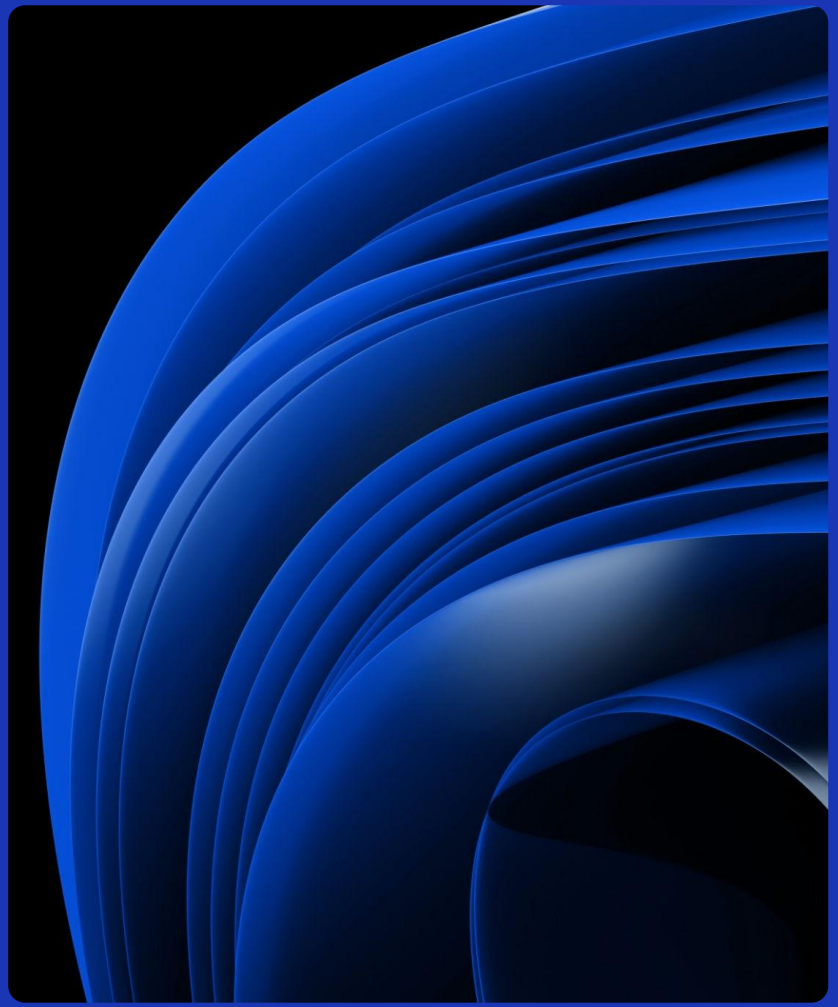
A My Path MN and BDPATCF Collaboration

Intro to Python

Led by William Munnich
Friday 2/22/2025



Class Timeline and Topics | Part A

Level A

- **2/22/25**
  - AI as a Learning Tool, Syntax Cheat Sheet, GitHub Setup and Basics
  - **PROJECT 1:** Portfolio Site (Hosted on GitHub)
  - First Push to Repository From Web and VSCode
- **3/1/25**
  - Variables, Data Types, For Loops, While Loops, If Statements, Libraries
- **3/8/25**
  - Types of Programming Languages, Python Programming Styles, Versatility of Python
- **3/15/25**
  - Python Problem Solving, How to Debug, Stack Overflow, Reading Source Documentation
  - The Power of Pseudocode
- **3/22/25**
  - Functions, Programs, File Types, Referencing Functions in a Different File, Databases

Class Timeline and Topics | Part B

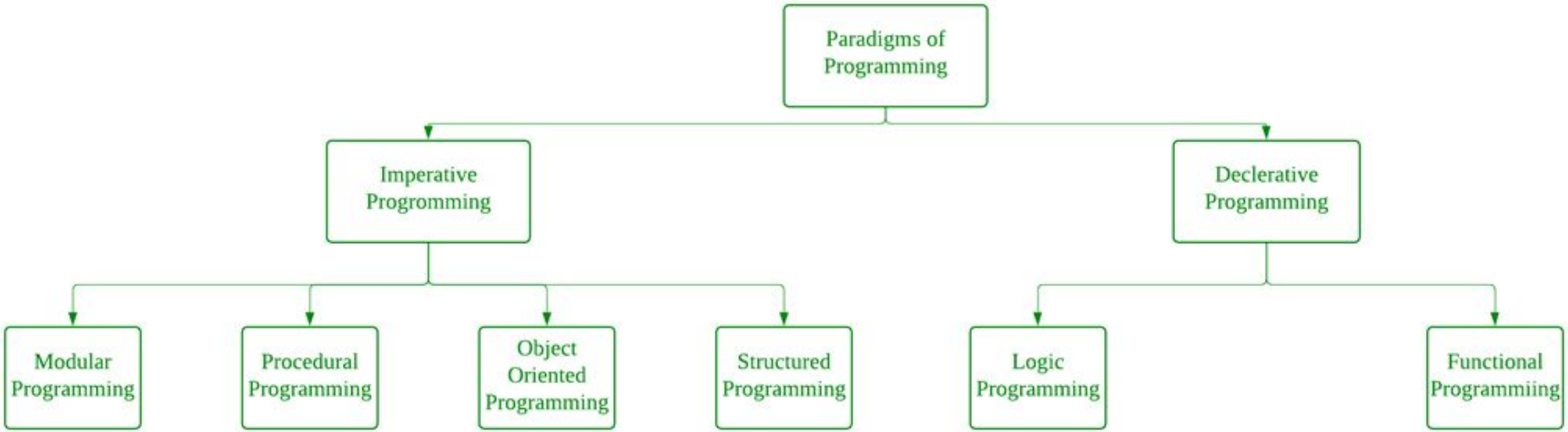
Level B

- **3/29/25**
  - **PROJECT 2:** Web Scraping → Saving To a File → Changing File Type → Deleting Files
- **4/5/25**
  - **PROJECT 3:** Local Web-Based Media Server With Supabase Local (Starting)
- **4/12/25**
  - **PROJECT 3:** Local Web-Based Media Server With Supabase Local (Wrapping Up)

Programming Paradigms

Programming paradigms are different ways to classify programming languages based on their features and how they work.

- **Imperative:** Focuses on describing how a program operates step-by-step.
  - **Object-Oriented:** Centers around creating objects that contain data and code.
- **Declarative:** Focuses on what the program should accomplish without specifying the exact steps.
  - **Functional:** Treats computation as mathematical functions and avoids changing state and mutable data.



Imperative Programming

Examples: C, C++, Java, BASIC, ASSEMBLY, Javascript

Specifies what to do and how to get it.

- Running Entire Operating Systems (like Windows or macOS)
- Launching Rockets into Space
- Powering Self-Driving Cars
- Building Massive Video Games (like Fortnite, Roblox, Minecraft)
- Managing Global Bank Transactions
- Controlling Factory Robots
- Running Air Traffic Control Systems
- Creating Movie Special Effects

How to Make a Peanut Butter and Jelly Sandwich

Step 1: Take 2 slices of bread
Step 2: Pick up a knife
Step 3: Open the peanut butter jar
Step 4: Scoop peanut butter with the knife
Step 5: Spread peanut butter on 1 slice of bread
Step 6: Clean the knife
Step 7: Open the jelly jar
Step 8: Scoop jelly with the knife
Step 9: Spread jelly on the other slice of bread
Step 10: Put the 2 slices together
Step 11: Enjoy your PB&J sandwich!

Declarative Programming

Examples: HTML, CSS, SQL

Specifies what you want, not how to get it.

-Structuring Websites (like Google or YouTube)
-Styling Webpages Globally (like Amazon’s look and feel)
-Managing Massive Databases (like school records or -Netflix recommendations)
-Powering Search Engines (like Google’s search results)
-Solving Complex Puzzles (like scheduling school classes)
Extracting Data from Files (like finding specific info in a library catalog)

Specifications for a Peanut Butter and Jelly Sandwich

I want a PB&J sandwich:
- 2 slices of bread
- Peanut butter on one slice
- Jelly on the other slice
- Put them together

Modular Programming

Examples: Python, Java, C++, Ruby, JavaScript

Specifies breaking code into reusable pieces.

- Building Large Websites (like Facebook or Twitter)
- Developing Complex Apps (like Snapchat or TikTok)
- Creating Video Game Levels (like in Minecraft or Zelda)
- Managing School Management Systems
- Designing Car Software Systems
- Building Online Shopping Platforms (like Amazon)
- Running Scientific Simulations
- Developing Medical Device Software

```
Module getBreadAndKnife():
  Step 1: Take 2 slices of bread
  Step 2: Pick up a knife

Module spreadPeanutButter():
  Step 3: Open the peanut butter jar
  Step 4: Scoop peanut butter with the knife
  Step 5: Spread peanut butter on 1 slice of bread

Module spreadJelly():
  Step 6: Clean the knife
  Step 7: Open the jelly jar
  Step 8: Scoop jelly with the knife
  Step 9: Spread jelly on the other slice of bread

Module assembleSandwich():
  Step 10: Put the 2 slices together
  Step 11: Enjoy your PB&J sandwich!
```

Run the Sandwich-Making Program:

```
getBreadAndKnife()
spreadPeanutButter()
spreadJelly()
assembleSandwich()
```

Procedural Programming

Examples: C, COBOL, Python, JavaScript

Specifies organizing code into step-by-step procedures.

- Running Computer Programs (like Microsoft Office)
- Simulating Weather Forecasts
- Operating Spacecraft Navigation
- Developing Mobile Games (like Angry Birds)
- Processing Online Payments
- Controlling Industrial Machinery
- Managing Hospital Patient Records
- Generating Animated Films

How to Make Any Sandwich (Using Reusable Modules)

```
Module getBreadAndKnife():
  Step 1: Take 2 slices of bread
  Step 2: Pick up a knife

Module spreadTopping(topping, slice):
  Step 3: Open the topping jar
  Step 4: Scoop the topping with the knife
  Step 5: Spread the topping on the slice of bread
```

```
Run the Sandwich-Making Program (for a PB&J Sandwich):
  getBreadAndKnife()
  spreadTopping("peanut butter", "slice 1")
  cleanKnife()
  spreadTopping("jelly", "slice 2")
  assembleSandwich()
```

```
Module getBreadAndKnife():
  Step 1: Take 2 slices of bread
  Step 2: Pick up a knife

Module spreadTopping(topping, slice):
  Step 3: Open the topping jar
  Step 4: Scoop the topping with the knife
  Step 5: Spread the topping on the slice of bread
```

```
Run the Sandwich-Making Program (for a Ham and Cheese Sandwich):
  getBreadAndKnife()
  spreadTopping("mayo", "slice 1")
  cleanKnife()
  spreadTopping("mustard", "slice 2")
  assembleSandwich()
  (Add ham and cheese slices before assembling!)
```

Object-Oriented Programming

Object-Oriented Programming

Examples: Python, Java, C++, C#, Swift, JavaScript (ES6+), Ruby

Encapsulates data and behavior into objects. Focuses on **reusability, modularity, and abstraction**.

Used in:

- Developing Mobile & Web Applications
- Game Development (Unity, Unreal Engine)
- Designing Graphical User Interfaces (GUIs)
- Large-Scale Software Systems
- Financial Modeling & Simulations
- AI & Machine Learning Applications
- Automating Business Processes

How to Make a Peanut Butter and Jelly Sandwich (OOP Style)

1. **Create a `Bread` class**
2. **Create a `Knife` class**
3. **Create a `PeanutButterJar` and `JellyJar` class**
4. **Define methods for `open()`, `scoop()`, and `spread()`**
5. **Instantiate objects: `bread1`, `bread2`, `knife`, `pb_jar`, `jelly_jar`**
6. **Call methods on objects** (e.g., `knife.scoop(pb_jar)`, `knife.spread(bread1)`)
7. **Assemble sandwich with `Sandwich.combine(bread1, bread2)`**
8. **Enjoy your PB&J sandwich!**

Structured Programming

Examples: C, Python, Java, C++, JavaScript

Breaks down a program into small, manageable **functions and procedures** that follow a clear, linear flow with **sequence, selection (decision-making), and iteration (loops).**

Used in:

- Embedded Systems & Firmware Development
- Business Software & Enterprise Applications
- Web Development & APIs
- Database Management Systems
- Scientific Computing & Simulations
- Automation & Scripting

How to Make a Peanut Butter and Jelly Sandwich (Structured Style)

1. **Define functions:**
   - `take_bread()`
   - `open_jar(jar_type)`
   - `scoop_ingredient(ingredient)`
   - `spread_ingredient(ingredient, bread_slice)`
   - `assemble_sandwich()`

python

```
take_bread()
open_jar("peanut butter")
scoop_ingredient("peanut butter")
spread_ingredient("peanut butter", "bread1")
open_jar("jelly")
scoop_ingredient("jelly")
spread_ingredient("jelly", "bread2")
assemble_sandwich()
```

Paradigm	Focus	Code Organization	Example Analogy
Structured	Logical Flow	Step-by-step, no `GOTO`	Following a recipe
Procedural	Function-Based	Functions for tasks	A restaurant kitchen
Modular	Reusability	Independent, reusable modules	A fast-food franchise

Logic Programming

Examples:

- **Prolog** – A logic-based language using facts, rules, and queries for AI and expert systems.
- **Datalog** – A Prolog subset optimized for database queries, graph analysis, and security policies.
- **Answer Set Programming (ASP)** – A constraint-solving paradigm for AI planning and optimization problems.

Definition: A programming paradigm based on **formal logic**. Instead of defining explicit steps (like in imperative programming), it **declares facts and rules** and lets the system infer the solution through logical deduction.

Used in:

- Artificial Intelligence & Expert Systems
- Natural Language Processing (NLP)
- Automated Theorem Proving
- Knowledge-Based Systems
- Constraint Solving (e.g., scheduling, Sudoku solving)

Making a PB&J Sandwich with Logical Thinking

Step 1: Tell the Computer What We Have

- We have **bread, peanut butter, jelly, and a knife**
- We can **spread things with a knife**

Step 2: Tell the Computer the Rules

- **Peanut butter can go on bread**
- **Jelly can go on bread**
- **If two things can go on bread, we can make a sandwich**

Step 3: Ask the Computer a Question

💡 *"Can I make a peanut butter and jelly sandwich?"*

Step 4: Let the Computer Think...

✅ **Yes! You can make a PB&J sandwich!** 🎉

Functional Programming

Examples: Haskell, Lisp, Clojure, Scala, F#, JavaScript (ES6+ with functional features), Python (functional programming style)

Uses pure functions and avoids changing variables or state. Instead of giving step-by-step instructions, functions transform data and return results.

Used in:

- Data Analysis & Big Data Processing
- Machine Learning & AI
- Parallel & Concurrent Computing
- Financial Systems & Risk Analysis
- Functional Web Development (e.g., React.js)
- Cryptography & Blockchain

```python
# Pure functions
def spread(ingredient, bread):
    return f"{bread} with {ingredient}"

def make_sandwich(bread1, bread2):
    return f"Sandwich made with {bread1} and {bread2}"

# Directly print the result without storing variables
print(make_sandwich(spread("peanut butter", "bread"), spread("jelly", "bread")))
```

Where Do HTML and CSS Fit in?

- **HTML (HyperText Markup Language) → Not a Programming Language**
  - HTML is a **markup language**, not a programming paradigm.
  - It is used to **structure content** on web pages.
  - If we had a category for **Markup Languages**, HTML would fit there.
- **CSS (Cascading Style Sheets) → Declarative Programming**
  - CSS **describes how things should look** without specifying step-by-step instructions.
  - It is similar to **Logic Programming**, as it relies on **rules** and **selectors** rather than sequential steps.

Python Programming Paradigms

Paradigm	Python Supports?	Notes
Imperative Programming	✅ Yes	Python allows step-by-step instructions using statements like loops and conditionals.
Procedural Programming	✅ Yes	Python supports writing procedures (functions) for structured programming.
Object-Oriented Programming (OOP)	✅ Yes	Python supports classes, objects, inheritance, and encapsulation.
Functional Programming	✅ Yes	Python supports first-class functions, higher-order functions, `map`, `reduce`, and `lambda`.
Modular Programming	✅ Yes	Python encourages code organization into reusable modules with `import`.
Declarative Programming	⚠️ Partially	Python supports some declarative approaches, like database queries (SQLAlchemy) and decorators.
Logic Programming	❌ No	Python does not have built-in logic programming like Prolog, but libraries (e.g., PySWIP) enable it.
Markup Language (Programming)	❌ No	Python is not a markup language like HTML or XML; it is a general-purpose programming language.