

# Day 1 Variables, constants, strings and numbers

Ted Lasso

Why Swift? Modern, easy to write, less accidental errors.

; optional  
Necessary only if  
two code pieces go same  
line.

How to follow along? Type the code, experiment.

How to create vars & constants?  
(Preferred, if possible)

Why Swift has constants? Avoid problems. (Swift helps in the contract).  
(It even recommends if value of var does not change)

How to create Strings?

- Use `"` for other `"`
- Use `"` for multiline strings (with line breaks).  
(Opening & closing delimiter must be on their own line.)

Functionality (Let string name = str)

- `str.count` (return length) Including newline character in multiline strings. *read some data. No (). (Later whole truth)*
- `str.uppercased()` *do some work* return uppercased string, `lowercased()`.
- `str.hasPrefix("Pre")` return true/false, `hasSuffix("fix")`.

• Might avoid multiline strings if code is to be shared for error searching.

How to store whole numbers? We can use `_` instead of `,` & Swift ignores it. (Improves readability)  
(Integer)

Functionality

`120.isMultiple(of: 3)`

(even \_\_\_\_\_ consecutive are ignored)  
or `name.isMultiple(of: anotherInt)`  
true/false

How to store decimal numbers? (Floating point)

The name comes from the surprisingly complex way the numbers are stored by your computer: it tries to store very large numbers such as 123,456,789 in the same amount of

space as very small numbers such as 0.0000000001, and the only way it can do that is by moving the decimal point around based on the size of the number.

- Swift uses type 'Double'. (Double precision floating point num.)
  - Wholly different type than Integers. Can't put together unless specifically asked. (Type Safety).  $\text{Int} + \text{Int}$  (Double data) or  $\text{Double} + \text{Double}$  (Int)
  - Once Swift decides data type of a const, var, it must always hold the same type.
- $$0.1 + 0.2 = 0.30000000000000004$$

Many older APIs use a slightly different way of storing decimal numbers, called CGFloat. Fortunately, Swift lets us use regular Double numbers everywhere a CGFloat is expected, so although you will see CGFloat appear from time to time you can just ignore it.

the reason floating-point numbers are complex is because computers are trying to use binary to store complicated numbers. (Such as 1/3)

---

Why Swift needs both Double & Integer?

The answer is that Swift is playing it safe: we can both see that 1 plus 1.0 will be 2, but your double is a variable so it could be modified to be 1.1 or 3.5 or something else. How can Swift be sure it's safe to add an integer to a double – how can it be sure you won't lose the 0.1 or 0.5?

The answer is that it can't be safe, which is why it isn't allowed.

---

Why is Swift type safe?  
Avoid mistakes with data.