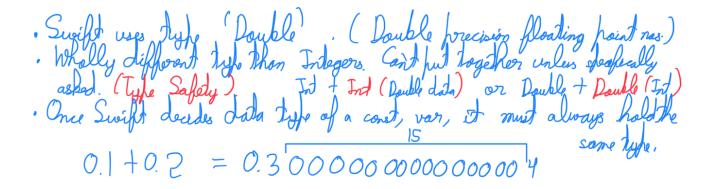
Vorsables, constants, strungs and numbers  Jakland  Meenoy only if
Why Sweft ? Modern, easy to write, less accedental express, two codepieces answers
How to follow along? Tyle the code, experiencent.
How to create vars & constants ? (Preferred, if possible)
Why Sweift has constants? Avoid problems, (Sweift help us in the contract), CIt even recommends if value of var does not change)
How to overte String?  Use "" for other"  ( with line breaks).  ( ) Opening & Josep delineter must be on their over line.)
Functionally (Let string name = stor)  Tread some data. No (). (Lator whole truth)  Etr, count (return length Including newline characters in multilune strings.  Stor. whereoud () do some work return whereased string loweraged ().  Dr. hastrefix ("fore") greturn true / false, has suffix ("fir").
· Might avoid multiline sterings if code is to be shared for ever searching.
How to store whole numbers? We can use _ instead of { Swift ignoress. (Improves roodability)  Functionality  [20, is Multiple (of: 3)) or name. is Multiple (of: onather Ind)  Terms / false
How to store decimal numbers ? (Floating point)

the name comes from the surprisingly compley way the numbers are stored by

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 0.000000001, and the only way it can do that is by moving the decimal point around based on the size of the number.



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 metakes with data