1. Cos'è Swift?

Swift è un linguaggio di programmazione sviluppato da Apple per semplificare lo sviluppo di app. È open-source e combina prestazioni elevate con una sintassi intuitiva, rendendolo accessibile sia ai principianti sia agli sviluppatori esperti.

Installazione di Swift

Per iniziare con Swift, hai bisogno di **Xcode**, l'ambiente di sviluppo integrato (IDE) di Apple, che include tutti gli strumenti necessari per scrivere, testare e eseguire codice Swift.

1. **Scarica Xcode**: disponibile gratuitamente su Mac App Store.
2. **Playground di Xcode**: permette di testare codice Swift rapidamente senza dover creare un progetto completo.

2. Primi Passi in Swift

Sintassi Base

Per dichiarare variabili e costanti si usano le parole chiave `var` e `let`. Swift è un linguaggio tipizzato, ma può determinare automaticamente il tipo di variabile se il valore è assegnato direttamente.

swift
Copia codice
```
var nome = "Mario"       // Variabile di tipo String
let età = 30             // Costante di tipo Int
```

> Nota: Usare `let` per le costanti che non cambiano valore durante l'esecuzione del programma.

Stampa di Output

Usiamo `print()` per mostrare output in Swift.

swift
Copia codice
```
print("Ciao, mondo!")
```

Commenti

Swift supporta commenti su singola linea `//` e multi-linea `/* ... */`.

3. Tipi di Dato

Swift supporta diversi tipi di dato principali:

- **Int**: numeri interi.
- **Double e Float**: numeri decimali.
- **String**: testo.
- **Bool**: valori booleani (vero o falso).

Esempio:

swift
Copia codice
```swift
let pi = 3.1415           // Double
let isOpen = true         // Bool
let message = "Benvenuto" // String
```

4. Operatori

Swift include operatori aritmetici (`+`, `-`, `*`, `/`, `%`), di confronto (`==`, `!=`, `<`, `>`, `<=`, `>=`) e logici (`&&`, `||`, `!`).

Esempio:

swift
Copia codice
```swift
let a = 5
let b = 10
let somma = a + b         // 15
let isEqual = (a == b)    // false
```

5. Strutture di Controllo

Condizionali

Le strutture `if`, `else if` e `else` permettono di eseguire codice in base a condizioni.

swift
Copia codice
```swift
let età = 18
if età >= 18 {
    print("Sei maggiorenne")
} else {
```

```
    print("Sei minorenne")
}
```

Cicli

Swift offre i cicli `for`, `while`, e `repeat-while`.

swift
Copia codice
```
// Ciclo for
for numero in 1...5 {
    print(numero)
}

// Ciclo while
var count = 1
while count <= 5 {
    print(count)
    count += 1
}

// Ciclo repeat-while
repeat {
    print(count)
    count -= 1
} while count > 0
```

6. Funzioni

Le funzioni in Swift sono dichiarate con `func`, seguite dal nome della funzione, dai parametri e dal tipo di ritorno.

swift
Copia codice
```
func saluta(nome: String) -> String {
    return "Ciao, \(nome)!"
}

print(saluta(nome: "Mario"))
```

I parametri possono avere valori predefiniti e nomi interni (per chiarezza) ed esterni (per migliorare la leggibilità).

7. Array e Dizionari

Array

Gli array memorizzano elenchi ordinati di elementi dello stesso tipo.

swift
Copia codice
```swift
var numeri = [1, 2, 3, 4]
numeri.append(5)           // Aggiunge 5 alla fine dell'array
print(numeri[2])           // Stampa 3
```

Dizionari

I dizionari memorizzano coppie chiave-valore.

swift
Copia codice
```swift
var capitale = ["Italia": "Roma", "Francia": "Parigi"]
capitale["Spagna"] = "Madrid"
print(capitale["Italia"]!) // Roma
```

8. Programmazione Orientata agli Oggetti (OOP)

Swift è un linguaggio orientato agli oggetti e supporta classi, strutture, ereditarietà, e protocolli.

Classi e Oggetti

Le classi definiscono il blueprint per oggetti, e possono avere proprietà e metodi.

swift
Copia codice
```swift
class Persona {
    var nome: String
    var età: Int

    init(nome: String, età: Int) {
        self.nome = nome
        self.età = età
```

```
    }

    func saluta() -> String {
        return "Ciao, \(nome)!"
    }
}

let mario = Persona(nome: "Mario", età: 30)
print(mario.saluta())
```

Strutture (Struct)

Le `struct` sono simili alle classi, ma non supportano l'ereditarietà.

swift
Copia codice

```
struct Punto {
    var x: Int
    var y: Int

    func distanza() -> Double {
        return sqrt(Double(x * x + y * y))
    }
}

let punto = Punto(x: 3, y: 4)
print(punto.distanza()) // 5.0
```

Protocolli

I protocolli definiscono metodi e proprietà che una classe o una struttura devono implementare.

swift
Copia codice

```
protocol Descrivibile {
    func descrivi() -> String
}

class Automobile: Descrivibile {
    var marca: String
```

```
    init(marca: String) {
        self.marca = marca
    }

    func descrivi() -> String {
        return "Automobile marca \(marca)"
    }
}

let auto = Automobile(marca: "Fiat")
print(auto.descrivi())
```

9. Opzionali

In Swift, una variabile può essere **opzionale** (cioè può avere o non avere un valore) dichiarandola con `?`. Usando `!` si può forzare l'unwrapping di un opzionale per accedere al suo valore.

swift
Copia codice

```
var nome: String? = "Mario"
print(nome!)           // Mario

// Uso dell'optional binding per evitare forzature
if let nomeNonNullo = nome {
    print(nomeNonNullo)
} else {
    print("Nome è nullo")
}
```

10. Gestione degli Errori

Swift offre una gestione degli errori robusta con `try`, `catch`, e `throws`.

swift
Copia codice

```
enum ErroreCalcolo: Error {
    case divisionePerZero
}

func dividi(_ a: Int, per b: Int) throws -> Int {
```

```
    guard b != 0 else {
        throw ErroreCalcolo.divisionePerZero
    }
    return a / b
}

do {
    let risultato = try dividi(10, per: 0)
    print(risultato)
} catch ErroreCalcolo.divisionePerZero {
    print("Errore: divisione per zero!")
}
```

11. SwiftUI e Interfaccia Utente

Swift include **SwiftUI**, un framework che permette di creare interfacce utente in modo dichiarativo.

swift
Copia codice

```
import SwiftUI

struct ContenutoView: View {
    var body: some View {
        VStack {
            Text("Ciao, mondo!")
                .font(.largeTitle)
                .padding()
            Button(action: {
                print("Bottone premuto")
            }) {
                Text("Premi qui")
            }
        }
    }
}

@main
struct AppEsempio: App {
    var body: some Scene {
        WindowGroup {
```

```
            ContenutoView()
        }
    }
}
```

Questa guida copre le basi di Swift, un linguaggio versatile e moderno. Esplorare SwiftUI e utilizzare strumenti come **Xcode** ti permetterà di creare applicazioni complete per i sistemi Apple. Buon apprendimento!