1. Problem Analysis

Programming is a process of problem solving. Different people use different techniques to solve problems. To be a skillful problem solver, and, therefore, to become a skillful programmer, you must use good problem-solving techniques. One common problem-solving technique includes following steps:

1. **Analyzing a problem**
2. Outlining the problem requirements
3. Designing (algorithms, flowcharts etc.)
4. Implement the algorithm in a programming language, such as Python.
5. Verify that the algorithm works.
6. Maintain the program by using and improving it, and modifying it if the problem domain changes.

Analyzing the problem is the first, and most important, step in the problem-solving technique. This step requires that you do the following:

- Thoroughly understand the problem.
- Understand the problem requirements. Requirements can include whether the program requires interaction with the user, whether it manipulates data, whether it produces output, and what the output looks like. If the program manipulates data, the programmer must know what the data are and how they are represented. To do this, you need to look at sample data.
- If the program produces output, you should know how the results should be generated and formatted.
- If the problem is complex, divide the problem into subproblems and repeat Steps 1 and 2 by analyzing each subproblem and understanding each subproblem's requirements. You also need to know how the sub-problems relate to each other.

Your overall programming experience will benefit if you spend enough time to thoroughly complete the problem analysis before attempting to write the programming instructions. Thoroughly analyzed and carefully designed program is much easier to follow and modify. Even the most experienced programmers spend a considerable amount of time analyzing a problem.

2. Algorithms and Flowcharts

Before writing computer programs we use different tools to design it. This design helps the programmer to write computer programs more easily, correctly, and quickly. Some most useful program design tools are: **algorithm**, **flowchart**, and **pseudocode**.

2.1. Algorithm

An algorithm is a finite sequence of instructions for solving a stated problem. A stated problem is a well-defined task that has to be performed and must be solvable by an algorithm. The algorithm must eventually terminate, producing the required result. For example, the algorithm to calculate simple interest (if principal is greater than or equal to 100000, interest is 5%, otherwise, interest is 3%) is given below:

1. Enter values of Principal, and Time.
2. If Principal is greater than or equal to 100000, then Interest = Principal × Time × 5 / 100.
3. Otherwise, Interest = Principal × Time × 3 / 100.
4. Display Interest as a result.

Properties of a good algorithm:

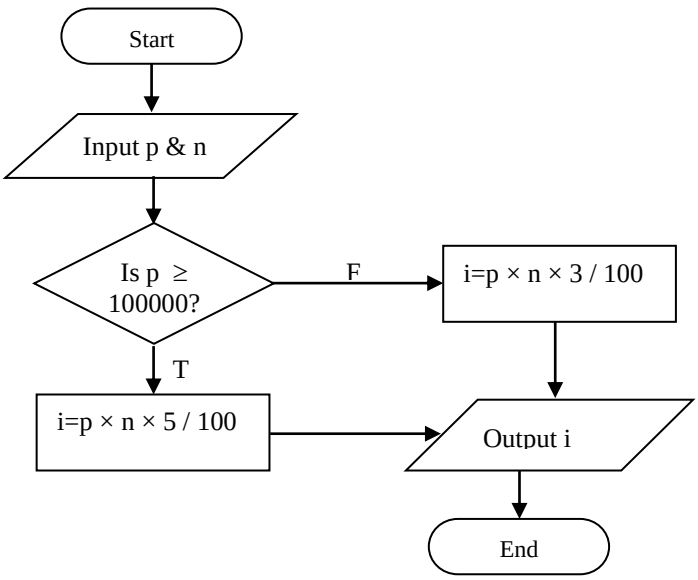
- **Input:** A number of quantities are provided to an algorithm initially before the algorithm begins. These quantities are inputs which are processed by the algorithm.
- **Definiteness:** The processing rules specified in the algorithm must be unambiguous and lead to a specific action.
- **Effectiveness:** Each instruction must be sufficiently basic such that it can, in principle, be carried out in a finite time by a person with paper and pencil.
- **Finiteness:** The total time to carry out all the steps in the algorithm must be finite.
- **Output:** An algorithm must have output.
- **Correctness:** Correct set of output must be produced from the set of inputs. For the same input, it must always produce the same output.

2.2. Flowchart

A **flowchart** is a common type of chart that represents an algorithm or a computer program showing the steps as boxes of various kinds, and their order by connecting these with arrows. Flowcharts are used in analyzing, designing, documenting or managing a process or program in various fields. A typical flowchart may have the following kinds of symbols.

- **Circles, ovals, or rounded rectangles:** Usually containing the word "Start" or "End", or another phrase signaling the start or end of a process.
- **Arrows:** Shows flow control. An arrow coming from one symbol and ending at another symbol represents that control passes to the symbol the arrow points to.
- **Rectangle:** Represents processing steps.
- **Parallelogram:** Represents input and output.
- **Diamond:** Represents condition or decision. These typically contain a Yes/No question or True/False test. This symbol is unique in that it has two arrows coming out of it, one corresponding to Yes or True, and one corresponding to No or False. The arrows should always be labeled.

For example, the flowchart to calculate simple interest (if balance is greater than or equal to 100000, interest is 5%, otherwise, interest is 3%) is given below:



2.3. Pseudocode

Pseudocode is an informal way of designing programs that does not require any strict programming language syntax or underlying technology considerations. It is used for creating an outline or a rough draft of a program. Pseudocode summarizes a program's flow, but excludes underlying details. Pseudocode is written to ensure that programmers understand requirements and align code accordingly.

At first glance, pseudocode looks like a modern programming language. The difference between pseudocode and a real programming language lies in the use of narrative text embedded directly within pseudocode statements. In pseudocode, we can use language constructs like IF, SWITCH, FOR, WHILE, and DO-WHILE along with Natural English. For example, the pseudocode to calculate simple interest (if balance is greater than or equal to 100000, interest is 5%, otherwise, interest is 3%) is given below:

```
GET Principal and Time
IF Principal >= 100000
    Interest = Principal × Time × 5 / 100
ELSE
    Interest = Principal × Time × 3 / 100
PRINT Interest
```

3. Coding

Coding, sometimes called computer programming, is how we communicate with computers. Code tells a computer what actions to take, and writing code is like creating a set of instructions. By learning to write code, you can tell computers what to do or how to behave in a much faster way. You can use coding skill to make different applications, websites and apps etc.

4. Compiler vs Interpreter

We generally write a computer program using a high-level language. A high-level language is one that is understandable by us, humans. This is called **source code**. However, a computer does not understand high-level language. It only understands the program written in **0**'s and **1**'s in binary, called the **machine code**. To convert source code into machine code, we use either a **compiler** or an **interpreter**. Both compilers and interpreters are used to convert a program written in a high-level language into machine code understood by computers. However, there are differences between how an interpreter and a compiler works.

Interpreter	**Compiler**
Translates program one statement at a time.	Scans the entire program and translates it as a whole into machine code.
Interpreters usually take less amount of time to analyze the source code. However, the overall execution time is comparatively slower than compilers.	Compilers usually take a large amount of time to analyze the source code. However, the overall execution time is comparatively faster than interpreters.
No Object Code is generated, hence are memory efficient.	Generates Object Code which further requires linking, hence requires more memory.
Programming languages like JavaScript, Python, Ruby use interpreters.	Programming languages like C, C++, Java use compilers.

5. Modern Computer System

A modern computer system consists of *hardware* and *software*. **Hardware** consists of the physical devices required to execute algorithms. **Software** is the set of these algorithms, represented as programs, in particular programming languages.

5.1. Computer Hardware

Any computer system has three essential components, namely, **input unit**, **central processing unit (CPU)** and **output unit**. The CPU itself has three components, namely, **arithmetic logic unit** (**ALU**), **control unit** (**CU**), and **memory unit** (**MU**). In addition, computers also have **secondary storage devices** (also called **auxiliary storage** or **backing storage**) that are used to store data and programs on a long-term basis.

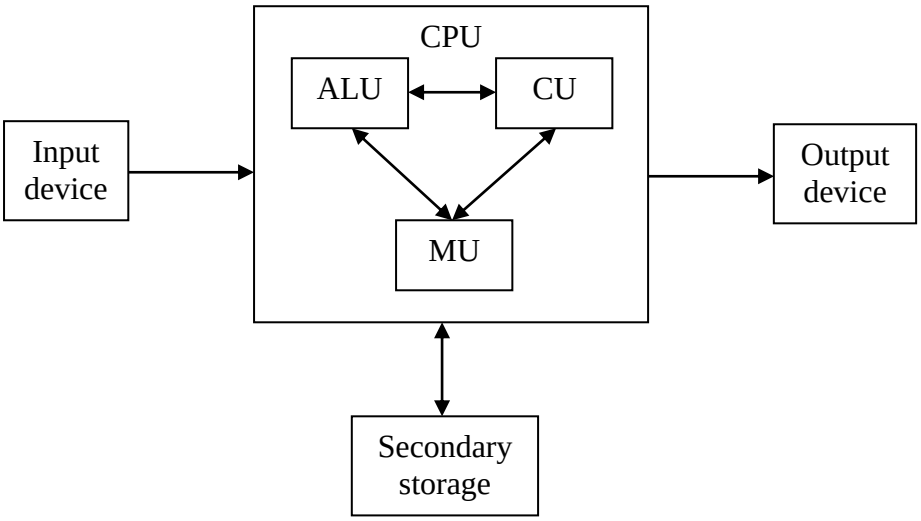


Fig: Building blocks of Computer

5.1.1. Input Devices

Input devices are used to transfer data into the memory unit of a computer. This information is then transferred from the memory to the ALU where comparisons and calculations are done and the results are sent back to the memory unit. Some common input devices are: keyboard, mouse, scanner, touch screen, and joystick.

5.1.2. Central Processing Unit (CPU)

The part of the computer that executes program instructions is known as processor or central processing device (CPU). It consists of three components. These components are arithmetic logic unit (ALU), control unit (CU), and memory unit (MU).

Arithmetic logic unit (ALU) performs fundamental mathematical operations consisting of addition, subtraction, multiplication, and division. In addition, it also performs logical operations that consist of comparisons like equal to, less than, greater than etc.

Control unit (CU) coordinates and controls the operations of a computer system. It controls the activities between memory and ALU and between CPU and input/output devices.

A memory unit (MU) is also called primary memory or main memory or RAM (random access memory). It holds data for processing, instructions for processing data (program), and information (processed data). The contents of main memory are lost when the computer is turned off.

5.1.3. Output Devices

The results that are stored in the memory can be transformed into a form that can be understood by users of a computer system by means of an output device. Some common output devices are monitor, printer, speaker etc.

5.1.4. Secondary Storage Devices

These devices are also called **auxiliary storage devices**. These devices are any storage other than main memory. Unlike main memory, these are long-term, non-volatile memory. That is, these devices store and retain the programs and data after the computer is switched off.

There are two types of secondary storage devices. This classification is based on the type of data access: **sequential** and **random**. In case of sequential access devices, the data stored in these devices can only be read in a sequence and to get to a particular point on the media, we have to go through all the preceding points. For example, **magnetic tapes** are sequential access media. In case of direct-access devices, the data stored can be accessed randomly at any point without passing through intervening points. **Magnetic disks** and **optical disks** are most common examples of direct-access devices.

- **Magnetic Tape:** It is a magnetically coated strip of plastic on which data can be stored. Storing data on tapes is considerably cheaper than storing data on disks. Accessing data on tapes, however, is much slower than accessing data on disks. Because, tapes are so slow, they are generally used for long-term storage and backup.
- **Magnetic Disk:** Magnetic disks store data on disks rather than in tapes. The most popular magnetic disks are **floppy disk** and **hard disk**.

  Floppy disk is a soft magnetic disk that is commonly used to move files between different computers, load new programs onto the computer, or store backup of data and small programs. It is slower to access data than hard disks and have less storage capacity, but is less expensive and portable. It is not very reliable and can be damaged easily. Floppies come in two basic sizes: 5.25 inch and 3.5 inch. 5.25 inch was common before 1978 and can hold 100KB and 1.2MB of data. The most common were 360KB and 1.2MB. 3.5 inch can hold 1.44MB of data. Now days, floppies are used infrequently.

  Hard disk is a magnetic disk on which we store computer data. It holds more data and is faster than floppy disks. It can store several gigabytes (GB) of data. A single hard disk usually consists of several platters. Each platter requires two read/write heads, one for each side. All read/write heads are attached to a single access arm so that they cannot move independently. Each platter has the same number of tracks and a track location that cuts across all platters is called a cylinder.
- **Optical Disk:** Optical disks are the storage medium from which data is read and to which data is written by lasers. There are two types of optical disks: **CD (compact disk)** and **DVD (digital versatile disk)**.

  CD can hold vast amount of information such as videos, music, animation, text etc. it is portable and its capacity usually ranges from 650 to 750MB. It is more reliable for long-term storage.

  DVD is primarily used to store movies or music. However, it can hold any type of information. It is similar to a CD but has a larger capacity. It can store about 17GB of data.

5.2. Computer Software

A computer needs both **hardware** and **software** for its proper functioning. Hardware (components like input, processing, output, and storage that can be physically handled) alone cannot perform any particular function without software.

Software is a computer program (set of instructions) that causes the hardware to do work. The software acts as an interface between the user and the computer. Software as a whole can be divided into a number of categories based on the types of work done. The two primary software categories are **system software** and **application software**.

5.2.1. System Software

System software includes **operating systems (OS)** and **utility programs** that operate and maintain a computer system and provide resources for applications programs. System software consists of low-level programs that interact with the computer at a very basic level. System software also includes software like **compilers**, **assemblers**, **debuggers**, and **file management tools**.

Operating systems are the most important programs that run on a computer. An operating system manages and coordinates the functions performed by the computer hardware. Every general-purpose computer must have an operating system to run other programs. Operating systems perform basic tasks, such as recognizing input from input devices, sending output to output devices, keeping track of files and directories on the disk, and controlling peripheral devices such as disk drives and printers. Most commonly used operating systems include Microsoft Windows, MS-DOS, UNIX, Solaris etc.

Utility is a program that performs a very specific task, usually related to managing system resources. Operating system contains a number of utilities for managing disk drives, printers and other devices.

5.2.2. Application Software

Application software (also called **end-user programs**) includes programs designed to help people to perform specific functions. Depending on the work for which it was designed, application software can manipulate text, numbers, graphics, or a combination of these elements. Application software can be a program used for accounting control in business, a program used for engineering design etc. Some basic examples are **word processors**, **spreadsheets**, **presentation graphics**, **image processors**, and **database management systems (DBMS)**.

A word processor is a program that enables us to perform word processing functions. Word processors use a computer to create, edit, and print documents. Of all computer applications, word processors are the most common. A word processor enables you to create a document, store it electronically on a disk, display it on a screen, modify it by entering commands and characters from the keyboard, and print it on a printer. Some of the commonly used word processors are Microsoft Word, WordStar, WordPerfect, AmiPro etc.

A spreadsheet is a table of values arranged in rows and columns. Spreadsheet applications or spreadsheets are computer programs that let you create and manipulate spreadsheets electronically. In a spreadsheet application, each value sits in a cell. You can define what type of data is in each cell and how different cells depend on one another. The relationships between cells are called formulas and the names of the cells

are called labels. These applications also support graphic features like charts and graphs. The most famous spreadsheet application is Microsoft Excel.

Presentation Graphics enable users to create highly stylized slides for presentation. This software includes functions for creating various types of charts and graphs and for inserting text in a variety of fonts. The most famous presentation software is Microsoft PowerPoint.

Image processors or graphics programs enable you to create, edit, manipulate, add special effects, view, print, and save images. Image processors include paint programs, draw programs, and image editors.

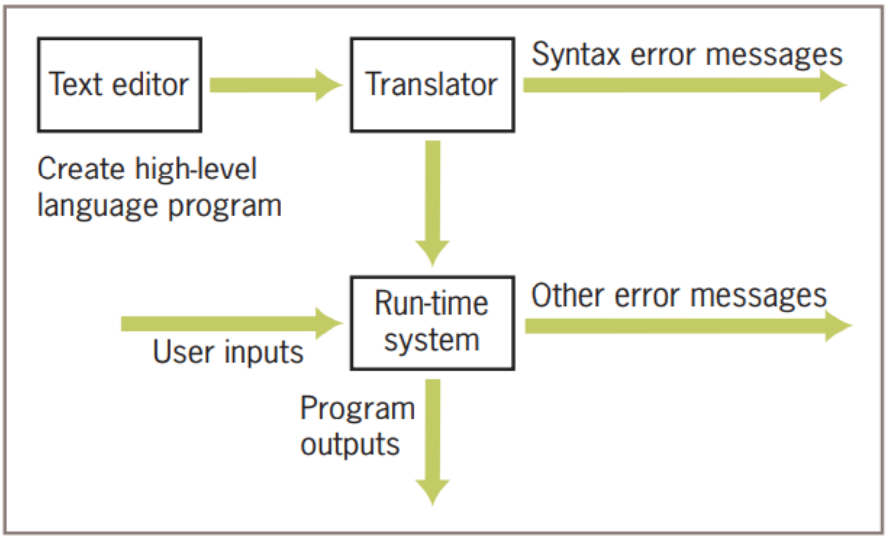
A DBMS is a collection of programs that enable you to store, modify, and extract information from a database (a systematically arranged collection of computer data). Some examples of DBMS are Microsoft Access, Oracle, Sybase, DB2 etc.

6. The Coding Process

A modern computer system consists of *hardware* and *software*. **Hardware** consists of the physical devices required to execute algorithms. **Software** is the set of these

Computer hardware can execute only instructions that are written in binary form – that is, in machine language. Writing a machine language program, however, would be an extremely tedious, error-prone task. To ease the process of writing computer programs, computer scientists have developed high-level programming languages for expressing algorithms. These languages resemble English and allow the author to express algorithms in a form that other people can understand.

A programmer typically starts by writing high-level language statements in a text editor. The programmer then runs another program called a translator to convert the high-level program code into executable code. Because it is possible for a programmer to make grammatical mistakes even when writing high-level code, the translator checks for syntax errors before it completes the translation process. If it detects any of these errors, the translator alerts the programmer via error messages. The programmer then has to revise the program. If the translation process succeeds without a syntax error, the program can be executed by the run-time system. The run-time system might execute the program directly on the hardware or run yet another program called an interpreter or virtual machine to execute the program.



The Coding Process

7. Installing and running Python

Download installer from http://www.python.org/download/ and install it. To launch an interactive session with Python's shell from a terminal command prompt, open a terminal window, and enter python at the prompt. To end the session on Unix machines (including macOS), press the Control+D key combination at the session prompt. To end a session on Windows, press Control+Z, and then press Enter.

To run a script, enter python, followed by a space, followed by the name of the script's file (including the .py extension), followed by any command-line arguments that the script expects.

You can also launch an interactive session with a Python shell by launching IDLE (Integrated Development and Learning Environment). IDLE is a default editor that comes with python. There are many advantages to using an IDLE shell rather than a terminal-based shell, such as color-coded program elements, menu options for editing code and consulting documentation, and the ability to repeat commands. IDLE also helps you manage program development with multiple editor windows. You can run code from these windows and easily move code among them.

The are several other free and commercial IDEs (Integrated Development Environments) with capabilities that extend those of IDLE. Some popular python IDEs are PyCharm, Spyder, Visual Studio Code, Atom, Jupyter etc.

8. Why Python?

Python is a widely-used, interpreted, object-oriented, and high-level programming language with dynamic semantics, used for general-purpose programming. It was created by **Guido van Rossum**, and first released on February 20, 1991.

While you may know the python as a large snake, the name of the Python programming language comes from an old BBC television comedy sketch series called **Monty Python's Flying Circus**.

One of the amazing features of Python is the fact that it is actually one person's work. Usually, new programming languages are developed and published by large companies employing lots of professionals, and due to copyright rules, it is very hard to name any of the people involved in the project. Python is an exception.

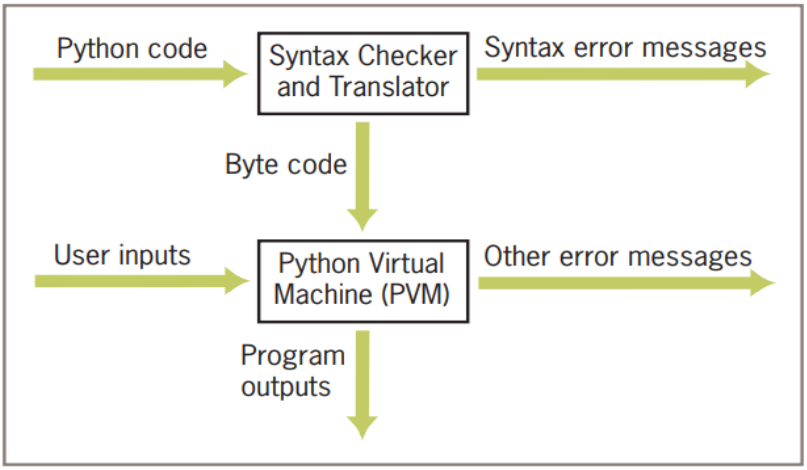
Of course, van Rossum did not develop and evolve all the Python components himself. The speed with which Python has spread around the world is a result of the continuous work of thousands (very often anonymous) programmers, testers, users (many of them aren't IT specialists) and enthusiasts, but it must be said that the very first idea (the seed from which Python sprouted) came to one head. Python has the following pedagogical benefits:

- Python has simple, conventional syntax. Python statements are very close to those of pseudocode algorithms, and Python expressions use the conventional notation found in algebra. Thus, students can spend less time learning the syntax of a programming language and more time learning to solve interesting problems.
- Python has safe semantics. Any expression or statement whose meaning violates the definition of the language produces an error message.
- Python scales well. It is very easy for beginners to write simple programs in Python. Python also includes all of the advanced features of a modern programming language, such as support for data structures and object-oriented software development, for use when they become necessary.

- Python is highly interactive. Expressions and statements can be entered at an interpreter's prompts to allow the programmer to try out experimental code and receive immediate feedback. Longer code segments can then be composed and saved in script files to be loaded and run as modules or standalone applications.
- Python is general purpose. In today's context, this means that the language includes resources for contemporary applications, including media computing and networks.
- Python is free and is in widespread use in industry. Students can download Python to run on a variety of devices. There is a large Python user community, and expertise.

9. How Python Works

Whether you are running Python code as a script or interactively in a shell, the Python interpreter does a great deal of work to carry out the instructions in your program.



Steps in interpreting a Python program

1. The interpreter reads a Python expression or statement, also called the source code, and verifies that it is well formed. As soon as the interpreter encounters error, it halts translation with an error message.
2. If a Python expression is well formed, the interpreter then translates it to an equivalent form in a low-level language called byte code. When the interpreter runs a script, it completely translates it to byte code.
3. This byte code is next sent to another software component, called the Python virtual machine (PVM), where it is executed. If another error occurs during this step, execution also halts with an error message.