Lesson Plan – Python Lesson 4 (1 Hour)

Objective

By the end of this lesson, students will:

- Install and manage Python packages using pip.
- Create and manipulate numerical data using NumPy arrays.
- Generate basic visualizations with matplotlib.
- Build simple GUI applications using tkinter.
- Build a project: a **Graphical Task Management App** with data visualization.

1. Package Management with pip (8 min)

Installing Packages

Explain that pip is Python's package installer that lets us add powerful tools to Python.

Demo Commands:

```
>>> # Check pip version
>>> python3 -m pip --version
>>>
>>> # See installed packages
>>> python3 -m pip list
>>>
>>> # Get info about a package
>>> python3 -m pip show numpy
>>>
>>> # Install packages
>>> python3 -m pip install numpy matplotlib
>>>
>>> # Uninstall a package
>>> python3 -m pip uninstall package_name
```

Version Specifiers: Briefly mention >=, <=, <, >, == for controlling package versions.

Quick Practice: Have students check their pip version and list installed packages.

2. NumPy Basics & Matrices (7 min)

What is a Matrix?

Think of it as a spreadsheet with rows and columns of data.

Demo:

```
>>> import numpy as np
>>>
>>> # Create a matrix
>>> matrix = np.array([[1, 2, 3],
>>>                    [4, 5, 6],
>>>                    [7, 8, 9]])
>>>
>>> # Access elements
>>> print(matrix[0][1])  # 2 (first row, second column)
>>>
>>> # Matrix operations
>>> print(matrix * 2)     # multiply each element by 2
```

Key Concept: Matrices let us work with lots of numbers at once efficiently.

3. Data Visualization with matplotlib (10 min)

Basic Plotting

matplotlib turns numbers into pictures (charts and graphs).

Demo:

```
>>> from matplotlib import pyplot as plt
>>>
>>> # Simple line plot
>>> plt.plot([1, 2, 3, 4, 5])
>>> plt.show()
```

Multi-line Plot Example

```
>>> import numpy as np

>>> days = np.arange(0, 21)
>>> other_site = np.arange(0, 21)
>>> real_python = other_site ** 2
```

```
>>> plt.plot(days, other_site, label="Other Site")
>>> plt.plot(days, real_python, label="Real Python")
>>> plt.legend()
>>> plt.show()
```

Different Chart Types

Show quick examples of:

- `plt.plot()` → line graphs
- `plt.bar()` → bar charts
- `plt.hist()` → histograms

Quick Practice: Have students plot a simple list of numbers.

4. GUI with tkinter (20 min)

What is a GUI?

Graphical User Interface - windows, buttons, and text boxes instead of just typing commands.

Basic Elements

```
>>> import tkinter as tk
>>>
>>> # Create main window
>>> window = tk.Tk()
>>> window.title("My First GUI")
>>>
>>> # Create a label (non-editable text)
>>> label = tk.Label(window, text="Hello!", fg="blue", bg="white")
>>> label.pack()
>>>
>>> # Create a button
>>> def on_click():
>>>     print("Button clicked!")
>>>
>>> button = tk.Button(window, text="Click Me", command=on_click)
>>> button.pack()
>>>
```

```
>>> # Create an entry (single-line input)
>>> entry = tk.Entry(window, width=20)
>>> entry.pack()
>>>
>>> # Run the app
>>> window.mainloop()
```

Key Components Explained:

- **tk.Tk()** → main window
- **tk.Label()** → displays text (can't edit)
- **tk.Button()** → clickable button
- **tk.Entry()** → single-line text input
- **tk.Text()** → multi-line text input
- **tk.Frame()** → container to group elements

Common Properties:

- `text` → what the element displays
- `fg` (foreground) → text color
- `bg` (background) → background color
- `width`, `height` → size in text units
- `command` → function to run when button clicked

Layout Managers:

- `.pack()` → stack elements vertically (default)
- `.pack(side=tk.LEFT)` → arrange horizontally
- `.place(x=10, y=20)` → exact positioning
- `.grid(row=0, column=1)` → spreadsheet-style layout

Getting Input:

```
>>> # Entry widget
>>> text = entry.get()       # get what user typed
>>> entry.delete(0, tk.END)  # clear the entry
>>> entry.insert(0, "text")  # put text in
```

```
>>>
>>> # Text widget
>>> text = text_box.get("1.0", tk.END)  # get all text
>>> text_box.delete("1.0", tk.END)      # clear all
>>> text_box.insert("1.0", "hello")     # insert at start
```

Events & Event Handlers:

- **Event** → an action (button click, key press)
- **Event Handler** → function that runs when event happens
- **Binding Events:**

```
>>> button.bind("<Button-1>", some_function())  # left mouse click
```

5. Mini-Project: GUI Task Manager with Graphs (15 min)

Build on the Task Logger from Lesson 1, but now with a graphical interface and charts!

Project Requirements:

1. GUI with entry fields for:
    - Task name
    - Difficulty (1-5)
    - Due date
2. Buttons to:
    - Add task
    - Mark task complete
    - Display graphs
3. Text display showing all tasks and their status
4. Graphs showing:
    - Number of tasks per difficulty
    - Number of tasks per status (pending/complete)

Stretch Challenges:

- Add a "Mark Complete" button that changes task status
- Create a bar chart showing pending vs completed tasks
- Add input validation for difficulty (must be 1-5)
- Use `.grid()` layout for better organization

6. Wrap-Up (5 min)

Recap:

- pip installs powerful packages
- NumPy handles numerical data efficiently
- matplotlib creates visualizations
- tkinter builds graphical interfaces
- Event handlers respond to user actions

Homework:

- Experiment with different matplotlib chart types
- Add more features to the Task Manager GUI
- Try different tkinter layout managers
- Research common tkinter events (Google "tkinter event types")

Teacher Guide / Aide

Timing Breakdown

- pip & Package Management → 8 min
- NumPy Basics → 7 min
- matplotlib Visualization → 10 min
- tkinter GUI Basics → 20 min
- Project → 15 min
- Wrap-up → 5 min

Teaching Tips

Engagement:

- Show visually impressive matplotlib examples first to hook interest
- Let students customize colors and button text in their GUIs
- Have faster learners help slower ones with GUI layout

Demonstrations:

- **pip**: Show live installation of a package
- **matplotlib**: Display several chart types side-by-side
- **tkinter**: Build a simple GUI live, step by step
- **Project**: Code along with students, troubleshoot together

Analogies:

- **pip** = App Store for Python
- **Matrix** = Spreadsheet of numbers
- **GUI** = The difference between DOS and Windows
- **Event Handler** = Motion sensor on automatic door

Hands-On:

- After each tkinter widget intro, have students add it to their own window
- Pause frequently for students to catch up with typing
- Encourage experimentation with colors, sizes, text

Stretch for Faster Learners:

- Implement complete task marking functionality
- Add multiple graph types (bar + histogram)
- Use `.grid()` layout instead of `.pack()`
- Save/load tasks to a file
- Add task editing and deletion

Common Pitfalls

pip Issues:

- Windows users might need `python` instead of `python3`
- Permission errors → may need admin rights or `--user` flag
- Students forget `-m` before pip

NumPy:

- Confusion between lists and arrays
- Off-by-one indexing errors

matplotlib:

- Forgetting `plt.show()` → no graph appears
- Not importing pyplot as plt

tkinter:

- Forgetting `window.mainloop()` → window closes immediately
- Button `command` with parentheses → `command=func()` instead of `command=func`
- Text widget indices ("1.0" vs 0) causing confusion
- Trying to use `return` in event handlers
- Not updating display after adding tasks

General:

- Import errors if packages not installed
- Mixing tabs and spaces causing IndentationError

Materials Needed

- Projector for live coding demos
- Pre-installed numpy, matplotlib on demo machine
- Backup: sample GUI screenshots in case live demo fails
- Handout with tkinter widget cheat sheet (optional)
- Example charts from matplotlib gallery to show possibilities

Pre-Lesson Preparation

- Test that pip, numpy, matplotlib, and tkinter work on all machines
- Have installation troubleshooting steps ready
- Prepare a finished version of the Task Manager GUI to show as inspiration
- Create a simple example GUI to demonstrate if time is tight

Troubleshooting Quick Reference

- **tkinter not found**: Usually included with Python, but some Linux systems need `python3-tk` package
- **matplotlib not showing plots**: Try `plt.show(block=True)` or check backend with `matplotlib.get_backend()`
- **Window not responding**: Make sure `mainloop()` is called and code isn't blocking the main thread