Welcome to Programming Fundamentals!

Before anyone can begin their coding journey, it is essential to understand the basic concepts and why so many programming languages exist. This course will walk you through the crucial things that any programmer must know before getting hands-on.

What is Coding or Programming?

"Programming is how *you* get computers to solve problems."

There are two key phrases here that are important:

- **You**: without the programmer (you), the computer is useless. It does what **you** tell it to do.
- **Solve problems**: computers are tools. They are complex tools, admittedly, but they are not mysterious or magical: they exist to make tasks easier.

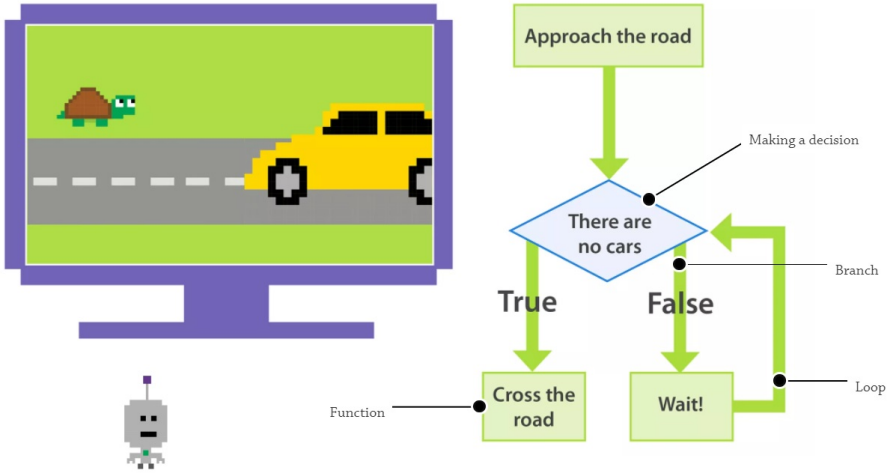
Computer programs (or software) are what make computers work. Without software, modern computers are just complicated machines for turning electricity into heat. It's software on your computer that runs your operating system, browser, email, games, movie player – just about everything.

Programming is Creative

Programming is a creative task: there is no right or wrong way to solve a problem, in the same way that there is no right or wrong way to paint a picture.

There are choices to be made, and one way may seem better than another, but that doesn't mean the other is wrong! With the right skills and experience, a programmer can craft software to solve an unlimited number of problems – from telling you when your next train will arrive to playing your favourite music.

The possibilities are constrained only by your imagination. That's what is best about programming.



When you create a program for a computer, you give it a set of instructions, which it will run one at a time in order, precisely as given. If you told a computer to jump off a cliff, it would!

1. *turn and face the cliff*
2. *walk towards the cliff*
3. *stop at the edge of the cliff*
4. *jump off the cliff*

To stop computers from constantly falling off cliffs, they can also make choices about what to do next:

If I won't survive the fall, don't jump off the cliff.

Computers never get bored and are really good at doing the same thing over and over again. Instruction 2 above might look in more detail like this:

2a. *left foot forward*

2b. *right foot forward*

2c. *go back to 2a*

These three concepts are the basic logical structures in computer programming:

1. **Sequence**: running instructions in order
2. **Selection**: making choices
3. **Repetition**: doing the same thing more than once, also called *iteration*

Add to these concepts the ability to deal with inputs and outputs and store data, and you have the tools to solve the majority of all computing problems.

Programs are often referred to as **code**, and hence programming is also known as **coding**.

Programming languages

Unfortunately, computers don't understand languages like English or Spanish, so we have to use a **programming language** they know to give them instructions.

There are many different programming languages, all of which have their own merits, and specific languages are better suited to particular types of tasks. Still, no one language is the 'best'.

Some popular programming languages are Python, C, C++, Java, etc.

Basic Coding Concepts

Now that you have a general idea about what a programming language is and what it does let us learn about some important concepts that everyone has to understand before learning a programming language. Every programming language has its way of presenting things, like how every language humans speak is different from each other.

In human languages, there are various sets of rules that are common in almost every language, such as grammar, style and pronunciation. Similarly, in computer languages, many concepts are nearly the same in all languages.

1. Variables

As the foundation of any computer programming language, **variables** act as “containers” that “hold” information. These containers then store this information for later use.

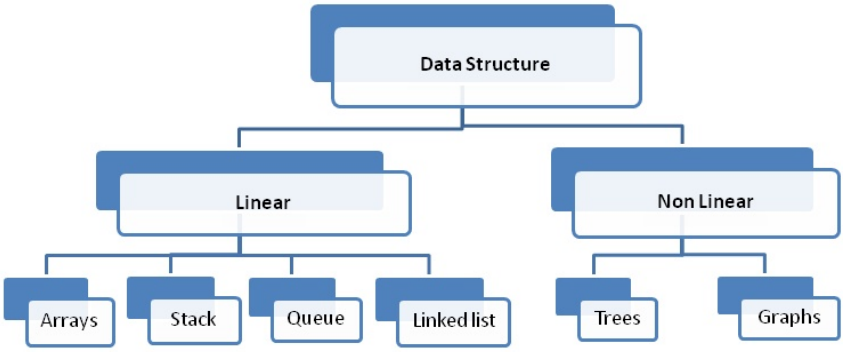
For example, imagine you are visiting the homepage of a website. Once you land on this page, a dialogue box pops into view with this simple greeting: “Hi! What’s your name?” This dialogue box is a variable! In this code, the programmer could name this variable “visitorName.” This means that when you type your name into the form and hit submit, your information will be stored in the “visitorName” variable. The programmer could then reference this variable at any time to access the information it contains.

2. Data Structures

Data structures allow programmers to streamline data collection when a large amount of related information is involved. Let's go back to our "visitorName" variable from above, but imagine the computer programmer needs to store and reference 10 different visitors' names rather than just one.

Rather than creating 10 different variables for each new visitor — which would increase the sheer amount of text in the program and make adding or removing new contacts difficult — the programmer could simply use a data structure to contain all related variables. In this case, the data structure would be a *List*.

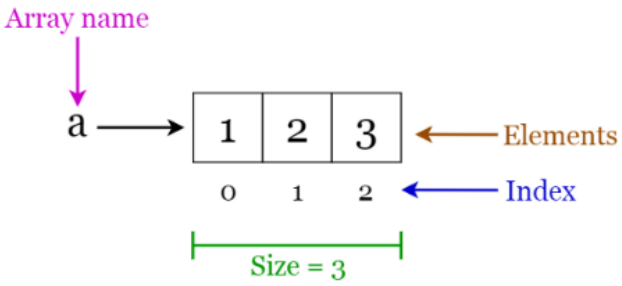
With this *List* data structure, the programmer only needs to create one variable rather than 10, which means the code would be much more flexible to change.



Common Data Structures:

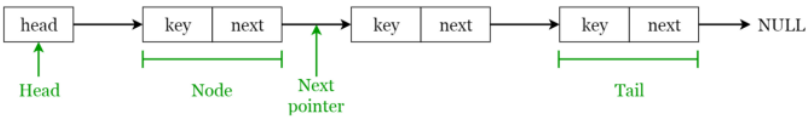
a) Arrays

An **array** is a structure of fixed size, which can hold items of the same data type. It can be an array of integers, an array of floating-point numbers, an array of strings or even an array of arrays (such as *2-dimensional arrays*). Arrays are indexed, meaning that random access is possible.



b) Linked Lists

A linked list is a sequential structure that consists of a sequence of items in linear order which are linked to each other. Hence, you have to access data sequentially and random access is not possible. Linked lists provide a simple and flexible representation of dynamic sets.

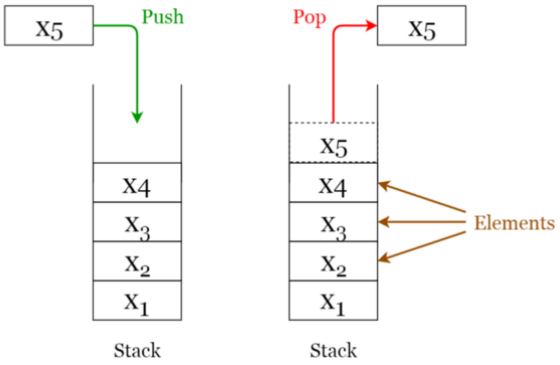


Let's consider the following terms regarding linked lists. You can get a clear idea by referring to the figure.

- Elements in a linked list are known as nodes.
- Each node contains a key and a pointer to its successor node, known as the next.
- The attribute named head points to the first element of the linked list.
- The last element of the linked list is known as the tail.

c) Stacks

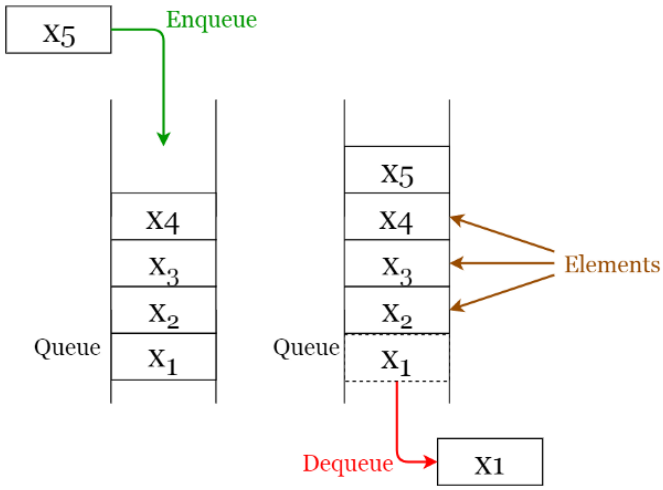
A **stack** is a **LIFO** (Last In First Out — the element placed at last can be accessed at first) structure which can be commonly found in many programming languages. This structure is named a "stack" because it resembles a real-world stack — a stack of plates.



- **Push**: Insert an element onto the top of the stack.
- **Pop**: Delete the topmost element and return it.

d) Queues

A **queue** is a **FIFO** (First In First Out — the element placed at first can be accessed at first) structure which can be commonly found in many programming languages. This structure is named a “queue” because it resembles a real-world queue — people waiting in a queue.



- **Enqueue**: Insert an element at the end of the queue.
- **Dequeue**: Delete the element from the beginning of the queue.

3. Control Structures

A **control structure** analyses variables and selects a direction in which to go determined from the given parameters. For example, when a computer program is running, the code is being read by the computer line by line from top to bottom and (for the most part) left to right.

As the code is being read, the computer will reach a point where it needs to make a “decision” (based on strict rules set by the computer programmer). At this point, the code could do things like jump to a different part of the program, re-run a certain piece of code again, or simply skip a block of code altogether.

Whatever parameters are set by the programmer will affect the code flow. Think of control structures as the directions your program needs to allow it to make choices and execute commands under different conditions.

- **Sequential**: default mode. Sequential execution of code statements (one line after another) -- like following a recipe
- **Selection**: used for decisions, branching -- choosing between 2 or more alternative paths. These are the some of the selection statements:
    - if
    - if/else
    - switch

- **Repetition**: used for looping, i.e. repeating a piece of code multiple times in a row. These are some of the types of loops:
    - while
    - do/while
    - for

4. Syntax

Just like in the English language, computer programming follows a **syntax** or a set of rules that define particular layouts of letters and symbols. Proper syntax ensures the computer reads and interprets code accurately. For example, let's consider a simple email address and its required syntax.

Email addresses are understood by readers and computers immediately due to their format. Typically, email addresses must consist of a string of letters and numbers, followed by the "@" symbol, and finally a website domain (e.g., bob_smith@companyname.com). This structure is known as the standard email syntax! It's easy to imagine that if the email address were not syntactically correct (company@.comnamebob_smith), computers would not be able to process it.

In a similar fashion, each computer programming language has its own syntax or appropriate order in how code should be written for the program to understand what it is supposed to do.

5. Tools

In the physical world, tools allow workers to perform tasks that would otherwise be extremely difficult (think of how a hammer helps drive a nail into a piece of wood and what this job would be like without tools). Similarly, a **tool** in computer programming is a piece of software that helps programmers write code much faster. These tools are generally called IDEs (Integrated Development Environment).

An integrated development environment is a software application that provides comprehensive facilities to computer programmers for writing code and running them. Each programming language has its own IDE which can convert the code which you've written into a form that can be understood by the computer.

Example: Python code can be written on IDLE, PyCharm or Visual Studio Code. Java can be written on BlueJ, NetBeans or Eclipse. These are the names of the IDEs that can allow the user to write and execute codes.

Key Terms

boolean: A data type representing logical true or false.

floating point: A data type representing numbers with fractional parts.

Integer: A data type representing whole numbers.

string: A data type represents a sequence of characters.