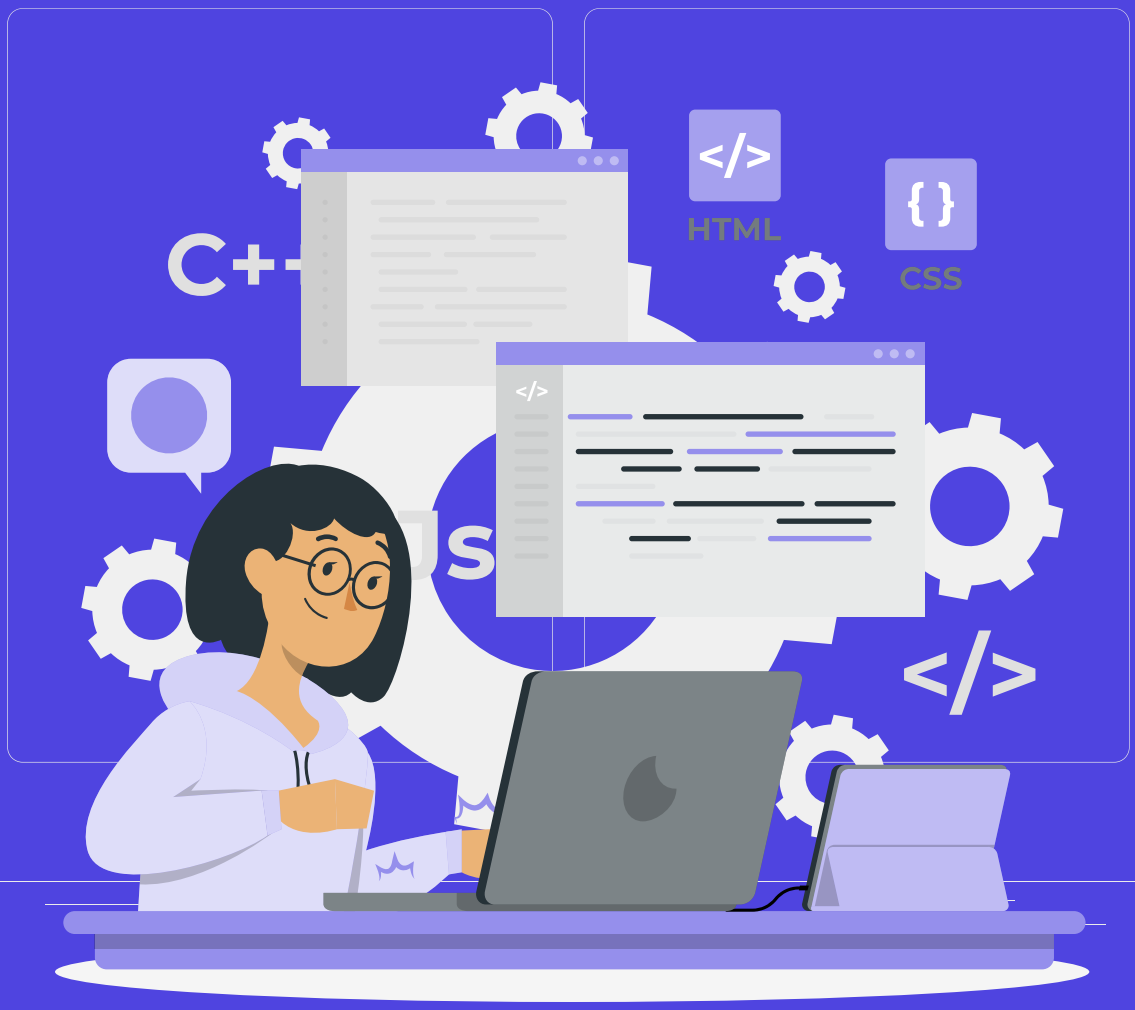
Introduction to Programming - BE
Lesson Plan



Relevel
by Unacademy

Recapitulation (2 min)

- Software Engineering is the process of developing and deploying a software end to end. It starts with feasibility study, requirement analysis, design, coding, testing and deployment/maintenance.
- A backend Engineer is someone who designs and develops the APIs, Database connection and server.
- Node JS, Express JS, SQL/No SQL Database & JavaScript are the trinity of backend development.

Content Outline (5 min)

- Recapitulation
- Introduction
- Different types of Programming paradigms.
- Difference between scripted and compiled languages.
- What is JavaScript?
- Assignment
- MCQ

Introduction (20 min)

We have learnt what software engineering is, the different scopes available for a software engineer and setting up our development environment through VS Code.

In this lesson we will learn the very essence of what it takes to develop a software, and it starts with something called programming.

What is Programming?

Programming is the set of instructions that we give to our computers to carry out a certain task.

We know that computers don't understand English, they only understand binary i.e. 0 & 1, but humans cannot speak binary, we speak languages that contain many words, vowels, phrases and letters.
But we need the computer to understand our instructions so that we can have the computer perform a task, since the computer doesn't understand English and since we cannot speak binary, we had to find a middle ground.

To make computers understand our instructions, we take the help of compilers(programs responsible for translating human language to machine code).

So compilers are essentially programs that act as translators and help us speak to the machine directly without learning how to communicate using 0s and 1s.

So programming is the act of laying out a set of commands that is given to the computer which is translated to machine code and then the task is carried out by the computer.

Why is programming required?

Since the dawn of civilization, human beings have evolved and created things to do work for us, be it donkeys carrying tons of bricks or a Boeing carrying tons of people, a juicer grinding fruits or a smartphone letting us call miles away, we have always invented things for our comfort and convenience that have allowed us to progress as a civilization.

Computers are no different, they are machines invented for doing complex calculations, video rendering, image processing, algorithmic analysis and for solving extremely complex problems.

Programming allows us to make use of computers to solve complex mathematical equations or render a 4K video to be uploaded on YouTube or using Google Docs to write this document that you are reading currently.

Everything and anything that can be done on a computer is done using programming, anything you do on a computer, be it a game you play(Counter Strike/ Project IGI), video you watch on YouTube or an application you run like MS Word, it is done using programming, each of these items contains thousands of lines of instructions telling the computer what to do and thus we get to run them smoothly on our computers.

Not everything can be done manually by human beings, certain tasks require automation, and there are computations that the human mind cannot always accurately state.

Thus to reduce manual work, improve efficiency and automate labor, programming was introduced as a means to solve the problem.

Programming can be done using a variety of computer programming languages, such as JavaScript, Python, and C++.

Different types of Programming Paradigms (1 hr)

We have learnt what programming is and why it is necessary. We will now learn all about the different types of programming paradigms.

There are basically four types of programming:

- Procedural
- Object Oriented
- Functional
- Declarative

Procedural Programming (15 min)

Procedural programming is a programming paradigm that evolved from structured programming and is based on the concept of invoking procedures. Procedures, often known as routines, subroutines, or functions, are essentially a set of instructions to be followed. Any procedure in a programme can be invoked at any time during execution, either by other procedures or by the programme itself.

As the name implies, a procedural language relies on specified and well-organized procedures, functions, or sub-routines in a program's architecture to express all the steps that the computer must follow to attain a particular state or result.

A programme is divided into variables, functions, statements, and conditional operators using the procedural language. To complete a task, procedures or functions are applied to the data and variables. These procedures can be called/invoked from any point in the programme hierarchy, as well as from other procedures. One or more processes can be found in procedural language software.

Procedural language is one of the most common types of programming languages in use, with notable languages such as C/C++, Java, ColdFusion and PASCAL.

When it comes to the following situations, procedural programming is often the best option:

- There's a complicated operation with inter - dependencies and a need for clear visibility of multiple program states ('SQL loading,"SQL loaded,'Network online,'No audio hardware,' and so on). This is usually the best way to start and stop an application.
- The program is one-of-a-kind, with only a few elements in common.
- The program is set in stone and is unlikely to change significantly over time.
- Over time, it is envisaged that no or just a few features would be added to the project.

When we have a lot of simple separate activities to do or a user interface to handle, we probably don't want to use it.

Object Oriented (25 min)

Object-oriented programming is a programming paradigm based on the concept of objects, which contain data as well as code to manipulate it. Object-oriented programming imitates many of the characteristics of real-world objects.

Java, C++, and Ruby are some of the most extensively used object-oriented programming languages. Many languages that are not strictly object-oriented, such as Python and JavaScript, provide features like classes and objects that are influenced by object-oriented programming. Simula and Smalltalk were the first object-oriented programming languages.

- Objects are made up of data and methods that describe their state and behaviour. These two entities are encapsulated in each object.
- The user cannot see how object methods are implemented inside. This allows objects to alter their abstract state using a simple external API.
- Classes have instances, which are objects. Classes serve as blueprints for constructing objects. An object's type is also its class.
- Other classes' state and behaviour can be inherited by classes. Objects of the subclass can be cast into objects of the parent class based on this concept.
- Polymorphism is a result of this type of casting. An object of a class can be implicitly cast to an object of the class's predecessors by the programme.

Let us now discuss 4 pillars of OOPS:

1. Abstraction:
The property of data abstraction is that just the most important details are displayed to the user. The user is not shown the units that are trivial or non-essential. For example, a car is considered as a whole rather than its distinct parts.

The practice of identifying only the required attributes of an object while discarding the irrelevant information is known as data abstraction. The features and behaviors of an object assist to distinguish it from other things of the same sort, as well as classify and group them.

Consider the case of a man at the wheel of a car. The man only knows that pressing the accelerators will increase the car's speed and that applying the brakes will stop it, but he has no idea how the speed is increased when the accelerator is pressed, nor does he understand the car's inner mechanism or how the accelerator, brakes, and other controls are implemented in the car. This is the definition of abstraction.

2. Encapsulation:
It's described as the grouping of information into a single unit. It's the glue that holds code and the data it manipulates together. Encapsulation can also be thought of as a protective shield that prevents data from being accessible by code outside of the shield.

- Encapsulation means that a class's variables or data are concealed from other classes and can only be accessible through the member functions of the class in which they are specified.
- Data-hiding is similar to encapsulation in that the data in a class is concealed from other classes.
- Declare all variables in the class as private and write public methods in the class to set and get the values of variables to accomplish encapsulation.

3. Inheritance:
Inheritance is a crucial component of OOP (Object Oriented Programming). It is a Java mechanism to allow one class to inherit the characteristics (fields and methods) of another.

Let's go over some of the most commonly used key terminologies:

- Superclass: A superclass is a class whose characteristics are inherited (or a base class or a parent class).
- Subclass: A subclass is a class that inherits from another class (or a derived class, extended class, or child class). In addition to the superclass fields and methods, the subclass can add its own fields and methods.
- Inheritance supports the concept of "reusability," which means that if we want to create a new class but there is already one that contains some of the code we need, we can derive our new class from the old one. We're reusing the old class's fields and functions in this way.

4. Polymorphism:
It refers to an OOPs programming language's ability to efficiently distinguish between things having the same name. Java accomplishes this through the signature and declaration of these entities.

OOP is frequently the better choice when:
- We have a team of programmers who don't need to know everything about every component.
- There's a lot of code out there that can be shared and reused.
- The project is expected to alter frequently and be expanded over time.
- Different resources, such as datasets or hardware, can assist different portions.

When we have numerous developers that need to share implementation details, we probably don't want to use it. For example, if we were creating a keyboard driver, we wouldn't want to break it up into pieces that would hide implementation details from a developer working on the driver.

Functional Programming (15 min)

Functional programming languages were created specifically for symbolic computing and list processing. Mathematical functions are the foundation of functional programming. Lisp, Python, Erlang, Haskell, Clojure, and other functional programming languages are some of the most popular.

The following are the most notable features of functional programming:

Functional programming languages are based on the concept of mathematical functions that execute computations using conditional expressions and recursion.

Higher-order functions and slow evaluation are supported by functional programming.

Flow is not supported by functional programming languages. Loop statements and conditional statements such as If-Else and Switch Statements are examples of controls. They use functions and functional calls directly.

Abstraction, encapsulation, inheritance, and polymorphism are all supported by functional programming languages, just like they are in OOP.

When should functional programming be used:

Domain specific languages (DSLs): If the issue domain is so complex that traditional industry languages like Java and C/++ fail to address it efficiently, a DSL might be the answer. We wouldn't use a DSL to construct our entire system, but we could use it, like the 5ESS switch, to code a critical function in a way that is easier to understand and maintain, ensuring its quality. DSL creation is a breeze with functional languages.

Declarative Programming (5 min)

Declarative programming is defined as writing code that defines what you want to achieve rather than how you want to do it. It's up to the compiler to find out how to do it.

Declarative programming refers to a set of problems for which the language implementation is responsible for finding a solution. Some parallel processing systems benefit from declarative programming since it simplifies the programming.

SQL and Prolog are two declarative programming languages.

Declarative programming is useful in the following situations:

- When we want to keep the code factorable and clean.
- When readability of the code is to be given utmost importance.
- When database queries need to be placed.

Difference between a Scripting Language and Compiled Language (25 min)

Scripting languages are, in essence, programming languages. The distinction between the two is that scripting languages do not require compilation and instead are interpreted. A C program, for example, must be compiled before running, whereas a scripting language like JavaScript or PHP does not require compilation.

Compiler programs are generally faster than interpreter programs since they are transformed from native machine code first. Furthermore, compilers examine and interpret code just once, reporting any faults that the code may contain collectively, whereas the interpreter reads and analyses code statements each time it encounters them, stopping immediately if there is a mistake. In fact, the gap between the two is blurring due to modern hardware's better calculation capabilities and innovative coding approaches.

Some scripting languages traditionally used without an explicit compilation step are JavaScript, PHP, Python, VBScript.

Some programming languages traditionally used with an explicit compilation step are C, C++.

Scripting Languages in Practice:

1. To programmatically automate some operations
2. Information extraction from a data set
3. Requires less code than standard programming languages.

Programming Language Applications:

1. They are usually run within a parent application, similar to scripts.
2. More consistent with mathematical models when integrating code
3. Programming languages such as JAVA can be compiled and run on any platform.

What is JavaScript? (1 hr)

JavaScript is a computer language that is largely utilized by Web browsers to provide users with a dynamic and interactive experience. JavaScript is used to develop the majority of the features and apps that make the Internet so important in modern life.

- It is lightweight and can be interpreted in an object-oriented manner.
- JavaScript was initially called LiveScript, but Netscape changed it to JavaScript.
- JavaScript is a programming language that provides an interface to the server-side. It allows developers to create interactive and non-standard web applications.

Why is JavaScript Important?

JavaScript is used to create an API:

- Create an API and connect to database using javascript libraries.
- Large support in open source community
- Easily integrate all the other services with the backend of any of the application
- Easy to learn other popular javascript framework and libraries since these are build on top of the Javascript
- Playing audio and video.
- Validating input from the backend.

While JavaScript is a server and client-side language, it has asynchronous interaction with a distant server as one of its most powerful capabilities. Asynchronous simply implies that JavaScript can connect with the server in the background without interfering with the forefront user interaction.

Take, for example, a search engine. Almost all search engines now have an autocomplete feature. As soon as the user types a word into the search box, a list of potential search terms or phrases shows underneath. It's a smooth experience. Without reloading the page, suggested search terms appear.
JavaScript reads the letters as the user inputs them, sends them to a remote server, and the server responds with suggestions.

JavaScript DataTypes

- Data types are used to store specific types of data and can be used to manipulate it accordingly.

Eg:

```
var x; // x is undefined
x = 12; // x is a Number
x = "John"; // x is a String
```

- **Primitive Data Types:** can only store a single value.
- **String:** used for storing text values in enclosed quotes

```
var firstName = 'John';
var lastName = "Doe";
var fullName = "My name is " + firstName + " " + lastName;
console.log(fullName);
// Output:
// My name is John Doe
```

- **Number:** used to store numbers (-2^53 - 1 to 2^53 - 1)

```
var a = 12; // number, integer
var b = 12.1; // number , floating point number
var c = 16/0; // Infinity
var d = 16/ -0; // - Infinity
var e = "text"/2 // NaN
```

- **Boolean:** can hold two values: “true” or “false”

```
var value = true;
```

- **Composite Data Types:** can hold complex values and collection of values.
- **Objects:** can store collection of data in a key-pair syntax

```
var car = {
  "modal": "XUV 700",
  "color": "black",
  "seats": 7
}
```

MCQs (10 min)

1. What kind of a language JavaScript is?
A. Scripting
B. Compiled
C. Static
D. None of the above
Answer: A

2. Which language type doesn't need an interpreter?
A. Compiled
B. Scripting
C. Dynamic
D. None of the above
Answer: A

3. Which programming paradigm uses objects?
A. Procedural
B. Functional
C. OOP
D. All of the above
Answer: C

4. Which programming language is used for placing queries to Database?
A. Functional
B. Declarative
C. Procedural
D. None of the above
Answer: B

5. Which one of the following is not an OOP concept?
A. Encapsulation.
B. Abstraction.
C. Polymorphism.
D. Reducer
Answer: D