Overview

- Map UI is done through MapKit
- The newer SwiftUI version is just a Map(), but this lacks some simple functionality like drawing a path on a map
    - As a result, we can use the older MKMapView() and create a UIViewRepresentable for it
        - This comes with MKMapViewDelegate which the Coordinator for the UIViewRepresentable can conform to
    - We usually can just use the empty initializer for MKMapView, and then set its region and delegate (if needed) afterwards
        - An MKCoordinateRegion just tells the map where to be centered and how zoomed in we should be
- Coordinates are represented as CLLocationCoordinate2D which just has a latitude and a longitude

Live Maps

- For showing user's current location and traveled path so far
- Location permissions and everything to do with a user's location is pretty much all handled by CLLocationManager
    - We have to check user's permissions and update the PList before we can start tracking the user's location
- There is a CLLocationManagerDelegate which contains all the delegate methods such as:
    - The user changed their location permissions
    - The user's location did update (gives us a list of new CLLocations)
        - This method is what allows us to update the user's traveled path in our view state
        - We can construct a CLLocationCoordinate2D with a CLLocation's .coordinate property
            - I believe there are more aspects of a CLLocation we can utilize, such as altitude, speed, etc.
- Make sure to call startUpdatingLocation and stopUpdatingLocation to make sure the location delegate methods are called
    - These can be hidden behind viewEvents which the view sends in an onAppear and onDisappear
- Since we are using a UIViewRepresentable, we have updateUIView which will allow us to draw the path the user has traveled
    - The UIViewRepresentable's Coordinator can act as the MKMapViewDelegate which has a method rendererFor which can add view overlays to the map
        - The map overlay for a traveled path is an MKPolyline

Static Map

- For showing a user's completed route after a workout
- We don't need the user's location at all, so we can just draw a map and use addOverlay to add a list of traveled coordinates to the map
    - We need to use the delegate method rendererFor to actually draw the MKPolyline
- We can add annotations to the map (like dropped pins)
    - There is a delegate method viewFor annotation which determines which MKAnnotationView to create for the annotation
        - This is similar to a UIKit TableView, where we need to deque an MKAnnotationView for MapKit optimization purposes
    - From what I worked on, the easiest way to get a custom view for this was to just set the image of the MKAnnotationView to something custom
        - We can also make our own class which subclasses MKAnnotationView which allows us to make any UIView for our MKAnnotationView