Simple Values

`let` for constants; `var` for variables

Constants: value doesn't need to be known at compile time, but must assign a value exactly once

Rule for types: constant or variable must have the same type as the value you want to assign it.

Inferring Types: Type can be inferred if value provided when creating constant/var.

Specifying types: Write after the variable, separated by colon

```
let explic itD ouble: Double = 70
```

Values are never implicitly converted to another type. If conversion required, must be done explicitly

```
let lottoS tring = " Today's winning lotto number is "
 let lottoN umber = 94
 let lottoS entence =  lottoS tring + String (lo tto Numbe
r)
```

Backslash string interpolation

```
let friend Count = 2
let friend Cou ntS tring = "I have \(frie ndC ount) number
 of
 friend s."
```

Use three double quotation marks for strings that take up multiple lines.

Create arrays and dictionaries using brackets, and access their elements by writing the index or key in brackets.

```
var toys = ["la bub u", " sim ski ", "furby"]
 toys[1] = " fug gle r"
 toys.a ppe nd( " jel lyc at")
 var bestIt emO fThing = [ " summer fruit" : " man go",
 bestIt emO fTh ing ["summer fruit"] = " pea ch"
```

Arrays grow automatically.

For empty array, write `excell ent Fruits = []`. For an empty dictionary, write `medioc reF ruits = [:]`.

If assigning empty array or dict to new var, you need to specify type.

```
let arrayO fGo odC off eeC afes: [String] = []
 let dictCa fes Wit hMa tch aAn dRa tings: [String: Float]
 = [:]
```

Control Flow

Conditionals: `if`, `switch`

Loops: `for-in`, `while`, `repeat - while`

Parentheses around condition/loop variable are optional, braces are rec

```
let croiss ant Scores = [1.2, 2.3, 2.2, 4]
 for score in croiss ant Scores {
     if score > 2.5 {
       print( " Great croiss ant !!")
     }
 }
```

In `if` statement, conditional must be a boolean expression. `if score }` is an error, not implicit comparison to zero.

You can assign if conditions to an assignment

```
 let scoreD eco ration = if teamScore > 10 {
" □"
 } else {
   " "
 }
print( " Sco re: ", teamScore, scoreD eco ration)
```

Optionals: either contains a value or contains `nil` to indicate a value is A question mark after type of value marks it as optional.

```
var option alS tring: String? = " Hel lo"
print( opt ion alS tring == nil)
 // prints " fal se"
```

Use `if` and `let` to work with missing values. If optional value is `nil`, c ional is false and code in braces is skipped. Otherwise, optional value is unwrapped and assigned to constant after `let`, which makes the unwra value available inside block of code.

```
var option alName: String? = "Big Dog"
 var greeting = " Hel lo! "
 if let name = option alName {
   greeting = " Hello, \(name )"
 } else {
     greeting = " Wow !"
 }
```

`??` operator: If optional value missing, default value used instead

```
let holida yInNov: String? = nil
 let holida yInDec: String = " Bal i"
 let welcomeMsg = " Enjoy \(holi day InNov ?? holida
ec )"
```