LECTURE 43

THREADS IN TKINTER GUIS

MCS 260 Fall 2021

Emily Dumas

REMINDERS

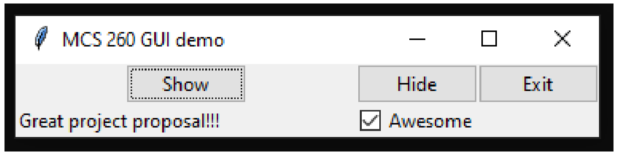
- Please complete the course evaluation.
- TA award nominations are open.
- Project 4 is due today at 6pm central, then you're done with MCS 260!
- Worksheet 15 solutions coming soon.

AFTER TODAY

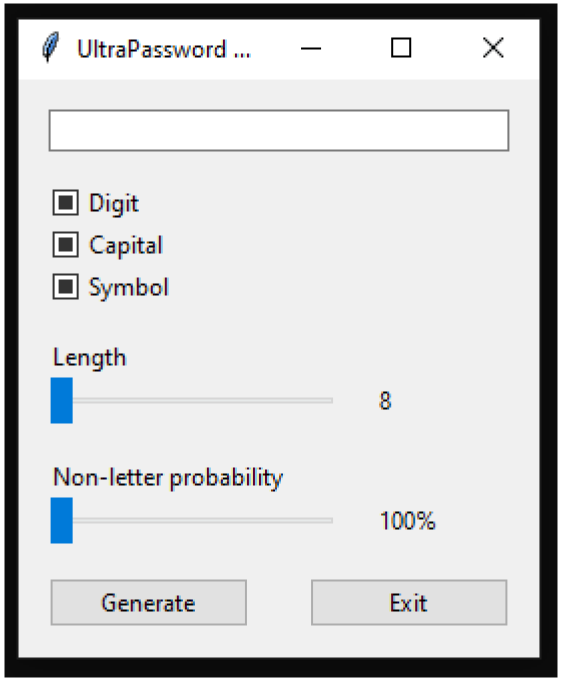
- No course activities.
- No office hours.
- Blackboard site stays up until Dec 31.
- My MCS 260 site stays up "forever" and will soon have everything except the videos.
- Hope to finish grading by Fri 10 Dec.
- Look for blackboard announcement about grades and archival site readiness.

GUI WORK SO FAR

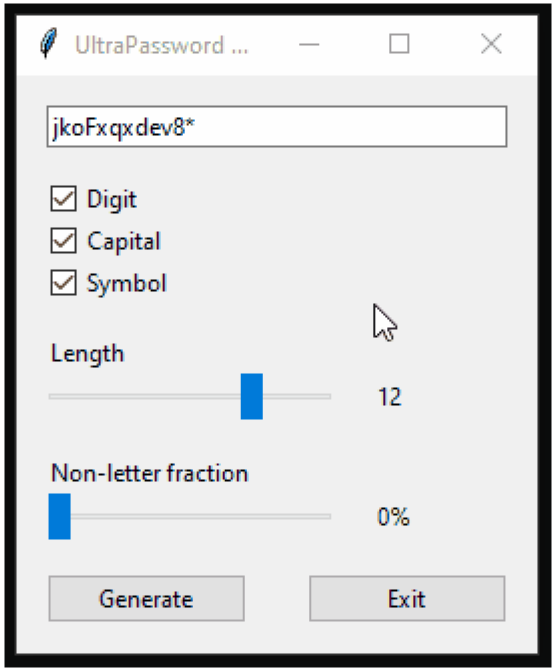
Lecture 36: Let's make buttons and labels in a window



Lecture 38: Let's make lots of widgets in a nice layout

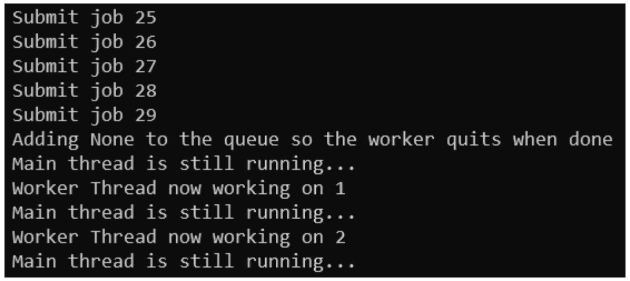


Lecture 40: Let's make the widgets do stuff (password generator)



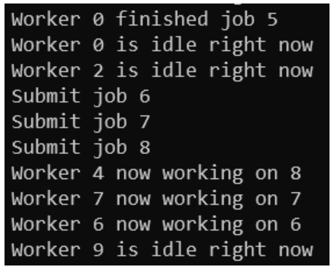
THREADING SO FAR

- Lecture 39 — Let's make some threads



```
Submit job 25
Submit job 26
Submit job 27
Submit job 28
Submit job 29
Adding None to the queue so the worker quits when done
Main thread is still running...
Worker Thread now working on 1
Main thread is still running...
Worker Thread now working on 2
Main thread is still running...
```

- Lecture 42 — Let's make the threads cooperate



```
Worker 0 finished job 5
Worker 0 is idle right now
Worker 2 is idle right now
Submit job 6
Submit job 7
Submit job 8
Worker 4 now working on 8
Worker 7 now working on 7
Worker 6 now working on 6
Worker 9 is idle right now
```

TODAY

The password generator application relies on a single function to generate a random password meeting certain specifications.

What if that function was slow?

Even if that's not realistic in this case, many GUI applications need to perform actions of long or uncertain duration.

THE PROBLEM

The GUI is unresponsive while the program is waiting for password generation to finish.

In addition to being a bad user experience, this can cause all sorts of problems.

A SOLUTION

Move the slow part to a separate "worker" thread.

Worker thread waits for signal that work is needed.

Main thread uses a `threading.Event` to signal it.

Main thread runs the GUI and remains responsive even if the worker is busy.

WHY NOT A QUEUE?

The producer-consumer pattern, with a queue, is useful if every piece of work dispatched by the main thread needs to be done.

But in this case, we only ever want to work on the most recent request for an update, ignoring any older ones that didn't get processed.

WHAT ELSE COULD WE DO?

- Perform encoding using an API (the way most actions in mobile apps occur)
- Make auto-update an option (settings menu?) with button to manually update if auto is disabled
- Make an application icon
- Make a help/about popup window

REFERENCES

- Official threading module documentation
- Official tkinter documentation
- Tk docs tutorial demonstrates lots of features with Python code
- Unofficial reference manual by John Shipman

REVISION HISTORY

- 2021-12-02 Initial publication

```
print("Have a good winter break!")
mcs260.exit(0)
```