Coding Language:

In modern programming, code is processed through a system of inputs and outputs called a coding language. While different languages have different purposes, strengths, and limitations, all coding languages aim to do the same thing: translate what a coder *wants* into what a computer *does*.

## **What’s Apex?**

Apex is a programming language used by Salesforce developers. It’s part of the Salesforce platform, and you can access it through the developer console.

The language has a range of specific features that set it apart from other coding languages, and you learn more about those in a bit. For now, the main thing to keep in mind is that Apex lets developers access the back-end power of Salesforce to create their own applications and do more with the platforms they already use for their business.

Similar to Java, Apex marries the ease of use programmers look for in any language with the ability to leverage the Salesforce API. This makes Apex a powerful resource for app development at any scale.

## **What Gets Coded with Apex?**

Developers use Apex to do more with data. From automating reactions to events, like button clicks, and related record updates to generating insights from large data sets, using Apex allows a wide range of operations that translate the needs of a business into coded solutions. In other words, it lets developers accomplish tasks that go beyond the out-of-the-box functionality of Salesforce products. Here’s a few things Apex can do for you.

* Perform complex, simultaneous validation over multiple objects, and perform custom validation implementation.
* Expand the functionality of existing workflows.
* Create web services that integrate other systems.
* Build email services for use cases like email blast or email setup.
* Apply personalized transactional logic (logic that occurs over the entire transaction, not just with a single record or object).
* React when a record is modified or modify the related object's record based on a triggering event.

Of course, it’s also important to keep in mind that code is a tool, not a cure-all for any problem. In many cases, you’ll find that a simple[Flow](https://help.salesforce.com/s/articleView?id=sf.flow.htm&language=en_US&type=5) or existing app can often meet your needs quicker and with less need to iterate than a from-scratch coding solution. What matters is understanding all your options – and Apex makes the things you do have to code easy to integrate with your business.

While there is a great deal of functionality built into Salesforce, understanding Apex lets you go that extra mile by crafting additional workflows and solutions that help your organization succeed.

## **Apex Features**

Apex was designed for easy use by developers. For this reason, it has several key features that make it a good fit when building business logic into systems. Specifically, Apex is:

### Object-Oriented

Apex’s syntax mirrors popular programming languages like Java, making it easy to use for experienced coders. For example, variable declaration, loop syntax and conditional statements all work the same in Apex as they do in Java.

### Based in Data

Apex is data-focused and designed to execute multiple queries and DML (Data Manipulation Language) statements at the same time.

### Strongly Typed

Apex is a strongly typed language. It uses direct reference to objects, like [sObject,](https://developer.salesforce.com/docs/atlas.en-us.apexref.meta/apexref/apex_methods_system_sobject.htm?_ga=2.92354075.709245842.1745256261-1022981372.1742905468) and any invalid reference quickly fails if it is deleted or uses the wrong data type. Unlike some other languages, Apex is also generally case-insensitive.

### Run in a Multitenant Environment

Apex runs in a multitenant environment, and its runtime engine is designed to avoid problems like runaway code – where a query takes too long to resolve. A combination of easy-to-understand error messages and automatic failure for code that violates resource limits allows you to keep mistakes from monopolizing your processing power. When code would normally cause a problem because of an error or inefficiency related to resources, Apex helps you find and fix the issue before it’s running rampant.

### Built with Integrated Support

DML operations like INSERT, UPDATE, DELETE and also DML Exception handling all have built-in support with Apex. It also supports inline SOQL (Salesforce Object Query Language) and SOSL (Salesforce Object Search Language) query handling.

### Automatically Upgraded

Apex’s functionality is upgraded alongside Salesforce’s capabilities, with no need to update Apex itself separately. However, each data Class has an API version that isn’t updated each release, which prevents code from breaking every time there’s a new version of Salesforce.

### Easily Testable

Built-in support for unit test creation and execution helps you refine your projects, alongside test results that tell you how much code is covered. With easy access to quick, information-rich testing, it’s easier to feel confident in the code you create with Apex.

## **Data Types in Apex**

To write code with Apex, you use a number of different data types. These include:

### Primitives

Primitives are some of the simplest and most common data types in code. They include Integers, Long, Date, Datetime, String, ID, and Boolean values. Learn more about [Primative Data Types](https://developer.salesforce.com/docs/atlas.en-us.apexcode.meta/apexcode/langCon_apex_primitives.htm?_ga=2.100523800.709245842.1745256261-1022981372.1742905468)here.

### Collections

Made up of Lists, Sets, and Maps, collections can store multiple records.

### sObject

A database table that stores specific information about a business, like accounts, contacts, or leads.

### Enums

An abstract data type that defines a set of named constant values which don't have a numerical order.

### Classes, Objects and Interfaces

As in Java, you can create classes in Apex. A class is a template or blueprint from which objects are created. An object is an instance of a class. An interface is like a class in which none of the methods have been implemented—the method signatures are there, but the body of each method is empty.

Interfaces help you separate the implementation of a method (or a specific block of code) from its declaration, which lets you create different implementations of a method for different applications.

These different data types are the bedrock of coding in Apex, and understanding how to use them is what allows you to build applications for your own use cases.

## **Apex Logic**

Apex is designed to be familiar. If you’ve worked with other coding languages, you’ll probably see similarities in the syntax you’ll use with Apex. But if not, don’t worry! Once you understand the basic types of statements Apex coders use, you can apply them to accomplish your goals. Some important parts of that syntax to keep in mind are:

### Loop Statement

A loop statement is used for iterating over a list or iterating over a piece of code for a specified number of times. It allows you to run code until a condition is met. Loops use logic like **for** and **while** to run specific code in response to specific conditions set by the programmer. For example, here’s a looping statement that prints the integers 1 through 10 to the debug log:

Integer count = 1;

do {

System.**debug**(count);

count++;} while (count < 11);

Copy

### Flow Control Statement

Apex uses the **if** or **switch** statements for flow control. This allows you to stop or start executing code based on a condition. You could run code if a record is updated, for example. Here’s an example of a control statement that differentiates between medals based on the placement of an individual.

if (place == 1) {

medal\_color = **'gold'**;} else if (place == 2) {

medal\_color = **'silver'**;} else if (place == 3) {

medal\_color = **'bronze'**;} else {

medal\_color = null;}

DML Statements and SQL Queries

Queries fetch the data from Salesforce database. This usually means looking for data from specific fields or objects, such as the Account object.

DML—Data Manipulation Language—statements allow you to update, delete, or otherwise alter records.

Put together, these statements let you take full advantage of the data stored in your iteration of Salesforce and perform operations that keep your data up to date with ease. For example, here’s a code block that inserts an account record and associated contact record:

Account accObject = new Account();

accObject.Name = 'Test Account’;

accObject.Website =’www.salesforce.com’;

Insert accObject ;

List<Contact> contactList = new List<Contact>();

Contact con1 = new Contact();

con1.FirstName =’first’;

con1.LastName =’contact’;

con1.AccountId = accObject.Id;

contactList.**add**(con1);

Insert contactList;