**Picklenotes**

8/20/2023

The code for the pickle machine is in github https://github.com/deebuck/pickleball. It is installed on oregano and tarragon.

On oregano it is checked out in /home/dee/Documents/Pickleball.

On tarragon it is checked out in /home/dee/git/pickleball.

On oregano, there is a user crontab, under dee, and it runs the code as user dee with:

python3 Documents/Pickleball/liz5.py -p CvT2 CvT1 ChT2 ChT1 -s 30.1230 -z -d -o 1

On tarragon, there are a couple of additional issues. One, we want it to run as user dee, since that is where the code lives, but user dee isn't always logged on so can't use a user crontab so we have to use the system crontab. Second, some of the modules we need aren't available from apt on Debian 12, so we run in a Python virtual environment where we can use pip without messing up the system Python libraries. To solve these two issues, there is a supporting script called picklerun in /usr/local/bin/picklerun, which takes the same parameters as the main picklemachine. Picklerun is run from system crontab under user dee, and it invokes the virtual environment and then runs the code with the parameters.

The main python code is in the file liz5.py, which is the picklemachine. It takes the following parameters:

-p –preferences: desired court preferences in descending order, specified as e.g. CvT1 or ChT2, where Cv means Cavalier Trail courts and Ch means Cherry Street.

-s –session: desired duration and start time. Specified as <duration>.<start time> where duration is 10, 15, 20, 25, 30 (1,1.5,2,2.5,3 hours), and starttime is in 24 hour clock.

-g –debug: debugging run. Emits a few extra bits of data during the run, and sends messages and emails only to dee

-v –verbose: when running, print out all messages on the console, in addition to logging and sending by email

-z –headless: do not attempt to open a browser window on the screen

-i –immediate: do not sleep until midnight. make a reservation for tomorrow

-d –dry-run: do not actually make the reservation with the city, do all the steps but that

-x –width: if not headless, the width in pixels of the screen

-y –height: if not headless, the height in pixels of the screen

-o –offset: integer value, add this many additional 5 minute delays after midnight.

The code knows of 3 Falls Church authorized users: Liz, Sue and Brenda. A user is chosen randomly. If the requested court duration requires that we make more than 1 reservation, then a second user is chosen randomly from the remaining 2.

The code also waits a random amount of time after midnight, from 20 seconds to 4 minutes, so that it doesn't look quite so much like a robot is making the reservations.

As it runs, the code makes “notes” in a “picklejuice” file, which is nominally set at /tmp/picklejuice.log but easily changed in the code.

At the end of a run, the program notifies “some people” by email and by text. Who gets notified and what they receive is a function of: a) whether the run was successful, and b) whether debugging (-g) was set. If debugging is on, then only dee is sent any notifications at all. Otherwise email and text are sent to a list of recipients specified in the program (email: Dee and Liz, text, usually Dee and Liz, occasionally Sue instead. The program doesn't know Sue's email, and doesn't know either email or phone number for Brenda).

Sending a text message is actually done by sending an email to the carrier's email-to-text gateway for the carrier that has that users phone number. So to add a user to the list of people to get a text requires knowing the phone number, the carrier, and the email address of the gateway for that carrier.

If the reservation was successful, the email contains the picklejuice file as its body. If unsuccessful then in some cases (particularly if the program experiences a timeout or exception of some kind, and doesn't know exactly what caused the problem), the email will also contains an attached screenshot. In either case a text will be sent indicating simply whether there was success or failure.

There is some logic that if the program gets an unknown exception when it wakes up at midnight, it will try again 3 times to get the initial website screens up. This is because the city is changing over the data available at midnight and we aren't sure how long that might take.

The program tries all the court prefernces specified in the -p parameter until it finds an available court or exhausts the options. However it does not have a facility to try different times or durations. That is an improvement that's been discussed but not implemented.

There is a known issue that if the program experiences an exception when it is trying to make a reservation and it has already logged in as a certain user, then it somewhat loses its mind, and doesn't currently have the ability to log that user out. Thus an attempt to rerun the program right away will fail if it randomly selects that same user on the next run, and tries to log in, because Falls Church thinks the user is still logged in.

There are 3 common reasons for a failure.

1) the program fails just after logging in as a chosen user because the Falls Church website thinks that user is already logged in, and pops up a “what do you want to do” screen that the program isn't expecting.

2) the program fails to make a reservation as the chosen user because the Falls Church website think that user already has a reservation pending withing the time-windows (i.e. max next 48 hours).

3) the program fails because of some problem I don't understand that causes some element it is looking for to be “stale”. This is not well understood and I haven't been able to get good data on it. However, this usually only occurs during debugging when I am doing a lot of runs of the program and some are failing.

If the program fails in one of these ways, it SHOULD attach a screenshot of the page it currently has when the failure occurs.