Introduction to Computers and Python

INF 605: Introduction to Programming - Python

Prof. Rongyu Lin

Lecture 1

Quinnipiac University
School of Computing and Engineering

Reading: Deitel Ch. 1 (pp. 1-30)

Today's Learning Journey

Part I: Getting to Know Each Other (15 min)

- Icebreaker activities
- Programming in daily life
- Course goals and expectations

Part II: Python Programming with Google Colab (60 min)

- Google Colab setup and introduction
- Why Python? Programming fundamentals
- First programs and calculations
- Variables and data types
- Preview of Python's power

Goal: By the end of today, you'll be confident writing Python programs with variables and data types!

Learning Objectives

By the end of this lecture, you will be able to:

1. **Explain** the relationship between hardware, software, and programming
2. **Understand** why Python is an excellent first programming language
3. **Set up** and use Python development environments (Google Colab, Jupyter)
4. **Write** and execute Python programs with proper syntax
5. **Create and use** variables to store different data types
6. **Perform** mathematical calculations including order of operations
7. **Format output** using f-strings and print statements
8. **Use** basic math library functions
9. **Appreciate** Python's role in data science and real-world applications

Let’s Get to Know Each Other!

Programming Background Survey

- Raise your hand if you’ve programmed before
- What programming languages have you tried?
- What was your first programming experience like?
- Don’t worry if you’re a complete beginner - you’re in great company!

Fun Fact: Every expert programmer started exactly where you are now!

”The best time to plant a tree was 20 years ago. The second best time is now.”

Programming is Everywhere!

Think-Pair-Share: Where do you encounter programming in daily life?

Examples you might not have considered:

- **Your smartphone:** Every app, every swipe, every notification
- **Social media:** Algorithms deciding what you see
- **Transportation:** GPS navigation, ride-sharing apps, traffic lights
- **Entertainment:** Netflix recommendations, Spotify playlists, video games
- **Shopping:** E-commerce sites, price comparisons, inventory management
- **Education:** Learning management systems, online courses, digital textbooks

Python Powers: Instagram, Netflix, Spotify, NASA, Google, financial institutions!

Programming isn't just for "tech people" - it's for everyone!

Your Programming Goals

Share with a neighbor: What do you hope to build or accomplish?

Possible goals and what Python can help you achieve:

- **Data Analysis:** Analyze sports statistics, financial data, research findings
- **Web Development:** Create websites and web applications
- **Automation:** Automate repetitive tasks, organize files, send emails
- **Games & Apps:** Build games, mobile apps, desktop applications
- **AI & Machine Learning:** Create intelligent systems, chatbots, image recognition
- **Career Advancement:** Add valuable skills to any field of study

Fun Python Facts:

- Named after "Monty Python's Flying Circus"
- Most popular programming language in the world (2024)
- Used by 8.2 million developers worldwide

Welcome to INF 605!

Introduction to Programming - Python

- **Practical Focus:** Real-world applications
- **Interactive Learning:** Hands-on coding every class
- **Modern Approach:** Industry-standard tools and practices
- **Data Science Ready:** Foundation for analytics and AI
- **Project-Based:** Build portfolio-worthy applications

Course Philosophy:

”Learning programming is like learning to drive - you need practice behind the wheel!”

Every class: Code → Practice → Build

What is Programming?

Programming is...

- **Problem Solving:** Breaking down complex tasks into simple steps
- **Communication:** Giving precise instructions to computers
- **Creativity:** Building solutions that didn't exist before
- **Logical Thinking:** Organizing ideas in a structured way
- **Automation:** Making computers do repetitive work

Programming is NOT:

- Only for "math geniuses"
- Memorizing syntax
- Working alone in dark rooms
- Just for computer science majors

Real-World Analogy: Programming is like cooking!
Recipe → Program, Ingredients → Data, Instructions → Code

Computer Fundamentals: Hardware & Software

Hardware (Physical Components):

- **CPU:** The ”brain” that executes instructions
- **Memory (RAM):** Temporary storage for active programs
- **Storage:** Long-term storage (SSD, hard drives)
- **Input/Output:** Keyboard, mouse, screen, network

Software (Instructions):

- **Operating System:** Windows, macOS, Linux
- **Applications:** Web browsers, games, productivity tools
- **Programming Languages:** Python, Java, C++, etc.

Key Relationship: Software controls Hardware

Amazing Fact: Today’s smartphones have more computing power than the computers used for the Apollo moon landing!

Types of Programming Languages

The Evolution of Programming Languages:

1. **Machine Language**
   - Binary (0s and 1s)
   - Directly executed by CPU
   - Very difficult for humans
   - Hardware-specific
2. **Assembly Language**
   - Human-readable mnemonics
   - Still low-level
   - One-to-one with machine code
   - Requires assembler
3. **High-Level Languages**
   - Human-friendly syntax
   - Platform independent
   - Python, Java, C++, JavaScript
   - Easier to learn and use
   - More productive programming

Which would you rather write?

Google Colab: Our Python Playground

What is Google Colab?

- **Free:** No installation or setup required
- **Cloud-based:** Access anywhere with internet
- **Powerful:** Free GPU and TPU resources
- **Collaborative:** Share and work together
- **Ready-to-use:** Pre-installed Python libraries

Why We Use Colab in This Course:

- No complicated software installation
- Same environment for everyone
- Access from any computer/tablet
- Easy to share assignments and projects
- Professional data science environment

Getting Started:

- Visit: `colab.research.google.com`
- Sign in with your Google account
- Create a new notebook
- Start coding immediately!

Colab Features for Learning Python

Key Features We'll Use:

Code Cells:

- Write and run Python code
- Instant execution with Shift+Enter
- See output immediately below

Text Cells (Markdown):

- Add explanations and notes
- Format with headers, lists, and links
- Mix documentation with code

Built-in Libraries:

- NumPy, Pandas, Matplotlib pre-installed
- No need to install packages
- Ready for data science projects

Live Demo: Let's open Colab and create our first program!

All course materials and assignments will be in Colab format

Why Python? The Perfect First Language

Python's Advantages:

- **Readable:** Code looks like English
- **Beginner-Friendly:** Gentle learning curve
- **Powerful:** Can build anything from websites to AI
- **Popular:** Number 1 programming language worldwide
- **Versatile:** Web, data science, automation, games
- **Great Community:** Massive support and resources
- **Rich Libraries:** Don't reinvent the wheel

Industry Usage:

- Google, Netflix, Instagram, Spotify
- NASA, CERN, Financial institutions
- Data scientists and AI researchers

Python Philosophy: *"Simple is better than complex. Readability counts."*

Your First Python Program

Let's write our first Python program together!

The Traditional "Hello, World!" Program: [Jupyter Compatible]

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

Let's make it more personal: [Jupyter Compatible]

```
print("Hello, Quinnipiac University!")
print("Welcome to INF 605!")
```

Adding some calculations: [Jupyter Compatible]

```
print("2 + 2 =", 2 + 2)
print("Python is awesome!")
```

What's happening here?

- **print():** A function that displays text
- **Strings:** Text in quotes
- **Numbers:** No quotes needed
- **Operations:** +, -, *, /

Python as a Powerful Calculator

Python can perform all kinds of mathematical operations:

Basic Arithmetic:

- Addition: `15 + 25` gives 40
- Subtraction: `50 - 18` gives 32
- Multiplication: `6 * 7` gives 42
- Division: `84 / 4` gives 21.0
- Exponentiation: `2 ** 10` gives 1024

More Operations:

- Floor division: `17 // 4` gives 4 (integer result)
- Modulus (remainder): `17 % 4` gives 1
- Order of operations: `2 + 3 * 4` gives 14 (not 20!)
- Use parentheses: `(2 + 3) * 4` gives 20

Working with Variables: [Jupyter Compatible]

- `price = 29.99`
- `quantity = 3`
- `total = price * quantity`
- `print(f"Total: ${total:.2f}")`

Advanced Math with Libraries: [Jupyter Compatible]

- `import math`
- `math.sqrt(16)` gives 4.0
- `math.pi` gives 3.14159...

Understanding Python Data Types

Python automatically recognizes different types of data:

Numbers: [Jupyter Compatible]

- **Integers:** `age = 20`, `year = 2025`
- **Floats:** `gpa = 3.75`, `price = 29.99`

Text (Strings): [Jupyter Compatible]

- `name = "Alice Johnson"`
- `university = "Quinnipiac University"`
- Must be in quotes!

True/False (Booleans): [Jupyter Compatible]

- `is_enrolled = True`
- `has_scholarship = False`

Key Point: Python is smart - you don't need to declare types!

Creating and Using Variables

Variables are like labeled boxes that store values

Creating Variables:

- `student_name = "Alice Johnson"`
- `student_age = 20`
- `student_gpa = 3.75`

Variable Naming Rules:

- Use descriptive names: `price`, not `p`
- Use underscores: `first_name`, not `firstName`
- Start with letter or underscore, not numbers
- No spaces or special characters

Using Variables in Calculations: [Jupyter Compatible]

- `price = 29.99`
- `quantity = 3`
- `total = price * quantity`
- `print(f"Total: ${total:.2f}")`

Let's create some variables together!

Formatting Output with F-Strings

F-strings make it easy to include variables in text

Basic F-String Syntax: [Jupyter Compatible]

- Put `f` before the quotes
- Put variables inside curly braces {}
- `print(f"Hello, {name}!")`

Practical Examples: [Jupyter Compatible]

- `name = "Alice"`
- `age = 20`
- `print(f"My name is {name} and I am {age} years old.")`

Formatting Numbers: [Jupyter Compatible]

- `price = 29.99567`
- `print(f"Price: ${price:.2f}")`
- Output: "Price: $29.96" (rounded to 2 decimal places)

F-strings make your output look professional!

Preview: The Power of Python Libraries

Libraries give Python superpowers!

Math Library Example: [Jupyter Compatible]

- `import math`
- `math.sqrt(16)` gives 4.0 (square root)
- `math.pi` gives 3.14159... (pi constant)
- `math.sin(math.pi/2)` gives 1.0

Real-World Calculation: [Jupyter Compatible]

- `radius = 7.5`
- `area = math.pi * radius ** 2`
- `print(f"Circle area: {area:.2f}")`

Coming Soon: You'll use libraries for data analysis, web development, machine learning, and more!

Don't reinvent the wheel - use libraries!

Python in Data Science & Beyond

Where you'll see Python in action:

Data Science & Analytics:

- **Pandas:** Analyze spreadsheet-like data
- **Matplotlib:** Create stunning charts and graphs
- **NumPy:** Fast mathematical computations
- **Jupyter:** Interactive data exploration

Web Development:

- **Flask/Django:** Build web applications
- **Requests:** Interact with web APIs

Artificial Intelligence:

- **TensorFlow:** Machine learning models
- **OpenCV:** Computer vision and image processing

By the end of this course, you'll be ready to explore all of these!

Libraries: Standing on the Shoulders of Giants

What are Libraries?

- Pre-written code you can use
- Solve common problems
- Save time and effort
- Tested and reliable
- Community-contributed

Popular Python Libraries:

- **NumPy:** Fast mathematical computations
- **Pandas:** Data analysis and manipulation
- **Matplotlib:** Creating charts and graphs
- **Requests:** Web APIs and downloading data
- **Flask/Django:** Building web applications
- **TensorFlow:** Machine learning and AI
- **OpenCV:** Computer vision and image processing

This would be hundreds of lines without libraries!

Development Environments

How do we write and run Python code?

1. **Text Editor + Terminal**
   - Write code in any text editor
   - Run with `python filename.py`
   - Simple but basic
2. **Integrated Development Environment (IDE)**
   - VS Code, PyCharm, Thonny
   - Built-in features: debugging, syntax highlighting
   - Professional development
3. **Interactive Environments**
   - **IPython:** Enhanced interactive shell
   - **Jupyter Notebooks:** Web-based, great for learning
   - Perfect for experimentation

Today's Goal: Get familiar with IPython and create our first notebook!

Jupyter Notebooks: The Ultimate Learning Tool

What makes Jupyter special?

- **Interactive:** Run code as you write it
- **Visual:** Mix code, text, and charts
- **Shareable:** Easy to share with others
- **Educational:** Perfect for learning step-by-step
- **Professional:** Used by data scientists worldwide

Jupyter Features:

- Code cells and Markdown cells
- Rich output and inline plots
- Easy export and version control
- Interactive widgets

Benefits for Students:

- See results immediately
- Document your thinking
- Save your progress
- Portfolio of work

We'll create our first notebook together!

Today's Takeaways & Next Steps

What We Covered Today:

- ✓ Got to know each other and shared programming goals
- ✓ Computer fundamentals and programming concepts
- ✓ Python's advantages as a first language
- ✓ Created variables and used different data types
- ✓ Performed calculations with proper syntax
- ✓ Learned about f-string formatting
- ✓ Preview of Python's power in data science

Hands-On Experience:

- ✓ Set up Google Colab environment for Python programming
- ✓ Wrote Python programs with variables
- ✓ Used Python for mathematical calculations
- ✓ Applied proper variable naming conventions
- ✓ Formatted output professionally

For Next Class:

1. Read Deitel Chapter 2 (Variables and Simple Data Types)
2. Try creating your own variable examples
3. Install Python on your personal computer (optional)

Questions?

Congratulations!

You've taken your first steps into the world of programming

Remember:

"Every expert was once a beginner.
Every master was once a disaster.
Every pro was once an amateur."

The key is to keep practicing and never stop learning!

See you Wednesday for more Python adventures!