Scenario-Based Course Tasks: Python Fundamentals for AI in Agriculture

Scenario Background

You are a beginner data scientist working with an **agricultural research team**. The team is collecting simple crop yield data from small farms to understand productivity trends and prepare for future AI-based automation. As part of the training, you will build small Python programs to **simulate, process, analyze, and visualize** agricultural data.

Each task builds on the previous one, gradually taking you from **basic Python logic** to **applied data analysis and visualization**.

Task 1: Fundamentals of Python (Logic & Data Representation)

Scenario: The research team needs a simple program to record farmer details. You are tasked to implement:

- A script that prints **"Welcome to Crop Yield Analyzer"**.
- Variables to store farmer's name, farm size (in acres), and main crop.
- Demonstrate data types: integers, floats, strings, and booleans.
- Implement **if-else**: e.g., check if farm size > 10 acres → "Large farm", else → "Small farm".
- Implement **loops** to generate a sample list of crops: ["Wheat", "Rice", "Corn"].

Learning outcome: Data types, variables, conditionals, loops.

Task 2: Static Data Processing (Pre-Generated Dataset)

Scenario: Researchers have collected **static sample data** of crop yields. The dataset looks like this:

Farmer	Crop	Acres	Yield (tons)	Region
Farmer 1	Wheat	5	12	North
Farmer 2	Rice	8	18	South
Farmer 3	Corn	4	9	East
Farmer 4	Wheat	6	14	West

Your tasks:

- Store this dataset as a **list of dictionaries** or a **list of tuples**.
- Write Python code to:
    - Print all farmers growing **Wheat**.
    - Calculate **total yield across all farmers**.
    - Find the **farmer with maximum yield**.

Learning outcome: Lists, dictionaries, loops, basic aggregation.

Task 3: Data Analysis (Beginner Level)

Scenario: Now the team wants **basic analysis** of the yield data.

- Implement **functions** to:
    - Calculate **average yield per crop type**.
    - Count how many farmers are in each region.
- Add **exception handling**: If data is missing (e.g., no yield for a farmer), handle it gracefully.

Learning outcome: Functions, error handling, reusable code.

Task 4: Statistical Analysis

Scenario: Researchers want to know the **statistical distribution** of yields.

- Use Python to calculate:
    - Mean, Median, Mode of crop yields.
    - Standard deviation to check variability.
    - Correlation between **acres vs yield**.

Sample Output Example:

Mean Yield = 13.25 tons

Median Yield = 13 tons

Standard Deviation = 3.5

Correlation (Acres, Yield) = 0.89

Learning outcome: Basic statistics in Python (statistics module, NumPy optional).

Task 5: Data Visualization (Graphs & Tables)

Scenario: Present results visually for non-technical researchers.

- Use **Matplotlib / Pandas** to:
    - Create a **bar chart**: Average yield per crop.
    - Create a **pie chart**: Percentage of farmers by region.
    - Create a **scatter plot**: Acres vs Yield.
- Show results in a simple **table format** using Pandas DataFrame.

Learning outcome: Beginner data visualization, tables, clear presentation.

Extension (Optional Advanced Mini-Project)

If students finish early:

- Predict **expected yield** for a farmer using a **simple formula**:
- Predicted Yield = Acres × Average Yield per Acre (based on dataset)
- Display results for each farmer.

Overall Outcome:
By the end, learners will be able to:

- Use **Python fundamentals** (variables, conditionals, loops).
- Work with **static datasets** (lists, dicts, tuples).
- Apply **basic analysis & statistics**.
- Produce **visualizations** that simulate beginner-level AI data processing.
- Work in a **realistic agricultural AI scenario** while still at a **fundamental Python level**.
- Upload final working code along with technical document guide in a new repository using GitHub Desktop.
- Prepare a submission document and include screenshots of output and link to GitHub repository for code review.

Technical Guide: Python Fundamentals for AI in Agriculture

This guide will help you solve the **Crop Yield Analysis Scenario** step by step. Each task builds on the previous one. Follow the instructions carefully, write Python code, and test your results.

---

1. Setup Your Environment

- Install **Python 3.9+**
- Recommended tools:
    - Anaconda (comes with Jupyter Notebook & libraries pre-installed)
    - OR install manually:
    - pip install numpy pandas matplotlib
- Use **Jupyter Notebook**, **VS Code**, or **PyCharm** to run your code.

---

2. Task 1: Fundamentals of Python

Create a file task1_basics.py

Practice:

```
# Print message
print("Welcome to Crop Yield Analyzer")


# Variables
farmer_name = "Farmer 1"
farm_size = 12
crop = "Wheat"


# Conditionals
if farm_size > 10:
   print("Large farm")
```

```
else:
    print("Small farm")

# Loop example
crops = ["Wheat", "Rice", "Corn"]
for c in crops:
    print(c)
```

Output: Program prints a welcome message, farm classification, and available crops.

---

3. Task 2: Static Data Processing

Store dataset in **list of dictionaries**:

```
data = [
    {"Farmer": " Farmer 1", "Crop": "Wheat", "Acres": 5, "Yield": 12, "Region": "North"},
    {"Farmer": " Farmer 2", "Crop": "Rice", "Acres": 8, "Yield": 18, "Region": "South"},
    {"Farmer": " Farmer 3", "Crop": "Corn", "Acres": 4, "Yield": 9, "Region": "East"},
    {"Farmer": " Farmer 4", "Crop": "Wheat", "Acres": 6, "Yield": 14, "Region": "West"},
]
```

Example operations:

```
# Print wheat farmers
for record in data:
    if record["Crop"] == "Wheat":
        print(record["Farmer"])

# Total yield
total_yield = sum([record["Yield"] for record in data])
print("Total Yield:", total_yield)
```

```
# Max yield
max_farmer = max(data, key=lambda x: x["Yield"])
print("Top Farmer:", max_farmer["Farmer"], "with", max_farmer["Yield"], "tons")
```

---

4. Task 3: Data Analysis

Define **functions** for analysis:

```
def average_yield(crop):
    values = [r["Yield"] for r in data if r["Crop"] == crop]
    return sum(values) / len(values) if values else 0


def count_by_region():
    regions = {}
    for r in data:
        regions[r["Region"]] = regions.get(r["Region"], 0) + 1
    return regions
```

Run:

```
print("Avg Wheat Yield:", average_yield("Wheat"))
print("Farmers by Region:", count_by_region())
```

This introduces reusable **functions** and **counts**.

---

5. Task 4: Statistical Analysis

Use Python's statistics library:

```
import statistics as stats


yields = [r["Yield"] for r in data]
```

```
acres = [r["Acres"] for r in data]

print("Mean:", stats.mean(yields))
print("Median:", stats.median(yields))
print("Std Dev:", stats.stdev(yields))

# Simple correlation
correlation = sum((a - stats.mean(acres)) * (y - stats.mean(yields))
                  for a, y in zip(acres, yields)) / (len(acres) - 1)
print("Correlation (Acres vs Yield):", correlation)
```

Students learn **statistics** and correlation basics.

---

6. Task 5: Data Visualization

Use **Matplotlib & Pandas**:

```
import pandas as pd
import matplotlib.pyplot as plt

df = pd.DataFrame(data)

# Bar chart: Average yield per crop
df.groupby("Crop")["Yield"].mean().plot(kind="bar", title="Avg Yield per Crop")
plt.show()

# Pie chart: Farmers by region
df["Region"].value_counts().plot(kind="pie", autopct="%1.1f%%", title="Farmers by Region")
plt.show()
```

```
# Scatter plot: Acres vs Yield
plt.scatter(df["Acres"], df["Yield"])
plt.xlabel("Acres")
plt.ylabel("Yield")
plt.title("Acres vs Yield")
plt.show()

# Table
print(df)
```

Produces **charts and tables** for presentation.

---

7. Optional Mini-Project: Yield Prediction

Add simple formula:

```
avg_yield_per_acre = sum(df["Yield"]) / sum(df["Acres"])
df["Predicted_Yield"] = df["Acres"] * avg_yield_per_acre
print(df[["Farmer", "Crop", "Acres", "Predicted_Yield"]])
```

Introduces the **concept of prediction**, a step toward AI.

Checklist for Students

- Install Python & libraries
- Complete **Task 1 → 5** step by step
- Run all outputs & verify results
- Submit Python files or Jupyter Notebook

This guide is **practical, step-by-step, and ensures completion**.