Becoming a core Python developer without attending college or an institute is absolutely possible! Many successful developers are self-taught. Here's a step-by-step guide to help you achieve your goal:

1. Understand the Role of a Core Python Developer

A core Python developer works on:

- Writing efficient, scalable, and maintainable Python code.
- Contributing to the Python language itself (e.g., CPython development).
- Building libraries, frameworks, or tools for the Python ecosystem.
- Solving complex problems using Python.

2. Learn Python Fundamentals

Start with the basics and build a strong foundation:

- **Free Resources**:
  - Python.org Official Tutorial
  - Automate the Boring Stuff with Python
  - Real Python Tutorials
- **Key Topics**:
  - Syntax, data types, variables, and operators.
  - Control flow (if-else, loops).
  - Functions, modules, and packages.
  - File handling and exception handling.
  - Object-Oriented Programming (OOP) in Python.

3. Master Advanced Python Concepts

Once you're comfortable with the basics, dive deeper:

- **Topics to Learn**:
  - Decorators, generators, and context managers.
  - Python's memory model and garbage collection.
  - Concurrency and parallelism (threading, multiprocessing, asyncio).
  - Metaprogramming (e.g., `__dunder__` methods).
  - Python's data model and how Python works under the hood.
- **Resources**:
  - Fluent Python by Luciano Ramalho
  - Effective Python by Brett Slatkin

4. Contribute to Open Source

Contributing to open-source projects is a great way to gain real-world experience:

- Start with small contributions (e.g., bug fixes, documentation).
- Explore Python's official repositories on GitHub.
- Look for beginner-friendly issues labeled "good first issue."
- Learn Git and GitHub workflows.

5. Build Projects

Apply your knowledge by building projects:

- **Beginner Projects**:
  - A to-do list app.
  - A web scraper.
  - A simple calculator.
- **Intermediate Projects**:
  - A blog platform using Flask/Django.
  - A data visualization tool using Matplotlib/Seaborn.
  - A chatbot using NLP libraries.
- **Advanced Projects**:
  - A custom Python package or library.
  - A machine learning model using Scikit-learn or TensorFlow.
  - A contribution to CPython or a popular Python library.

6. Learn Python Internals

To become a core developer, you need to understand how Python works internally:

- **Topics to Study**:
  - Python's bytecode and the Python Virtual Machine (PVM).
  - CPython's source code and C API.
  - Python's garbage collection mechanism.
  - Python's Global Interpreter Lock (GIL).
- **Resources**:
  - CPython Internals Book
  - Python Developer's Guide

7. Practice Problem-Solving

Develop strong problem-solving skills:

- Solve problems on platforms like:
  - LeetCode
  - HackerRank
  - Codewars
- Participate in coding competitions (e.g., Google Code Jam, Codeforces).

8. Network with the Python Community

Engage with the Python community to learn and grow:

- Join forums like Reddit's r/Python.
- Attend Python meetups and conferences (e.g., PyCon).
- Follow Python experts on Twitter and LinkedIn.
- Contribute to discussions on Python Discord.

9. Build a Portfolio

Showcase your skills to potential employers or collaborators:

- Create a GitHub profile and upload your projects.
- Write blog posts or tutorials about Python.
- Share your contributions to open-source projects.

10. Apply for Jobs or Freelance Work

Once you're confident in your skills, start applying:

- Look for Python developer roles on job boards like LinkedIn, Indeed, or AngelList.
- Apply for freelance gigs on platforms like Upwork or Fiverr.
- Highlight your projects, open-source contributions, and problem-solving skills.

11. Keep Learning

Python is constantly evolving, so stay updated:

- Follow Python Enhancement Proposals (PEPs).
- Read the Python documentation.
- Explore new libraries and frameworks.

Sample Learning Roadmap

1. **Month 1-2**: Learn Python basics and build small projects.
2. **Month 3-4**: Master advanced Python concepts and start contributing to open source.
3. **Month 5-6**: Dive into Python internals and build advanced projects.
4. **Month 7-8**: Focus on problem-solving and networking.
5. **Month 9-12**: Build a portfolio and start applying for jobs.

With dedication, consistent practice, and active participation in the Python community, you can become a core Python developer without formal education. Good luck!