Introduction

Preliminaries

1. Syllabus

2. Textbook – You should have a copy

3. Motivation for this class – What this class is all about
    a. More than a programming class
    b. It's about "Computational Thinking"
    c. I want you to be able to:
        - Solve problems
        - Design algorithms
        - Think about efficiency
        - Translate your algorithm → Program
4. Why Python

    Pascal → C++ → Java → Python

    Python is:
    - Easy to use – to get started quickly
    - Not much overhead (extra typing)
        a. Allows you focus on the problem – algorithm
    - Easy to experiment – best way to learn – INTERACTIVE MODE
    - Plenty of 3rd party modules

5. Is Python a "Real" Language – Yes
    a. **Industrial Light and Magic** – Star Wars Episode I
    b. **Spotify** – Data Science & Data Analysis (Pandas)
    c. **Google** – Web Development, official server-side language
    d. **Netflix** – Data analysis on the server side
    e. **Instagram** – Implement its "Business Logic" & data analysis
    f. **Dropbox** – For its desktop client
    g. **Reddit** – Entire site relies on Python libraries
    h. **NASA** - Real-Time systems and simulation
    i. Etc.

6. Careers?
    a. In 2020 there will be 1.4 million computer-science related jobs available
    b. With only 400,000 graduates to fill those roles.

7. Python 2 vs. Python 3
    a. All software has versions
    b. Java is now on version 14
    c. Backward compatibility
    d. Python 2 and Python 3 are not compatible
    e. As of 1/1/20, for Python 2 – there will be:
        - no new bug reports
        - no fixes
        - no changes
    f. Python 2 is no longer supported
    g. Current version is Python 3.8.
        - That's what I'll be using in class
    h. You must have Python 3.7 or later for this class

Chapter 1, Miller 3rd ed., Introduction to Python

Section 1.4, Problem Solving Strategies

1. Problem Solving Rule #1 – Think Before you Code
    a. Programming: 80% Thinking - 20% Coding

Big Picture	Program Layout	Coding
Strategy	Tactics	Tools

    b. When starting a new program, worst thing you can do first → Start coding

    c. How can you possible tell the computer how to solve a problem if you don't know how to solve it?

2. Example Problem
   a. Class of 12 students – Each shakes hand with everyone else
   b. How many handshakes are there?

Person #1 arrives	0 handshakes
Person # 2 arrives	1 handshake
Person # 3 arrives	2 handshakes
Person # 4 arrives	3 handshakes
…	
Person # 12 arrives	11 handshakes
	ADD These Up

For 12 people: 1 + 2 + … 11 = 66

In general, for X people, number of handshakes

1 + 2 + … + (X-1)

It is well known that

$$\sum_{i=1}^{n} i = \frac{n(n+1)}{2}$$

Can be easily verified, example with n = 10

1 + 2 + 3 + 4 + 5 + 6 + 7 + 8 + 9 + 10

	1	2	3	4	5	6	7	8	9	10
+	10	9	8	7	6	5	4	3	2	1
	11	11	11	11	11	11	11	11	11	11

Add up the sums. There are n terms, each one has value n+1
So, the sum = n(n+1).
But it is twice as big, so the original sum = n(n+1) / 2

Back to our original problem, with 12 people

$$Handshakes = \sum_{i=1}^{11} i = \frac{(11)(12)}{2} = (11)(6) = 66$$

Much more satisfying solution.

Generalizes to any number of people.