Introduction to Python Programming

Chapter 1: Getting Started with Programming

Welcome!

This guide is designed to help you reflect on what we've covered so far and reinforce the key concepts. Programming is a skill that develops over time, so don't worry if things don't click immediately - that's completely normal!

---

1. What is Computer Programming?

Programming is about problem-solving. At its core, programming is the process of:

1. **Recognizing a problem** that needs to be solved
2. **Identifying the steps** (an algorithm) a person could follow to solve it
3. **Translating those steps** into a programming language the computer understands

Understanding Algorithms

An **algorithm** is simply a series of steps to solve a problem. Think of it like a recipe - you follow specific steps in order to achieve a result.

Here's the key insight: Coming up with *any* algorithm is one challenge. Coming up with an algorithm that *works with Python* is another. This takes practice and experience!

Important points to remember:

- There are usually **many different algorithms** that can solve the same problem
- Some algorithms work better with certain programming languages than others
- **The hardest part of programming is thinking through the problem** and breaking it down into steps
- Once you can think algorithmically, you can adapt your thinking to *any* programming language

From Problem to Algorithm: Examples

Let's look at some examples to understand how we break down problems into algorithms.

Example 1: The Game of Nim

In the game of Nim, we start with 12 marbles. Two players take turns choosing 1, 2, or 3 marbles. The player who takes the last marble wins.

Breaking down the problem:

```
ALGORITHM: Play Nim
1. Start with 12 marbles
2. REPEAT until no marbles remain:
   a. Show current number of marbles
   b. Ask current player: "How many marbles do you want (1, 2, or 3)?"
   c. Check if choice is valid (1-3 AND not more than remaining)
      - If invalid, ask again
      - If valid, remove that many marbles
   d. Switch to the other player
3. Announce the winner (whoever took the last marble)
```

Notice: This algorithm uses concepts we can implement in Python:

- Variables (to store marble count, player turn)
- Loops (REPEAT until...)
- Conditionals (IF choice is valid...)
- Input/Output (asking questions, showing results)

Example 2: Tower of Hanoi

The Tower of Hanoi puzzle has three pegs and several disks of different sizes. All disks start on one peg, and the goal is to move them all to another peg following these rules:

- Move only one disk at a time
- Never place a larger disk on a smaller disk

Breaking down the problem (for 3 disks):

```
ALGORITHM: Solve Tower of Hanoi (3 disks)
1. Move small disk from Start to Destination
2. Move medium disk from Start to Spare
3. Move small disk from Destination to Spare
4. Move large disk from Start to Destination
5. Move small disk from Spare to Start
6. Move medium disk from Spare to Destination
7. Move small disk from Start to Destination
```

For a more general solution with any number of disks, we'd need recursion (we'll learn this later):

```
ALGORITHM: Move N disks from Start to Destination (using Spare)
IF only 1 disk:
   - Move it from Start to Destination
ELSE:
   - Move (N-1) disks from Start to Spare (using Destination as spare)
   - Move the largest disk from Start to Destination
   - Move (N-1) disks from Spare to Destination (using Start as spare)
```

Key Takeaway: Notice how we can describe the solution in plain language first, then think about how to express it in code. This is the essence of algorithmic thinking!

---

2. Introduction to Python

What Kind of Language is Python?

Python is an **interpreted programming language**. But what does that mean?

Interpreted vs. Compiled Languages:

The key difference is **when your code gets converted to machine code** (the 1s and 0s the processor understands):

- **Compiled languages** (like C++ or Java): Your code is converted to machine code *before* you run the program. This is a separate step called "compiling."
- **Interpreted languages** (like Python): Your code is converted to machine code *while* the program is running, line by line.

Pros of Interpreted Languages (Python):

- Easier to write and test code
- Make changes and see results immediately
- More flexible and forgiving for beginners

Cons of Interpreted Languages:

- Generally slower than compiled programs
- Need the interpreter installed to run your code

For learning programming, Python's advantages far outweigh the disadvantages!

---

3. Setting Up Your Python Environment

Installing Python

Python needs to be installed on your computer before you can run Python programs.

For Windows:

1. Go to python.org
2. Download the latest Python 3.x version
3. Run the installer
4. **IMPORTANT:** Check "Add Python to PATH" during installation
5. Click "Install Now"

For macOS:

1. Go to python.org
2. Download the latest Python 3.x version for macOS
3. Open the downloaded .pkg file
4. Follow the installation wizard

Verify Installation:
Open Command Prompt (Windows) or Terminal (macOS) and type:

```
python --version
```

You should see something like "Python 3.12.x"

Visual Studio Code (VS Code)

Important Distinction: VS Code is NOT Python. VS Code is NOT a programming language.

VS Code is a code editor - a tool for writing code. Think of it like Microsoft Word, but for programmers. Python is one of *many* programming languages you can use with VS Code.

Installing VS Code:

1. Go to code.visualstudio.com
2. Download for your operating system
3. Install and open VS Code

Setting Up Python in VS Code:

1. Open VS Code
2. Go to Extensions (Ctrl+Shift+X on Windows, Cmd+Shift+X on Mac)
3. Search for "Python" by Microsoft
4. Click "Install"

How to Know VS Code Recognizes Python:

- Look at the **bottom-right corner** of VS Code

- You should see the Python version (e.g., "Python 3.12.x")
- This shows which Python interpreter VS Code is using
- Click it to change interpreters if you have multiple Python versions

Other Options:
VS Code is our choice for this course, but you should know there are alternatives:

- **PyCharm** - Popular Python-specific IDE
- **Sublime Text** - Lightweight editor
- **IDLE** - Comes with Python installation
- **Jupyter Notebooks** - Great for data science

For this course, please stick with VS Code so we're all using the same tools.

---

4. Organizing Your Course Files

A good folder structure is *essential* for staying organized throughout this course.

Recommended Structure:

```
PythonCourse/
├── Week01/
├── Week02/
├── Week03/
├── Week04/
├── Labs/
│   ├── Lab01/
│   ├── Lab02/
│   └── Lab03/
├── Assignments/
│   ├── Assignment01/
│   └── Assignment02/
└── Practice/
```

Creating Your Folder:

Windows:

1. Open File Explorer
2. Navigate to Documents (or wherever you want to store course files)
3. Right-click → New → Folder
4. Name it "PythonCourse"
5. Inside, create subfolders for Labs, Assignments, etc.

macOS:

1. Open Finder
2. Navigate to Documents (or your preferred location)
3. File → New Folder
4. Name it "PythonCourse"
5. Inside, create subfolders for Labs, Assignments, etc.

Opening Your Project in VS Code

The key to success: Always open the *folder* in VS Code, not individual files.

Method 1 - From VS Code:

1. Open VS Code
2. File → Open Folder
3. Navigate to your PythonCourse folder
4. Click "Select Folder"

Method 2 - From File Explorer/Finder:

1. Right-click on your PythonCourse folder
2. Select "Open with Code" (if available)

Method 3 - From Command Line/Terminal:

1. Open Command Prompt (Windows) or Terminal (macOS)
2. Navigate to your folder:

   ```
   cd Documents/PythonCourse
   ```

3. Type: `code .` (the dot means "current folder")

Understanding File Paths

Knowing where your files are saved is crucial!

Windows Path Example:

```
C:\Users\YourName\Documents\PythonCourse\Labs\Lab01\program.py
```

macOS Path Example:

```
/Users/YourName/Documents/PythonCourse/Labs/Lab01/program.py
```

In VS Code:

- The **Explorer panel** (left side) shows your folder structure
- The **file tab** at the top shows the current file name
- The **status bar** at bottom shows the full file path (hover over Python version)

Quick Check: If you can see your project folder structure in VS Code's Explorer panel, you're in the right place!

Reflection Questions

Before moving on, take a moment to reflect:

1. Can you explain in your own words what an algorithm is?
2. What's the difference between an interpreted and compiled language?
3. Do you understand that VS Code and Python are two different things? Can you explain the difference?
4. Have you successfully set up your course folder structure?
5. Can you open your project folder in VS Code and see your files?

If you answered "no" or "I'm not sure" to any of these questions, that's okay! This is your signal to review that section or ask for help.