# *Advanced Python Programming*

Workshop 2 - Instructor: Tanya Khanna - 09/16/2024

# Workshop Details

**Email**: tk759@scarletmail.rutgers.edu

**Workshop Materials:**

- **Github Link:** https://github.com/Tanya-Khanna/DataScienceWorkshop_Fall-2024_NBL
- Spring 2024 Workshops: Link

# Workshop Schedule

Fall 2024 workshops calender

| | |
|---|---|
| **Introduction to Python Programming** | **September 9, 2024; 4 - 5:30 PM** |
| **Advanced Python Programming** | **September 16, 2024; 4 - 5:30 PM** |
| **Web Scraping with Python** | **September 23, 2024; 4 - 5:30 PM** |
| **Mastering Data Analysis: Pandas and Numpy** | **September 30, 2024; 4 - 5:30 PM** |
| **Data Management with Python: SQL and NoSQL** | **October 7, 2024; 4 - 5:30 PM** |
| **Python for Visualization and Exploration** | **October 14, 2024; 4 - 5:30 PM** |
| **Introduction to Machine Learning: Supervised Learning** | **October 21, 2024; 4 - 5:30 PM** |
| **Introduction to Machine Learning: Unsupervised Learning** | **October 28, 2024; 4 - 5:30 PM** |
| **Deploying Machine Learning Models** | **November 4, 2024; 4 - 5:30 PM** |
| **Ethical AI and Responsible Data Science** | **November 11, 2024; 4 - 5:30 PM** |

# Table of Contents

# Introduction

**This workshop is designed to help you tackle common challenges when writing Python code. We'll focus on a few important areas to make you more efficient and confident as a programmer:**

1. **Debugging and Error Handling:**
   - Spot and fix mistakes in your code, whether it's a typo or a logic error.
   - Learn about the different kinds of errors you might run into (like a variable not being found or trying to use the wrong type of data).
   - Use techniques to handle these errors in a way that prevents your programs from crashing and keeps them running smoothly.
2. **Object-Oriented Programming (OOP):**
   - Organize your code using classes, which are like blueprints for objects (think of an object as a real-world item, like a "Car," and a class as the blueprint for making different cars).
   - Understand key ideas like:
     - ***Encapsulation***: Keeping related data and functions together in one place.
     - ***Inheritance***: Making new objects that build on existing ones.
     - ***Polymorphism***: Writing code that can work with different types of objects.
   - These techniques help you write code that is easier to manage, reuse, and extend for larger projects.

**3. Functional Programming (FP) in Python:**

- Use functional programming principles, which focus on writing clear and predictable code by avoiding changes to data and focusing on the functions that process it.
- Understand key ideas like:
  - ***Immutability*:** Keeping data unchanged.
  - ***First-class functions*:** Treating functions like any other piece of data, meaning you can pass them around and use them in your programs.
  - ***Pure functions*:** Functions that always give the same result if you give them the same input, without changing anything else.
- Learn how to use Python's built-in tools like lambda, map(), filter(), and reduce() to write shorter, more efficient code that processes data.

**4. Data Structures:**

Finally, we'll do a quick look at some important data structures in Python, such as:

- Stacks and Queues: Useful for organizing tasks or items in an "in-out" manner. (like stacking plates or lining up people for a ticket).

# Importance of Error Handling

Programming always involves mistakes, especially as your projects get bigger. That's why learning how to debug and handle errors is so important:

- ***Find and fix bugs:*** Debugging helps you track down exactly where things are going wrong in your code, so you can fix them quickly.
- ***Keep your programs stable:*** By using error-handling tools like try-except blocks, you can stop your programs from crashing when something unexpected happens.
- ***Prevent bigger problems:*** Good error handling makes sure that one small issue in your code doesn't turn into a complete failure, allowing your program to keep running smoothly for users.

In short, knowing how to debug and handle errors makes your code more reliable, saves time, and reduces the frustration of tracking down issues. It helps keep your software running without interruptions.

# Importance of Object Oriented Programming

Object-Oriented Programming helps you organize your code better. It's essential for:

- ***Breaking code into reusable parts:*** With OOP, you can create "objects" that store related data and functionality together, which makes your code easier to understand and reuse in other projects.
- ***Managing complexity:*** Instead of one giant piece of code, you can use classes and objects to break down complex tasks into smaller, manageable pieces.
- ***Making your code flexible:*** OOP lets you use features like inheritance (where new objects can build on existing ones) and polymorphism (writing code that works with different types of objects), making it easier to extend and update your programs without starting from scratch.

Mastering OOP means you'll be able to write cleaner, more organized, and easier-to-maintain code. This is especially helpful when working on big projects or with a team.

# Importance of Functional Programming

Functional programming is another way to think about coding, focusing on using functions to build programs. It's useful because:

- ***Shorter, clearer code:*** Functional programming encourages writing smaller, more readable blocks of code that are easier to follow and reuse.
- ***Fewer surprises in your code:*** FP promotes immutability—keeping data unchanged, which helps avoid unexpected side effects in your program.
- ***Efficiently processing data:*** FP tools like map, filter, and reduce allow you to work with data (like lists or dictionaries) more effectively, often in fewer lines of code.

Learning functional programming helps you write faster, less buggy, and easier-to-test code. It pushes you to focus on what your program needs to do, rather than worrying about how to do it step-by-step, making the code simpler and more elegant.