Apex
è un linguaggio di programmazione fortemente tipizzato, orientato agli oggetti.
Utilizza la sintassi simile a Java e consente agli sviluppatori di aggiungere la logica aziendale alla maggior parte degli eventi di sistema, inclusi i click sui pulsanti, gli aggiornamenti dei record correlati e le pagine Visualforce.
Apex è un linguaggio case insensitive.
•È un linguaggio fortemente tipato, ciò significa che ogni volta che si dichiara una variabile, occorre dichiararla con il suo tipo.
L'inizializzazione di una variabile in Apex è composta da 4 step (i primi due per la dichiarazione gli altri per inizializzare la variabile):
•Specificare il Tipo di dato: tipo della variabile che si sta dichiarando.
•Specificare il Nome della variabile
•Utilizzare il carattere di assegnamento '='.
•Specificare il valore da salvare.
I nomi delle variabili non possono avere:
•Spazi Bianchi
•Caratteri speciali (+,-, &, *, ...)
•Non possono essere parole riservate come:string,integer, …
Bisogna dare sempre un nome con un significato alle variabili e usare la camelCase annotation
Best-practice per dichiarare i nomi delle variabili usando la camelCase annotation:
•Iniziare con la lettera minuscola e utilizzare la lettera maiuscola per tutte le parole che seguono la prima
La dichiarazione di un insieme di variabili mette da parte un pezzo di memoria utilizzato per archiviare i dati solo durante l'esecuzione del programma.
Se si dichiara una variabile e non la si inizializza con un valore, il valore sarà null.
Null indica l'assenza di un valore o di un dato.
È possibile assegnare null a qualsiasi variabile con un tipo primitivo.
•Variabili di Classe
•Variabili di istanza
•Variabili statiche
Variabili locali
•Interne ai metodi (come parametri o dentro al metodo)
•Dentro ogni blocco di codice come ad esempio il ciclo for,if,etc...
Le variabili possono essere definite in qualsiasi punto in un blocco, e possono essere utilizzate da quel punto in poi.
I figli possono accedere alle variabili, e non possono definire un'altra variabile con lo stesso nome che è stato già usato in un blocco padre.
I blocchi paralleli non hanno accesso alla variabile, e possono riusare il nome della variabile.
Una costante è una variabile il cui valore non cambia dopo che è stato inizializzato una volta.
Le costanti possono essere definite usando la parola chiave final.
La parola chiave final indica che la variabile può essere inizializzata al massimo una volta:
•Stesso nella dichiarazione
•Con un metodo statico se la costante è definita in una classe
Un'espressione è una componente essenziale in ogni programma.
Un'espressione è un costrutto composto da:
•Variabili
•Operatori
•Invocazioni a metodo che valutano un valore
InApex, un'espressione è sempre una di questiseguenti tipi:
•Un'espressione letterale:1 + 1
•Una nuovosObject,Apexobject, list, set, omap:
    •Case myCase = New Case();

- •MyClass myClassInstance = New MyClass ();
- •List <Account> myAcc = New List<Account>();
- •Set <String> myStr = New Set<String>();
- •Map<Integer,String> myMap = New Map<Integer,String>();

•Tutti i valori presenti a sinistra dell'operatore di assegnazione:

- •i
- •myList[3]
- •myContact.name

•Ogni riferimento al campo disObjectche non è un L-value
•Una query SOQL o SOSL circondata da parentesi quadrate
•Un'invocazione a metodo istanza o statico: System.assert(true)
Apex utilizza tipi di dati, variabili e costrutti del linguaggio correlati come enumerazioni, costanti, espressioni, operatori e istruzioni di assegnazione.
In Apex, tutte le variabili e le espressioni hanno un tipo di dato, come sObject, primitive o enum
Un primitivo, può essere un integer, un double, un long, un date, un datetime, un string, un ID o un boolean.
Un oggetto sObject , può essere un sObject generico o un sObject specifico , ad esempio un account, un contatto oMyCustomObject__c
Una collezione, tra cui:
•Un elenco (o array) di primitive, oggetti sObjects,oggetti definiti dall'utente, oggetti creati da classi Apex o collezioni
•Un set di primitive
•Una mappa da primitiva a primitiva, sObject o collezione
•Un elenco di valori tipizzato, noto anche come enum
•Oggetti creati da classi Apex definite dall'utente
•Oggetti creati dalle classi Apex fornite dal sistema
•Null(per la costante nulla, che può essere assegnata a qualsiasi variabile).
I metodi possono restituire valori di uno qualsiasi dei tipi elencati, o non restituire alcun valore ed essere di tipo Void.
STRING
Usato per contenere qualsiasi carattere: lettere,numeri ... ecc.
Dichiarata con una sola virgoletta,non doppia virgoletta!
Strings = 'Valore stringa';
Usa sequenze diescapeper indicare caratteri speciali, come\' per una virgoletta singola,\r per il ritorno a capo,\n per l'avanzamento di riga ... ecc.
A differenza di Java, le Stringhe in Apex supportano l'utilizzo degli operatori di confronto: ==, !=, <, <=, >,>=.
INTEGER
Utilizzato per contenere un numero senza posizione decimale, a 32 bit.
Integer myInteger= 365;
LONG
Utilizzato per contenere un numero senza posizione decimale a 64-bit.

```
Long l = 2147483648L;
```

DECIMAL
Utilizzato per contenere un numero con 1 o più posizioni decimali, a 32 bit.
Decimal Mydecimal= 3.14;
DOUBLE
Utilizzato per contenere un numero con 1 o più posizioni decimali, è a 64-bi
Utilizzato quando è necessario un intervallo più ampio del decimale.

```
Double d = 3.14159;
```

DATE
•Un valore che indica un giorno particolare.

•I valori della data non contengono informazioni sul tempo.
•I valori della data devono sempre essere creati con un metodo statico di sistema.
•È possibile aggiungere o sottrarre un valore intero da un valore Date, restituendo un valore Date
DATETIME
Un valore che indica un particolare giorno + ora,come un timestamp.I valori di Datetime devono sempre essere creati con un metodo statico di sistema.
È possibile aggiungere o sottrarre un valore Integer o Double da un valore Datetime, restituendo un valore Datetime.
TIME
Un valore che indica una particolare ora.
I valori di tipo Time devono sempre essere creati con un metodo statico di sistema.
Time myTime = Time.newInstance(18, 30, 2, 20);
BOOLEAN
Un valore può essere vero (True) o falso (False) o nullo (null).
Boolean isWinner = true;
ID
Qualsiasi identificatore di record valido di 18caratteri.
Ogni carattere può essere uno dei 62 valori possibili:26 lettere minuscole, 26 lettere maiuscole, 10 cifre.
ID id='00300000003T2PGAA0';
Se si imposta ID su un valore di 15 caratteri,Apex converte il valore nella sua rappresentazione di 18caratteri.
Tutti i valori ID non validi vengono rifiutati con un'eccezione di runtime
SOBJECT
Apex supporta i tipi di dati primitivi (come Integer), le classi personalizzate definite dall'utente, il tipo generico sObjecto un tipo specifico sObject(come Account).
Tutti i tipi di dati Apex ereditano da Object.
Per esempio: Object obj= 10;
// Assegnare un oggetto a una variabile di tipo intero.
Integer i = (Integer)obj;
System.assertEquals(10, i);
Collections
Le Collection in Apex possono essere:
•Lists.
•Sets.
•Maps
LIST
Una lista è una raccolta ordinata e indicizzata di elementi non unici che si distinguono per i loro indici che partono da 0.
Le liste sono equivalenti agli Array.
Gli elementi dell'elenco possono essere di qualsiasi tipo di dati: tipi primitivi,sObject, tipi definiti dall'utente e tipi Apex incorporati.
Possono contenere anche alcune collezioni e possono essere innestate una nell'altra e diventare multidimensionali.
Per accedere agli elementi in un elenco, utilizzare i metodi di List forniti da Apex.
Quando si utilizzano elenchi monodimensionali di primitive o oggetti, è possibile utilizzare anche la notazione di array più tradizionale per dichiarare e fare riferimento a elementi di elenco.
Ad esempio, è possibile dichiarare un elenco unidimensionale di primitive o oggetti seguendo il nome del tipo di dati con i caratteri:
String[] colors = new List <String> ();
Queste due affermazioni sono equivalenti alla precedente:
List <String> colors = newString[1]

;String[] colors = newString[1];
Per fare riferimento a un elemento di un elenco unidimensionale, è possibile anche seguire il nome dell'elenco con la posizione dell'indice dell'elemento nelle parentesi quadre.

colors [0] = 'Green';

È possibile ordinare gli elementi dell'elenco e l'ordinamento dipende dal tipo di dati degli elementi.
Utilizzando il metodo nome_lista.sort(), è possibile ordinare gli elementi in un elenco.
L'ordinamento è in ordine ascendente per elementi di tipi di dati primitivi, come le stringhe.

SET

Un set è una raccolta non ordinata di elementi che non contengono duplicati.
Non si utilizzano gli indici.
Gli elementi set possono essere di qualsiasi tipo di dati: tipi primitivi,sObject, tipi definiti dall'utente e tipi Apex built-in.
I set possono contenere collezioni che possono essere annidate l'una nell'altra.
Ad esempio, è possibile avere una serie di elenchi di insiemi di numeri interi.
Un set può contenere fino a quattro livelli di raccolte nidificate al suo interno, ovvero fino a un massimo di cinque livelli.
Per dichiarare un set, utilizzare la parola chiave Set seguita dal nome del tipo di dati primitivi all'interno di caratteri <> .
Set<String>myStringSet= new Set<String>();
Per accedere agli elementi di un set, utilizzare i metodi di sistema forniti da Apex.

MAP

Una map è una raccolta di coppie chiave-valore in cui ogni chiave univoca si associa a un singolo valore non unico.
Le chiavi sono non ordinate e uniche (equivalenti al Set).
I valori sono ordinati e non unici (equivalenti a List).
La chiavi di tipo stringa sono case sensitive.
Chiavi e valori possono essere qualsiasi tipo di dati:tipi primitivi, raccolte, oggetti sObject, tipi definiti dall'utente e tipi Apex incorporati.
Le chiavi e i valori della mappa possono contenere qualsiasi raccolta e possono essere nidificate e diventare multidimensionali.
Ad esempio, è possibile avere una mappa di numeri interi per le mappe, che, a loro volta, associano le stringhe agli elenchi.
Le chiavi della mappa possono contenere solo fino a quattro livelli di raccolte nidificate.
Per dichiarare una mappa, utilizzare la parola chiave Map seguita dai tipi di dati della chiave e il valore compreso tra <>.
Come con le liste e i set, è possibile popolare le coppie chiave-valore della mappa quando la mappa viene dichiarata utilizzando la sintassi della parentesi graffa ({}).
All'interno delle parentesi graffe, specificare prima la chiave, quindi specificare il valore per quella chiave usando =>.