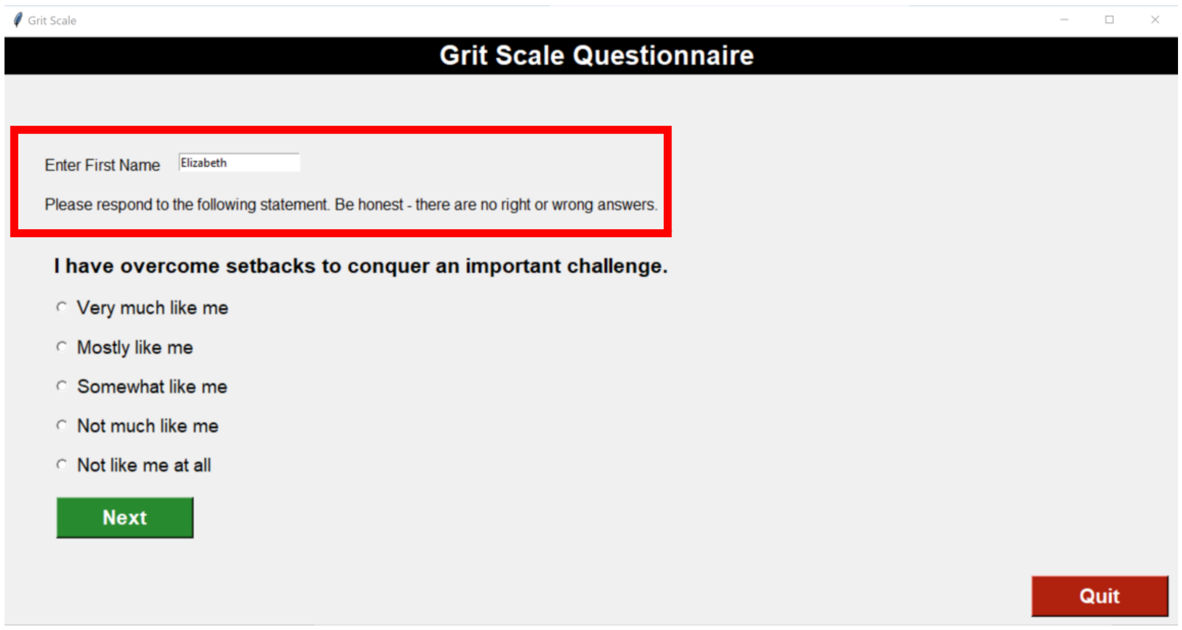
Elizabeth Schlievert
HCI 584
Summer 2022
Developer Guide

Overview:

- The Grit Scale measures the extent to which individuals can maintain focus and interest and persevere in obtaining long-term goals, a key element of technical aptitude. The assessment serves up a series of statements that would prompt the end user to respond using a 5-point Likert-type scale. The responses to each statement are associated with a numeric grit score. Once all questions are answered, the individual response scores are compiled and divided by the number of statements to deliver an average Grit score.

User Flow:

- **Task #1: Access Assessment**
    - User pulls up assessment program
    - User enters First Name
    - User reads Instructions



Enter First Name Elizabeth

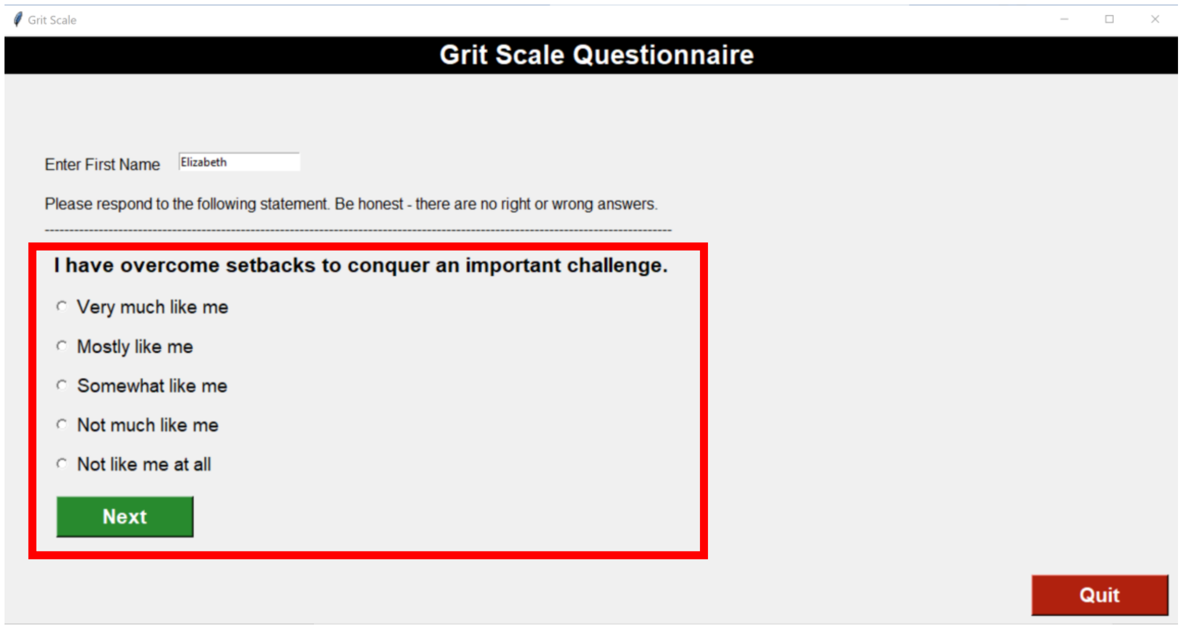
Please respond to the following statement. Be honest - there are no right or wrong answers.

I have overcome setbacks to conquer an important challenge.

- Very much like me
- Mostly like me
- Somewhat like me
- Not much like me
- Not like me at all

- **Task #2: Complete Assessment**
    - User is given the same options for answering each of the 12 statements:
        a) Very much like me
        b) Mostly like me
        c) Somewhat like me
        d) Not much like me
        e) Not like me at all
    - User reads 12 questions, responding a, b, c, d or e for each:
        1) I have overcome setbacks to conquer an important challenge.

2) New ideas and projects sometimes distract me from previous ones.
3) My interests change from year to year.
4) Setbacks don't discourage me.
5) I have been obsessed with a certain idea or project for a short time but later lost interest.
6) I am a hard worker.
7) I often set a goal but later choose to pursue a different one.
8) I have difficulty maintaining my focus on projects that take more than a few months to complete.
9) I finish whatever I begin.
10) I have achieved a goal that took years of work.
11) I become interested in new pursuits every few months.
12) I am diligent.


Grit Scale

Grit Scale Questionnaire

Enter First Name: Elizabeth

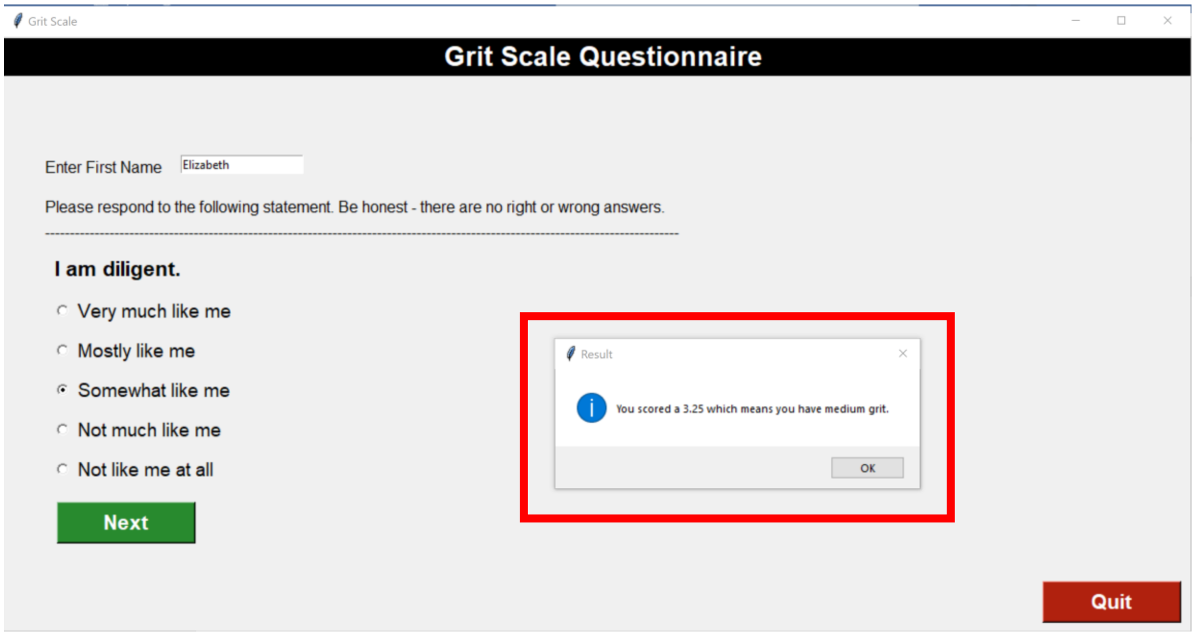
Please respond to the following statement. Be honest - there are no right or wrong answers.

I have overcome setbacks to conquer an important challenge.

- Very much like me
- Mostly like me
- Somewhat like me
- Not much like me
- Not like me at all

- **Task #3: Review Results**
  - Scores are tabulated
    - For questions 1, 4, 6, 9, 10 and 12 the following points will be assigned:
      - Very much like me = 5
      - Mostly like me = 4
      - Somewhat like me = 3
      - Not much like me = 2
      - Not like me at all = 1
    - For questions 2, 3, 5, 7, 8 and 11 the following points will be assigned:
      - Very much like me = 1
      - Mostly like me = 2
      - Somewhat like me = 3
      - Not much like me = 4
      - Not like me at all = 5

        - The score for all questions will be added and divided by 12, with a maximum score of 5 (extremely high grit) and a low score of 1 (extremely low grit).
    - Scores are displayed onscreen and stored in a CSV file.



Grit Scale

Grit Scale Questionnaire

Enter First Name Elizabeth

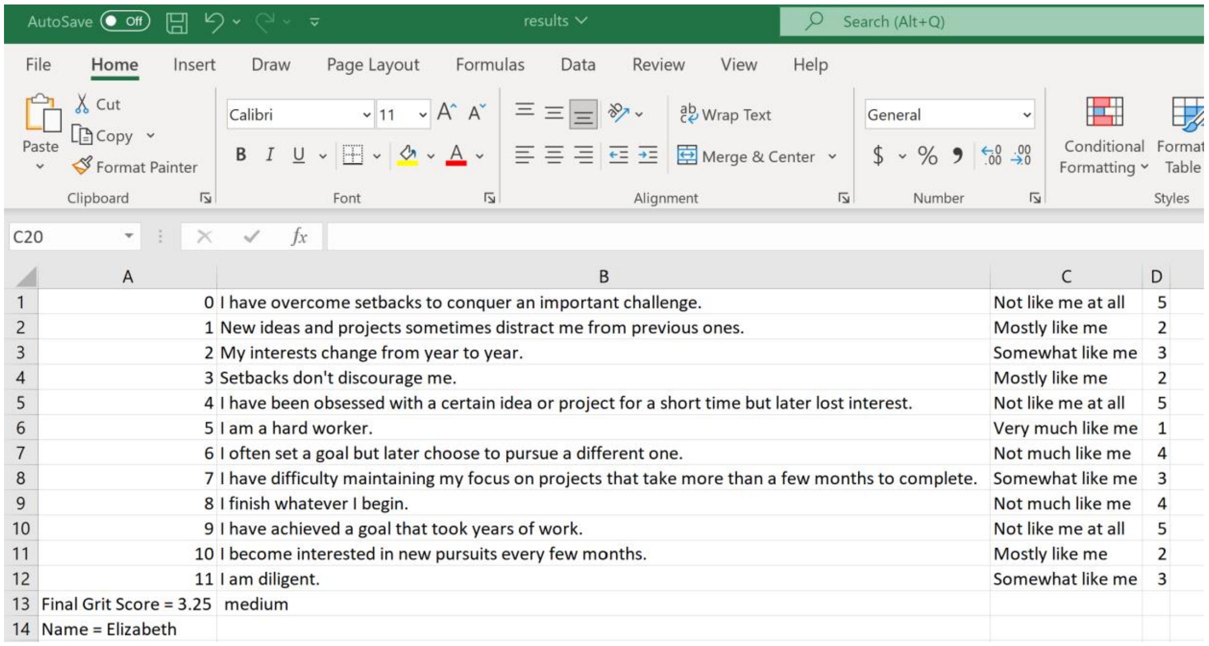
Please respond to the following statement. Be honest - there are no right or wrong answers.

I am diligent.

- Very much like me
- Mostly like me
- [✓ Somewhat like me]
- Not much like me
- Not like me at all

Result

You scored a 3.25 which means you have medium grit.



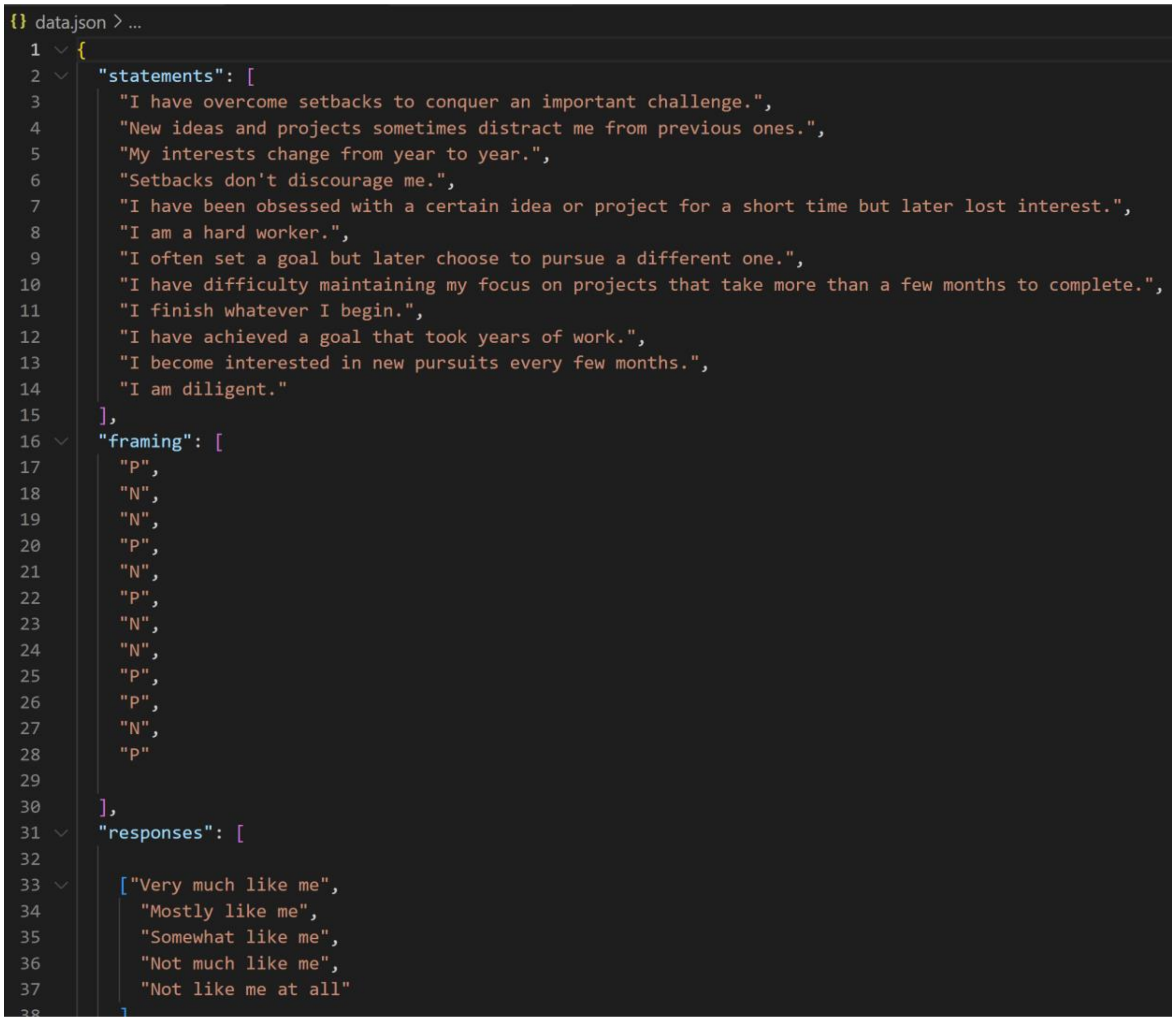
	A	B	C	D
1	0	I have overcome setbacks to conquer an important challenge.	Not like me at all	5
2	1	New ideas and projects sometimes distract me from previous ones.	Mostly like me	2
3	2	My interests change from year to year.	Somewhat like me	3
4	3	Setbacks don't discourage me.	Mostly like me	2
5	4	I have been obsessed with a certain idea or project for a short time but later lost interest.	Not like me at all	5
6	5	I am a hard worker.	Very much like me	1
7	6	I often set a goal but later choose to pursue a different one.	Not much like me	4
8	7	I have difficulty maintaining my focus on projects that take more than a few months to complete.	Somewhat like me	3
9	8	I finish whatever I begin.	Not much like me	4
10	9	I have achieved a goal that took years of work.	Not like me at all	5
11	10	I become interested in new pursuits every few months.	Mostly like me	2
12	11	I am diligent.	Somewhat like me	3
13	Final Grit Score = 3.25	medium		
14	Name = Elizabeth			

Installation and Deployment

- Read “Getting Started” in User Guide.
- All modules used are included in Python 3 standard packages.

Code Overview:

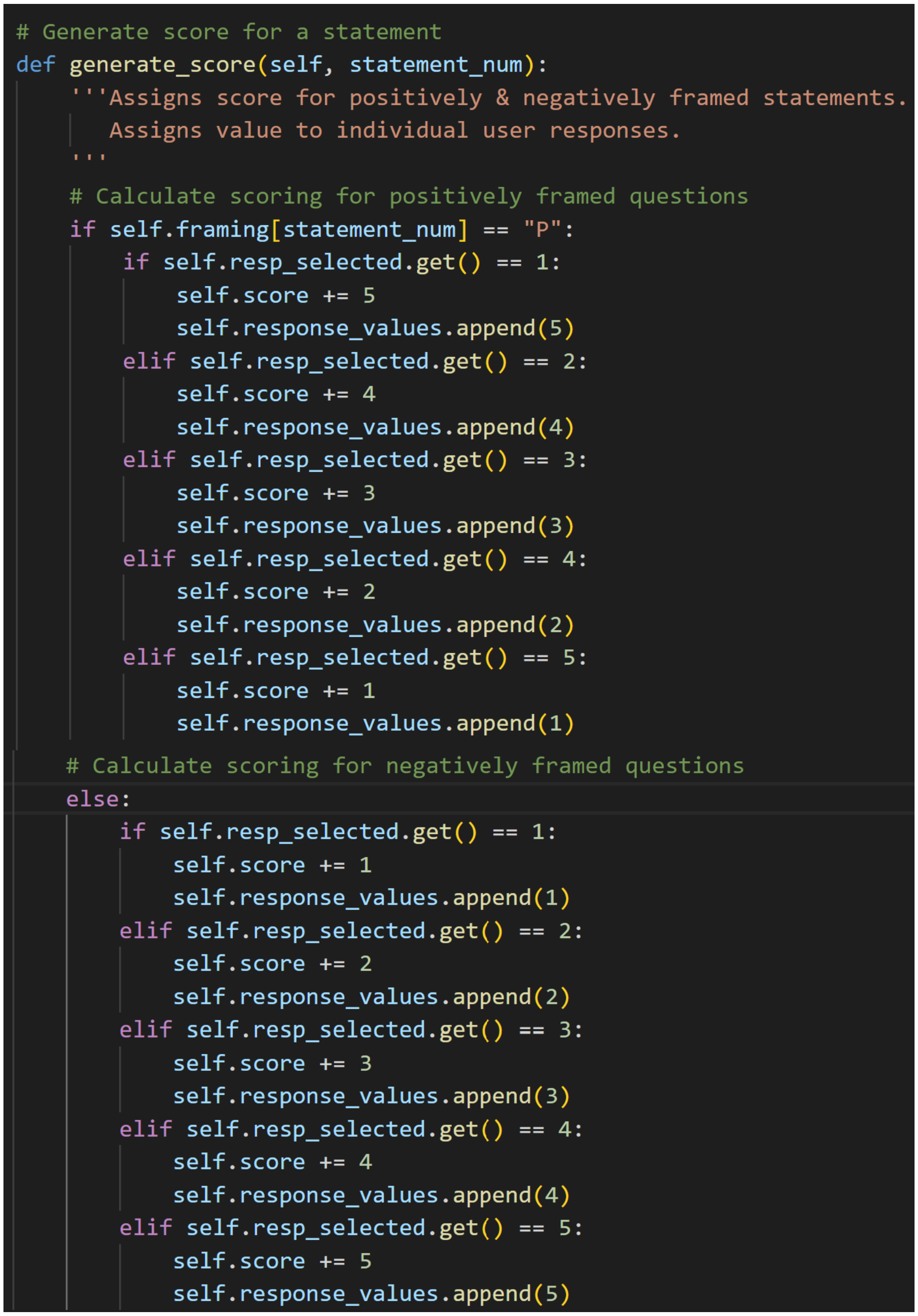
- Statements, statement framing, and responses are stored in json file intitled data.json. This file could be updated to include any number of statements or responses:



data.json > ...

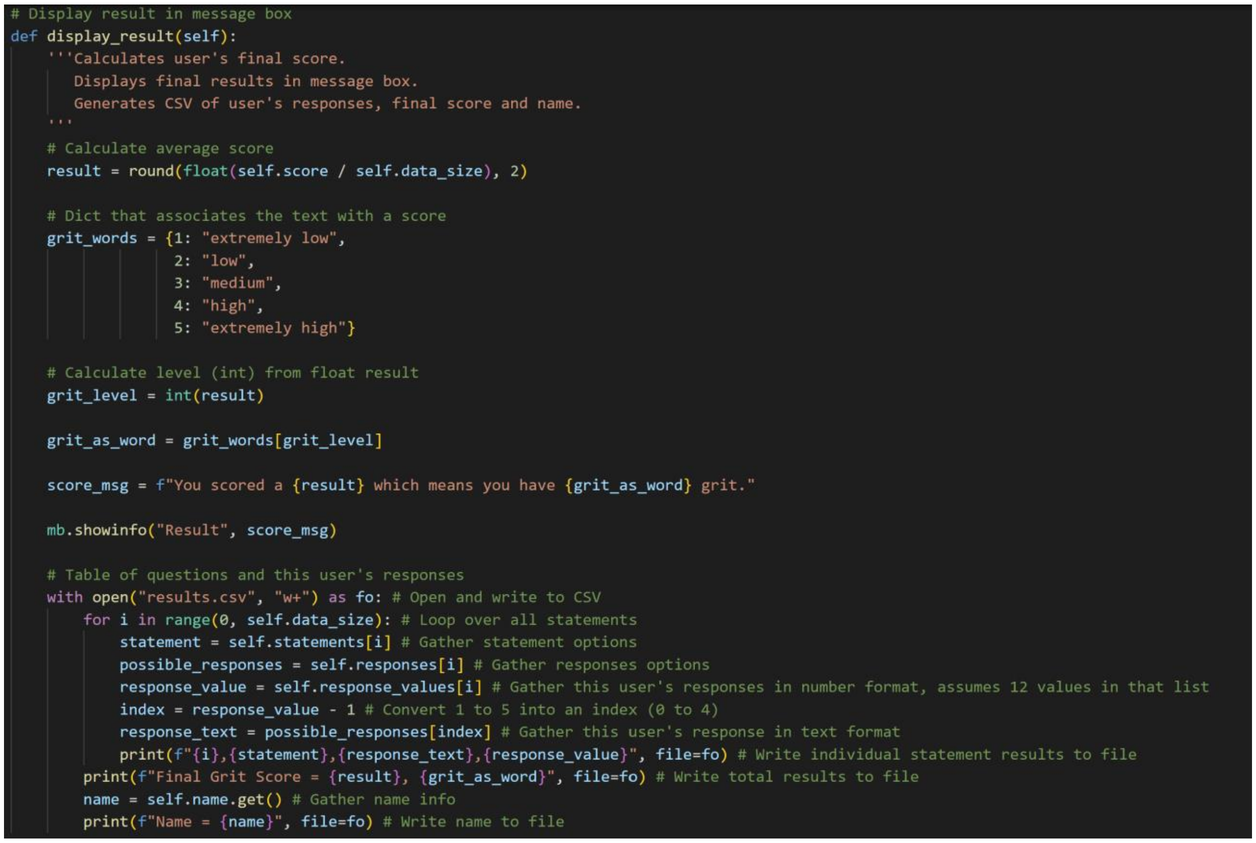
```
{
  "statements": [
    "I have overcome setbacks to conquer an important challenge.",
    "New ideas and projects sometimes distract me from previous ones.",
    "My interests change from year to year.",
    "Setbacks don't discourage me.",
    "I have been obsessed with a certain idea or project for a short time but later lost interest.",
    "I am a hard worker.",
    "I often set a goal but later choose to pursue a different one.",
    "I have difficulty maintaining my focus on projects that take more than a few months to complete.",
    "I finish whatever I begin.",
    "I have achieved a goal that took years of work.",
    "I become interested in new pursuits every few months.",
    "I am diligent."
  ],
  "framing": [
    "P",
    "N",
    "N",
    "P",
    "N",
    "P",
    "N",
    "N",
    "P",
    "P",
    "N",
    "P"

  ],
  "responses": [

    ["Very much like me",
      "Mostly like me",
      "Somewhat like me",
      "Not much like me",
      "Not like me at all"
```

- The statement framing and scoring are associated using **def generate_score**:



```python
# Generate score for a statement
def generate_score(self, statement_num):
    '''Assigns score for positively & negatively framed statements.
       Assigns value to individual user responses.
    '''
    # Calculate scoring for positively framed questions
    if self.framing[statement_num] == "P":
        if self.resp_selected.get() == 1:
            self.score += 5
            self.response_values.append(5)
        elif self.resp_selected.get() == 2:
            self.score += 4
            self.response_values.append(4)
        elif self.resp_selected.get() == 3:
            self.score += 3
            self.response_values.append(3)
        elif self.resp_selected.get() == 4:
            self.score += 2
            self.response_values.append(2)
        elif self.resp_selected.get() == 5:
            self.score += 1
            self.response_values.append(1)
    # Calculate scoring for negatively framed questions
    else:
        if self.resp_selected.get() == 1:
            self.score += 1
            self.response_values.append(1)
        elif self.resp_selected.get() == 2:
            self.score += 2
            self.response_values.append(2)
        elif self.resp_selected.get() == 3:
            self.score += 3
            self.response_values.append(3)
        elif self.resp_selected.get() == 4:
            self.score += 4
            self.response_values.append(4)
        elif self.resp_selected.get() == 5:
            self.score += 5
            self.response_values.append(5)
```

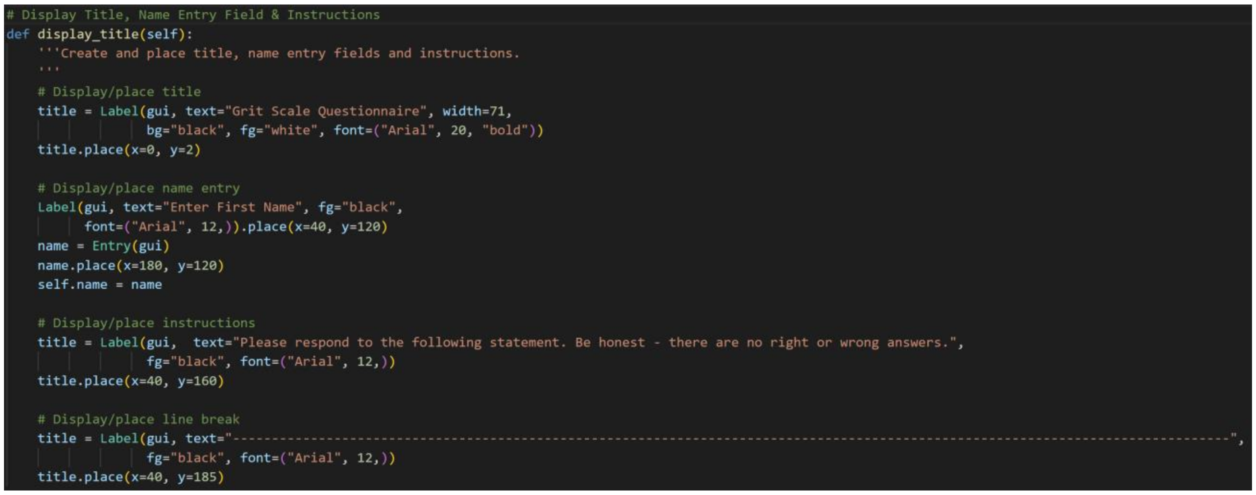
- **def display_results** then takes the users responses and calculates a final grit score, with associated language, and delivers the responses to each question, as well as the final Grit Score and first name to a CSV file titled Results.csv:



```python
# Display result in message box
def display_result(self):
    '''Calculates user's final score.
       Displays final results in message box.
       Generates CSV of user's responses, final score and name.
    '''
    # Calculate average score
    result = round(float(self.score / self.data_size), 2)

    # Dict that associates the text with a score
    grit_words = {1: "extremely low",
                  2: "low",
                  3: "medium",
                  4: "high",
                  5: "extremely high"}

    # Calculate level (int) from float result
    grit_level = int(result)

    grit_as_word = grit_words[grit_level]

    score_msg = f"You scored a {result} which means you have {grit_as_word} grit."

    mb.showinfo("Result", score_msg)

    # Table of questions and this user's responses
    with open("results.csv", "w+") as fo: # Open and write to CSV
        for i in range(0, self.data_size): # Loop over all statements
            statement = self.statements[i] # Gather statement options
            possible_responses = self.responses[i] # Gather responses options
            response_value = self.response_values[i] # Gather this user's responses in number format, assumes 12 values in that list
            index = response_value - 1 # Convert 1 to 5 into an index (0 to 4)
            response_text = possible_responses[index] # Gather this user's response in text format
            print(f"{i},{statement},{response_text},{response_value}", file=fo) # Write individual statement results to file
        print(f"Final Grit Score = {result}, {grit_as_word}", file=fo) # Write total results to file
        name = self.name.get() # Gather name info
        print(f"Name = {name}", file=fo) # Write name to file
```

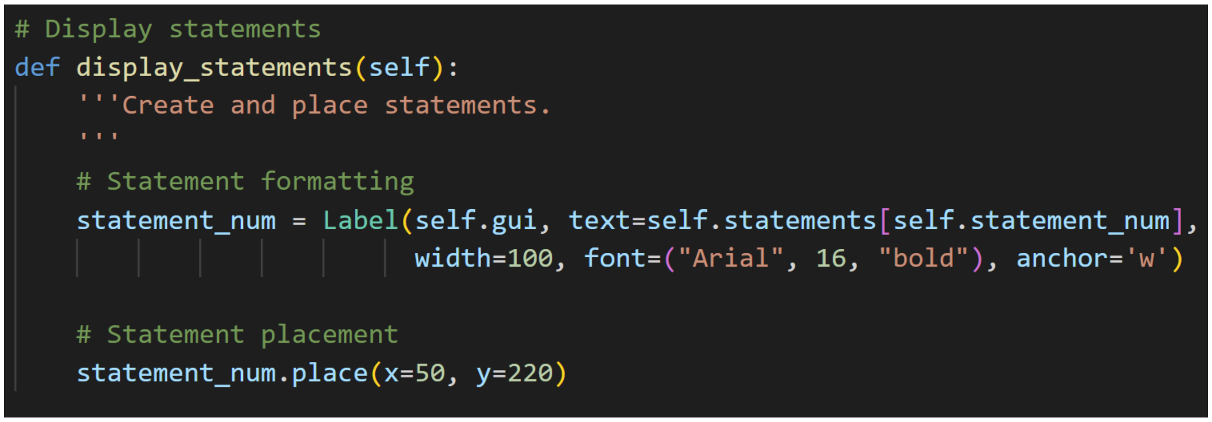
- The remaining functions populate the user interface and behaviors.

- **def display_title** populates the page title, the First Name entry field and the instructions with associated styling:



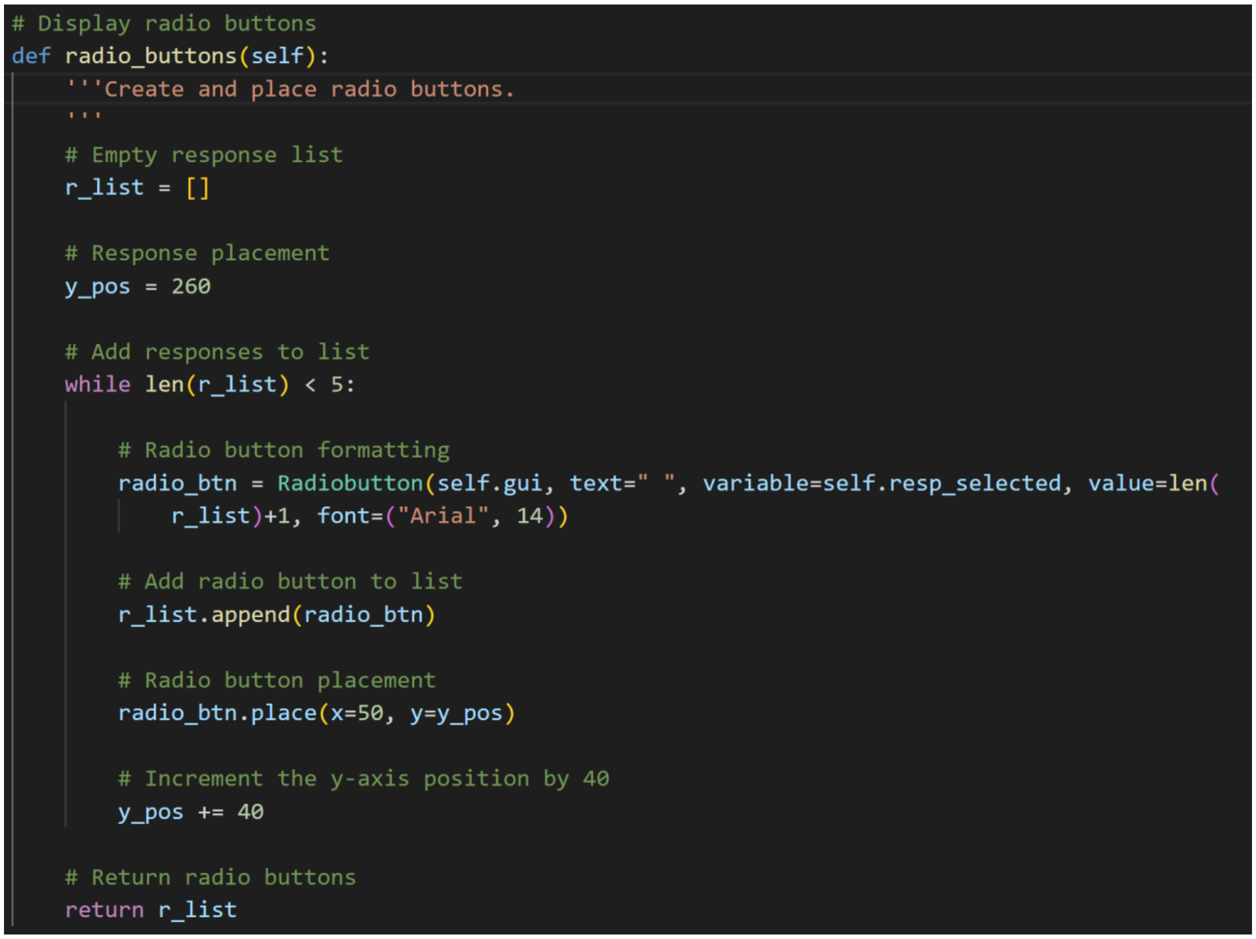
```python
# Display Title, Name Entry Field & Instructions
def display_title(self):
    '''Create and place title, name entry fields and instructions.
    '''
    # Display/place title
    title = Label(gui, text="Grit Scale Questionnaire", width=71,
                  bg="black", fg="white", font=("Arial", 20, "bold"))
    title.place(x=0, y=2)

    # Display/place name entry
    Label(gui, text="Enter First Name", fg="black",
          font=("Arial", 12,)).place(x=40, y=120)
    name = Entry(gui)
    name.place(x=180, y=120)
    self.name = name

    # Display/place instructions
    title = Label(gui,  text="Please respond to the following statement. Be honest - there are no right or wrong answers.",
                  fg="black", font=("Arial", 12,))
    title.place(x=40, y=160)

    # Display/place line break
    title = Label(gui, text="-------------------------------------------------------------------------------------------------------------------------------",
                  fg="black", font=("Arial", 12,))
    title.place(x=40, y=185)
```

- **def display_statements** populates the statements, with associated styling:



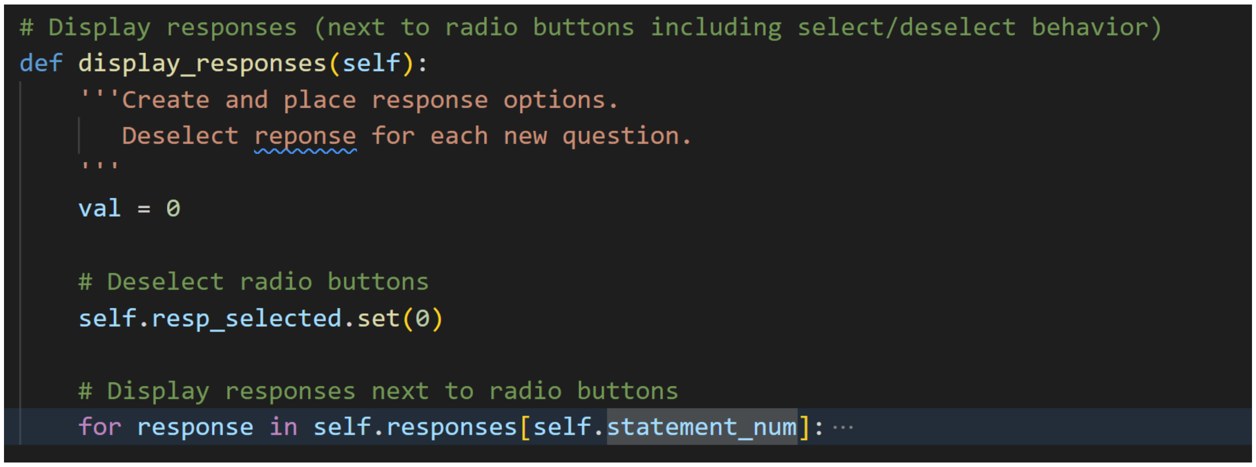
```python
# Display statements
def display_statements(self):
    '''Create and place statements.
    '''
    # Statement formatting
    statement_num = Label(self.gui, text=self.statements[self.statement_num],
                          width=100, font=("Arial", 16, "bold"), anchor='w')

    # Statement placement
    statement_num.place(x=50, y=220)
```

- **def radio_buttons** populates the radio buttons next to the responses:



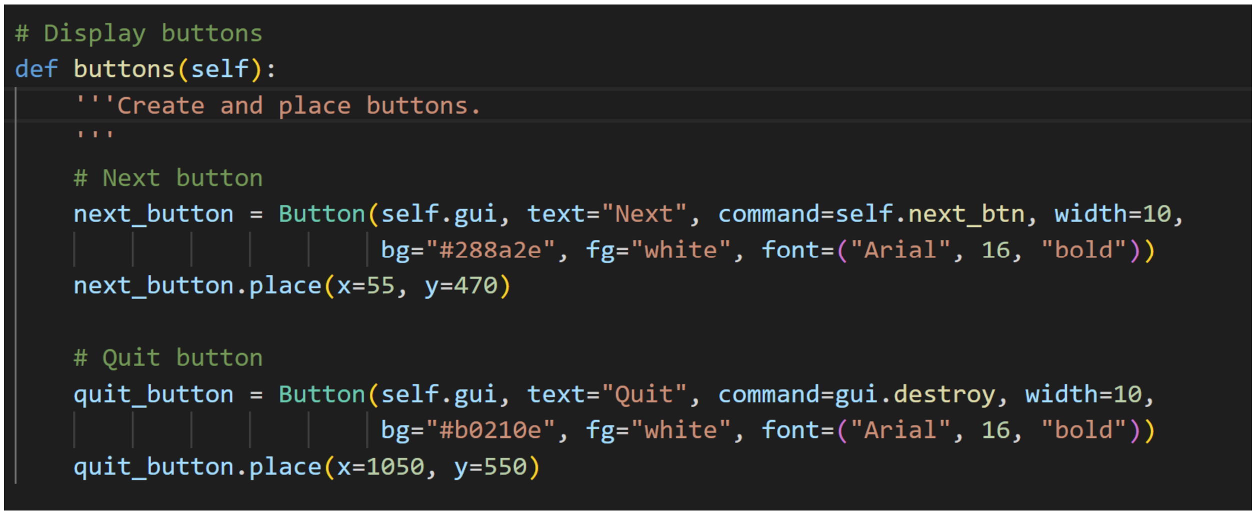
```python
# Display radio buttons
def radio_buttons(self):
    '''Create and place radio buttons.
    '''
    # Empty response list
    r_list = []

    # Response placement
    y_pos = 260

    # Add responses to list
    while len(r_list) < 5:

        # Radio button formatting
        radio_btn = Radiobutton(self.gui, text=" ", variable=self.resp_selected, value=len(
            r_list)+1, font=("Arial", 14))

        # Add radio button to list
        r_list.append(radio_btn)

        # Radio button placement
        radio_btn.place(x=50, y=y_pos)

        # Increment the y-axis position by 40
        y_pos += 40

    # Return radio buttons
    return r_list
```

- **def display_responses** populates the responses next to the radio buttons and deselects the radio buttons when moving to a new statement:



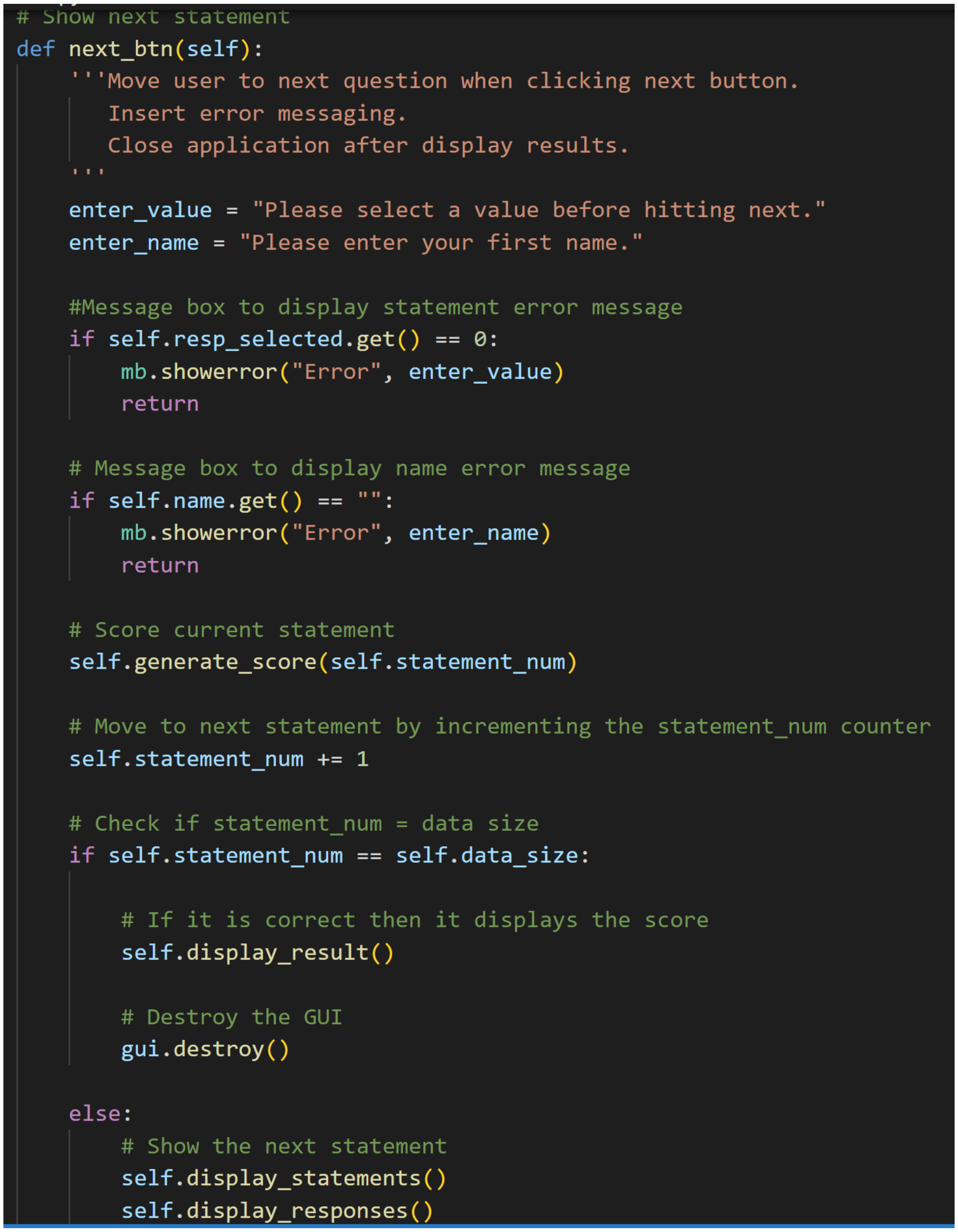
```python
# Display responses (next to radio buttons including select/deselect behavior)
def display_responses(self):
    '''Create and place response options.
       Deselect reponse for each new question.
    '''
    val = 0

    # Deselect radio buttons
    self.resp_selected.set(0)

    # Display responses next to radio buttons
    for response in self.responses[self.statement_num]: ···
```

- **def buttons** populates the Next and Quit buttons:



```python
# Display buttons
def buttons(self):
    '''Create and place buttons.
    '''
    # Next button
    next_button = Button(self.gui, text="Next", command=self.next_btn, width=10,
                         bg="#288a2e", fg="white", font=("Arial", 16, "bold"))
    next_button.place(x=55, y=470)

    # Quit button
    quit_button = Button(self.gui, text="Quit", command=gui.destroy, width=10,
                         bg="#b0210e", fg="white", font=("Arial", 16, "bold"))
    quit_button.place(x=1050, y=550)
```

- **def next_btn** offers the behavior to move users to the next question after hitting next, as well as offers error messaging in the absence of a First Name or selected Response. Finally, it displays the final score message and shuts down the application once complete:



```python
# Show next statement
def next_btn(self):
    '''Move user to next question when clicking next button.
       Insert error messaging.
       Close application after display results.
    '''
    enter_value = "Please select a value before hitting next."
    enter_name = "Please enter your first name."

    #Message box to display statement error message
    if self.resp_selected.get() == 0:
        mb.showerror("Error", enter_value)
        return

    # Message box to display name error message
    if self.name.get() == "":
        mb.showerror("Error", enter_name)
        return

    # Score current statement
    self.generate_score(self.statement_num)

    # Move to next statement by incrementing the statement_num counter
    self.statement_num += 1

    # Check if statement_num = data size
    if self.statement_num == self.data_size:

        # If it is correct then it displays the score
        self.display_result()

        # Destroy the GUI
        gui.destroy()

    else:
        # Show the next statement
        self.display_statements()
        self.display_responses()
```

Known Issues:

- The user does not have the capability to go back and change their response to a previous statement. The full questionnaire must be completed again to change responses.

Future work:

- Below are potential enhancements to the current application
    - Expand statement set that allows for sub-category scores for:
        - Consistency of Interest
        - Perseverance of Effort
        - Ambition
    - Offer a graphic representation of score on visual scale.
    - Shuffle the order of the questions.
    - Show users their score comparative to national averages.

Ongoing deployment/development:

- To save time in confirming the scoring model accuracy, write a unit test that automatically simulates randomized user responses rather than manually clicking through 12 statements.