20.

Task 2: Simple Calculator

- Ask the user to input **two numbers**.
- Perform and print **addition, subtraction, multiplication, division, integer division, modulus, and exponent**.

Task 3: List Operations

- Create a **list of 7 numbers** (can be hardcoded).
- Print the **sum**, **maximum**, and **minimum** of the list.
- Print only the numbers **greater than 50** using a loop.

Task 4: String Manipulation

- Ask the user to input a **sentence**.
- Print:
    - The sentence in **uppercase** and **lowercase**
    - The **number of words**
    - The **number of vowels**

Task 5: Loops and Conditionals (FizzBuzz)

- Print numbers from 1 to 20.
- For multiples of 3, print "Fizz"
- For multiples of 5, print "Buzz"
- For numbers divisible by both 3 and 5, print "FizzBuzz"

Task 6: Functions

- Write a function square(num) that returns the **square of a number**.
- Write another function is_even(num) that returns True if the number is even, else False.
- Test both functions with **3 different numbers**.

Optional Bonus: Nested Loops

- Print a **multiplication table** from 1 to 5 using nested loops.