***Read all of the following information before starting the exam:***

* The exam is open book, open notes, open Python documentation, open internet, etc.
* You **MAY NOT** use any form of technology to communicate with, send to, or receive information from another person (e.g., classmates, other instructors, anonymous or known persons on the internet). HOWEVER, you are **encouraged** to submit written questions to the professor or TA by email and/or have a private help session with the instructor through ZOOM.
* **MODULES/PACKAGES:** You may now use any Python packages you wish. However, you must follow specific instructions even if a package/library may be the easier way. You may use/reuse code (with proper attribution; e.g., “this function is modified from Dr. Smay’s.py file” or “this import is from my HW1 file”)
* **COMMENTS/DOCUMENTATION:** Your functions (especially constructors for classes) should use docstrings and other comments inside the function as necessary.
* **SUBMISSION:** You must create a private github repository for your work and invite the TA and instructor as collaborators.
* **GRADING:** When we grade your assignment, we will run your program with those given numerical values, looking for correct answers. Then we will change the numerical values (including changing the SIZES of the arrays) and look for correct answers for those modified values as well. We will only use numerical values, array sizes and functions that make sense. We will not be testing your program to see how it handles bad data.

1. (30 points) (main themes tested: GUI creation/use and reuse of code) Problem 2 from exam 2 analyzed the transient response of the RLC circuit below when a sinusoidal driving voltage *v*(*t*) was switched on. Here, you should write a GUI program that displays fields (line edit widgets) where the user can alter the values for L, R, C, and magnitude, frequency and phase of the sinusoidal driving voltage and that simulates the circuit and displays the resulting transient behavior. Your program should:
   * 1. display a picture of the RLC circuit inside a group box along with the labeled line edits for the user inputs. (see EX3P1SP22.py). The defaults from exam 2 were: R=10Ω, L=20H, C=0.05F, *v*(*t*)=20⋅sin(20⋅t+0)
     2. simulate the transient response of the circuit by using *scipy.optimize.solve\_ivp* to solve the differential equations based on user input.
     3. plot the transient behavior for *i1*, *i2* and *vc* (voltage across the capacitor)as a function of time with a matplotlib graph on your gui that includes the toolbar for exploring/editing the plot.

Diagram, schematic

Description automatically generated

1. (35 points) (Main themes tested: reading/writing/parsing text file, drawing schematics on GUI, MVC) Create program that can read a circuit text file that contains sufficient information to describe the RLC circuit in Problem 1 and that draws a diagram of the circuit using Graphics View Framework with symbols similar to those in the figure of Problem 1.

Program requirements:

i. You should write your own text file consisting of resistors, inductors, capacitors, voltage sources, nodes and loops. I recommend following the structure of the files for the latest pipe network program that uses pairs of hypertext tags such as <node>…</node>, etc. I found that using a spacing of 20 units for node positions worked well (i.e., TopLeft node at -40,20, BottomRight node at 40,-20)

ii. Your program should read from and parse this file to get the necessary information to draw the diagram.

iii. Fix the function for drawing the resistor. Don’t worry about labeling the circuit elements or drawing the loop currents.

Note: For an example of how to create compound graphics objects such as a resistor, you should explore the code used in the quarter car model to springs.

1. (35 points) (Main themes tested: modeling a thermodynamic cycle by adapting previous Otto cycle program and MVC design pattern) We’ve modeled the Rankine cycle using the MVC design pattern and utilized pyXSteam to find steam properties. Here, you should create a GUI program using the MVC design pattern to model both an Otto cycle **and** diesel cycle. Your program should have the following characteristics:

i. A GUI that allows the user to input all the necessary information to specify either the Otto cycle or the diesel cycle in either English or SI units. The GUI should change appropriately with selection of the cycle from either a combo box widget or a radio button widget.

ii. A calculate button that analyses the Otto or diesel cycle and outputs the results to the GUI by way of line edit objects as well as a matplotlib graph embedded in the GUI.

iii. Utilize a good MVC design pattern where all GUI inputs interact only with the proper controller object and the controller is responsible for updating the Model and the View.

**Note**: Working versions of the Otto and Diesel cycles have been placed in the folder for use. In the Otto program, air is treated as an ideal gas with reference state at T=0oC, P=1atm where molar specific properties (u=0, h=0, s=0, v=vo) and properties at other states are calculated by appropriately integrating the specific heat functions. The class air in Air.py makes these calculations simple by specifying two independent thermodynamic properties to the set function.

**About the air standard diesel cycle:**

An air standard diesel cycle consists of four, internally reversible processes: **1.** An isentropic compression from state 1 (at bottom dead center, V1) to state 2 (at top dead center, V1=r⋅V2), **2.** A constant pressure heat addition from state 2 to state 3 (P3=P2, V3 = rc⋅V2), **3.** An isentropic expansion from state 3 to state 4 (at bottom dead center, V4=V1), and **4.** A constant volume heat rejection.

An example diesel cycle has a compression ratio of r=18, a cutoff ratio of rc = 2, and a state 1 of T1 = 300K, P1=0.1MPa.