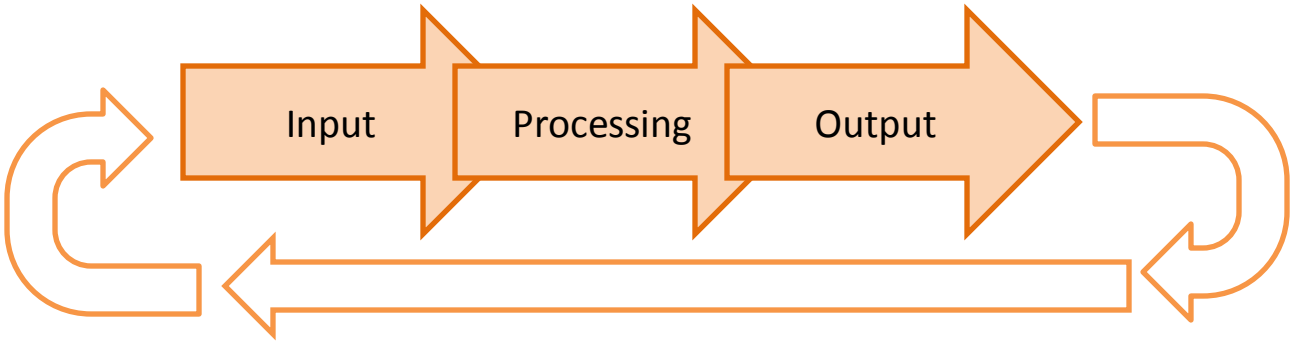
# Computer Programming in Python

Chapter 1

Introduction

# Chapter 1 Introduction

- Computers are simply ***data processing devices***
  - Machines that receive input, process it in some way, and produce output
  - They need to be told what to do and in what order



# Chapter 1 Introduction

- Computers Programs
  - Statements that tell a computer what to do
  - A sequence of small steps for the computer to execute and produce the desired result
  - Millions or even billions of these small steps are being executed by the computer every second
  - Combine to provide what appear to be seamless operations as we interact with the computer

# Chapter 1 Introduction

- Software
  - Computer programs are referred to as ***software***
  - Programs are needed to make the computer useful…tell it what to do and in what order
  - People who design, write, and test software are commonly referred to as ***software engineers***, software developers, or computer programmers

# Chapter 1 Introduction

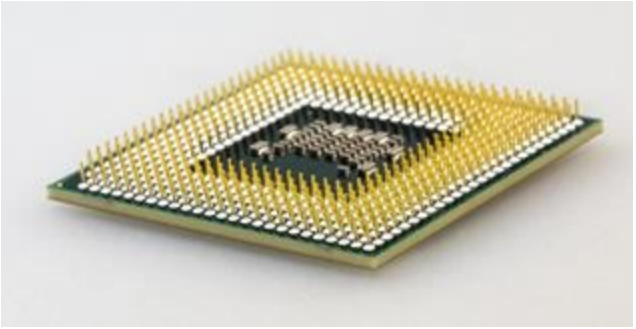
- Parts of the Computer - ***Hardware***
  - Any part of the computer that we can physically touch (including the parts inside)
  - Inside, is a circuit board referred to as the ***Motherboard***
    - Holds most of a computer's electronic devices

# Chapter 1 Introduction

- Parts of the Computer - Hardware
  - The main piece of computing hardware is the Central Processing Unit - ***CPU***
    - Mounted on the Motherboard
    - The “Brains” of the computer
      - Performs basic instructions and controls computer operations

# Chapter 1 Introduction

- Parts of the Computer - Hardware
  - The CPU has two parts
    - **Control Unit** - retrieves and decodes instructions, and coordinates computer activity
    - **Arithmetic Logic Unit** (ALU) - basic arithmetic and data comparison



# Chapter 1 Introduction

- Parts of the Computer - Hardware
  - Main Memory (RAM) - **Random Access Memory**
    - Modules with a series of memory addresses that store information
    - The memory addresses are ***volatile***
      - When the computer is turned off, RAM is cleared
    - The CPU can access RAM very quickly and programs and files are copied into RAM for this reason

# Chapter 1 Introduction

- Parts of the Computer – Hardware
  - ***Secondary Storage***
    - Hard Drives, External Drives, Flash Drives, etc.
    - ***Non-volatile*** - information is retained when the power is off
    - Data access speeds are slower

# Chapter 1 Introduction

- Hardware - Input and Output Devices
  - ***Input device*** - anything that provides input or data for a computer
    - Keyboard, mouse, microphone, camera, etc.
  - ***Output device*** - anything that accepts computer output
    - Monitors, speakers, printers, etc.

*Storage devices can be used for reading data into a computer or writing output from a computer, so they could be considered both input and output devices.*

# Chapter 1 Introduction

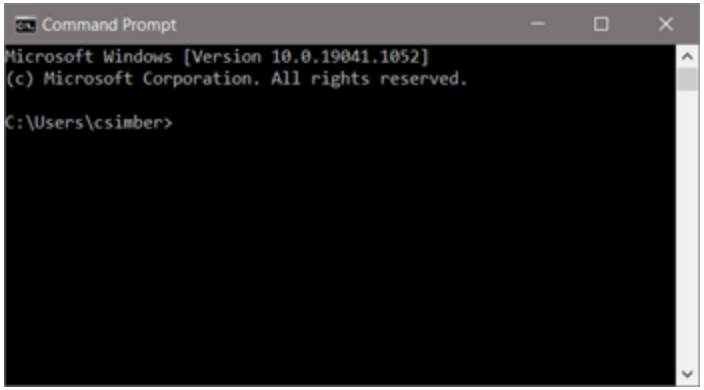
- Software – Computer Programs
  - Computers follow an input, processing, output sequence of operations
  - Computers need to be told what to do and in what order to do it
  - This is accomplished with ***software***
    - The instructions for the computer

# Chapter 1 Introduction

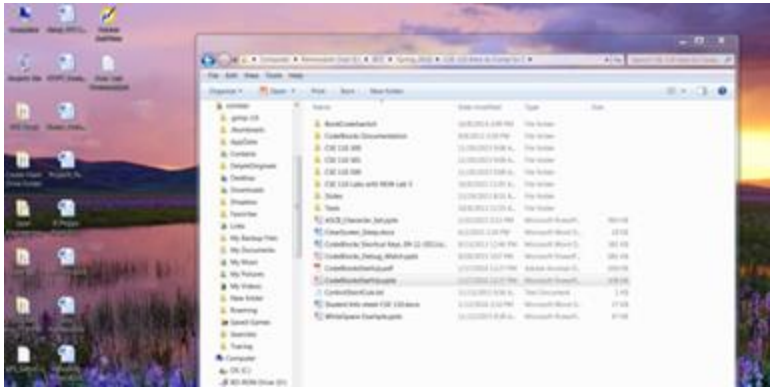
- Software – Two Types
  - ***System*** (Operating System) Software
    - Provide an interface for us to more easily use computers
    - A computer does not need an OS
    - It is much easier for us to interact with the computer through an operating system

# Chapter 1 Introduction

- Operating System evolution
  - Command Line
  - Menu driven
  - Graphical User Interface (GUI)




Command Line Interface (GUI)



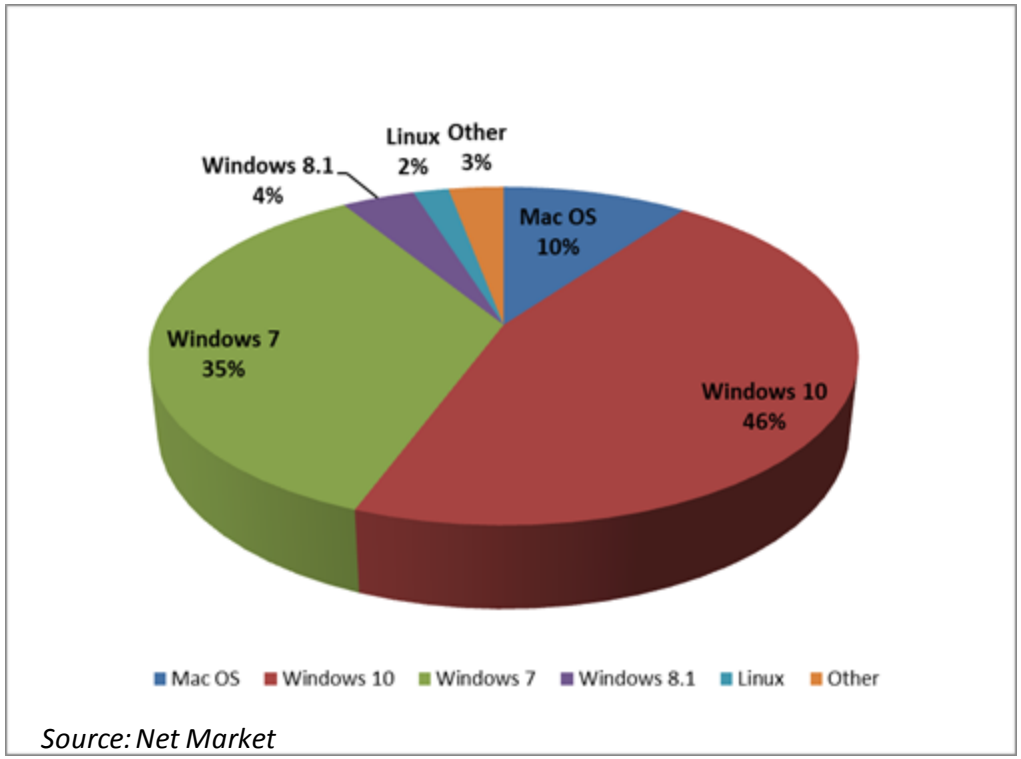
Graphical User Interface

# Chapter 1 Introduction

- Software – two Types
  - System (Operating System) Software
    - Acts like an orchestra conductor
      - Controlling computer hardware
      - Managing devices connected to the computer
      - Interacting with programs that are running

# Chapter 1 Introduction

- Popular Operating Systems
  - Windows, macOS, and Linux.




*Source: Net Market Share*

- Software – two Types
  - ***Application programs*** (everything else)
    - Software that we commonly use to accomplish work on a computer
      - Word processors
      - Spreadsheet applications
      - Gaming software
      - Presentation programs
      - etc.

# Chapter 1 Introduction

- The Language of Computers - ***binary language***
  - ***bit*** (or binary digit) - in computing represents a logical state that can be 0 or 1
    - It is the smallest information representation
  - ***Byte*** is a combination of 8 bits of 0s and 1s
    - Each letter, number, and special character consists of a binary representation

"K" = 01001011

# Chapter 1 Introduction

- The Language of Computers
  - ***ASCII*** (pronounced askee) stands for the American Standard Code for Information Interchange which was developed as a character encoding standard

| Decimal | Binary | ASCII |
|---|---|---|
| 64 | 0100 0000 | @ |
| 65 | 0100 0001 | A |
| 66 | 0100 0010 | B |
| 67 | 0100 0011 | C |
| 68 | 0100 0100 | D |

# Chapter 1 Introduction

- The Language of Computers
  - The ASCII table consists of binary representations
    - 26 uppercase and 26 lowercase letters
    - 9 digits
    - Special characters
    - Punctuation marks
    - Other symbols and keyboard keys

*Reference Appendix A of the text*

# Chapter 1 Introduction

- The Language of Computers
  - ***Unicode Standard*** - incorporates the ASCII 256 items
  - Expands to include the binary representations for symbols and the text used in most of the world's languages
  - Unicode 13.0 contains binary representations for 143,859 characters.

# Chapter 1 Introduction

- The Language of Computers
  - Numbers
    - Numeric integer values (whole numbers)
      - Represented in computers using the positions of the bits and powers of 2 at those bits
        - » Starting from right and working toward the left

# Chapter 1 Introduction

- The Language of Computers
  - Numbers example: 0000 1110 = 14


$$1 \times 2^3 + 1 \times 2^2 + 1 \times 2^1 + 0 \times 2^0$$

$$8 + 4 + 2 + 0$$

$$14$$

# Chapter 1 Introduction

- The Language of Computers
  - Additional Bytes and other numbering schemes are used for larger numbers
  - Images are made up of pixels and are stored by converting each pixel to a numeric value which is then stored in binary
  - Sound is stored using samples that are converted to the nearest numeric value

# Chapter 1 Introduction

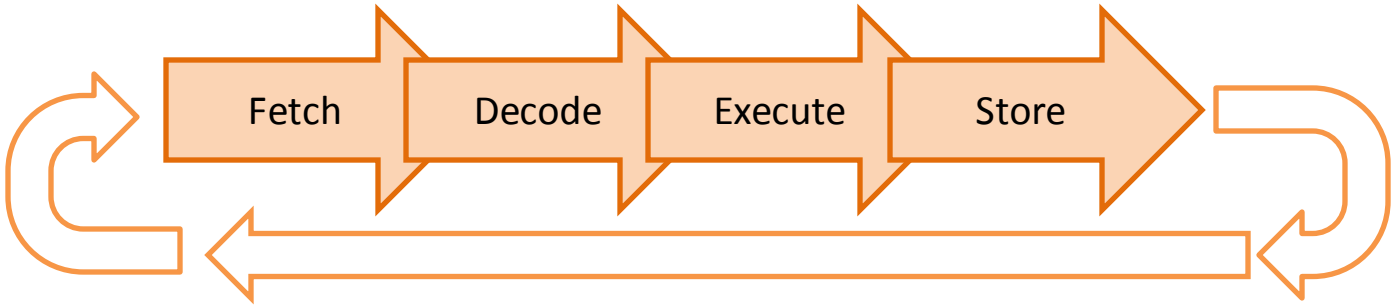
- Computer - machine made up of the various parts
  - Needs to be told what to do and in what order to do it…how do we do this?
  - We program the computer to perform various tasks
    - The computer wants machine language
      - This is tedious for us
    - ***Programming languages*** provide an easier way for us to communicate with the machine

# Chapter 1 Introduction

- Programming Languages – Two Types
  - ***Low-level languages***
    - Closer to machine language
      - Machine and Assembly
  - ***High-level***
    - Closer to our language
      - All other programming languages

- Programming Languages – ***Low-level***
  - Mirror computer (CPU) operations
  - CPU operations are repeated many times very quickly
    - Goes through a ***machine cycle***
      - Fetch, decode, execute, and store
      - Occurring millions or even billions of times per second
  - CPU processing power is measured in hertz (cycles-per-second)
    - *A one-gigahertz (1 GHz) CPU can execute one billion cycles (instructions) per second*

- ***Machine cycle***
  - Fetches the required piece of data or instruction from memory
  - Decodes the instruction
  - Executes the instruction
  - Stores the result of the instruction



- Programming Languages - ***High-level languages***
  - Introduced to make programming easier and more efficient
  - Multiple instructions are combined into a single statement
  - There are hundreds of high-level languages
    - Approximately 250 programming languages currently in use
    - About 700 languages in total

# Chapter 1 Introduction

- Programming Languages
  - A few High-level Languages and their intended use

| | |
|---|---|
| Ada | Department of Defense programs |
| BASIC | Beginners All-purpose Symbolic Instruction Code |
| C, C++ | powerful general-purpose programming |
| COBOL | Common Business-Oriented Language |
| FORTRAN | FORmula TRANslator for math and science |
| Pascal | teaching programming |
| Java | applications running over the internet |
| Python | general-purpose applications and data handling |

# Chapter 1 Introduction

- Programming Languages
  - Writing in high-level languages is much easier, but the computer wants binary
  - Software must be ***translated*** for the computer
    - A ***compiler*** translates the high-level language into a separate machine language program
      - Compiling or "building" the program
      - A "Build" is a compiled version of the software

*Java and C++ use a compiler*

- Programming Languages
  - An ***interpreter*** on the other hand, reads, translates, and executes the program one line at a time
  - As an instruction in the program is read by the interpreter, it converts the instruction into machine language and then executes that instruction
    - One line of code at a time

*Python uses an interpreter*

# Chapter 1 Introduction

- Programming Languages
  - The instructions that programmers write in a high-level languages is referred to as ***source code***
  - The code is written using a text editor typically in an ***Integrated Development Environment (IDE)***
  - IDE's are software applications that include a suite of integrated tools for software developers to write, execute, debug, and test software

- Programming Languages - Components
  - Programming languages have some characteristics and rules that must be followed when writing programs in that language
    - ***Syntax*** refers to the rules for properly combining symbols, operators, and punctuation, and the proper use of operators
    - ***Grammar*** determines the structure of the sentences containing the symbols, operators, and punctuation that make up the instructions for the computer

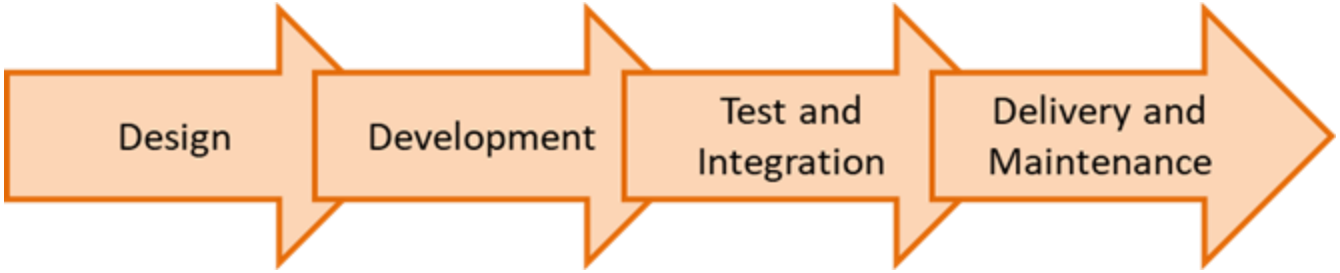
- Programming Languages - components
  - ***Keywords*** are reserved by the language for a specific use and cannot be used for another purpose
    - Some Python keywords

| | | | | |
|---|---|---|---|---|
| False | except | else | import | in |
| True | class | finally | is | return |
| and | continue | for | lambda | try |
| as | def | from | nonlocal | while |
| assert | del | global | not | with |
| async | elif | if | or | yield |

*Most IDEs will color code keywords to highlight them*

# Chapter 1 Introduction

- Developing Software
  - Software Project Process
    1. Requirements gathering and decomposition
    2. The Software Development Life Cycle (SDLC)

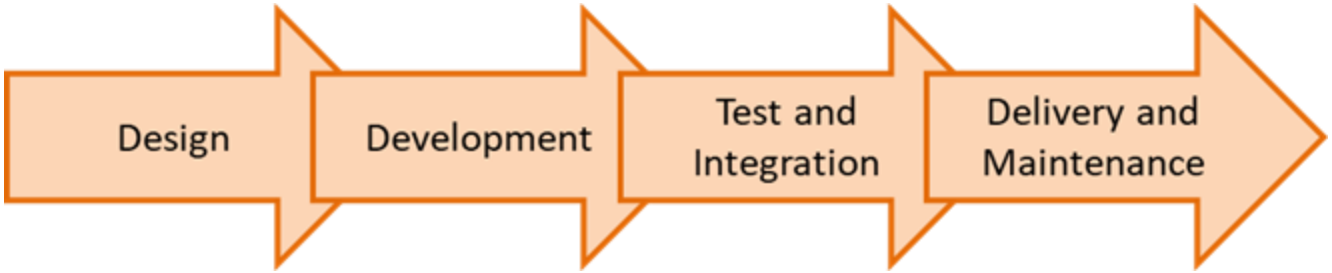


# Developing Software Projects and the Software Development Process

# Chapter 1 Introduction

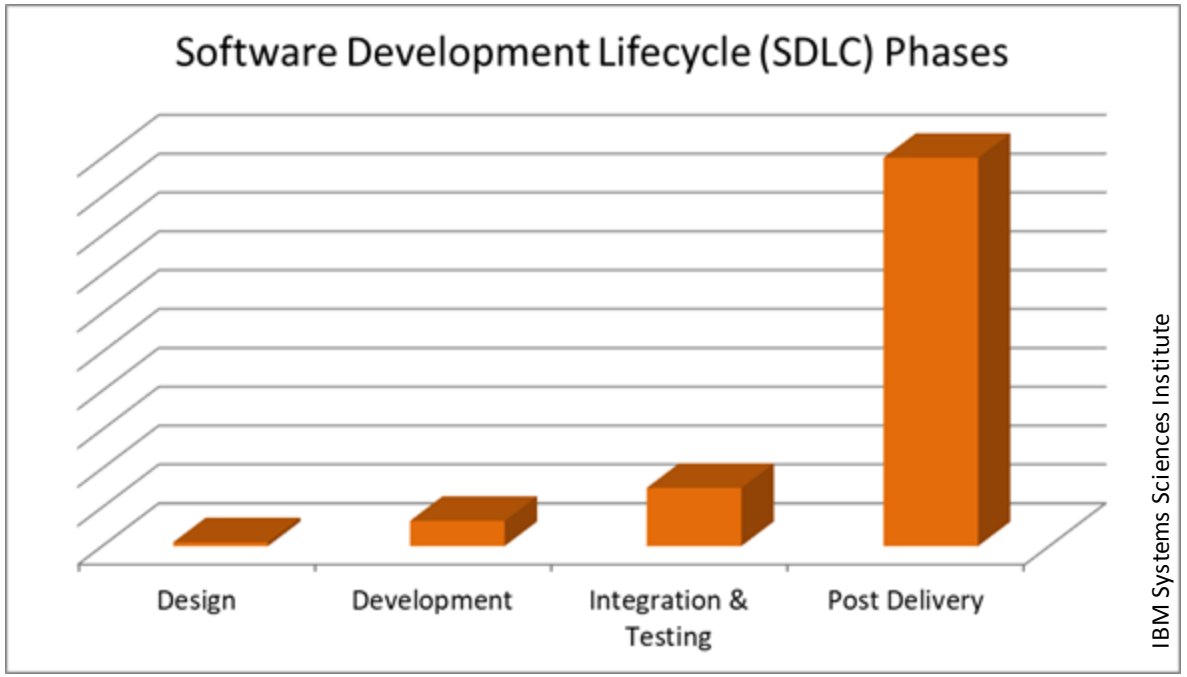
- Developing Software - ***Requirements Decomposition***
  - Prior to the planning and design phase
  - Complete understanding of what the program is supposed to do
    - How it will do what it is supposed to do will be determined as the design phase is completed (in the software development) phase
  - This process also assists in decomposing the project into manageable “chunks” for development

- Developing Software
  - Software Development Lifecycle (SDLC) Phases
    1. Design
    2. Development
    3. Test and Integration
    4. Delivery and Maintenance



- Software Developing Process – ***Design Phase***
  - After requirements are decomposed
  - Design the overall program
  - Break down the requirements and the project into smaller pieces
  - Determine program flow
    - How will the program be used
    - What output will be provided

- Software Developing Process – ***Design Phase***
  - Critical phase due to the cost increase of changes and fixing errors further on in the process



- Software Development Process ***Tools***
  - Software engineering tools that assist in the design (and development stage as well) include:
  - ***Pseudocode*** (sort of code)
    - Short-hand version of the steps and order of operations for a program

Do this
Then do this
Then do this
etc.

- Software Development Process Tools
  - Pseudocode example:
    - Consider a requirement to write a computer program that calculates *Recommended Net Worth*
    - Recommended Net Worth equation:

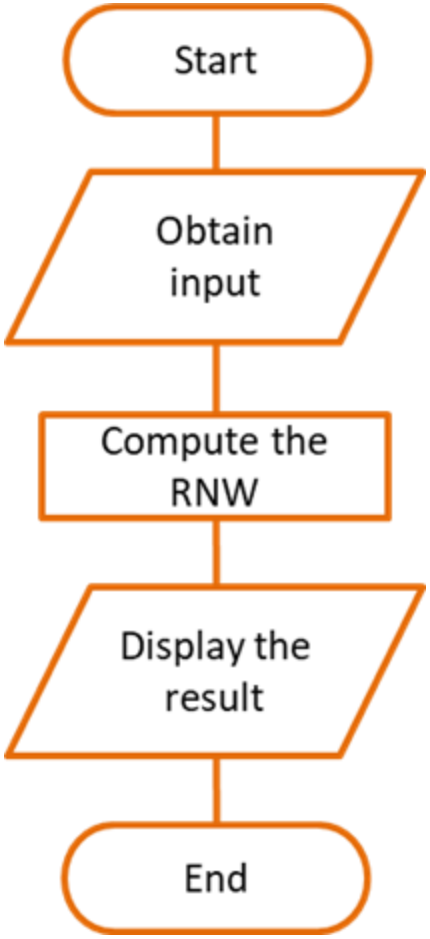
      Age *times* annual salary *divided* by 10

- Software Development Process Tools
  - Pseudocode example:
    - The pseudocode for the solution might be:

Step 1 Start the program
Step 2 Obtain age and salary information
Step 3 Compute the RNW (age x salary divided / 10)
Step 4 Display the output
Step 5 End the program

# Chapter 1 Introduction

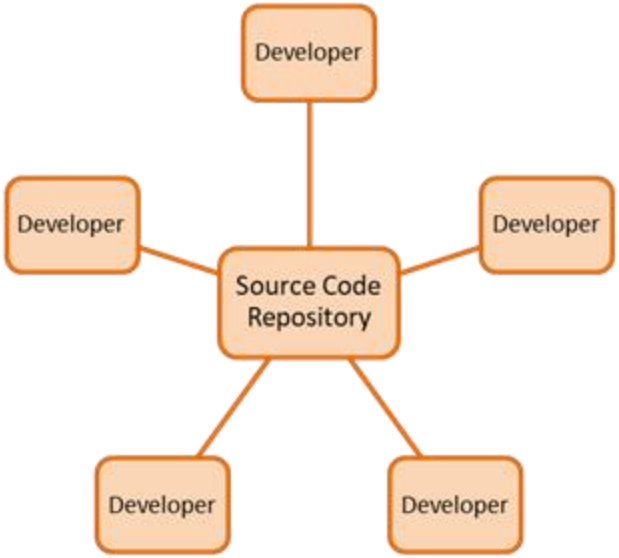
- Software Development Process Tools - ***Flowcharts***
  - Provide a faster and often clearer depiction of the ***algorithm*** (solution)
    - We think in pictures, not text
  - Help to ensure that steps are not overlooked



# Chapter 1 Introduction

- Software Development – ***Development Phase***
  - Includes writing the code that will be executed to produce the desired result and meet the requirements
  - Often divided among multiple programmers and requires collaboration and regular discussion to ensure a cohesive solution
  - A ***Configuration Management System*** (CMS) is used with a source code repository that stores all of the programs files

- Configuration Management System (CMS)
  - Tool suite for managing and developing software projects
  - Source code repository maintained by the Configuration Manager



- Configuration Management System (CMS)
  - Programmers access this repository to obtain a copy of a file and add functionality or make modifications to the code
  - The code is written in the copied file, and this changed file is tested with the other files in the source code repository
  - After testing, the modified file is placed into the repository and is used by all of the other programmers in place of the original file
  - The original file is retained by the configuration management tool as a version control mechanism

# Chapter 1 Introduction

- Configuration Management System (CMS)
  - Many configuration management systems have integrated suites that include:
    - Project scheduling
    - Task assignments
    - Progress tracking and reporting
    - Defect reporting and tracking systems
    - Cost reporting
    - Test and Quality Control data

# Chapter 1 Introduction

- Development Process Types
  - The ***Agile Development Process*** is a popular method in use today
    - Go by various names, but all are iterative and incremental software methodologies
    - Scrum – regular meetings, periodic cycles called sprints
    - Crystal - methodology, techniques, and policies
    - Dynamic Systems Development Method (DSDM)
    - Lean Development
    - Feature-Driven Development (FDD)

*Commonly referred to as Iterative Enhancement*

- Development Process Types – Agile
  - Evidenced-based empirical approach
  - Assumes that the problem cannot be fully understood or completely defined up front (before work begins)
  - Focuses on how to maximize the ability to deliver quickly, respond to emerging/changing requirements, and adapt to evolving technologies and changes in market conditions.

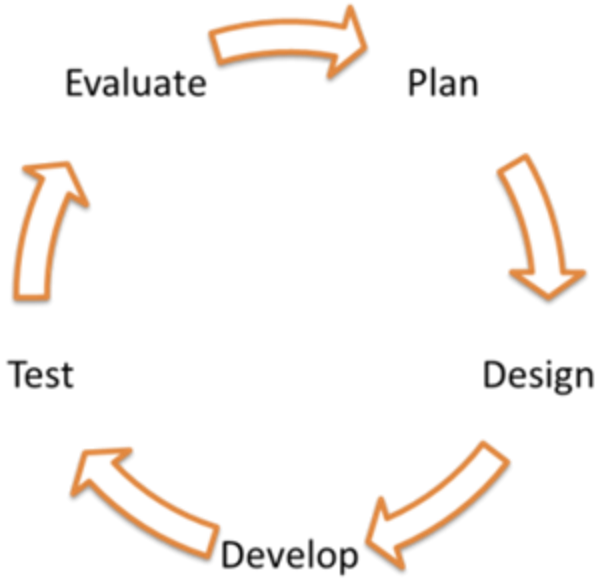
# Chapter 1 Introduction

- Development Process Types – Agile
  - A framework for managing software development
  - An iterative process that relies on weekly or bi-weekly development cycles called “sprints”
    - In many cases, there are also short daily stand-up meetings

- Development Process Types - Agile
  - the ***Sprint*** is a working period between scrums
  - Sprint meetings (Scrums)
    - Tasks completed from the previous sprint plan are reviewed
    - Completed work is demonstrated to stakeholders for feedback and approval
    - Tasks that were not completed from the previous sprint plan are reviewed with a course of action (re-plan)
    - The scope of work that will be completed during the next sprint cycle is planned, and engineers are assigned to the tasks

# Chapter 1 Introduction

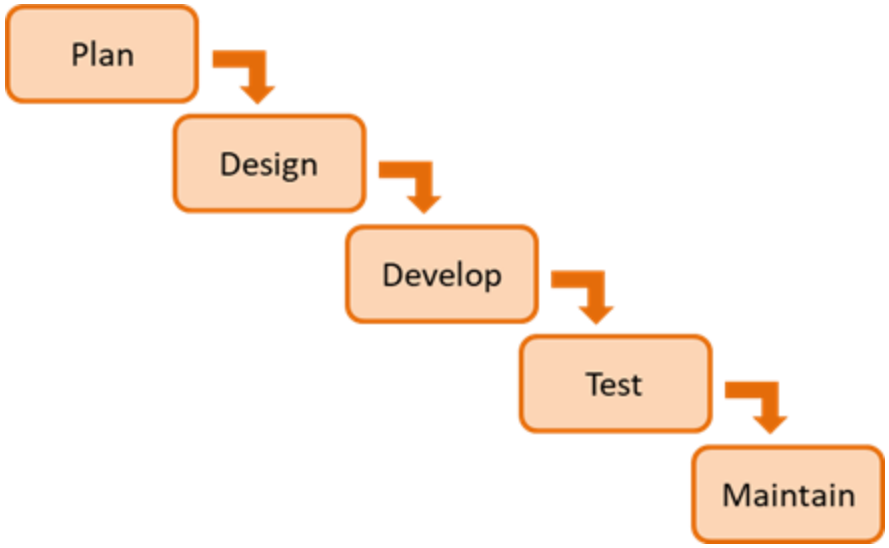
- Development Process Types – Agile Process




*Typically a weekly cycle*

# Chapter 1 Introduction

- Development Process Types – ***Waterfall Model***
  - Phases are sequential, may overlap, but are non-repeating
  - Each phase depends on the completion of the previous phase
  - Used extensively in the past, but replaced for the most part in recent years by the Agile Process



- Software Development – ***Test & Integration***
  - Program runs and is tested to ensure that there are no errors in the code, and that it performs correctly
    - Meets the requirements
  - In large organizations, there is a Test and Integration Team responsible for this phase
    - Any errors found by the team are relayed back to the developer

- Software Development – ***Test & Integration***
  - If the code is part of a larger project, it must be integrated into the overall project and tested again as a complete program
    - Most CMS tools provide this capability

- Software Development – ***Test & Integration***
  - There are two types of errors looked for during the test phase in addition to verifying the output
    - ***Syntax errors*** have to do with language specific rules like indentation and punctuation
      - Are found by the compiler or interpreter and the code will not be compiled or executed

- Software Development – ***Test & Integration***
  - There are two types of errors looked for during the test phase in addition to verifying the output
    - ***Logic errors*** are errors in the algorithm or the way that the algorithm was written by the programmer

Example: if the requirement is that the program multiply a number by two only if it is greater than ten, and the programmer writes the code so that a number is multiplied by two if it is less than ten, that would be a logic error.

# Chapter 1 Introduction

- Software Development – ***Delivery & Maintenance***
  - Final phase of the software development life-cycle
  - The program is delivered to the client or customer and a period of maintaining the program begins
  - Maintenance includes:
    - Updates or patches that fix errors or security issues found after delivery
    - Upgrades that provide additional functionality or capability

*Updates to software are commonplace today*

*Chapter 1 Introduction*