Programming 2 Midterm Part 2

Allowed Materials

The following resources are allowed:

- Python Library documentation on your computer or at https://docs.python.org/3/library/
- Github to view your exam repo, but no other Github repositories.
- Your own code on your computer for the tkinter assignments done in this course. But no other code.
- These Tkinter sites:
  - ☐ John Shipman's Tkinter Ref: https://tkdocs.com/shipman/ or PDF https://tkdocs.com/shipman/tkinter.pdf
  - ☐ Tkinter tutorial & ref: https://python-course.eu/tkinter/
  - ☐ Tkinter tutorial & ref: https://www.pythontutorial.net/tkinter/
  - ☐ Widget Attributes https://anzeljg.github.io/rin2/book2/2405/docs/tkinter/std-attrs.html
  - ☐ Python 3 Tkinter Docs https://docs.python.org/3/library/tkinter.html
  - ☐ Tkdocs https://tkdocs.com and tutorial https://tkdocs.com/tutorial,

Procedure

1. Disconnect any auxiliary monitors. Use only one monitor during the exam.

2. Join the TAs voice channel on Discord and share your screen. Don't watch anyone else's screen.

3. Start recording a video using OBS or Webex.

4. Accept the assignment on Github and do it.

5. Submit a link to your video on this form: https://forms.gle/ZPYUUXgL2dN9oUcp9

Good luck.

What To Submit

1. Commit your work and push to the remote repository (on Github). Be careful to include all the Python files you add. It's a good idea to check that Github has your final submission.

2. After you finish, rename your video file as Yourname-labexam1.mkv, upload to Google Drive, and share with these people:

j.brucker@ku.th

poomtum.r@ku.th

thanatibordee.s@ku.th

vitvara.v@ku.th

nabhan.s@ku.th

3. Please inform the TA monitoring your exam when you are done. Thanks.

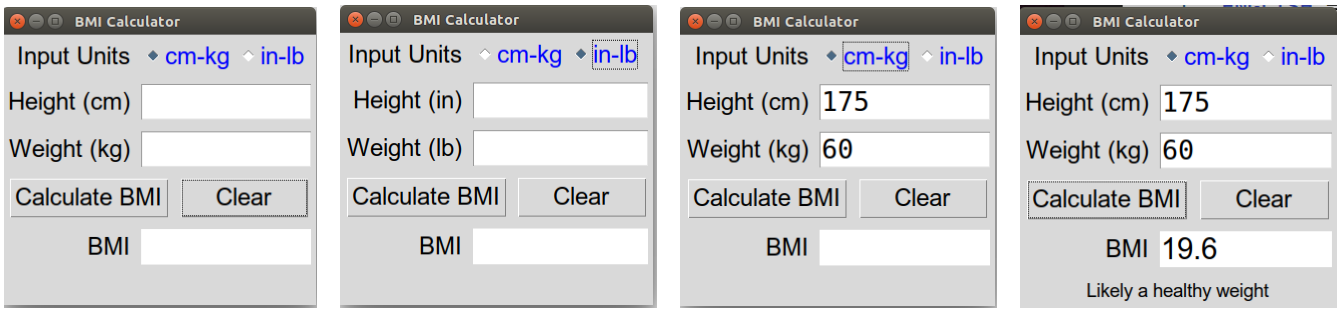
TAs: Please record finish time on Google sheet.

Problem: Write the UI for a BMI Calculator

Write a Tkinter UI to compute the Body Mass Index (BMI) in either Metric (cm - kg) or English (in - lb) units. See example below.

Requirements:

1. The UI is a subclass of tkinter.Tk.
2. UI has Radio buttons to select units as cm-kg or in-lb.
3. UI has buttons to calculate the BMI and clear the input fields.
4. When the user changes the input units (using radio buttons), the labels on the Length and Weight fields immediately change to show the units.
    - you can change the labels on the left side, or add separate labels for the units to the right of the input fields.
5. The starter code contains a class BMICalculator to compute the BMI from height, weight, and units. Your UI should use this class to compute the BMI.
6. Write a **`main.py`** that does: a) create an instance of BMICalculator, b) create a UI and set the BMICalculator into the UI (*dependency injection*) as done in the unit converter UI, c) run the UI.
7. Your code should never crash or print messages on the console. If the user inputs an invalid number, either ignore it or print a message in the label at the bottom of the UI.
8. Use a grid layout and leave space between components. Don't use place (absolute) layout.



`Initial display` `Change units` `Enter height & weight` `Click "Calculate"`

Programming Hints

1. BMICalculator has 2 constants for the units:

```
ENGLISH = "in-lb"
METRIC = "cm-kg"
```

when you call the bmi function, you must use these values for the "units" parameter:

```
bmicalculator.bmi(height, weight, units)
```

To simplify your code, use ENGLISH and METRIC as both the text and the value of the Radiobuttons.

2. To tie the 2 Radio buttons together (so only one is selected at a time) they need to share the same Control Variable. You should define a Control Variable (`tk.StringVar`) and use it on both radio buttons.

3. You should have a method that is invoked when the user selects either radio button (for example, `update_units(self)`) and this method updates the UI as needed.

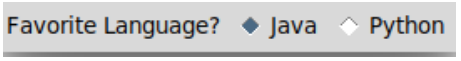
4. Good Radiobutton example: https://www.pythontutorial.net/tkinter/tkinter-radio-button/

5. Use ttk components (ttk.Label, ttk.Radiobutton) and *styles* to avoid a lot of duplicate code for styling the components. The default style names are "TLabel", "TButton", "TEntry", and "TRadiobutton". If you define a default style, it is automatically applied to all components. For example:

```
style = ttk.Style(root)   # 'root' is your top-level component, may be self
```

```
# set the style for all Labels
style.configure('TLabel', font=('Arial',16))
# style for all radio buttons
style.configure('TRadiobutton', font=('Somefont',15), foreground='blue', ...)
```

In my experiments, the 'TEntry' style did not set the font on Entry fields, but other styles work.

Radio Buttons: 


```
language = tk.StringVar()
language.set("Java")
rb1 = ttk.Radiobutton(root, text="Java", value="Java", variable=language, ...)
rb2 = ttk.Radiobutton(root, text="Python", value="Python", variable=language, ...)
```