Let’s go step by step.

---

1. What is Python?

Python is a **high-level, interpreted language** known for its:

- Simplicity and readability (like writing plain English)
- Huge standard library
- Strong community support
- Versatility — used in:
  - Web development (Django, Flask)
  - Data science & ML (NumPy, Pandas, TensorFlow)
  - Scripting and automation
  - APIs and backend development
    -

---

2. Setting Up Your Python Environment

Step 1: Install Python

- Go to https://python.org/downloads
- Download the **latest version** (≥ 3.12)

- While installing:
  - Check the box **“Add Python to PATH”**
  - Then click *Install Now*

Step 2: Verify Installation

Open a terminal / command prompt:

```
python --version
```

You should see something like:

```
Python 3.12.1
```

Step 3: Install VS Code (recommended IDE)

Install Visual Studio Code

Add **Python Extension** from the Extensions Marketplace.

---

3. Writing Your First Python Program

Create a file: `hello.py`

```
print("Hello, World!")
```

Run it in terminal:

```
python hello.py
```

✅ Output:

```
Hello, World!
```

---

4. Core Python Concepts You Should Learn

We'll cover topics in this logical order:

Step	Topic	Example
1	Variables & Data Types	x = 10, name = "Jamal"
2	Operators	a + b, a == b
3	Conditional Statements	if, elif, else
4	Loops	for, while
5	Functions	def greet(name):
6	Lists, Tuples, Sets, Dictionaries	Data collections
7	String Operations	"Hello".upper()
8	Modules & Packages	import math
9	File Handling	open("file.txt", "r")

Step	Topic	Example
10	**Error Handling**	try...except
11	**Classes & OOP**	class Person:
12	**Virtual Environments**	venv for isolated projects
13	**Pip & Libraries**	pip install requests

---

5. Your .NET-to-Python Mindset Shift

Concept	.NET (C#)	Python Equivalent
Variable declaration	int x = 5;	x = 5
Function	void SayHello()	def say_hello():
Class	public class MyClass	class MyClass:
String interpolation	$"Hello {name}"	f"Hello {name}"
Array/List	List<int> numbers	numbers = [1, 2, 3]
Main function	static void Main()	if __name__ == "__main__":

Python is **dynamically typed**, so you don't need to declare variable types explicitly.

---

6. Learn by Doing: Mini Projects (in sequence)

Level	Project Idea	Concepts Covered
Beginner	Calculator CLI App	Functions, input/output
Beginner	To-Do List (text file)	File handling
Intermediate	Weather App using API	HTTP requests (requests library)
Intermediate	Basic Flask Web API	Routing, JSON, APIs
Advanced	Data analysis with Pandas	Data science basics
Advanced	Django-based website	Full-stack web app

---

7. Recommended Learning Resources

Interactive Tutorials

https://www.learnpython.org/

https://www.w3schools.com/python/

https://www.programiz.com/python-programming

YouTube Channels

freeCodeCamp.org — 4-hour Python beginner tutorial

Tech With Tim — clear explanations + mini projects

Programming with Mosh — practical coding approach

-

Books

Automate the Boring Stuff with Python by Al Sweigart

Python Crash Course by Eric Matthes

---

8. After Basics — Choose Your Path

Once you're comfortable with syntax and basics, pick a specialization:

Area	Library / Framework	Example
Web Development	Flask / Django	Create APIs or web apps
Automation	pyautogui, os	Automate file tasks
Data Science	pandas, numpy	Analyze CSV/Excel data
Machine Learning	scikit-learn, tensorflow	Predictive models
API Integration	requests, FastAPI	Connect with REST APIs

---

30 days plan for learning Python….

Python concepts to what you already know from **C#, ASP.NET, and React**.

This plan assumes **~1.5–2 hours/day** and focuses on **real backend usage (Flask/FastAPI)**, not just theory.

30-Day Python Roadmap

Beginner → Intermediate (Tailored for .NET + React Developers)

WEEK 1: Python Core (Syntax → Data Handling)

Day 1 - Python Setup & Mental Model

Goal: Feel at home in Python

Install:

Python 3.11+

VS Code + Python extension

Learn:

Python philosophy vs C#

Indentation instead of `{ }`

-

Practice:

-

```
print("Hello World")
name = "Jamal"
age = 30
print(f"My name is {name}, age {age}")
```

Mapping from C#:

C#	Python
`Console.WriteLine()`	`print()`
`string`	`str`
`{}` blocks	Indentation

Day 2 - Variables & Data Types

Goal: Understand Python's dynamic typing

Types:

`int`, `float`, `str`, `bool`

Type checking:

```
type(10)
isinstance(10, int)
```

Key Difference from C#:

No `int x = 10;`

Python decides type at runtime

Day 3 - Lists, Tuples, Sets, Dictionaries

Goal: Master Python collections

```
numbers = [1, 2, 3]
user = {"id": 1, "name": "Jamal"}
unique = {1, 2, 3}
coords = (10, 20)
```

Mapping from C#:

C#	Python
List<T>	list
Dictionary<K,V>	dict
HashSet<T>	set
Tuple<>	tuple

Day 4 - Conditions & Loops

Goal: Flow control

```
if age > 18:
    print("Adult")
else:
    print("Minor")
```

```
for i in range(5):
    print(i)
```

React mindset: Similar to `.map()` loops but backend-oriented.

Day 5 - Functions

Goal: Reusable logic

```
def add(a, b=0):
    return a + b
```

Default parameters

Named arguments:

```
add(b=5, a=10)
```

Day 6 - Modules & Virtual Environments

Goal: Project structure

Create venv:

```
python -m venv venv
source venv/bin/activate
```

Install package:

```
pip install requests
```

Similar to **NuGet + project references**

Day 7 - Mini Project

Build:

CLI User Manager

Add users

List users

Delete users

Store in dictionary

WEEK 2: Object-Oriented Python (C# Friendly Zone)

Day 8 - Classes & Objects

```
class User:
    def __init__(self, name):
        self.name = name
```

Mapping from C#:

C#	Python
Constructor	`__init__`
this	self

Day 9 - Inheritance & Polymorphism

```
class Admin(User):
    def delete_user(self):
        pass
```

Day 10 - Encapsulation & Properties

```
class Account:
    def __init__(self):
        self._balance = 0

    @property
    def balance(self):
        return self._balance
```

Day 11 - Exceptions

```
try:
    x = int("abc")
except ValueError:
    print("Invalid number")
```

Equivalent to `try/catch` in C#.

Day 12 - File Handling

```
with open("data.txt", "w") as f:
```

```
    f.write("Hello")
```

Day 13 - JSON & APIs

```
import json

json.dumps({"id": 1})
json.loads('{"id":1}')
```

Perfect for **React ↔ Python API** work.

Day 14 - Mini Project

Build:
✅ **File-based User CRUD**

Save users in JSON

Read & update users

WEEK 3: Web Backend (Flask → FastAPI)

Day 15 - Flask Basics

```
from flask import Flask

app = Flask(__name__)

@app.route("/")
def home():
    return "Hello API"
```

Think **ASP.NET Web API (minimal)**.

Day 16 - REST APIs (CRUD)

```
@app.route("/users", methods=["POST"])
def create_user():
    return {"success": True}
```

Day 17 - Request, Response, JSON

```
from flask import request
```

```
data = request.json
```

Same as `Request.Body` in ASP.NET.

Day 18 - CORS & Auth Basics

`flask-cors`

JWT overview

You've already used this in your React + Flask work.

Day 19 - Database (MySQL)

```
import mysql.connector
```

Connect MySQL

Execute queries

Day 20 - ORM with SQLAlchemy

```
class User(db.Model):
    id = db.Column(db.Integer, primary_key=True)
```

Think **Entity Framework Core**.

Day 21 - Mini Project

Build:
✅ **User Management API**

Login

JWT auth

MySQL DB

Connect to React frontend

WEEK 4: Intermediate Python (Production Skills)

Day 22 - FastAPI (Upgrade Flask)

```
from fastapi import FastAPI
app = FastAPI()
```

Faster, typed, Swagger built-in.

Day 23 - Pydantic & Type Hints

```
from pydantic import BaseModel

class User(BaseModel):
    name: str
    age: int
```

Closest Python gets to **C# strong typing**.

Day 24 - Async & Await

```
async def get_data():
    return "data"
```

Like `async/await` in C#.

Day 25 - Background Tasks

Email sending

Jobs (you already used Airflow concepts)

Day 26 - Logging & Config

```
import logging
```

Equivalent to `ILogger`.

Day 27 - Testing

```
def test_add():
    assert add(2,3) == 5
```

Using `pytest`.

Day 28 - Packaging & Deployment

`requirements.txt`

`gunicorn`

Linux deployment

Day 29 - Final Project

Build:
Production-ready API

FastAPI

JWT

MySQL

Email service

React-ready

Day 30 - Review & Next Steps

Refactor code

Add caching (Redis)

Dockerize app
