```
<= // comparison operator
>= // comparison operator

// --- LOGICAL OPERATOR ---

&& // and
|| // or
! // not

// --- NIL-COALESCING OPERATOR ---

?? // nil-coalescing operator that can be called on an Optional datatype, where a default user-designated value is assigned to an Optional variable if its value is nil
```

Control structures

```
// ----- CONTROL STRUCTURE -----

// --- CONDITIONALS ---

// IF ELSE IF ELSE

if age >= 18 {
    print("Adult")
} else if age >= 13 {
    print("Teenager")
} else {
    print("Child")
}

// SWITCH CASE DEFAULT
    // provides some degree of basic pattern-matching as an alternative to if else if else constructs
    // default => the fall-through default case if all other predicate case conditions are unmet

let color = "red"
switch color {
case "red":
    print("The color is red")
case "blue":
    print("The color is blue")
default:
    print("Unknown color")
}

// --- LOOPS ---

// FOR IN LOOP
    // Swift's for in loops do allow for iteration over an iterable data structure
    // Swift does not provide for conventional C-style for loops, but the C-style output to the stdout can be replicated using the range operator
    // .. => range operator dynamically generates an iterable data structure from the given minimum to maximum range

var numbers: [Int] = [1, 2, 3, 4, 5]
for number in numbers {
    print(number)
}

for i in 0..<10 {
    print(i)
}

// WHILE LOOP

var counter = 0
while counter < 5 {
```