

# Shaping the Path

---

CONTROLLING THE FLOW OF OUR CODE



**Simon Allardice**

STAFF AUTHOR, PLURALSIGHT

@allardice [www.pluralsight.com](http://www.pluralsight.com)

**if / else**

```
// If

var score = 100

if (score > 10) {
    print("It's greater than 10")
} else {
    print("It's not greater than 10")
}
```

```
// If
```

```
var score = 100
```

```
if (score > 10) {  
    print("It's greater than 10")  
} else {  
    print("It's not greater than 10")  
}
```

// If

var score = 100

```
if (score > 10) {  
    print("It's greater than 10")  
} else {  
    print("It's not greater than 10")  
}
```

Parentheses aren't required

// If

var score = 100

```
if score > 10 {  
    print("It's greater than 10")  
} else {  
    print("It's not greater than 10")  
}
```

Parentheses aren't required

// If

var score = 100

```
if score > 10 {  
    print("It's greater than 10")  
} else {  
    print("It's not greater than 10")  
}
```

Parentheses aren't required

```
// If

var score = 100

if score > 10 {
    print("It's greater than 10")
} else {
    print("It's not greater than 10")
}
```



```
// If

var score = 100

if score > 10 {
    print("It's greater than 10")
} else {
    print("It's not greater than 10")
}
```

// If

var score = 100

```
if score > 10 {  
    print("It's greater than 10")  
} else {  
    print("It's not greater than 10")  
}
```

Braces are **always required**

# C-Style Single-line "if" Statements

# C-Style Single-line "if" Statements

```
if (score > 10)  
    printf("It's greater than 10");
```

# C-Style Single-line "if" Statements

```
// This is not allowed in Swift  
if (score > 10)  
    printf("It's greater than 10");
```

```
if somecondition {  
    print("It's true")  
} else {  
    print("It's false")  
}
```



```
if somecondition {  
    print("It's true")  
} else {  
    print("It's false")  
}
```

Conditions must be **true** or **false**

```
if somecondition {  
    print("It's true")  
} else {  
    print("It's false")  
}
```

Conditions must be **true** or **false**



```
if highScore > 10000 {  
    print("It's true")  
} else {  
    print("It's false")  
}
```

Conditions must be **true** or **false**

Conditions must be **true** or **false**

```
if highScore > 10000 {  
    print("It's true")  
} else {  
    print("It's false")  
}
```

```
if carModel == "Tesla" { ...
```



Conditions must be **true** or **false**

```
if highScore > 10000 {  
    print("It's true")  
} else {  
    print("It's false")  
}
```

```
if carModel == "Tesla" { ...
```

```
if myOptionalInt != nil { ...
```



Conditions must be **true** or **false**

```
if highScore > 10000 {  
    print("It's true")  
} else {  
    print("It's false")  
}
```

```
if carModel == "Tesla" { ...
```

```
if myOptionalInt != nil { ...
```

```
if userLoggedIn { ...
```

Conditions must be **true** or **false**

```
if highScore > 10000 {  
    print("It's true")  
} else {  
    print("It's false")  
}
```

```
if carModel == "Tesla" { ...
```

```
if myOptionalInt != nil { ...
```

```
if userLoggedIn { ...
```

```
if bonusEnabled { ...
```

# Conditions must be **true** or **false**

```
if highScore > 10000 {  
    print("It's true")  
} else {  
    print("It's false")  
}
```

```
if carModel == "Tesla" { ...
```

```
if myOptionalInt != nil { ...
```

```
if userLoggedIn { ...
```

```
if bonusEnabled { ...
```

```
if a == b { ...
```

```
if a != b { ...
```

```
if a > b { ...
```

```
if a < b { ...
```

```
if a >= b { ...
```

```
if a <= b { ...
```

# Conditions With Multiple Elements

# Conditions With Multiple Elements

// logical AND

```
if  a == b  &&  c != d  { ...
```



# Conditions With Multiple Elements

// logical AND

```
if a == b && c != d { ...
```

// logical OR

```
if a < b || a > c { ...
```

# Conditions With Multiple Elements

// logical AND

```
if a == b && c != d { ...
```

// logical OR

```
if a < b || a > c { ...
```

// use parens with complex situations

```
if score >= highScore && bonus == 0 ||  
   score * bonus - penalty >= highScore { ...
```

# Conditions With Multiple Elements

// logical AND

```
if a == b && c != d { ...
```

// logical OR

```
if a < b || a > c { ...
```

// use parens with complex situations

```
if (score >= highScore && bonus == 0) ||  
((score * bonus) - penalty) >= highScore { ...
```

# Basic Switch Statement in Swift

# Basic Switch Statement in Swift

```
var abbrev = "MB" // ... or "Kb" or "GB" or "TB", etc.
```

# Basic Switch Statement in Swift


```
var abbrev = "MB" // ... or "Kb" or "GB" or "TB", etc.  
  
// later ...  
switch abbrev {  
case "kB":  
    print("kilobyte")  
case "MB":  
    print("megabyte")  
case "GB":  
    print("gigabyte")  
case "TB":  
    print("terabyte")  
case "PB":  
    print("petabyte")  
case "EB":  
    print("exabyte")  
default:  
    print("Not a recognized abbreviation.")  
}
```

In Swift, a `switch` must be  
**exhaustive.**


In Swift, each case must  
contain **executable code**.



```
// In many C-style languages
switch (levelNumber) {
    case 1:
    case 2:
    case 3:
        printf("Beginner level");
        break;
    case 4:
    case 5:
    case 6:
        printf("Intermediate level");
        break;
    case 7:
    case 8:
        printf("Advanced level");
        break;
    default:
        printf("Incorrect level")
}
```

```
// In many C-style languages
switch (levelNumber) {
    case 1:
    case 2: fallthrough
    case 3: 
        printf("Beginner level");
        break;
    case 4:
    case 5:
    case 6:
        printf("Intermediate level");
        break;
    case 7:
    case 8:
        printf("Advanced level");
        break;
    default:
        printf("Incorrect level")
}
```

```
// In many C-style languages
switch (levelNumber) {
    case 1:
        case 2: fallthrough
        case 3:
            printf("Beginner level");
            break;
    case 4:
        case 5: fallthrough
        case 6:
            printf("Intermediate level");
            break;
    case 7:
    case 8:
        printf("Advanced level");
        break;
    default:
        printf("Incorrect level")
}
```



```
// In many C-style languages
switch (levelNumber) {
    case 1:
        case 2: fallthrough
        case 3:
            printf("Beginner level");
            break;
    case 4:
        case 5: fallthrough
        case 6:
            printf("Intermediate level");
            break;
    case 7:
        case 8: fallthrough
            printf("Advanced level");
            break;
    default:
        printf("Incorrect level")
}
```

```
switch abbrev {  
case "kB":  
case "KB":  
    print("kilobyte")  
case "MB":  
    print("megabyte")  
case "GB":  
    print("gigabyte")  
case "TB":  
    print("terabyte")  
case "PB":  
    print("petabyte")  
case "EB":  
    print("exabyte")  
default:  
    print("Not a recognized abbreviation.")  
}
```

```
switch abbrev {  
case "kB":  
case "KB":  
    print("kilobyte")  
case "MB":  
    print("megabyte")  
case "GB":  
    print("gigabyte")  
case "TB":  
    print("terabyte")  
case "PB":  
    print("petabyte")  
case "EB":  
    print("exabyte")  
default:  
    print("Not a recognized abbreviation.")  
}
```

```
switch abbrev {
```

```
case "kB":
```

```
case "KB":
```

```
    print("kilobyte")
```

```
case "MB":
```

```
    print("megabyte")
```

```
case "GB":
```

```
    print("gigabyte")
```

```
case "TB":
```

```
    print("terabyte")
```

```
case "PB":
```

```
    print("petabyte")
```

```
case "EB":
```

```
    print("exabyte")
```

```
default:
```

```
    print("Not a recognized abbreviation.")
```

```
}
```

• 'case' label in a 'switch' should have at least one executable statement

In Swift, there is  
**no automatic fallthrough.**

```
volcanoExplosivityIndex = 3

switch volcanoExplosivityIndex {
case 0:
    print("Effusive")
case 1:
    print("Gentle")
    print("Note: expect a plume of < 1 km")
case 2:
    print("Explosive")
case 3:
    print("Catastrophic")
case 4:
    print("Cataclysmic")
    print("For example: the 2010 eruption of Eyjafjallajökull")
case 5:
    print("Paroxysmic")
case 6:
    print("Colossal")
case 7:
    print("Super-colossal")
case 8:
    print("Mega-colossal")
    print("NOTE: 🤯🤯🤯")
default:
    print("Not a recognized index.")
}
```



```
volcanoExplosivityIndex = 3
```

```
switch volcanoExplosivityIndex {  
case 0:  
    print("Effusive")  
case 1:  
    print("Gentle")  
    print("Note: expect a plume of < 1 km")  
case 2:  
    print("Explosive")  
case 3:  
    print("Catastrophic")  
case 4:  
    print("Cataclysmic")  
    print("For example: the 2010 eruption of Eyjafjallajökull")  
case 5:  
    print("Paroxysmic")  
case 6:  
    print("Colossal")  
case 7:  
    print("Super-colossal")  
case 8:  
    print("Mega-colossal")  
    print("NOTE: 🤯🤯🤯")  
default:  
    print("Not a recognized index.")  
}
```

```
volcanoExplosivityIndex = 3

switch volcanoExplosivityIndex {
case 0:
    print("Effusive")
case 1:
    print("Gentle")
    print("Note: expect a plume of < 1 km")
case 2:
    print("Explosive")
case 3:
    print("Catastrophic")
case 4:
    print("Cataclysmic")
    print("For example: the 2010 eruption of Eyjafjallajökull")
case 5:
    print("Paroxysmic")
case 6:
    print("Colossal")
case 7:
    print("Super-colossal")
case 8:
    print("Mega-colossal")
    print("NOTE: 🤯🤯🤯")
default:
    print("Not a recognized index.")
}
```

```
volcanoExplosivityIndex = 3

switch volcanoExplosivityIndex {
case 0:
    print("Effusive")
case 1:
    print("Gentle")
    print("Note: expect a plume of < 1 km")
case 2:
    print("Explosive")
case 3:
    print("Catastrophic")
case 4:
    print("Cataclysmic")
    print("For example: the 2010 eruption of Eyjafjallajökull")
case 5:
    print("Paroxysmic")
case 6:
    print("Colossal")
case 7:
    print("Super-colossal")
case 8:
    print("Mega-colossal")
    print("NOTE: 🤯🤯🤯")
default:
    print("Not a recognized index.")
}
```

```
volcanoExplosivityIndex = 3

switch volcanoExplosivityIndex {
case 0:
    print("Effusive")
case 1:
    print("Gentle")
    print("Note: expect a plume of < 1 km")
case 2:
    print("Explosive")
case 3:
    print("Catastrophic")
case 4:
    print("Cataclysmic")
    print("For example: the 2010 eruption of Eyjafjallajökull")
case 5:
    print("Paroxysmic")
case 6:
    print("Colossal")
case 7:
    print("Super-colossal")
case 8:
    print("Mega-colossal")
    print("NOTE: 🤯🤯🤯")
default:
    print("Not a recognized index.")
}
```

```
volcanoExplosivityIndex = 3
```

```
switch volcanoExplosivityIndex {
```

```
case 0:
```

```
    print("Effusive")
```

```
case 1:
```

```
    print("Gentle")
```

```
    print("Note: expect a plume of < 1 km")
```

```
case 2:
```

```
    print("Explosive")
```

```
case 3:
```

```
    print("Catastrophic")
```

```
case 4:
```

```
    print("Cataclysmic")
```

```
    print("For example: the 2010 eruption of Eyjafjallajökull")
```

```
case 5:
```

```
    print("Paroxysmic")
```

```
case 6:
```

```
    print("Colossal")
```

```
case 7:
```

```
    print("Super-colossal")
```

```
case 8:
```

```
    print("Mega-colossal")
```

```
    print("NOTE: 🤯🤯🤯")
```

```
default:
```

```
    print("Not a recognized index.")
```

```
}
```

```
volcanoExplosivityIndex = 3
```

```
switch volcanoExplosivityIndex {
```

```
case 0:
```

```
    print("Effusive")
```

```
case 1:
```

```
    print("Gentle")
```

```
    print("Note: expect a plume of < 1 km")
```

```
case 2:
```

```
    print("Explosive")
```

```
case 3:
```

```
→ print("Catastrophic")
```

```
case 4:
```

```
    print("Cataclysmic")
```

```
    print("For example: the 2010 eruption of Eyjafjallajökull")
```

```
case 5:
```

```
    print("Paroxysmic")
```

```
case 6:
```

```
    print("Colossal")
```

```
case 7:
```

```
    print("Super-colossal")
```

```
case 8:
```

```
    print("Mega-colossal")
```

```
    print("NOTE: 🤯🤯🤯")
```

```
default:
```

```
    print("Not a recognized index.")
```

```
}
```

```
// In many C-style languages
switch (levelNumber) {
    case 1:
    case 2:
    case 3:
        printf("Beginner level");
        break;
    case 4:
    case 5:
    case 6:
        printf("Intermediate level");
        break;
    case 7:
    case 8:
        printf("Advanced level");
        break;
    default:
        printf("Incorrect level")
}
```



```
// In many C-style languages
```

```
switch (levelNumber) {
```

```
    case 1:
```

```
    case 2:
```

```
    case 3:
```

```
        printf("Beginner level");
```

```
        break;
```

```
    case 4:
```

```
    case 5:
```

```
    case 6:
```

```
        printf("Intermediate level");
```

```
        break;
```

```
    case 7:
```

```
    case 8:
```

```
        printf("Advanced level");
```

```
        break;
```

```
default:
```

```
    printf("Incorrect level");
```

In Swift, you do **not** need to  
**break** at the end of each case



```
// In many C-style languages
switch (levelNumber) {
    case 1:
    case 2:
    case 3:
        printf("Beginner level");
        break;
    case 4:
    case 5:
    case 6:
        printf("Intermediate level");
        break;
    case 7:
    case 8:
        printf("Advanced level");
        break;
    default:
        printf("Incorrect level")
}
```

```
// In many C-style languages  
switch (levelNumber) {
```

```
  case 1:  
  case 2: ↓ fallthrough  
  case 3: ↓  
    printf("Beginner level");  
    break;
```

```
  case 4:
```

```
  case 5:
```

```
  case 6:
```

```
    printf("Intermediate level");  
    break;
```

```
  case 7:
```

```
  case 8:
```

```
    printf("Advanced level");  
    break;
```

```
  default:
```

```
    printf("Incorrect level");
```



```
// In Swift
switch levelNumber {
case 1,2,3:
    print("Beginner level")
case 4,5,6:
    print("Intermediate level")
case 7,8:
    print("Advanced level")
default:
    print("Incorrect level!")
}
```



```
switch abbrev {  
  case "kB", "KB":  
    print("kilobyte")  
  case "mB", "MB":  
    print("megabyte")  
  
  // etc...  
  
  default:  
    print("Not a recognized abbreviation.")  
}
```

# Swift Ranges

# Swift Ranges

```
switch someNumber {  
case 1,2,3,4,5,6,7,8,9,10:  
    print("One through ten")  
case 11,12,13,14,15,16,17,18,19,20:  
    print("Eleven through twenty")  
case 21:  
    print("Twenty-one")  
default:  
    print("Anything else!")  
}
```



# Swift Ranges

```
switch someNumber {  
    case 1,2,3,4,5,6,7,8,9,10:  
        print("One through ten")  
    case 11,12,13,14,15,16,17,18,19,20:  
        print("Eleven through twenty")  
    case 21:  
        print("Twenty-one")  
    default:  
        print("Anything else!")  
}
```

# Swift Ranges

```
switch someNumber {  
  case 1...10:  
    print("One through ten")  
  case 11,12,13,14,15,16,17,18,19,20:  
    print("Eleven through twenty")  
  case 21:  
    print("Twenty-one")  
  default:  
    print("Anything else!")  
}
```

# Swift Ranges

# Swift Ranges

...

**range operator**

# Swift Ranges

*start* . . . *end*

**range operator**

# Swift Ranges

1 . . . 10

**range operator**

# Swift Ranges

100..**.**..**.**..**.**500

**range operator**

# Swift Ranges

...

**range operator**



# Swift Ranges

# Swift Ranges

```
switch someNumber {  
case 1...10:  
    print("One through ten")  
case 11...20:  
    print("Eleven through twenty")  
case 21:  
    print("Twenty-one")  
default:  
    print("Anything else!")  
}
```

# Swift Ranges

```
switch someNumber {  
case 1...10:  
    print("One through ten")  
case 11...20:  
    print("Eleven through twenty")  
case 21:  
    print("Twenty-one")  
default:  
    print("Anything else!")  
}
```

```
volcanoExplosivityIndex = 3

switch volcanoExplosivityIndex {
case 0:
    print("Effusive")
case 1:
    print("Gentle")
    print("Note: expect a plume of < 1 km")
case 2:
    print("Explosive")
case 3:
    print("Catastrophic")
case 4:
    print("Cataclysmic")
    print("For example: the 2010 eruption of Eyjafjallajökull")
case 5:
    print("Paroxysmic")
case 6:
    print("Colossal")
case 7:
    print("Super-colossal")
case 8:
    print("Mega-colossal")
    print("NOTE: 🤯🤯🤯")
default:
    print("Not a recognized index.")
}
```

```
volcanoExplosivityIndex = 3

switch volcanoExplosivityIndex {
case 0:
    print("Effusive")
case 1:
    print("Gentle")
    print("Note: expect a plume of < 1 km")
case 2:
    print("Explosive")
case 3:
    print("Catastrophic")
case 4:
    print("Cataclysmic")
    print("For example: the 2010 eruption of Eyjafjallajökull")
case 5:
    print("Paroxysmic")
case 6:
    print("Colossal")
case 7:
    print("Super-colossal")
case 8:
    print("Mega-colossal")
    print("NOTE: 🤯🤯🤯")
default:
    print("Not a recognized index.")
}
```

must be **exhaustive**

```
volcanoExplosivityIndex = 3

switch volcanoExplosivityIndex {
case 0:
    print("Effusive")
case 1:
    print("Gentle")
    print("Note: expect a plume of < 1 km")
case 2:
    print("Explosive")
case 3:
    print("Catastrophic")
case 4:
    print("Cataclysmic")
    print("For example: the 2010 eruption of Eyjafjallajökull")
case 5:
    print("Paroxysmic")
case 6:
    print("Colossal")
case 7:
    print("Super-colossal")
case 8:
    print("Mega-colossal")
    print("NOTE: 🤯🤯🤯")
default:
    print("Not a recognized index.")
}
```

must be **exhaustive**  
**executable code** in each case

```
volcanoExplosivityIndex = 3

switch volcanoExplosivityIndex {
case 0:
    print("Effusive")
case 1:
    print("Gentle")
    print("Note: expect a plume of < 1 km")
case 2:
    print("Explosive")
case 3:
    print("Catastrophic")
case 4:
    print("Cataclysmic")
    print("For example: the 2010 eruption of Eyjafjallajökull")
case 5:
    print("Paroxysmic")
case 6:
    print("Colossal")
case 7:
    print("Super-colossal")
case 8:
    print("Mega-colossal")
    print("NOTE: 🤯🤯🤯")
default:
    print("Not a recognized index.")
}
```

must be **exhaustive**  
**executable code** in each case  
no **automatic fallthrough**

# Creating Loops



# Creating Loops

```
// code to repeat  
call a function  
make a constant  
print a message  
call another function  
// ...
```

# Creating Loops



```
// code to repeat
```

```
call a function
```

```
make a constant
```

```
print a message
```

```
call another function
```

```
// ...
```

# Creating Loops

```
// code to repeat  
call a function  
make a constant  
print a message  
call another function  
// ...
```

# Creating Loops



```
// code to repeat
```

```
call a function
```

```
make a constant
```

```
print a message
```

```
call another function
```

```
// ...
```

# Creating Loops

```
// code to repeat  
call a function  
make a constant  
print a message  
call another function  
// ...
```

# Creating Loops



```
// code to repeat
```

```
call a function
```

```
make a constant
```

```
print a message
```

```
call another function
```

```
// ...
```

# Creating Loops

```
// code to repeat  
call a function  
make a constant  
print a message  
call another function  
// ...
```

# Creating Loops

```
// code to repeat  
call a function  
make a constant  
print a message  
call another function  
// ...
```



# Creating Loops

```
// C-style "while"
while (some_condition_is_true) {
    // code to repeat
    call a function
    make a constant
    print a message
    call another function
    // ...
}
```

# Creating Loops

```
// C-style "while"
while (itemsToProcess > 0) {
    // code to repeat
    call a function
    make a constant
    print a message
    call another function
    // ...
}
```


# Creating Loops

```
// Swift-style "while"
while itemsToProcess > 0 {
    // code to repeat
    call a function
    make a constant
    print a message
    call another function
    // ...
}
```

# Creating Loops

```
// Swift-style "while"  
while itemsToProcess > 0 {  
    // code to repeat  
    call a function  
    make a constant  
    print a message  
    call another function  
    // ...  
}
```


parentheses  
optional





# Creating Loops

```
// Swift-style "while"  
while itemsToProcess > 0 {  
    // code to repeat  
    call a function  
    make a constant  
    print a message  
    call another function  
    // ...  
}
```

parentheses  
optional



braces  
required



# Creating Loops

```
// code to repeat  
call a function  
make a constant  
print a message  
call another function  
// ...
```

# Creating Loops

```
// C-style "do-while"  
do {  
    // code to repeat  
    call a function  
    make a constant  
    print a message  
    call another function  
    // ...  
} while (itemsToProcess > 0)
```

# Creating Loops

```
// code to repeat  
call a function  
make a constant  
print a message  
call another function  
// ...
```



# Creating Loops

```
// Swift-style "repeat-while"  
repeat {  
    // code to repeat  
    call a function  
    make a constant  
    print a message  
    call another function  
    // ...  
} while itemsToProcess > 0
```

# Creating Loops

```
// code to repeat  
call a function  
make a constant  
print a message  
call another function  
// ...
```

# Creating Loops

```
// Classic C-style "for" loop
for ( int i = 0; i < 50; i++ ) {
    // code to repeat
    call a function
    make a constant
    print a message
    call another function
    // ...
}
```

# Creating Loops

```
// Classic C-style "for" loop
for ( int i = 0; i < 50; i++ ) {
    // code to repeat
    call a function
    make a constant
    print a message
    call another function
    // ...
}
```

initialization

condition

afterthought

# Creating Loops

```
// Classic C-style "for" loop
for ( int i = 0; i < 50; i++ ) {
    // code to repeat
    call a function
    make a calculation
    print the result
    call another function
    // ...
}
```

initialization

afterthought



# Swift Range Operators

# Swift Range Operators

...

# Swift Range Operators

...

..  
..>



# Swift Range Operators

1...10

..

# Swift Range Operators

1...10

1, 2, 3, 4, 5, 6, 7, 8, 9, 10

..

# Swift Range Operators

`1...10`

`1, 2, 3, 4, 5, 6, 7, 8, 9, 10`

`1..<10`

# Swift Range Operators

`1...10`

`1, 2, 3, 4, 5, 6, 7, 8, 9, 10`

`1..<10`

`1, 2, 3, 4, 5, 6, 7, 8, 9`

# Swift Range Operators

1...10

1, 2, 3, 4, 5, 6, 7, 8, 9, 10

**closed range operator**  
inclusive of both values

1..**<**10

1, 2, 3, 4, 5, 6, 7, 8, 9

# Swift Range Operators

`1...10`

`1, 2, 3, 4, 5, 6, 7, 8, 9, 10`

**closed range operator**  
inclusive of both values

`1..<10`

`1, 2, 3, 4, 5, 6, 7, 8, 9`

**half-open range operator**  
range will not include the value on the right

# Review: Loops in Swift

```
// code to repeat  
// ...  
// ...  
// ...
```

# Review: Loops in Swift

```
while some_condition_is_true {  
    // code to repeat  
    // ...  
    // ...  
    // ...  
}
```



# Review: Loops in Swift

```
repeat {  
    // code to repeat  
    // ...  
    // ...  
    // ...  
} while some_condition_is_true
```

# Review: Loops in Swift

```
for item in items {  
    // code to repeat  
    // ...  
    // ...  
    // ...  
}
```

# String Concatenation **in Java**

```
String fName = "Payton";  
String lName  = "Emery";  
  
// concatenate two strings with a space in the middle  
String fullName = fName + " " + lName;
```

# String Concatenation **in Java**

```
String fName = "Payton";  
String lName = "Emery";
```

```
// concatenate two strings with a space in the middle  
String fullName = fName + " " + lName;
```

# String Concatenation

```
let message = fName + " " + lName + " is currently the " +  
              status + " player with a score of: " + String(score)
```

# String Concatenation

```
let message = fName + "_" + lName + "_ is currently the_" +  
              status + "_player with a score of:_" + String(score)
```

# String Concatenation

```
let message = fName + "_" + lName + " is currently the " +  
              status + "_player with a score of:_" + String(score)
```

\ ( )



# String Interpolation

"Now Playing *TRACK-NAME* by *ARTIST-NAME* which is *DURATION* long"