Flow Control Structures

Overview

- Useful operators
- What is flow control?
- Conditional statements
- Loops

What is flow control?

- Repeat
- Skip or execute
- Exit early

Comparison operators

```
1 == 1   // true because 1 is equal to 1
2 != 1   // true because 2 isn't equal to 1
2 > 1    // true because 2 is greater than 1
1 < 2    // true because 1 is less than 2
1 >= 1   // true because 1 is greater than or equal to 1
2 <= 1   // false because 2 isn't less than or equal to 1
```

Logical operators

```
let value: Bool = false

!value           //this is true
value && false   // this is false
value || false   // this is false
```

Range operators

```
// Range operators
let closedRange = -2...5
let halfClosedRange = -2..<5
let oneSidedRangeLeft = ...5
let oneSidedRangeRight = 5...
let halfOpenOneSided = ..<5
```

Conditional statements

- If-else
- Switch
- Guard
- Ternary conditional operator

IF-ELSE

- Executes code or skips based on conditions
- Else provides fallback
- Else if allows chaining
- Only one block is executed
- Supports pattern matching

GUARD

- Acts like a tollbooth
- Needs a fallback
- Supports pattern matching

Ternary conditional operator

- Shorthand for IF-ELSE
- Used for value assignment or simple function calls
- Ideal for small, concise comparisons

SWITCH CASE

- Match values to patterns
- Exhaustive
- Executes first matching case without fallthrough
- Supports where clauses for additional filtering

Loops

- Repeat tasks

FOR-IN Loops

- Iterate over sequence
- Supports where clause
- Supports pattern matching
- Supports where clauses for additional filtering

WHILE

- Repeats while condition is true
- Evaluates condition first, then executes

REPEAT-WHILE

- Repeats while condition is true
- Executes first, then evaluates condition