Notes 7: functional programming

functional programming languages usually provide:
  first class functions
    first class
    can be passed into function
    can be returned from function
    can be set to a variable
  second class
    passed into function
  Third class
    none of the above
  higher-rder functions
    take a function as a parameter
    and or return a function
  polymorphism
    functions can work on lots of things
  list types and list operators
    lists are naturally recursive beasts or can be handled recursively with ease
  structured function returns
    return more than one thing
  constructors for structured objects
    make a block at one time
  garbage collection
    this is required if you have variables with unlimited extent. (which you get with closures)

Kotlin and Functional Programming
  first class — YES
  higher-order — YES
  Polymorph — YES — via object hierarchy and generics
  structure return — YES — especially with data classes
  Constructors — YES
  GC — absolutely

map/fold and lambda expressions
  lambda — a shorthand notation — most often used for single line anonymous functions
    lambda/simple.kt
      intro to lambda expressions
    lambda/map.kt
      lambda expressions and map/fold functions on list
        also filter, any, all, none, find, count
        these work on all collections (list, set, map)

Currying
  Suppose you have a function with 4 params. In one section of your code 3 of the 4 are always the same.
  Currying means to create a new function with the three preset!
  see curry/curry.kt

Compare speed of Kotlin and Go
  will use sorting of integer lists for the comparisons
Go: speed_go/
    sort a slice of structs
    by casting, can change the sort sort field
    10,000,000 sort int takes 2.2 seconds
    10,000,000 sort string (len 6) takes 7.9 sec (about 2 sec to build, 6 sec to sort)

Other than the particulars of Go, a fairly standard imperative program

Kotlin: speed_kt/

function programming and top down thinking
    “top down programming” is what you have been taught.
    start with statement of problem, design classes, design function interfaces, write ...
    Linked to a method of software development “waterfall”
    functional programming is “bottom up”.
    start by writing a program to do one little piece of task. Make sure it works.
    write another function ....
    The final program ends up being a fairly simple assembly of the pieces.
    You know it will work, because all of the pieces are easily and independently testable because each function depends only on its parameters
    In functional programming you ALWAYS have something that works.
    May not do everything, but it does things correctly

finally a full thing in functional Kotlin
the zip code lookup assignment
    hw2/hw2.kt

CONCLUSION
Functional programming does not preclude using loops, variables or mutable objects. You just have to use them thoughtfully. In particular, you MUST ensure that functions, on a give set of input, ALWAYS do that same thing.