Functional v OOP

like USA v China

the equivalent in comp programming is

functional paradigm

versus

object oriented programming

functional and oop are not mutually exclusive. func created object oriented concepts in the early days

immutable oop v mutable oop

there is a big difference between immutable oop v mutable oop

the later is the path we have been on with: c++, java, c#, JavaScript oop, ...

for immutable oop. look at the string class in all modern languages including the ones that promite mutable oop.

mutable oop is an antipattern because mutable data fullstoo is an antipattern.

to understand why you really need to be a programmer who has experienced both.

it essentially boils down to the fact that mutable programs on more difficukt to understand

they do not really support concurrency. or not in a wst that majes sense

immutable oop

in some ways this is compatible with functional orogrammer

think smalltalk, erlang

but do we need it?

essentially it is a form if packaging. linking a set of functions to a datatype

lets take a string as example.

witg an oop implementation the desiners of the standard languahe try and predict every functikn we are likely to need for strings. in general they do a good job with this BUT

why should strings we limited to this set of functiobs

there are lots more functions that people may want to create that work on strings.

why should we have to create a new string class to beable to define these new string functions

functions should not need to be tied to their datatyoes

they may be inextricably linked to tge datatyoe (they usually are)

but this is not the same as tieing them tovtge datatype in an ooo fashion

we have modules/library/namspace mechanisms in all modern programming languages. these allow you to tie and group functions that relate to a datatyoe together.