Conditionals

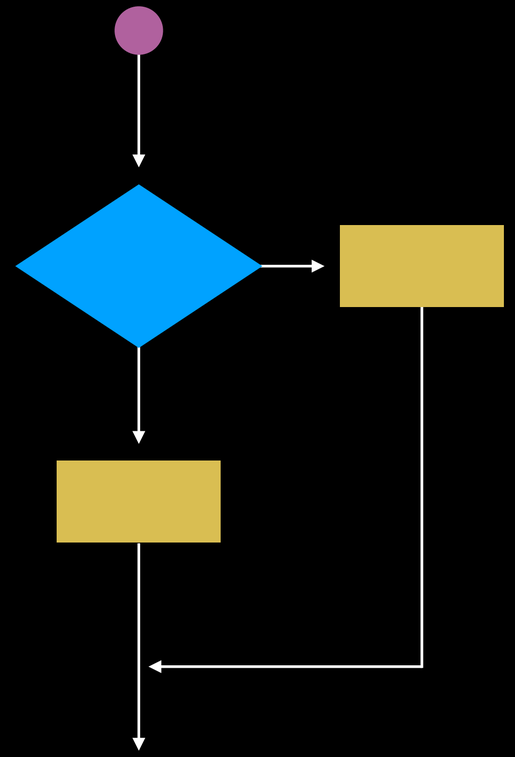
Conditionals

Execute different code depending on a "question"

Evaluate a boolean expression

Equal, different, contains, greater than…

if, **switch** and **ternary**



Boolean operators

==	Equal to	User password, points in a game
!=	Not equal to	Variable **is not** empty
<	Less than	Condition related to time
<=	Less than or equal to	Balance on saving account
>	Greater than	Condition related to temperature
>=	Greater than or equal to	Enabling user access due age restrictions
Boolean	Elements that return boolean values	contains(), isAnimating(), insert()

if

```swift
// Declaring elements
var userAge: Int = 16
let ageToDrive: Int = 18

// Checking a boolean expression
if userAge < ageToDrive {
    // Block executed if the expression is true
    print("Only bicycle ...")
} else {
    // Block executed if the expression is false
    print("I can LEGALLY drive!")
}
```

```swift
// Declaring elements
var userAge: Int = 16
let ageToDrive: Int = 18

// Checking a boolean expression
if userAge < ageToDrive {
    // Block executed if the expression is true
    print("Only bicycle ...")
} else {
    // Block executed if the expression is false
    print("I can LEGALLY drive!")
}
```

```swift
// Declaring elements
var userAge: Int = 16
let ageToDrive: Int = 18

// Checking a boolean expression
if userAge < ageToDrive {
    // Block executed if the expression is true
    print("Only bicycle ...")
} else {
    // Block executed if the expression is false
    print("I can LEGALLY drive!")
}
```

```swift
// Declaring elements
var userAge: Int = 16
let ageToDrive: Int = 18

// Checking a boolean expression
if userAge < ageToDrive {
    // Block executed if the expression is true
    print("Only bicycle ...")
} else {
    // Block executed if the expression is false
    print("I can LEGALLY drive!")
}
```

```swift
// Declaring elements
var userAge: Int = 16
let ageToDrive: Int = 18

// Checking a boolean expression
if userAge < ageToDrive {
    // Block executed if the expression is true
    print("Only bicycle ...")
} else {
    // Block executed if the expression is false
    print("I can LEGALLY drive!")
}
```

```swift
// Declaring elements
var userAge: Int = 16
let ageToDrive: Int = 18

// Checking a boolean expression
if userAge < ageToDrive {
    // Block executed if the expression is true
    print("Only bicycle ...")
} else {
    // Block executed if the expression is false
    print("I can LEGALLY drive!")
}
```

```swift
// Declaring elements
var userAge: Int = 16
let ageToDrive: Int = 18

// Checking a boolean expression
if userAge < ageToDrive {
    // Block executed if the expression is true
    print("Only bicycle ...")
} else {
    // Block executed if the expression is false
    print("I can LEGALLY drive!")
}
```

```swift
// Declaring elements
var userAge: Int = 16
let ageToDrive: Int = 18

// Checking a boolean expression
if userAge < ageToDrive {
    // Block executed if the expression is true
    print("Only bicycle ...")                    Only bicycle ...
} else {
    // Block executed if the expression is false
    print("I can LEGALLY drive!")
}
```

value >= 0 ?

true color green

false color red

Hands on

switch

```swift
// Declaring constant
let initials: String = "DA"

// Checking possible values for the expression
switch initials {
    case "DS":
        print("Danilo Santana")
    case "GD":
        print("Gilles Deltel")
    case "MN":
        print("Mark Nichols")
    default:
        print("None of the above")
}
```

```swift
// Declaring constant
let initials: String = "DA"

// Checking possible values for the expression
switch initials {
    case "DS":
        print("Danilo Santana")
    case "GD":
        print("Gilles Deltel")
    case "MN":
        print("Mark Nichols")
    default:
        print("None of the above")
}
```

```swift
// Declaring constant
let initials: String = "DA"

// Checking possible values for the expression
switch initials {
    case "DS":
        print("Danilo Santana")
    case "GD":
        print("Gilles Deltel")
    case "MN":
        print("Mark Nichols")
    default:
        print("None of the above")
}
```

```swift
// Declaring constant
let initials: String = "DA"

// Checking possible values for the expression
switch initials {
    case "DS":
        print("Danilo Santana")
    case "GD":
        print("Gilles Deltel")
    case "MN":
        print("Mark Nichols")
    default:
        print("None of the above")
}
```

```swift
// Declaring constant
let initials: String = "DA"

// Checking possible values for the expression
switch initials {
    case "DS":
        print("Danilo Santana")
    case "GD":
        print("Gilles Deltel")
    case "MN":
        print("Mark Nichols")
    default:
        print("None of the above")
}
```

```swift
// Declaring constant
let initials: String = "DA"

// Checking possible values for the expression
switch initials {
    case "DS":
        print("Danilo Santana")
    case "GD":
        print("Gilles Deltel")
    case "MN":
        print("Mark Nichols")
    default:
        print("None of the above")
}
```

```swift
// Declaring constant
let initials: String = "DA"

// Checking possible values for the expression
switch initials {
    case "DS":
        print("Danilo Santana")
    case "GD":
        print("Gilles Deltel")
    case "MN":
        print("Mark Nichols")
    default:
        print("None of the above")
}
```

```swift
// Declaring constant
let initials: String = "DA"

// Checking possible values for the expression
switch initials {
    case "DS":
        print("Danilo Santana")
    case "GD":
        print("Gilles Deltel")
    case "MN":
        print("Mark Nichols")
    default:
        print("None of the above")
}
```

```swift
// Declaring constant
let initials: String = "DA"

// Checking possible values for the expression
switch initials {
    case "DS":
        print("Danilo Santana")
    case "GD":
        print("Gilles Deltel")
    case "MN":
        print("Mark Nichols")
    default:
        print("None of the above”)          None of the above
}
```

Hands on

ternary

```swift
// Declaring elements
var userAge: Int = 16
let ageToDrive: Int = 18

// Checking a boolean expression
if userAge < ageToDrive {
    // Block executed if the expression is true
    print("Only bicycle ...")
} else {
    // Block executed if the expression is false
    print("I can LEGALLY drive!")
}
```

```swift
// Declaring elements
var userAge: Int = 16
let ageToDrive: Int = 18

// Checking a boolean expression
if userAge < ageToDrive {
    // Block executed if the expression is true
    print("Only bicycle ...")
} else {
    // Block executed if the expression is false
    print("I can LEGALLY drive!")
}

print( userAge < ageToDrive ? "Only bicycle ..." : "I can LEGALLY drive!" )
```

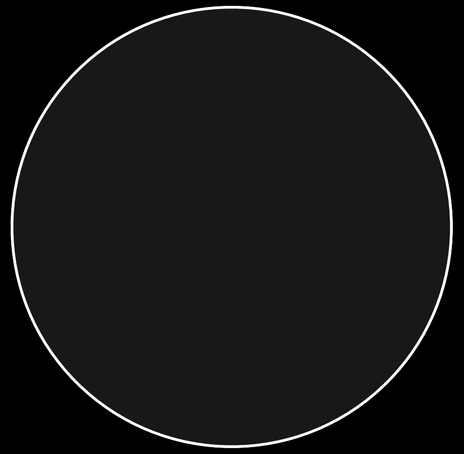
```swift
// Declaring elements
var userAge: Int = 16
let ageToDrive: Int = 18

// Checking a boolean expression
if userAge < ageToDrive {
    // Block executed if the expression is true
    print("Only bicycle ...")
} else {
    // Block executed if the expression is false
    print("I can LEGALLY drive!")
}

print( userAge < ageToDrive ? "Only bicycle ..." : "I can LEGALLY drive!" )
```

```swift
// Declaring elements
var userAge: Int = 16
let ageToDrive: Int = 18

// Checking a boolean expression
if userAge < ageToDrive {
    // Block executed if the expression is true
    print("Only bicycle ...")
} else {
    // Block executed if the expression is false
    print("I can LEGALLY drive!")
}

print( userAge < ageToDrive ? "Only bicycle ..." : "I can LEGALLY drive!" )
```

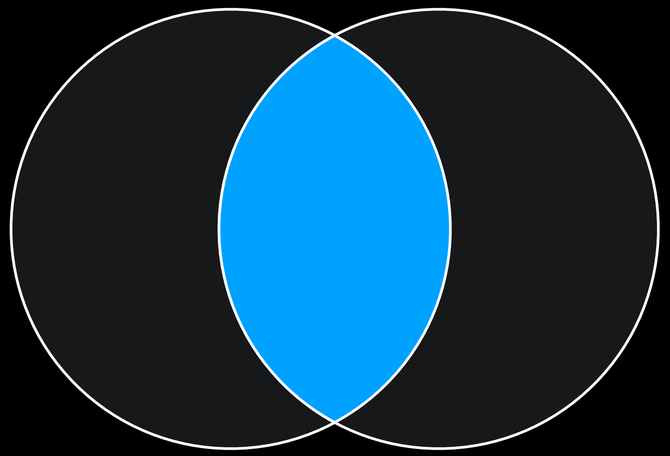
```swift
// Declaring elements
var userAge: Int = 16
let ageToDrive: Int = 18

// Checking a boolean expression
if userAge < ageToDrive {
    // Block executed if the expression is true
    print("Only bicycle ...")
} else {
    // Block executed if the expression is false
    print("I can LEGALLY drive!")
}

print( userAge < ageToDrive ? "Only bicycle ..." : "I can LEGALLY drive!" )
```

```swift
// Declaring elements
var userAge: Int = 16
let ageToDrive: Int = 18

// Checking a boolean expression
if userAge < ageToDrive {
    // Block executed if the expression is true
    print("Only bicycle ...")
} else {
    // Block executed if the expression is false
    print("I can LEGALLY drive!")
}

print( userAge < ageToDrive ? "Only bicycle ..." : "I can LEGALLY drive!" )
```

Logical operators



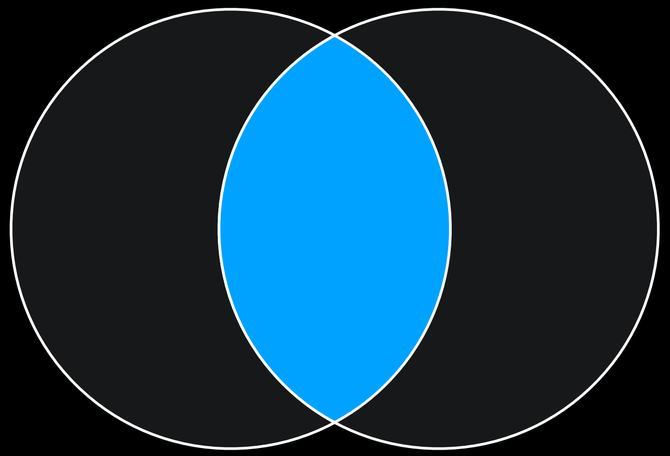
AND

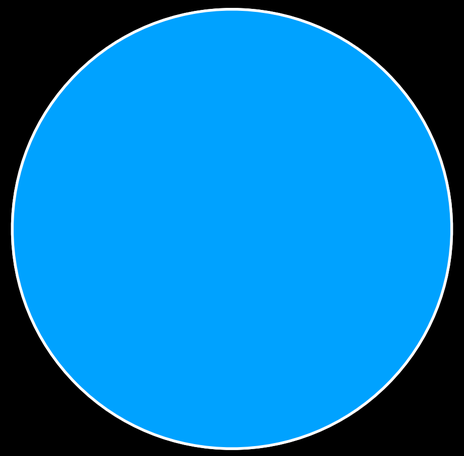


Both terms

username is true
AND
password is true

AND

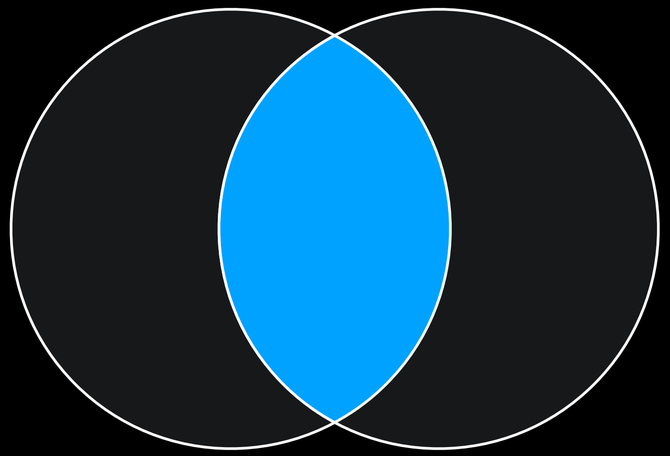




Both terms

username is true
AND
password is true

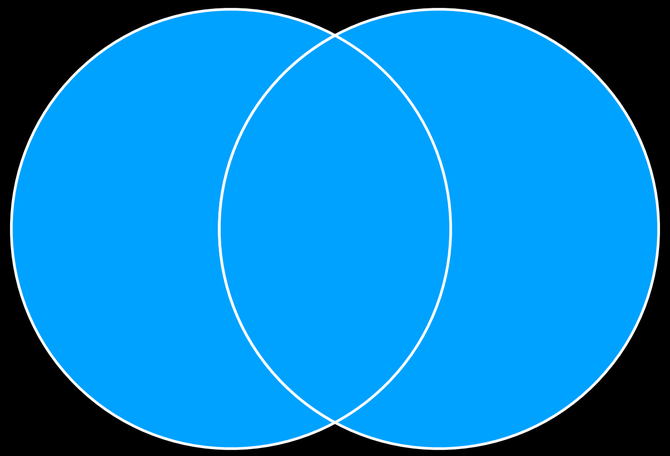
AND



Both terms

username is true
AND
password is true

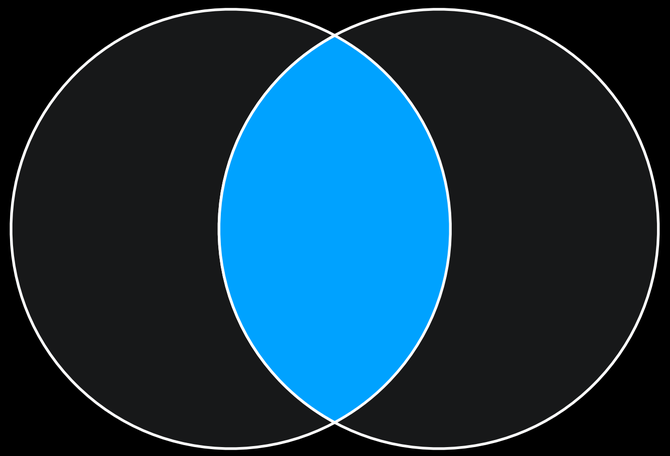
OR



Either terms

moves is 0
OR
time is 0

AND



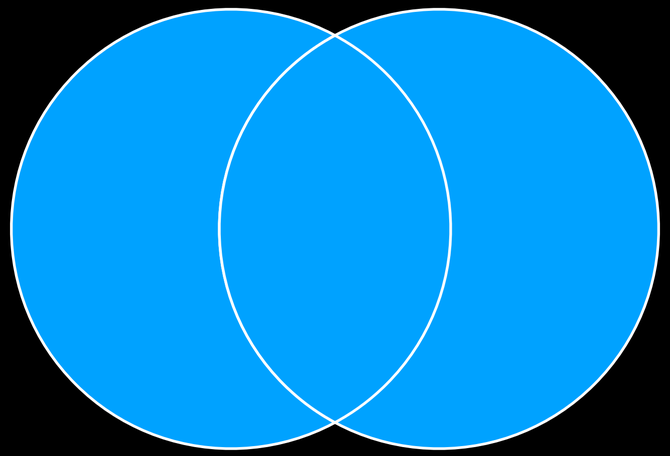
Both terms

username is true

AND

password is true

OR

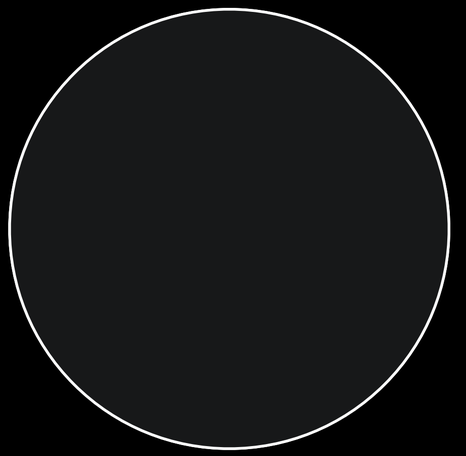


Either terms

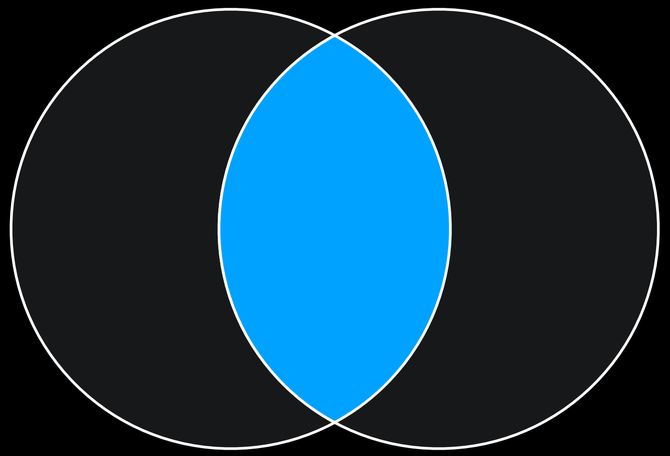
moves is 0

OR

time is 0



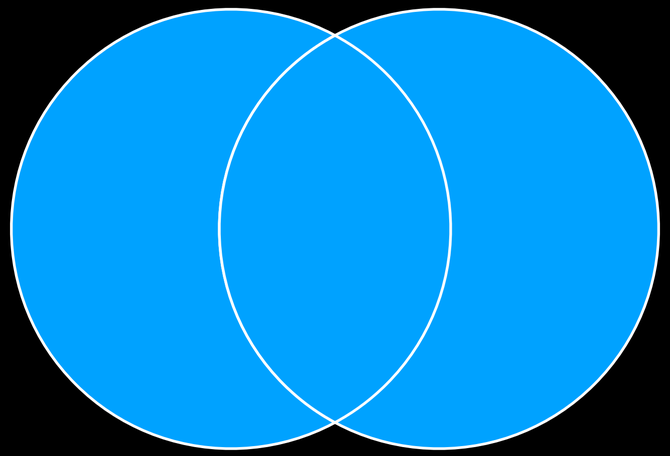
AND



Both terms

username is true
AND
password is true

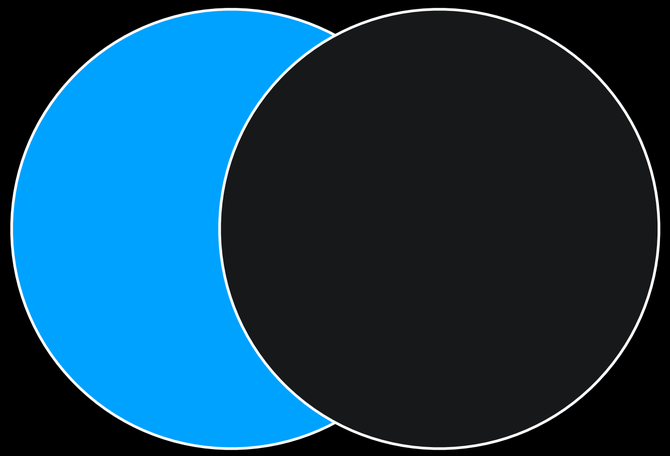
OR



Either terms

moves is 0
OR
time is 0

NOT



One term

message is
NOT
Danilo

Hands on