From "Proof of Concept" to an Actual Product

<p><b> Problems Beget Projects: </b><br>

In late January, I had a rough idea that I wanted to code something involving swim workouts. I was getting repeated requests from friends for new swim workouts that they could do on their own, and I quickly realized that this "problem" of "create new and fresh swim workouts every single day" might be something that I could frame in some computer code. </p>

<p><b>Where the Idea Came From:</b><br>

One friend in particular, would politely ask me if I had any workouts that I could send him. Being a swim coach for many years, and growing up swimming, I of course would seem to be the person who would have swim workouts. At first, I sent him some workouts from my workout notebook. I created this notebook while writing up a different workout each day for the high school swim team. I also sent him some email chains that Mark (another coach) and myself use to send each other workouts. </p>

<p>A couple of issues became painfully obvious. Firstly, the workouts that I had previously written down somewhere weren't easy to share electronically. Secondly, I also frankly didn't have that many workouts written down at all. Aside from the high school team's workouts, I usually don't write my workouts down. For the spring/summer/fall club team, I always have a basic outline of what a particular group's workout will be, and then I fill in the \_\_\_\_\_\_\_\_\_\_\_\_ spots. For early season, those \_\_\_\_\_\_\_\_\_\_\_\_\_ spots are filled with stroke drill progressions. For mid-season, they might be filled with training sets. Finally, the end of season might be filled race specific simulations; sets that teach race strategy. Because so much of my swim coaching has revolved around teaching, I never took the time build up a big repository of simple training based workouts. </p>

<p><b>My Workout Writing Heuristics</b><br>

Despite most of my coaching revolving around "swim technique" workouts. I certainly do know <em>how</em> to formulate good, interesting and effective <em>training</em> workouts. I believe that a lot of "writing good workouts" is actually about what <em>not</em> to write:

<ul>

<li><p>Don't automatically focus every practice on freestyle <br>(there are 3 other strokes!</p></li>

<li><p>Don't reuse the exact same test sets every week<br> (a test set of 20 x 50s on :40 can easily be compared to a performance on a test set such as 16 x 75s on 1:05!)</p></li>

<li><p>Don't always use the same "style" of set just because it might have been "proven" as the most effective and direct way to train a certain energy system<br> (5 x 100s descend from easy to all out, a 500 build, and 5 x 100s on a moderate paced interval can all can work the same fitness energy systems!)</p></li>

</ul></p>

<p>

What this list really says is, "people want variety!". Once someone advances beyond the private club level of swimming and into high school and senior level swimming, it's all about training. With masters swimming it's all about fitness. Training and fitness goals don't happen without long-term consistent commitment. The two major things that stop this from happening are injuries and burnout. People that are bored get "burn-out". People that over train the same muscle movements, in the same style and the same intensity, day in and day out tend to get injured more frequently. A lot of masters, high school and club workouts eventually just sort of blend together into "medium-fast endless freestyle swimming". A much better way is to always seek variety, while still tracking your fitness and training goals. From week to week, day to day, set to set, item to item (within the set), from coach to coach, things can and should be different. The target energy systems, the rest intervals, the strokes, and the 'style' of workout should all be different as often as possible. The same fitness and training goal can be served by multiple styles of sets.</p>

<p><b>The Elegance of Controlled Randomness</b><br>

My inner mad scientist began mapping out how a program of some sort could take user inputs, and spit out wildly different workouts from those same inputs, day after day, and week after week. The answer was randomness. Of course, we would want these different workouts to still be <em>good</em> workouts. So the randomness would have to be harnessed. I began writing up pseudo-code for how it could work. An extremely abbreviated version:</p>

<div class="code">

# User inputs to be converted <br>

# <br>

time\_input = number of minutes the user has to swim <br>

percent\_input = the % mixture of sprint versus endurance emphasis in the workout <br>

warm\_up\_time = 10 minutes (assumption made for the user) <br>

<br>

# Get time to be divided into sprint or endurance focus <br>

# <br>

if time\_input == "90 – 120 minutes":<br>

&emsp; &emsp; time\_for\_workout\_sets = time\_input – warm\_up\_time<br>

<br>

elif time\_input == "60 to 90 minutes":<br>

&emsp; &emsp; time\_for\_workout\_sets = time\_input – warm\_up\_time<br>

<br>

elif time\_input == "40 to 60 minutes":<br>

&emsp; &emsp; time\_for\_workout\_sets = time\_input – warm\_up\_time<br>

<br>

# Get decimal representations of sprint/endurance mixture <br>

#<br>  
if percent\_input == "50/50":<br>

&emsp; &emsp; sprint\_time\_float = 0.5<br>

&emsp; &emsp; endurance\_time\_float = 0.5<br>

<br>

elif percent\_input == "0/100: <br>

&emsp; &emsp; sprint\_time\_float = 0<br>

&emsp; &emsp; endurance\_time\_float = 1<br>

<br>

elif percent\_input == "100/0":<br>

&emsp; &emsp; sprint\_time\_float = 1<br>

&emsp; &emsp; endurance\_time\_float = 0<br>

<br>

elif percent\_input == "1/3 sprint 2/3 endurance":<br>

&emsp; &emsp; sprint\_time\_float = .34<br>

&emsp; &emsp; endurance\_time\_float = .66<br>

<br>

elif percent\_input == "2/3 sprint 1/3 endurance":<br>

&emsp; &emsp; sprint\_time\_float = .66<br>

&emsp; &emsp; endurance\_time\_float = .34<br>

<br>

# Get sprint time in minutes<br>

#<br>

sprint\_time = time\_for\_workout\_sets \* sprint\_time\_float<br>

<br>

# Get endurance time in minutes<br>

# <br>

endurance\_time = time\_for\_workout\_sets \* endurance\_time\_float<br>

<br>

# Randomly divide up sprint\_time into a number of sets<br>

# Randomly select the type of sprint set<br>

#<br>

sprint\_set\_number = sprint\_time / randint(1,4) <br>

<br>

for i in range(sprint\_set\_number): <br>

&emsp; &emsp; sprint\_swimsets.append(<br>

sprint\_set\_types[randint(0,5)] <br>

) <br>

<br>

# Randomly divide up endurance\_time into a number of sets<br>

# Randomly select the type of endurance set<br>

#<br>

endurance\_set\_number = endurance\_time / randint(1,4) <br>

<br>

for i in range(endurance\_set\_number): <br>

&emsp; &emsp; endurance\_swimsets.append(<br>

endurance\_set\_types[randint(0,5)] <br>

) <br>

<br>

# Get number of yards our user can swim in the number of minutes per set <br>

# (endurance\_time / endurance\_set\_number ) <br>

<br>

# the script currently solves a complex math formula to derive <br>

# the user's average pace based on a few test set inputs<br>

<br>

# randomly access the correct "bucket" based on <br>

# the yards and type of swim set called for<br>

#<br>

For swimset in sprint\_swimsets: <br>

&emsp; &emsp; if swimset > 1000 yards: <br>

&emsp; &emsp; &emsp; &emsp; get\_random\_swim\_workout(<br>

1000more\_yard\_bucket[swimset[type]][randint(0,100)] <br>

) <br>

&emsp; &emsp; elif swimset < 1000 yards and swimset > 500 yards: <br>

&emsp; &emsp; &emsp; &emsp; get\_random\_swim\_workout(<br>

500to1000\_yard\_bucket[swimset[type]][randint(0,100)] <br>

) <br>

&emsp; &emsp; elif swimset < 500 yardsL<br>

&emsp; &emsp; &emsp; &emsp; get\_random\_swim\_workout(<br>

500less\_yard\_bucket[swimset[type]][randint(0,100)] <br>

) <br>

<br>

# show swim sets to the user<br>

</div>

<p><b>Things Change:</b><br>

The pseudo-code that I started with was just to prove to myself that the concept could ever give a realistic workout. I wrote it up in python, and used some lists of dictionaries to store the workout "buckets". I filled the buckets with dummy sets. As I ran the script and tinkered with it, I realized that programmatically, the concept could work, but that I would need to write potentially 1000s of swimsets! Wanting to focus more on the coding than the swim sets, I recruited my friend, and also swim coach, Mark Shveyd to partner up on the project. Ever since then we've really come up with some more great ideas: customized intervals based on the user's test set, and allow the user to filter out types of sets, strokes and styles that they don't want. We looked into other swim workout options available on the web and we couldn't find a single one that we thought was actually good enough for someone to use (from all of the sites we found, it seems that many have indeed tried).</p>

<p> <b>Website? App? What is it?</b><br>

At first we were convinced that we wanted an app. I began learning how to use Kivy, which is a proven platform for writing android and iphone apps in python rather than objective-c or java. Kivy has a lot of promise, but there were a number of problems. Most of our problems were not with kivy, but with clarifying what we wanted to do. How would Mark input his swim sets, being non-programmer, at the same time I built the app? How would we update the workout buckets after uploading it to the app market? How would we dispay all of the options and settings in such a small screen without confusing a user?</p>

<p><b> Django!</b><br>

It was a real light bulb moment that seemed to happen all at once in my mind: "it should be a website, not a mobile app". I had been juggling all sorts of projects: our swim workout app, continuing to teach myself python, teach myself django, and re-learn html and css. I realized that constantly switching from one project to another was really inefficient. <a href="/blog/view/projectlearning">I also remembered that the best way to motivate oneself to learn is to have a project that you actually care about</a>. All the pieces began to fall into place:

<ul>

<li> <p>If we used django, I could build out the front end of the website, while Mark inputs swim sets into the database through the django admin. Using an html form would ensure that Mark's workouts would be properly formatted for the script to work with. No more trying to teach Mark how to hand insert his workouts into a document-type storage file.</p></li>

<li> <p>If we used a website, we could easily query the right workout from an SQL database of 10s of 1000's of workouts.</p></li>

<li><p> If we used a website, we could take advantage of html forms and lot's of screen real estate to help users give workout parameters.</p></li>

<li><p> If we used bootstrap 3, we could have a site that easily looks good both on a computer screen and a phone; there wouldn't necessarily ever need to be an app.</p></li>

</ul>

<p> Truthfully, we didn't "have to" use Django. However, at that time, Python was the only programming language I knew, and I had just learned how to build this blog site using django. The "app" phase of the project started in February, and it only took about one week (one week into March) to see that the website route was going to be about 5 times faster as I was about 5 times more productive than I had been trying to deal with the kivy emulator for android and iOS. I've since developed much of the site on my computer (running localhost):

<ul>

<li> <p> Set up django's views.py to easily talk with my swim set selecting script.</p></li>

<li> <p>Re-worked the swim set selecting script to directly SQL query the MySQL database without any django api.</p></li>

<li> <p>Set up the swim set models and admin forms to add new swim sets.</p></li>

<li><p> Added user registration, logins, and soon will add some "workout history" like functions for the user.</p></li>

<li> <p>Added a custom css file that over rides some aspects of bootstrap 3.</p></li>

</ul>

<p> The "actual product" still isn't quite ready for prime time. We'll need to test and calibrate some things. Mark will continue to write workouts into the database. Our plan is to test and eventually launch some sort of subscription plan to lap swimmers around the world.</p>

<p><a href="/blog/view/manytomanysql">See my other post about a specific challenge I had converting us from document storage to an SQL database</a></p>