21-Day Python Challenge (Beginner → Intermediate), the next topics should naturally prepare you for real-world programming.

I recommend the following syllabus:

- **Day 17:** Packages, Pip, Virtual Environment (venv)
- **Day 18:** Mini Project – Student Management System (Revision of Days 1–17)

These topics bridge the gap between learning Python syntax and building practical applications.

🚀 21-Day Python Challenge

Day 17 – Python Packages, Pip & Virtual Environment

GK InfoTech Solutions – The Institute of Future Computer Knowledge Engineers

🎯 Learning Objectives

After completing today's lesson, students will be able to:

- Understand what a **Package** is.
- Understand what **PIP** is.
- Install Python packages.
- Import installed packages.
- Understand what a **Virtual Environment (venv)** is.
- Know why professional developers use virtual environments.

📓 What is a Package?

A **Package** is a collection of Python modules that provide extra functionality.

Think of your mobile phone.

Initially, it has only basic features.

When you install WhatsApp, YouTube, or Instagram, your phone gains new abilities.

Python works in the same way.

📷 Picture 1 – Mobile Apps vs Python Packages

Explanation

```
Mobile Phone

↓

Install Apps

↓

More Features

-------------------

Python

↓

Install Packages

↓

More Features
```

📓 What is PIP?

PIP stands for **Preferred Installer Program**.

It is the package manager for Python.

Using PIP, we can install thousands of ready-made libraries.

Examples:

- numpy
- pandas
- matplotlib
- requests
- flask

📷 Picture 2 – PIP Downloads Packages

```
Internet

↓

PIP

↓

Python Package

↓

Your Computer
```

Checking PIP

Open Command Prompt and type:

```
pip --version
```

Example Output

```
pip 24.x.x
```

Installing a Package

Example

```
pip install requests
```

Meaning

```
Internet

↓

Download

↓

Install

↓

Ready to Use
```

Using an Installed Package

```
import requests

print("Requests Package Imported Successfully")
```

Installing NumPy

```
pip install numpy
```

Example

```
import numpy as np

numbers = np.array([10,20,30])

print(numbers)
```

Output

```
[10 20 30]
```

📷 Picture 3 – Package Installation Process

What is a Virtual Environment?

Imagine two school students.

Rahul needs Python 3.12.

Ravi needs Python 3.10.

If both use the same computer, software conflicts may happen.

A **Virtual Environment** gives each project its own isolated Python environment.

📷 Picture 4 – Virtual Environment

Explanation

```
Computer
|
├── Project A
│       Python + Packages
├── Project B
│       Python + Packages
└── Project C
        Python + Packages
```

Each project is independent.

Creating a Virtual Environment

```
python -m venv myproject
```

Activate (Windows)

```
myproject\Scripts\activate
```

Deactivate

```
deactivate
```

Practice Program

```
import math

print(math.sqrt(144))
```

Mini Activity

Create a program that imports:

```
import math
import random
import datetime
```

Display

- Current Date
- Random Number
- Square Root of 100

Interview Questions

- What is a Package?
- What is PIP?
- What is Virtual Environment?
- Why do we use Virtual Environment?

🌟 Day 17 Success Message

Professional programmers don't write everything from scratch—they use packages to save time and virtual environments to keep projects organized.

🚀 21-Day Python Challenge

Day 18 – Mini Project: Student Management System

GK InfoTech Solutions – The Institute of Future Computer Knowledge Engineers

Today we will use everything learned from Days **1–17**.

📓 Project Goal

Build a simple **Student Management System**.

Students will learn:

- Variables
- Input
- Conditions
- Loops
- Functions
- Lists
- Dictionaries
- File Handling
- Exception Handling
- OOP concepts

📷 Picture 5 – Student Management System

Menu

```
============================
 STUDENT MANAGEMENT SYSTEM
============================

1. Add Student
2. View Students
3. Search Student
4. Exit
```

Complete Program

```
students = {}

def add_student():
    roll = input("Enter Roll Number : ")
    name = input("Enter Student Name : ")
    marks = int(input("Enter Marks : "))

    students[roll] = {
        "Name": name,
        "Marks": marks
    }

    print("Student Added Successfully")


def view_students():

    if len(students) == 0:
        print("No Student Records")
        return

    print("\nStudent Records")

    for roll, data in students.items():

        print("----------------------")
        print("Roll :", roll)
        print("Name :", data["Name"])
        print("Marks :", data["Marks"])
```

```
def search_student():

    roll = input("Enter Roll Number : ")

    if roll in students:

        print(students[roll])

    else:

        print("Student Not Found")

while True:

    print("\n===== STUDENT MANAGEMENT =====")
    print("1.Add Student")
    print("2.View Students")
    print("3.Search Student")
    print("4.Exit")

    choice = input("Enter Choice : ")

    if choice == "1":
        add_student()

    elif choice == "2":
        view_students()

    elif choice == "3":
        search_student()

    elif choice == "4":
        print("Thank You")
        break

    else:
        print("Invalid Choice")
```

📷 Picture 6 – Program Flow

```
Start

↓

Menu

↓

Add Student

↓

View Student

↓

Search Student

↓

Exit
```

What Students Have Used

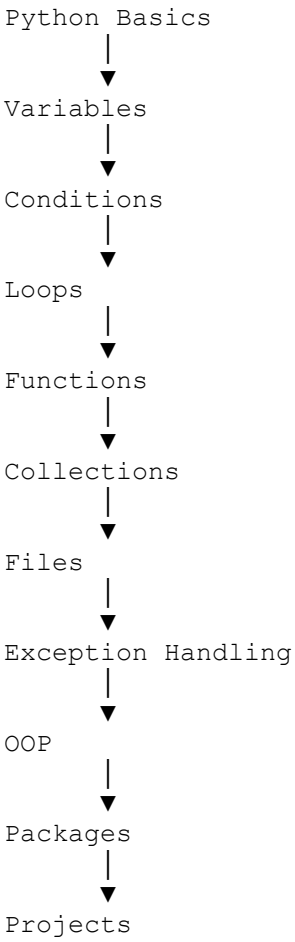
Concept	Used
Variables	✅
Data Types	✅
Conditions	✅
Loops	✅
Functions	✅
Dictionary	✅
Exception Handling	✅ (can be added for input validation)
OOP	Can be extended later
File Handling	Can be added to save records

Challenge Activity

Improve the project by adding:

- ✅ Delete Student
- ✅ Update Student
- ✅ Save Records to File
- ✅ Read Records from File
- ✅ Pass/Fail Result
- ✅ Total Students
- ✅ Highest Marks

📷 Picture 7 – Complete Python Learning Journey



🎓 Learning Outcome (Days 1–18)

By completing these 18 days, you can:

- Write Python programs confidently.
- Solve logical problems using loops and conditions.
- Store and process data using lists, tuples, sets, and dictionaries.
- Create reusable functions.
- Handle files and exceptions.
- Understand Object-Oriented Programming.
- Use Python packages and virtual environments.
- Build simple real-world applications.

"Programming is like learning to ride a bicycle. At first it feels difficult, but with daily practice, your hands, eyes, and brain work together naturally. Every program you write today becomes the foundation for the software you will build tomorrow."

This sequence provides a smooth transition from beginner concepts to practical, intermediate-level Python development before the final days focus on databases, GUI development, APIs, or capstone projects.