Project - A Learning Management System for Programming in Python [PL4Python]

John Erwin Bisa
College of Engineer
De La Salle University
Manila, Philippines
john_erwin_bisa@dlsu.edu.ph

Jefferson Levita
College of Engineer
De La Salle University
Manila, Philippines
jefferson_levita@dlsu.edu.ph

Myles Earvin Uy
College of Engineer
De La Salle University
Manila, Philippines
myles_uy@dlsu.edu.ph

I. Introduction

A. Background of the Project

This project is about making a learning management system. It will reflect an interactive type of teaching where each student is catered with different levels of module and teaching depending on their own learning phase and style, this is "personalized learning". As mentioned in [1], personalized learning focuses on creating the ideal learning environment with a personalized learning schedule accustomed to each student's perspective. Similarly in [2], Personalized learning can involve different levels in the educational process, which include personalization of the curriculum, the courses, as well as the support provided within the courses. Personalized knowledge can take place in regular (face-to-face) gaining knowledge of settings as well as in technology-enhanced learning settings.

The Program is designed to gauge the students strengths, skills, needs, interests and the amount of information absorbed in a certain length of time. It will keep track of every student's activities, progress, speed and profile in order to aid the program's response in adjusting a personalized learning path as Amanda discussed in [3]. The program is designed to adjust to the pacing and speed of the student depending on how the student progresses in every topic similar to the idea in [4]. As mentioned in [5], that In gauging the strengths, skills, and interests of the student it is important to dig deep into their profile, activities and portfolio because these things greatly reflect a student's knowledge in programming.

Personalized Learning has a lot of benefits and outstanding advantages which would really have great impacts on the students, teachers and administration but it does still have its own set of disadvantages as mentioned in [6]. However a fully coded program that this project plans to create would supplement in decreasing those disadvantages like the problem of having more work for teachers and being in close contact with the student as well as the hiring of teachers because this would rely solely in the program to assess the students and give out proper modules accordingly, with this in mind it is still really import to have human intervention to properly answer questions of students similar to the idea in [7].The only key factor in personalized learning is motivation. As stated in [8], Personalized learning is a powerful statement that has an objective to the student taking the process of gaining knowledge so the students would then need to think and reflect. Experiential learning is the practice grounds that learning is best done through experience. It involves students moving from theory to pure practice and experiencing them in real life.

Personalized learning is expensive and difficult to implement, but the emergence of technology has made it easier. Technology has made it more convenient for educators to develop and implement such learner-centered lessons[9]. Technology could solve the historic constraints of one to many teaching. Various algorithms could be developed and implemented to numerous mediums, and provide the same learning environment as a classroom setting [9]. Salman Khan, the founder of the massive online learning platform khanacademy.com, argued that a student should be remediated or be accelerated if they can in that scenario [10]. He argued that students should strive for mastery over the subject matter rather than being proficient at it. Personalized Learning solves this problem by meeting the student's individual learning needs while incorporating their interests and preferences [10]. This model takes into account individual student interests, learning needs and level of ability [10].

B. Objectives

- To write a program using Python Programming Language.
- To use "Tkinter" in Python to create a GUI for the program.
- To teach the users on how to program using Python, by using the principle of "Personalized Learning".
- To categorize the user's ability in programming.
- To create problem sets to test the user's knowledge in programming or coding in Python.

II. Questions for Interviewing the Students [Recommending Difficulty]

The program will recommend the difficulty of the problem set given to the user or the student. The program will recommend a difficulty by asking some questions about the student.

A. Did you have experience in coding?

1) If Yes:

a) How many programming languages did you use?

b) How much experience do you have in programming? Rate yourself from 1 – 5

c) When was the last time you had coded? In Number of Days

d) Did you learn to program formally (school, seminars, etc.) or by yourself?

i. If formal: How many programming classes have you had in the past?

2) If No:

a) How many programming languages have you heard of?

B. *If you were to make a program now, rate yourself about your level of confidence from 0 (lowest) to 5 (highest)?*

C. *Check whether the code has errors, write the line number that you think is incorrect: (Python Programming Language)*

```
01: items_correct = 0
02: total_items = 0
03: score_in_percent = 0
04: items_correct = input("Correct items: ")
05: total_items = input("Total items: ")
06: print ("score_in_percent")
07: score_in_percent = items_correct / total_items
```

D. *Do you think that you can code from scratch?*

E. *Rate yourself on about your interest in programming Rate yourself from 0 (lowest) – 5(highest)*

F. *At what stage or difficulty would you like to start programming (Beginner, Intermediate, Advanced)?*

III. Lessons to be Taught

A. *Introduction*

1) *Python Interpreters*
   a) *Anaconda*
   b) *PyCharm*
   c) *Geeks for Geeks Online Interpreter*
2) *Installing the Python Interpreters*
3) *Documentations / Manual*

B. *"return" Function*

C. *Modules*

D. *"print" Function*

E. *Variables*

F. *Basic Arithmetic Operations*

1) *Addition*
2) *Subtraction*
3) *Multiplication*
4) *Division*

G. *Data Types*

1) *String*
2) *List*
3) *Integer*
4) *Float*

H. *Operators*

1) *Arithmetic*
2) *Comparison*
3) *Logical*
4) *Membership*
5) *Identity*

I. *Palindrome Program*

J. *"len" Function*

K. *"global" Function*

L. *Using Imports and Numpy*

M. *Accessing and Creating Files*

N. *Logical (If) Statements*

O. *Looping Statements*

1) *For Loop*
2) *While Loop*

P. *"split" Function*

Q. *Text Formatting*

IV. Difficulty

This program is about personal learning, which means that the users, or the students, can learn programming in Python on their own pace. Since there are some students that already have knowledge or experience in programming, and some students who only just started programming, without personal learning, there would only be one problem set and some of the students may struggle or some may find it too easy to answer. This program has a seatwork where there are three (3) difficulties: Beginner, Intermediate, and Advanced.

A. *Beginner*

1) *The user has access to the "tips" button, where it shows some tips and reminders to help solve the problems.*
2) *The user can access the lecture notes while answering.*
3) *The user answers the problems by filling in the blanks.*

B. *Intermediate*

1) *The user has access to the "tips" button.*
2) *The user can access the lecture notes while answering the problem set.*
3) *The user answers the problems by coding from scratch, then the user will attach their code to the program for checking.*

C. *Advanced*

1) *The user cannot access the "tips" button.*
2) *The user cannot access the lecture notes while answering the problem.*
3) *The user user answers the problems by coding from scratch, then the user will attach their code to the program for checking.*

V. Problem Sets

A. *Problem Set 1*

1) *Beginner*
   a) *Write a program that returns "Hello World"*
   b) *Write a program that prints "Python".*
   c) *Write a program that returns "Hello World" It must be in a module named "X"*
      i. *If a plus b is c, how do you write this in Python? [Addition]*
      ii. *If a minus b is c, how do you write this in Python? [Subtraction]*

iii. *If a multiplied by b is c, how do you write this in Python? [Multiply]*

iv. *If a divided by b is c, how do you write this in Python? [Division]*

2) *Intermediate and Advanced*

a) *Write a program that returns "Hello World" It must be in a module named "X".*

b) *Write a program that prints and returns "Python!!!" It must be in a module named "program"*

c) *Write a program that can add the given values and return it. [module name: add]*

d) *Write a program that can subtract the given values and return it. [module name: subtract]*

e) *Write a program that can multiply the given values and return it. [module name: multiply]*

f) *Write a program that can divide the given values and return it. [module name: divide]*

g) *Items c) to f) must be in their own module with arguments. Ex. "a+b" → "add(a,b)"*

B. *Problem Set 2*

1) *Beginner*

a) *If a modulo b is c, how do you write this in Python?*

b) *If a square is b, how do you write this in Python?*

c) *Create a list and put it in a variable named "my_list", the values of the list are 2, 4, 6, 8, and 10 respectively.*

d) *Call the first element or item of the list you previously made.*

e) *Use the "len" function to give the number of elements in the list you previously made (my_list).*

f) *Write a program that is an integer data type with the variable "num1".*

g) *Write a program that is a float data type with the variable "num2".*

h) *The variable "a" must be declared as a "global variable".*

i) *Return the palindrome of the given string. The string is "message"*

2) *Intermediate and Advanced*

a) *Create a module with arguments that can modulo the given values, and return it. [module name: modulo]*

b) *Create a module with arguments that squares the given value, and return it. [module name: squared]*

c) *Create a module named "lists"; it must return a list with 5 integers, the integers are: 2, 4, 6, 8, and 10 respectively. The list must be declared as a global variable.*

d) *Create a module named "y", then call the 1st element from the list you made in the previous item.*

e) *Create a module named "lengths", use the "len" function and return the number of elements from the list you made in the previous item.*

f) *Create a module named "num1", it must return an integer data type.*

g) *Create a module named "num2", it must return a float data type.*

h) *Write a Python program that returns the palindrome of the given string. It must be in a module with arguments, the module must be named "palindrome".*

C. *Problem Set 3*

1) *Beginner*

a) *Write a program that returns a character or string that is repeated by a specific value it must be on a module with arguments. Arguments must be in this order: character, number of times repeated. Default value for the character is "A" and default number of times is 5*

b) *Write a program that modules a number, only by using addition, subtraction, multiplication, or division.*

i. *While loops/statements can only be used.*

ii. *This must be on a module with arguments, and the module named "modulo" and if "a modulo b" the arguments should be "(a,b)"*

iii. *This problem must be in a module named "problem1".*

c) *Write a program that square roots the given numbers using Numpy. This must be in a module with arguments and the module named as "importing".*

d) *Create a module that finds and counts the word "right" in the given paragraph, and return the result.*

i. *Use "if" statements & "for" statements only.*

ii. *The program should also find and count the word regardless of the case or capitalization.*

iii. *It must be on a module named "Paragraph".*

e) *Create a module that creates a text (.txt) file.*

i. *The contents of the file is a paragraph.*

ii. *Create a module that reads the text (.txt) file created from the previous item and return the contents of the file.*

iii. *It must be in a module named "creating".*

f) *Create a module that reads the text (.txt) file created from the previous item and return the contents of the file. It must be in a module named "accessing".*

2) *Intermediate and Advanced*

a) *Create a module that returns a character or string that is repeated by a specific value.*

i. *It must be on a module with arguments.*

ii. *Arguments must be in this order: character, number of times repeated respectively.*

iii. *Default value for the character is "A" and the default number of times is 5.*

This module must be named "problem1"

b) *Write a program that modulus a number, only by using addition, subtraction, multiplication, or division.*

i. *While loops/statements can only be used.*

ii. *This must be on a module with arguments, and named "modulo"*

iii. *Arguments must be in this order: dividend, divisor.*

c) Create a module that returns square roots from the given numbers using Numpy. This must be in a module with arguments, and the module named as "importing"

d) Create a module that finds and counts the word "right" in the given paragraph, and return the result.

- *i. Use "if" statements & "for" statements only.*
- *ii. The program should also find and count the word regardless of the case or capitalization.*
- *iii. It must be on a module named "Paragraph".*

e) Create a module that creates a text (.txt) file.

- *iv. The contents of the file is a paragraph.*
- *v. Create a module that reads the text (.txt) file created from the previous item and return the contents of the file.*
- *vi. It must be in a module named "creating".*

f) Create a module that reads the text (.txt) file created from the previous item and return the contents of the file. It must be in a module named "accessing".

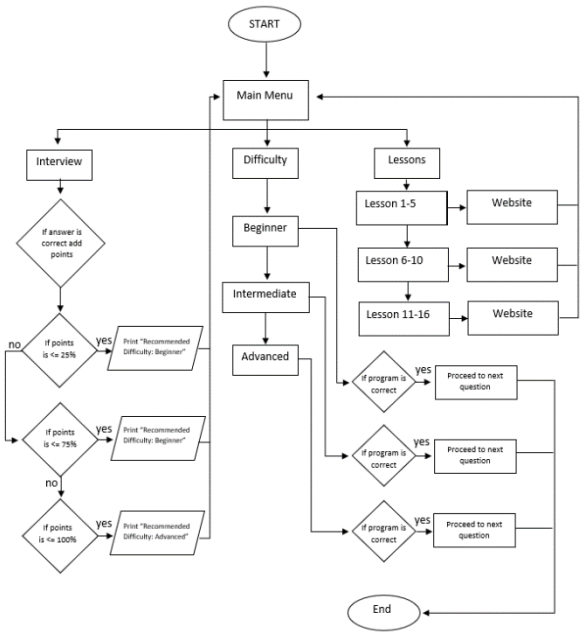
VI. Methodoloy

The software developers will make use of Python Programming Language to write a program about personalized learning. Furthermore, the programmers will make use of "Tkinter", a Python package add-on, to create a GUI or graphical user interface for the program.

A. IPO Chart

Input	Process	Output
• Interview Answers	• Answers are graded by a point system and computed to get percentage of a person's knowledge in coding.	• Determine the difficulty suggested to the user.
• Problem Set Code Answers	• Code will be checked for errors and syntax.	• Allow the user to proceed to the next problem.

B. Flowcharts



VII. Results

A. Screenshots

Figure 1: Main Menu

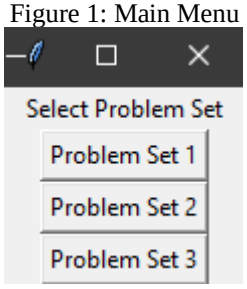


Figure 2: Problem Set Selection

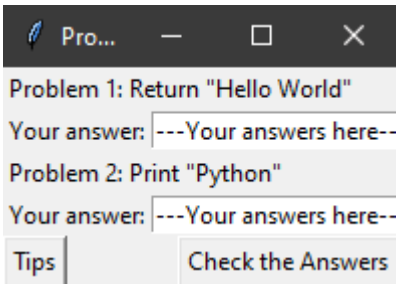


Figure 3: Beginner Problem Set 1 (Page 1)

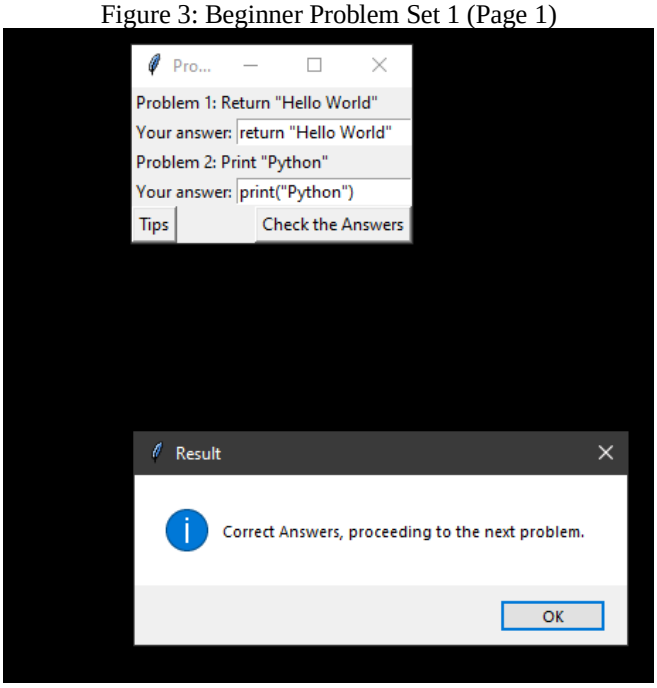


Figure 4: Beginner Problem Set 1 (Page 1) with Correct Answers Prompt

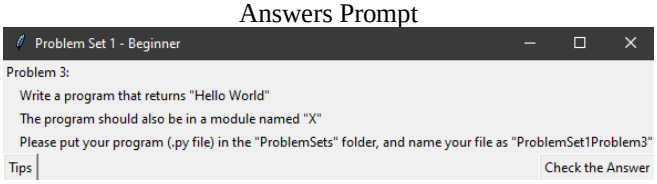


Figure 5: Beginner Problem Set 1 (Page 2)

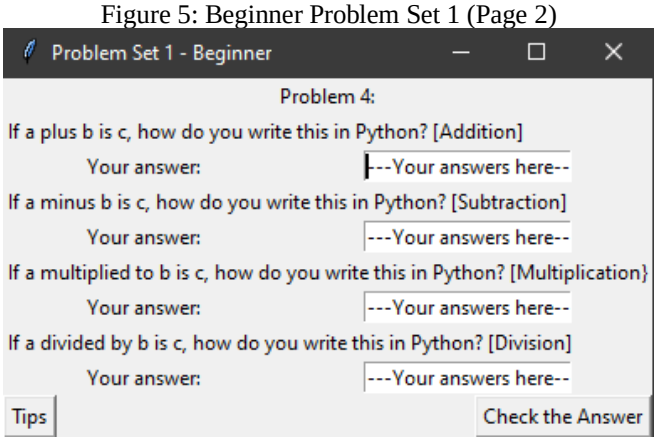


Figure 6: Beginner Problem Set 1 (Page 3)

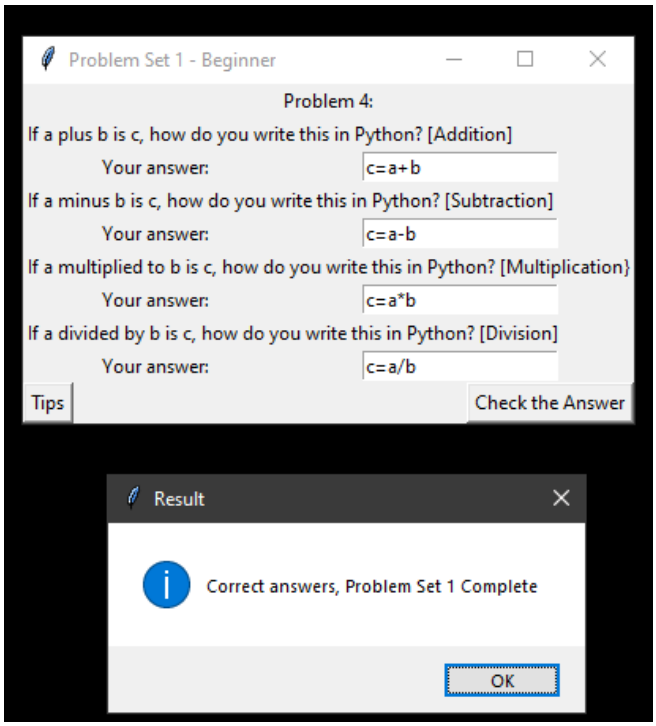


Figure 7: Beginner Problem Set 1 (Page 3) with Correct Answers Prompt

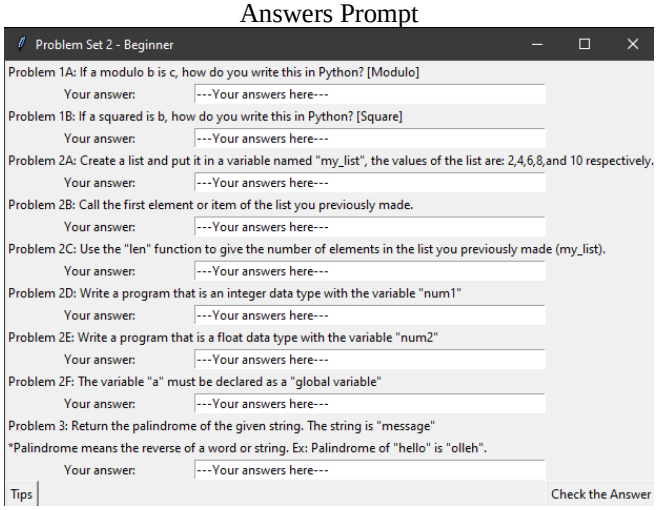


Figure 8: Beginner Problem Set 2

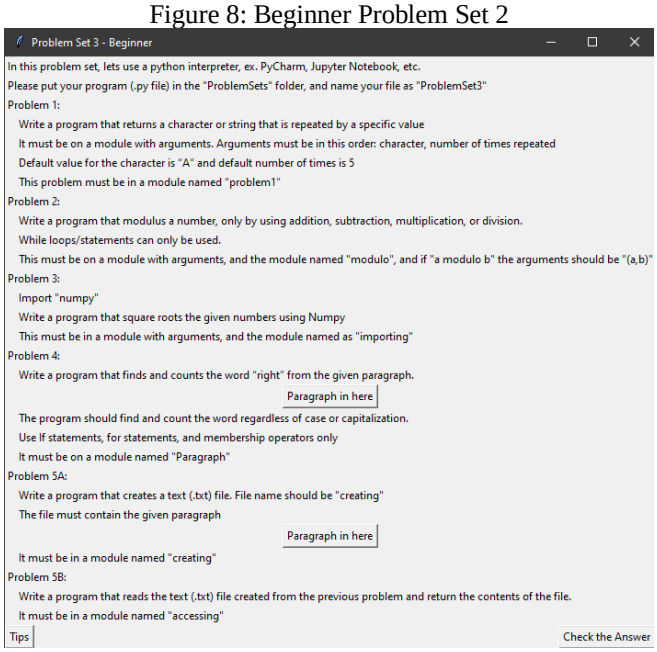


Figure 9: Beginner Problem Set 3

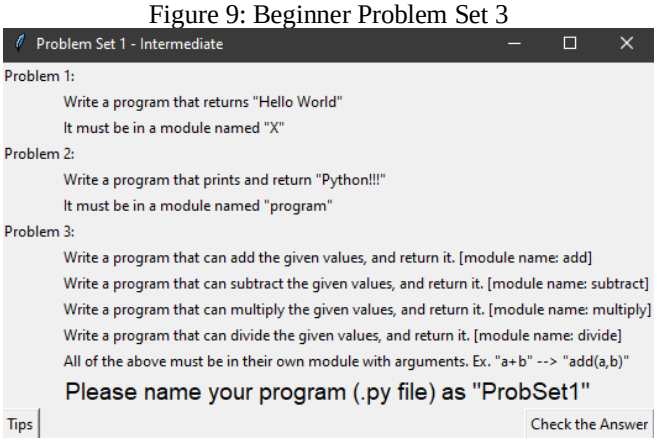


Figure 10: Intermediate Problem Set 1

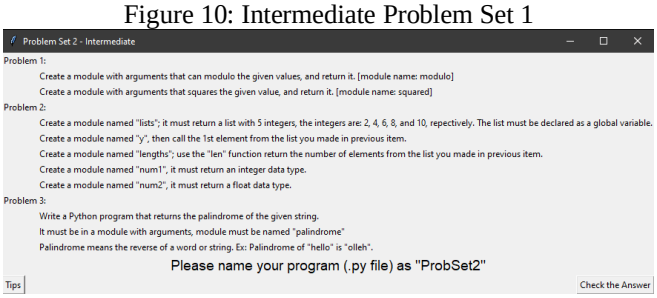


Figure 11: Intermediate Problem Set 2

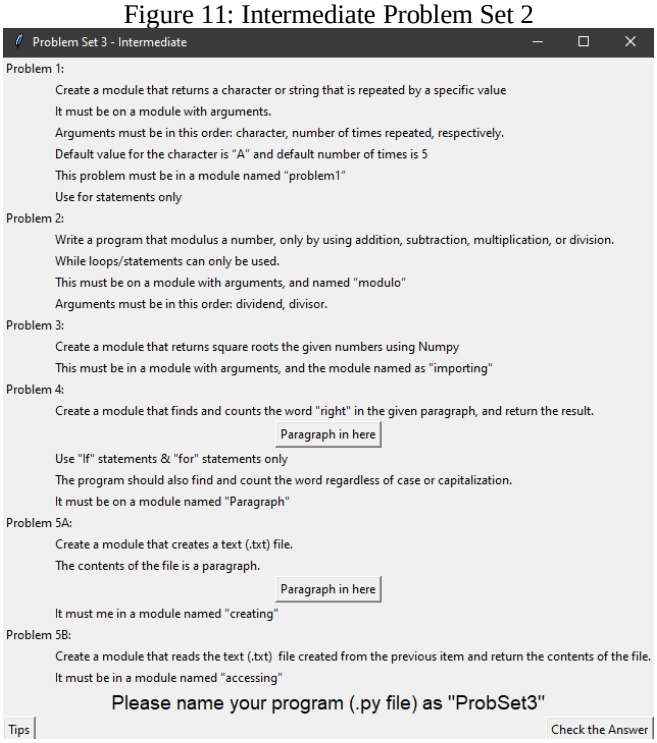


Figure 12: Intermediate Problem Set 3

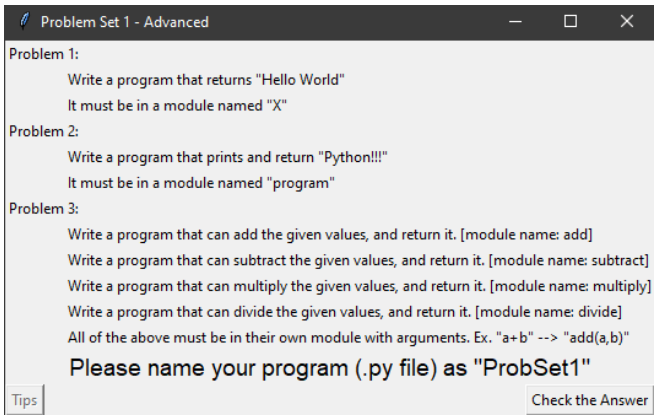


Figure 13: Advanced Problem Set 1

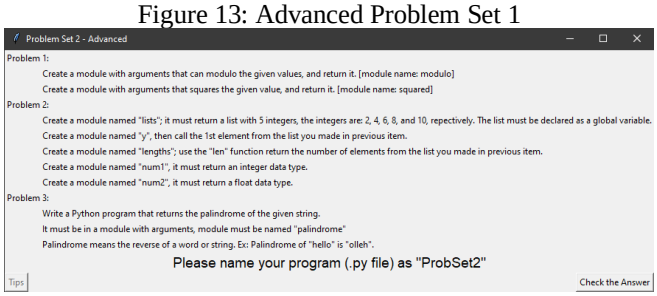


Figure 14: Advanced Problem Set 2

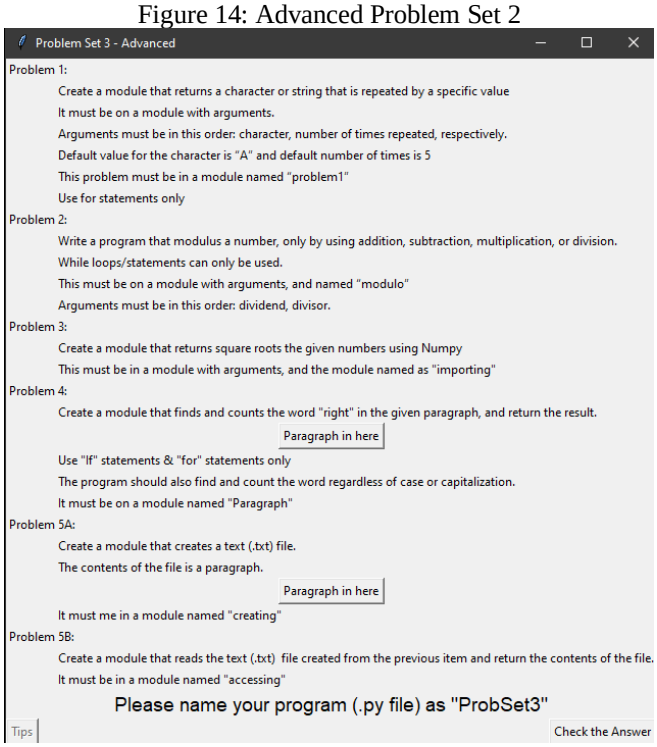


Figure 15: Advanced Problem Set 3

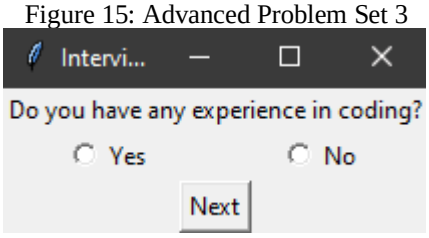


Figure 16: Interview Part 1

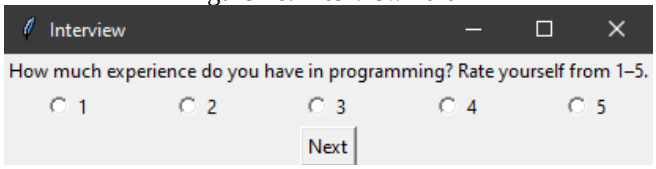


Figure 17: Interview Part 2

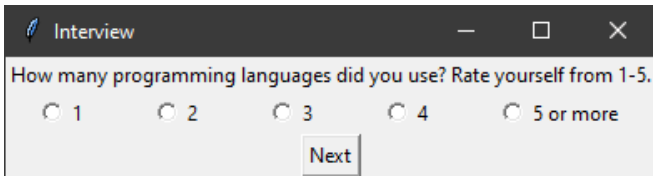


Figure 18: Interview Part 3

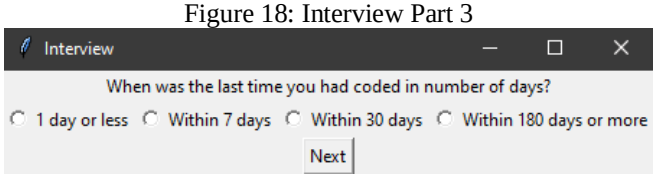


Figure 19: Interview Part 4

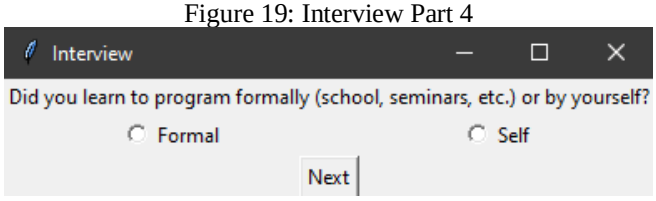


Figure 20: Interview Part 5

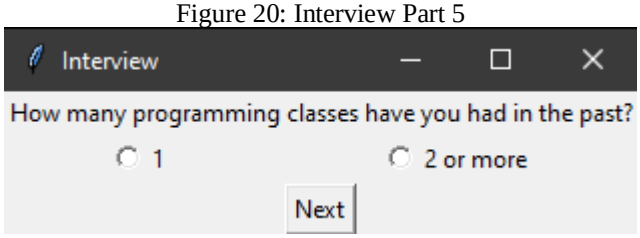


Figure 21: Interview Part 6

Figure 22: Interview Part 7

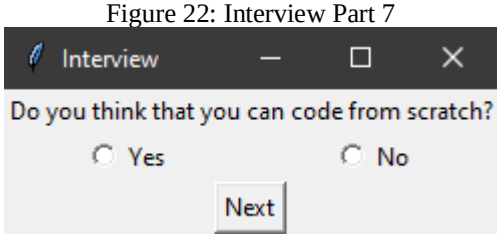


Figure 23: Interview Part 8

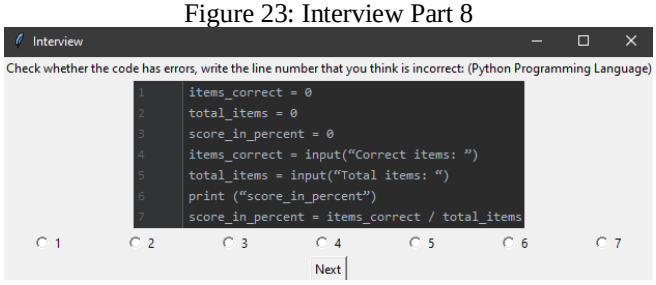


Figure 24: Interview Part 9

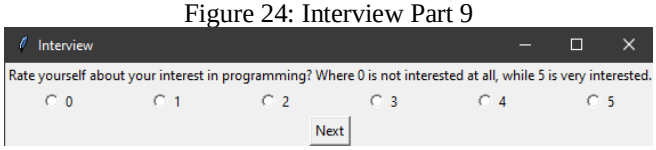


Figure 25: Interview Part 10

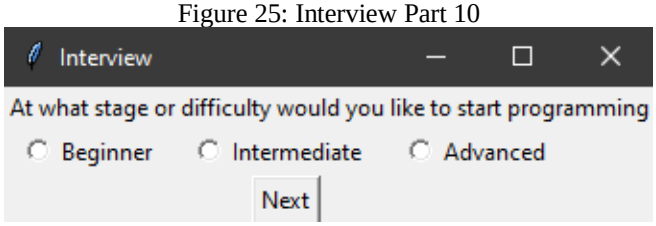


Figure 26: Interview Part 11

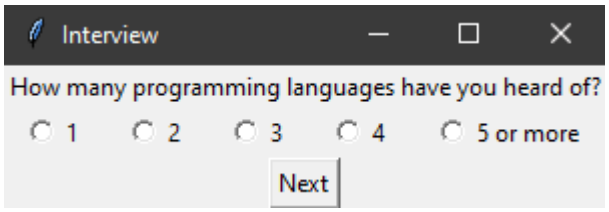


Figure 27: Interview Part 12

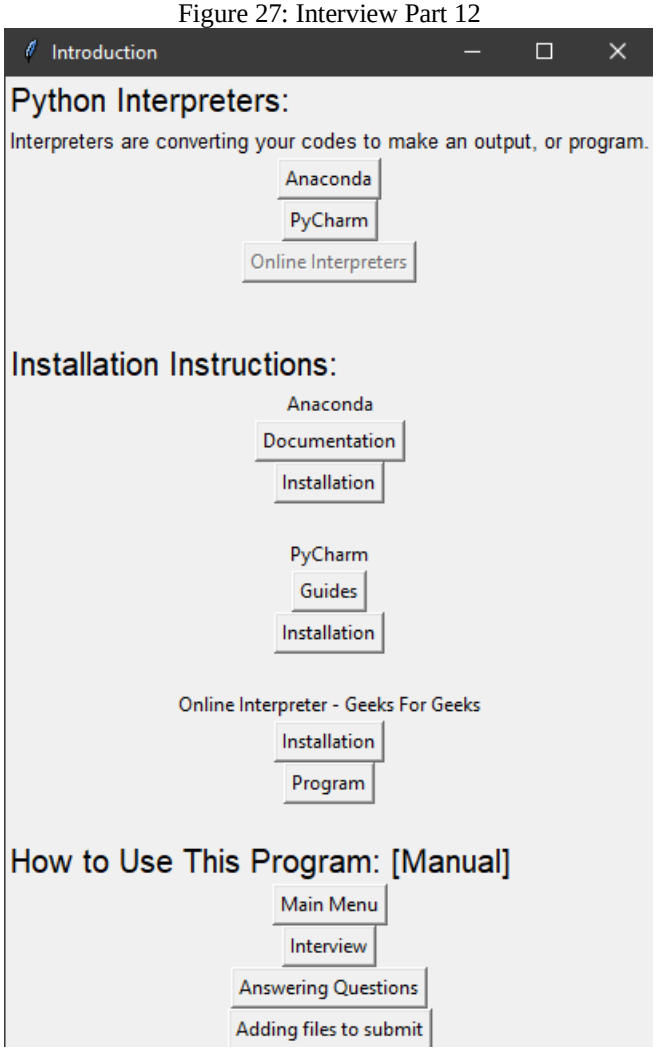


Figure 28: Introduction Lecture

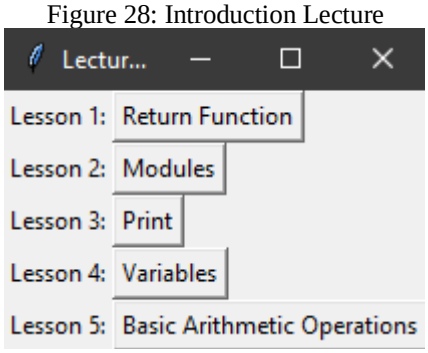


Figure 29: Lecture for Problem Set 1

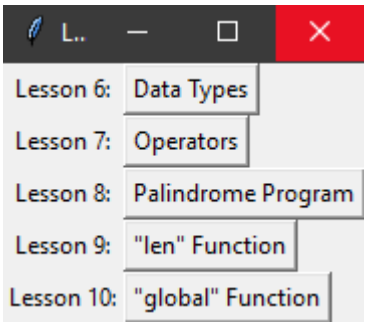


Figure 30: Lecture for Problem Set 2

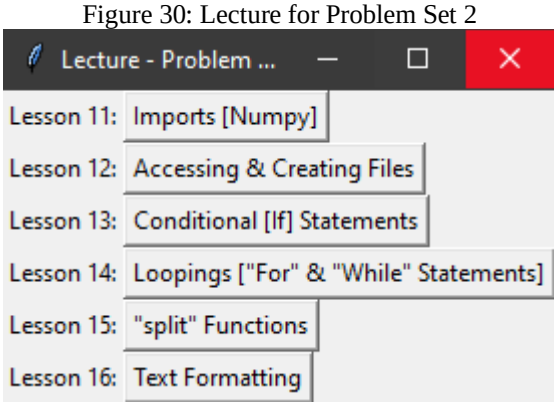


Figure 31: Lecture for Problem Set 3

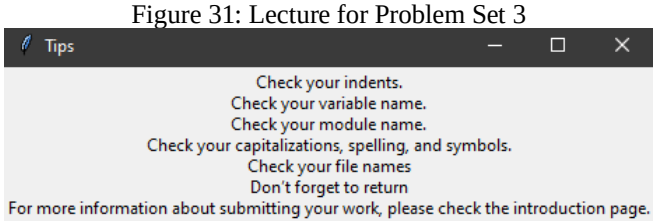


Figure 32: Tips Window

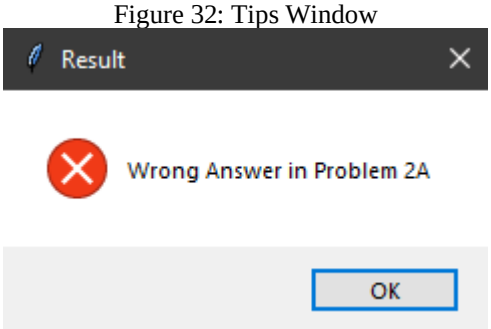


Figure 33: Wrong Answer Prompt

B. Codes

```
from tkinter import *
import tkinter.font as tkFont
from tkinter import messagebox
import os
import webbrowser
from importlib import reload
from PIL import ImageTk, Image
import numpy as np


def first_run():
    try:
        os.mkdir('ProblemSets')
    finally:
        return 0
```

```
def interview01():
    def com():
        value = r.get()
        global q_01, q_02, q_03, q_04, q_05, q_06, q_07
        if value == 1:
            q_01 = value
            q_07 = 0
            interview_w.destroy()
            interview02()
        elif value == 0:
            q_01 = value
            q_02 = q_03 = q_04 = q_05 = q_06 = 0
            interview_w.destroy()
            interview07()
        else:
            messagebox.showerror("Error", "Error")
            interview_w.destroy()
            interview01()

    r = IntVar()
    r.set(2)
    interview_w = Toplevel()
    interview_w.title("Interview")
    q = Label(interview_w, text="Do you have any
experience in coding?")
    q.grid(row=0, column=0, columnspan=2)
    Radiobutton(interview_w, text="Yes", variable=r,
value=1).grid(row=1, column=0)
    Radiobutton(interview_w, text="No", variable=r,
value=0).grid(row=1, column=1)
    qb = Button(interview_w, text="Next", command=com)
    qb.grid(row=2, column=0, columnspan=2)
    # max 1pt


def interview02():
    def com():
        value = r.get()
        global q_02
        if value >= 1 and value <= 5:
            q_02 = value
            interview_w.destroy()
            interview03()
        else:
            messagebox.showerror("Error", "Error")
            interview_w.destroy()
            interview02()

    r = IntVar()
    r.set(0)
    interview_w = Toplevel()
    interview_w.title("Interview")
    q = Label(interview_w, text="How much experience do
you have in programming? Rate yourself from 1–5.")
    q.grid(row=0, column=0, columnspan=5)
    Radiobutton(interview_w, text="1", variable=r,
value=1).grid(row=1, column=0)
    Radiobutton(interview_w, text="2", variable=r,
value=2).grid(row=1, column=1)
    Radiobutton(interview_w, text="3", variable=r,
value=3).grid(row=1, column=2)
```

```
    Radiobutton(interview_w, text="4", variable=r,
value=4).grid(row=1, column=3)
    Radiobutton(interview_w, text="5", variable=r,
value=5).grid(row=1, column=4)
    qb = Button(interview_w, text="Next", command=com)
    qb.grid(row=2, column=0, columnspan=5)
    # max 5pts


def interview03():
    def com():
        value = r.get()
        global q_03
        if value >= 1 and value <= 5:
            q_03 = value
            interview_w.destroy()
            interview04()
        else:
            messagebox.showerror("Error", "Error")
            interview_w.destroy()
            interview03()

    r = IntVar()
    r.set(0)
    interview_w = Toplevel()
    interview_w.title("Interview")
    q = Label(interview_w, text="How many programming
languages did you use? Rate yourself from 1-5.")
    q.grid(row=0, column=0, columnspan=5)
    Radiobutton(interview_w, text="1", variable=r,
value=1).grid(row=1, column=0)
    Radiobutton(interview_w, text="2", variable=r,
value=2).grid(row=1, column=1)
    Radiobutton(interview_w, text="3", variable=r,
value=3).grid(row=1, column=2)
    Radiobutton(interview_w, text="4", variable=r,
value=4).grid(row=1, column=3)
    Radiobutton(interview_w, text="5 or more", variable=r,
value=5).grid(row=1, column=4)
    qb = Button(interview_w, text="Next", command=com)
    qb.grid(row=2, column=0, columnspan=5)
    # max 5pts


def interview04():
    def com():
        value = r.get()
        global q_04
        if value >= 0 and value <= 3:
            q_04 = value
            interview_w.destroy()
            interview05()
        else:
            messagebox.showerror("Error", "Error")
            interview_w.destroy()
            interview04()

    r = IntVar()
    r.set(4)
    interview_w = Toplevel()
    interview_w.title("Interview")
    q = Label(interview_w, text="When was the last time you
had coded in number of days?")
    q.grid(row=0, column=0, columnspan=4)
    Radiobutton(interview_w, text="1 day or less",
variable=r, value=3).grid(row=1, column=0)
    Radiobutton(interview_w, text="Within 7 days",
variable=r, value=2).grid(row=1, column=1)
    Radiobutton(interview_w, text="Within 30 days",
variable=r, value=1).grid(row=1, column=2)
    Radiobutton(interview_w, text="Within 180 days or
more", variable=r, value=0).grid(row=1, column=3)
    qb = Button(interview_w, text="Next", command=com)
    qb.grid(row=2, column=0, columnspan=4)
    # max 3pts


def interview05():
    def com():
        value = r.get()
        global q_05, q_07
        if value == 1:
            q_05 = value
            interview_w.destroy()
            interview06()
        elif value == 0:
            q_05 = value
            q_07 = 0
            interview_w.destroy()
            interview08()
        else:
            messagebox.showerror("Error", "Error")
            interview_w.destroy()
            interview05()

    r = IntVar()
    r.set(2)
    interview_w = Toplevel()
    interview_w.title("Interview")
    q = Label(interview_w, text="Did you learn to program
formally (school, seminars, etc.) or by yourself?")
    q.grid(row=0, column=0, columnspan=2)
    Radiobutton(interview_w, text="Formal", variable=r,
value=1).grid(row=1, column=0)
    Radiobutton(interview_w, text="Self", variable=r,
value=0).grid(row=1, column=1)
    qb = Button(interview_w, text="Next", command=com)
    qb.grid(row=2, column=0, columnspan=2)
    # max 1pts


def interview06():
    def com():
        value = r.get()
        global q_06
        if value == 1:
            q_06 = value
            interview_w.destroy()
            interview08()
        elif value == 2:
            q_06 = value
            interview_w.destroy()
            interview08()
```

```
        else:
            messagebox.showerror("Error", "Error")
            interview_w.destroy()
            interview06()

    r = IntVar()
    r.set(0)
    interview_w = Toplevel()
    interview_w.title("Interview")
    q = Label(interview_w, text="How many programming
classes have you had in the past?")
    q.grid(row=0, column=0, columnspan=2)
    Radiobutton(interview_w, text="1", variable=r,
value=1).grid(row=1, column=0)
    Radiobutton(interview_w, text="2 or more", variable=r,
value=2).grid(row=1, column=1)
    qb = Button(interview_w, text="Next", command=com)
    qb.grid(row=2, column=0, columnspan=2)
    # max 2pts


def interview07():
    def com():
        value = r.get()
        global q_07
        if value >= 1 and value <= 4:
            q_07 = value
            interview_w.destroy()
            interview08()
        else:
            messagebox.showerror("Error", "Error")
            interview_w.destroy()
            interview07()

    r = IntVar()
    r.set(0)
    interview_w = Toplevel()
    interview_w.title("Interview")
    q = Label(interview_w, text="How many programming
languages have you heard of?")
    q.grid(row=0, column=0, columnspan=5)
    Radiobutton(interview_w, text="1", variable=r,
value=1).grid(row=1, column=0)
    Radiobutton(interview_w, text="2", variable=r,
value=2).grid(row=1, column=1)
    Radiobutton(interview_w, text="3", variable=r,
value=3).grid(row=1, column=2)
    Radiobutton(interview_w, text="4", variable=r,
value=4).grid(row=1, column=3)
    Radiobutton(interview_w, text="5 or more", variable=r,
value=5).grid(row=1, column=4)
    qb = Button(interview_w, text="Next", command=com)
    qb.grid(row=2, column=0, columnspan=5)
    # max 5pts


def interview08():
    def com():
        value = r.get()
        global q_08
        if value >= 0 and value <= 5:
            q_08 = value
            interview_w.destroy()
            interview09()
        else:
            messagebox.showerror("Error", "Error")
            interview_w.destroy()
            interview08()

    r = IntVar()
    r.set(6)
    interview_w = Toplevel()
    interview_w.title("Interview")
    q = Label(interview_w,
              text="If you were to make a program now, rate
yourself about your level of confidence from 0 (lowest) to 5
(highest)?")
    q.grid(row=0, column=0, columnspan=6)
    Radiobutton(interview_w, text="0", variable=r,
value=0).grid(row=1, column=0)
    Radiobutton(interview_w, text="1", variable=r,
value=1).grid(row=1, column=1)
    Radiobutton(interview_w, text="2", variable=r,
value=2).grid(row=1, column=2)
    Radiobutton(interview_w, text="3", variable=r,
value=3).grid(row=1, column=3)
    Radiobutton(interview_w, text="4", variable=r,
value=4).grid(row=1, column=4)
    Radiobutton(interview_w, text="5", variable=r,
value=5).grid(row=1, column=5)
    qb = Button(interview_w, text="Next", command=com)
    qb.grid(row=2, column=0, columnspan=6)
    # max 5pts


def interview09():
    def com():
        value = r.get()
        global q_09
        if value >= 1 and value <= 7:
            if value == 6 or value == 7:
                q_09 = 1
            else:
                q_09 = 0
            interview_w.destroy()
            interview10()
        else:
            messagebox.showerror("Error", "Error")
            interview_w.destroy()
            interview09()

    r = IntVar()
    r.set(0)
    interview_w = Toplevel()
    interview_w.title("Interview")
    q = Label(interview_w,
              text="Check whether the code has errors, write the
line number that you think is incorrect: (Python
Programming Language)")
    q.grid(row=0, column=0, columnspan=7)
    global q_09_img
    q_09_img =
ImageTk.PhotoImage(Image.open("Images/Interview/Q_09.
png"))
```

```
    Label(interview_w, image=q_09_img).grid(row=1,
column=0, columnspan=7)
    Radiobutton(interview_w, text="1", variable=r,
value=1).grid(row=2, column=0)
    Radiobutton(interview_w, text="2", variable=r,
value=2).grid(row=2, column=1)
    Radiobutton(interview_w, text="3", variable=r,
value=3).grid(row=2, column=2)
    Radiobutton(interview_w, text="4", variable=r,
value=4).grid(row=2, column=3)
    Radiobutton(interview_w, text="5", variable=r,
value=5).grid(row=2, column=4)
    Radiobutton(interview_w, text="6", variable=r,
value=6).grid(row=2, column=5)
    Radiobutton(interview_w, text="7", variable=r,
value=7).grid(row=2, column=6)
    qb = Button(interview_w, text="Next", command=com)
    qb.grid(row=3, column=0, columnspan=7)
    # max 1pt


def interview10():
    def com():
        value = r.get()
        global q_10
        if value == 1:
            q_10 = value
            interview_w.destroy()
            interview11()
        elif value == 0:
            q_10 = value
            interview_w.destroy()
            interview11()
        else:
            messagebox.showerror("Error", "Error")
            interview_w.destroy()
            interview10()

    r = IntVar()
    r.set(2)
    interview_w = Toplevel()
    interview_w.title("Interview")
    q = Label(interview_w, text="Do you think that you can
code from scratch?")
    q.grid(row=0, column=0, columnspan=2)
    Radiobutton(interview_w, text="Yes", variable=r,
value=1).grid(row=1, column=0)
    Radiobutton(interview_w, text="No", variable=r,
value=0).grid(row=1, column=1)
    qb = Button(interview_w, text="Next", command=com)
    qb.grid(row=2, column=0, columnspan=2)
    # max 1pt


def interview11():
    def com():
        value = r.get()
        global q_11
        if value >= 0 and value <= 5:
            q_11 = value
            interview_w.destroy()
            interview12()
        else:
            messagebox.showerror("Error", "Error")
            interview_w.destroy()
            interview11()

    r = IntVar()
    r.set(6)
    interview_w = Toplevel()
    interview_w.title("Interview")
    q = Label(interview_w,
              text="Rate yourself about your interest in
programming? Where 0 is not interested at all, while 5 is
very interested.")
    q.grid(row=0, column=0, columnspan=6)
    Radiobutton(interview_w, text="0", variable=r,
value=0).grid(row=1, column=0)
    Radiobutton(interview_w, text="1", variable=r,
value=1).grid(row=1, column=1)
    Radiobutton(interview_w, text="2", variable=r,
value=2).grid(row=1, column=2)
    Radiobutton(interview_w, text="3", variable=r,
value=3).grid(row=1, column=3)
    Radiobutton(interview_w, text="4", variable=r,
value=4).grid(row=1, column=4)
    Radiobutton(interview_w, text="5", variable=r,
value=5).grid(row=1, column=5)
    qb = Button(interview_w, text="Next", command=com)
    qb.grid(row=2, column=0, columnspan=6)
    # max 5pts


def interview12():
    def com():
        value = r.get()
        global q_12
        if value >= 0 and value <= 5:
            q_12 = value
            interview_w.destroy()
            recommend_diff()
        else:
            messagebox.showerror("Error", "Error")
            interview_w.destroy()
            interview12()

    r = IntVar()
    r.set(3)
    interview_w = Toplevel()
    interview_w.title("Interview")
    q = Label(interview_w, text="At what stage or difficulty
would you like to start programming")
    q.grid(row=0, column=0, columnspan=6)
    Radiobutton(interview_w, text="Beginner", variable=r,
value=0).grid(row=1, column=0)
    Radiobutton(interview_w, text="Intermediate",
variable=r, value=1).grid(row=1, column=1)
    Radiobutton(interview_w, text="Advanced", variable=r,
value=2).grid(row=1, column=2)
    qb = Button(interview_w, text="Next", command=com)
    qb.grid(row=2, column=0, columnspan=3)
    # max 2pts
```

```
def recommend_diff():
    r_diff = Toplevel()
    r_diff.title("Interview - Results")
    q_total = q_01 + q_02 + q_03 + q_04 + q_05 + q_06 +
q_07 + q_08 + q_09 + q_10 + q_11 + q_12
    if q_total >= 23 and q_total <= 31:  # 75%-100%
        diff = "Advanced"
    elif q_total >= 8 and q_total < 23:  # 25%-75%
        diff = "Intermediate"
    elif q_total >= 0 and q_total < 8:  # 0%-25%
        diff = "Beginner"
    else:
        diff = "Error"
    Label(r_diff, text="I recommend the following
difficulty:").pack()
    Label(r_diff, text=diff).pack()
    # all 36pts, all max 31 pts
    # print(q_total, q_01, q_02, q_03, q_04, q_05, q_06,
q_07, q_08, q_09, q_10, q_11, q_12)  # for testing


def sets():
    def close_sets():
        problem_sets_w.destroy()

    problem_sets_w = Toplevel()
    problem_sets_w.title("Problem Set Selection")
    Label(problem_sets_w, text="Select Problem Set").pack()
    if diff == "Beginner":
        Button(problem_sets_w, text="Problem Set 1",
command=lambda: [f() for f in [beginner_set1_1,
close_sets]]).pack()
        Button(problem_sets_w, text="Problem Set 2",
command=lambda: [f() for f in [beginner_set2_1,
close_sets]]).pack()
        Button(problem_sets_w, text="Problem Set 3",
command=lambda: [f() for f in [beginner_set3_1,
close_sets]]).pack()
    elif diff == "Intermediate" or diff == "Advanced":
        Button(problem_sets_w, text="Problem Set 1",
command=lambda: [f() for f in [uni_set1,
close_sets]]).pack()
        Button(problem_sets_w, text="Problem Set 2",
command=lambda: [f() for f in [uni_set2,
close_sets]]).pack()
        Button(problem_sets_w, text="Problem Set 3",
command=lambda: [f() for f in [uni_set3,
close_sets]]).pack()


def beginner():
    try:
        os.mkdir('ProblemSets/Beginner')
    except FileExistsError:
        nothing = None
        # print("\"ProblemSets/Beginner\" Already Exists")
    finally:
        global diff
        diff = "Beginner"
        intermediate_b.config(state=DISABLED)
        advanced_b.config(state=DISABLED)
        interview_b.config(state=DISABLED)
        sets()
        # print("Folder \"ProblemSets/Beginner\" Created")


def intermediate():
    global diff
    diff = "Intermediate"
    beginner_b.config(state=DISABLED)
    advanced_b.config(state=DISABLED)
    interview_b.config(state=DISABLED)
    sets()


def advanced():
    global diff
    diff = "Advanced"
    intermediate_b.config(state=DISABLED)
    beginner_b.config(state=DISABLED)
    interview_b.config(state=DISABLED)
    sets()


def beginner_set1_1():
    def chk():
        a = p1.get()
        b = p2.get()
        if a == "" or b == "" or a == "---Your answers here---"
or b == "---Your answers here---":
            messagebox.showerror("Result", "I cannot accept
blank answers.")
        else:
            fname = "ProblemSets/Beginner/s1p1.py"
            data = """def A():
                """ + str(a)
            with open(fname, "w") as f:
                f.write(data)

            from ProblemSets.Beginner import s1p1
            reload(s1p1)

            try:
                s1p1.A()
                assert s1p1.A() == "Hello World"
                correct1 = 1
            except:
                correct1 = 0
                messagebox.showerror("Result", "Wrong answer
in Problem 1")

            try:
                correct2 = 1
                assert b == "print(\"Python\")"
            except:
                messagebox.showerror("Result", "Wrong Answer
in Problem 2")
                correct2 = 0

            if correct1 == 1 and correct2 == 1:
                messagebox.showinfo("Result", "Correct
Answers, proceeding to the next problem.")
                beginner_w.destroy()
                beginner_set1_2()
```

```
    beginner_w = Toplevel()
    beginner_w.title("Problem Set 1 - Beginner")
    Label(beginner_w, text="Problem 1: Return \"Hello
World\"").grid(row=0, column=0, columnspan=2,
sticky=W)
    Label(beginner_w, text="Your answer:").grid(row=1,
column=0)
    p1 = Entry(beginner_w)
    p1.grid(row=1, column=1)
    p1.insert(0, "---Your answers here---")
    Label(beginner_w, text="Problem 2: Print
\"Python\"").grid(row=2, column=0, columnspan=2,
sticky=W)
    Label(beginner_w, text="Your answer:").grid(row=3,
column=0)
    p2 = Entry(beginner_w)
    p2.grid(row=3, column=1)
    p2.insert(0, "---Your answers here---")
    Button(beginner_w, text="Tips",
command=tips).grid(row=4, column=0, sticky=W)
    Button(beginner_w, text="Check the Answers",
command=chk).grid(row=4, column=1, sticky=E)


def beginner_set1_2():
    def chk():
        try:
            from ProblemSets import ProblemSet1Problem3
            reload(ProblemSet1Problem3)
        except:
            messagebox.showerror("Result", "Cannot find the
file.")

        try:
            ProblemSet1Problem3.X()
            assert ProblemSet1Problem3.X() == "Hello World"
            messagebox.showinfo("Result", "Correct Answer,
proceeding to the next problem.")
            beginner_set1_3()
            beginner_w.destroy()
        except:
            messagebox.showerror("Result", "Wrong Answer")

    beginner_w = Toplevel()
    beginner_w.title("Problem Set 1 - Beginner")
    Label(beginner_w, text="Problem 3:").grid(row=0,
column=0, columnspan=2, sticky=W)
    Label(beginner_w, text="    Write a program that returns
\"Hello World\"").grid(row=1, column=0, columnspan=2,
sticky=W)
    Label(beginner_w, text="    The program should also be
in a module named \"X\"").grid(row=2, column=0,
columnspan=2, sticky=W)
    Label(beginner_w, text="    Please put your program (.py
file) in the \"ProblemSets\" folder, and name your file as
\"ProblemSet1Problem3\"").grid(row=3, column=0,
columnspan=2)  # need tutorial attach submission
    Button(beginner_w, text="Tips",
command=tips).grid(row=4, column=0, sticky=W)
    Button(beginner_w, text="Check the Answer",
command=chk).grid(row=4, column=1, sticky=E)

def beginner_set1_3():
    def chk():
        a = p1.get()
        b = p2.get()
        c = p3.get()
        d = p4.get()
        if a == "" or b == "" or c == "" or d == "" or a == "---
Your answers here---" or b == "---Your answers here---" or
c == "---Your answers here---" or d == "---Your answers
here---":
            messagebox.showerror("Result", "I cannot accept
blank answers.")
        else:
            fname = "ProblemSets/Beginner/s1p3a.py"
            data = """def pt1(a,b):\n    """ + str(a) + "\n    " +
"return c"
            with open(fname, "w") as f:
                f.write(data)

            fname = "ProblemSets/Beginner/s1p3b.py"
            data = """def pt2(a,b):\n    """ + str(b) + "\n    " +
"return c"
            with open(fname, "w") as f:
                f.write(data)

            fname = "ProblemSets/Beginner/s1p3c.py"
            data = """def pt3(a,b):\n    """ + str(c) + "\n    " +
"return c"
            with open(fname, "w") as f:
                f.write(data)

            fname = "ProblemSets/Beginner/s1p3d.py"
            data = """def pt4(a,b):\n    """ + str(d) + "\n    " +
"return c"
            with open(fname, "w") as f:
                f.write(data)

            from ProblemSets.Beginner import s1p3a
            reload(s1p3a)
            try:
                s1p3a.pt1(1,2)
                assert s1p3a.pt1(1,2) == 1+2
                correct1 = 1
            except:
                correct1 = 0
                messagebox.showerror("Result", "Wrong Answer
in addition")

            from ProblemSets.Beginner import s1p3b
            reload(s1p3b)
            try:
                s1p3b.pt2(10,2)
                assert s1p3b.pt2(10,2) == 10-2
                correct2 = 1
            except:
                correct2 = 0
                messagebox.showerror("Result", "Wrong Answer
in subtraction")

            from ProblemSets.Beginner import s1p3c
```

```
        reload(s1p3c)
        try:
            s1p3c.pt3(5,2)
            assert s1p3c.pt3(5,2) == 5*2
            correct3 = 1
        except:
            correct3 = 0
            messagebox.showerror("Result", "Wrong Answer
in multiplication")

        from ProblemSets.Beginner import s1p3d
        reload(s1p3d)
        try:
            s1p3d.pt4(3,2)
            assert s1p3d.pt4(3,2) == 3/2
            correct4 = 1
        except:
            correct4 = 0
            messagebox.showerror("Result", "Wrong Answer
in division")

        if correct1 == 1 and correct2 == 1 and correct3 == 1
and correct4 == 1:
            messagebox.showinfo("Result", "Correct answers,
Problem Set 1 Complete")
            sets()
            beginner_w.destroy()
            beginner_b.config(state=NORMAL)
            intermediate_b.config(state=NORMAL)
            advanced_b.config(state=NORMAL)
            interview_b.config(state=NORMAL)

    beginner_w = Toplevel()
    beginner_w.title("Problem Set 1 - Beginner")
    Label(beginner_w, text="Problem 4:").grid(row=0,
column=0, columnspan=2)
    Label(beginner_w, text="If a plus b is c, how do you
write this in Python? [Addition]").grid(row=1, column=0,
columnspan=2, sticky=W)
    p1 = Entry(beginner_w)
    Label(beginner_w, text="Your answer:").grid(row=2,
column=0)
    p1.grid(row=2, column=1)
    p1.insert(0, "---Your answers here---")
    Label(beginner_w, text="If a minus b is c, how do you
write this in Python? [Subtraction]").grid(row=3, column=0,
columnspan=2, sticky=W)
    p2 = Entry(beginner_w)
    Label(beginner_w, text="Your answer:").grid(row=4,
column=0)
    p2.grid(row=4, column=1)
    p2.insert(0, "---Your answers here---")
    Label(beginner_w, text="If a multiplied to b is c, how do
you write this in Python? [Multiplication}").grid(row=5,
column=0, columnspan=2, sticky=W)
    p3 = Entry(beginner_w)
    Label(beginner_w, text="Your answer:").grid(row=6,
column=0)
    p3.grid(row=6, column=1)
    p3.insert(0, "---Your answers here---")
    Label(beginner_w, text="If a divided by b is c, how do
you write this in Python? [Division]").grid(row=7,
column=0, columnspan=2, sticky=W)
    p4 = Entry(beginner_w)
    Label(beginner_w, text="Your answer:").grid(row=8,
column=0)
    p4.grid(row=8, column=1)
    p4.insert(0, "---Your answers here---")
    Button(beginner_w, text="Tips",
command=tips).grid(row=9, column=0, sticky=W)
    Button(beginner_w, text="Check the Answer",
command=chk).grid(row=9, column=1, sticky=E)


def beginner_set2_1():
    def chk():
        a = p1.get()
        b = p2.get()
        c = p3.get()
        d = p4.get()
        e = p5.get()
        f_ = p6.get()
        g = p7.get()
        h = p8.get()
        i = p9.get()
        if a == "" or b == "" or c == "" or d == "" or a == "---
Your answers here---" or b == "---Your answers here---" or
c == "---Your answers here---" or d == "---Your answers
here---":
            messagebox.showerror("Result", "I cannot accept
blank answers.")
        else:
            fname = "ProblemSets/Beginner/s2p1a.py"
            data = "def pt1(a,b):\n    " + str(a) + "\n    " + "return
c"
            with open(fname, "w") as f:
                f.write(data)
            from ProblemSets.Beginner import s2p1a
            reload(s2p1a)
            try:
                s2p1a.pt1(8,2)
                assert s2p1a.pt1(8,2) == 8 % 2
                correct1 = 1
            except:
                correct1 = 0
                messagebox.showerror("Result", "Wrong Answer
in Problem 1A")

            fname = "ProblemSets/Beginner/s2p1b.py"
            data = "def pt2(a):\n    " + str(b) + "\n    " + "return
b"
            with open(fname, "w") as f:
                f.write(data)
            from ProblemSets.Beginner import s2p1b
            reload(s2p1b)
            try:
                s2p1b.pt2(2)
                assert s2p1b.pt2(2) == 2**2
                correct2 = 1
            except:
                correct2 = 0
```

```
            messagebox.showerror("Result", "Wrong Answer
in Problem 1B")

        fname = "ProblemSets/Beginner/s2p2a.py"
        data = "def pt3():\n    " + str(c) + "\n    return
my_list"
        with open(fname, "w") as f:
            f.write(data)
        from ProblemSets.Beginner import s2p2a
        reload(s2p2a)
        try:
            s2p2a.pt3()
            assert s2p2a.pt3() == [2, 4, 6, 8, 10]
            assert len(s2p2a.pt3()) == 5
            correct3 = 1
        except:
            correct3 = 0
            messagebox.showerror("Result", "Wrong Answer
in Problem 2A")

        fname = "ProblemSets/Beginner/s2p2b.py"
        data = "def pt4():\n    " + "my_list = [2,4,6,8,10]\n
" + "return " + str(d)
        with open(fname, "w") as f:
            f.write(data)
        from ProblemSets.Beginner import s2p2b
        reload(s2p2b)
        try:
            s2p2b.pt4()
            assert s2p2b.pt4() == 2
            correct4 = 1
        except:
            correct4 = 0
            messagebox.showerror("Result", "Wrong Answer
in Problem 2B")

        fname = "ProblemSets/Beginner/s2p2c.py"
        data = "def pt5():\n    " + "my_list = [2,4,6,8,10]\n
" + "return " + str(e)
        with open(fname, "w") as f:
            f.write(data)
        from ProblemSets.Beginner import s2p2c
        reload(s2p2c)
        try:
            s2p2c.pt5()
            assert s2p2c.pt5() == 5
            correct5 = 1
        except:
            correct5 = 0
            messagebox.showerror("Result", "Wrong Answer
in Problem 2C")

        fname = "ProblemSets/Beginner/s2p2d.py"
        data = "def pt6():\n    " + str(f_) + "\n    return num1"
        with open(fname, "w") as f:
            f.write(data)
        from ProblemSets.Beginner import s2p2d
        reload(s2p2d)
        try:
            s2p2d.pt6()
            assert type(s2p2d.pt6()) == int
            correct6 = 1
        except:
            correct6 = 0
            messagebox.showerror("Result", "Wrong Answer
in Problem 2D")

        fname = "ProblemSets/Beginner/s2p2e.py"
        data = "def pt7():\n    " + str(g) + "\n    return num2"
        with open(fname, "w") as f:
            f.write(data)
        from ProblemSets.Beginner import s2p2e
        reload(s2p2e)
        try:
            s2p2e.pt7()
            assert type(s2p2e.pt7()) == float
            correct7 = 1
        except:
            correct7 = 0
            messagebox.showerror("Result", "Wrong Answer
in Problem 2E")

        try:
            assert h == "global a"
            correct8 = 1
        except:
            correct8 = 0
            messagebox.showerror("Result", "Wrong Answer
in Problem 2F")

        fname = "ProblemSets/Beginner/s2p3.py"
        data = "def pt9():\n    " + "return " + str(i)
        with open(fname, "w") as f:
            f.write(data)
        from ProblemSets.Beginner import s2p3
        reload(s2p3)
        try:
            s2p3.pt9()
            assert s2p3.pt9() == "message"[::-1]
            correct9 = 1
        except:
            correct9 = 0
            messagebox.showerror("Result", "Wrong Answer
in Problem 3")

        if correct1 == 1 and correct2 == 1 and correct3 == 1
and correct4 == 1 and correct5 == 1 and correct6 == 1 and
correct7 == 1 and correct8 == 1 and correct9 == 1:
            messagebox.showinfo("Result", "Correct answers,
Problem Set 2 Complete")
            sets()
            beginner_w.destroy()
            beginner_b.config(state=NORMAL)
            intermediate_b.config(state=NORMAL)
            advanced_b.config(state=NORMAL)
            interview_b.config(state=NORMAL)

    beginner_w = Toplevel()
    beginner_w.title("Problem Set 2 - Beginner")
    Label(beginner_w, text="Problem 1A: If a modulo b is c,
how do you write this in Python? [Modulo]").grid(row=0,
column=0, columnspan=2, sticky=W)
    p1 = Entry(beginner_w, width=55)
```

```
    Label(beginner_w, text="Your answer:").grid(row=1, column=0)
    p1.grid(row=1, column=1, sticky=W)
    p1.insert(0, "---Your answers here---")
    Label(beginner_w, text="Problem 1B: If a squared is b, how do you write this in Python? [Square]").grid(row=2, column=0, columnspan=2, sticky=W)
    p2 = Entry(beginner_w, width=55)
    Label(beginner_w, text="Your answer:").grid(row=3, column=0)
    p2.grid(row=3, column=1, sticky=W)
    p2.insert(0, "---Your answers here---")
    Label(beginner_w, text="Problem 2A: Create a list and put it in a variable named \"my_list\", the values of the list are: 2,4,6,8,and 10 respectively.").grid(row=4, column=0, columnspan=2, sticky=W)
    p3 = Entry(beginner_w, width=55)
    Label(beginner_w, text="Your answer:").grid(row=5, column=0)
    p3.grid(row=5, column=1, sticky=W)
    p3.insert(0, "---Your answers here---")
    Label(beginner_w, text="Problem 2B: Call the first element or item of the list you previously made.").grid(row=6, column=0, columnspan=2, sticky=W)
    p4 = Entry(beginner_w, width=55)
    Label(beginner_w, text="Your answer:").grid(row=7, column=0)
    p4.grid(row=7, column=1, sticky=W)
    p4.insert(0, "---Your answers here---")
    Label(beginner_w, text="Problem 2C: Use the \"len\" function to give the number of elements in the list you previously made (my_list).").grid(row=8, column=0, columnspan=2, sticky=W)
    p5 = Entry(beginner_w, width=55)
    Label(beginner_w, text="Your answer:").grid(row=9, column=0)
    p5.grid(row=9, column=1, sticky=W)
    p5.insert(0, "---Your answers here---")
    Label(beginner_w, text="Problem 2D: Write a program that is an integer data type with the variable \"num1\"").grid(row=10, column=0, columnspan=2, sticky=W)
    p6 = Entry(beginner_w, width=55)
    Label(beginner_w, text="Your answer:").grid(row=11, column=0)
    p6.grid(row=11, column=1, sticky=W)
    p6.insert(0, "---Your answers here---")
    Label(beginner_w, text="Problem 2E: Write a program that is a float data type with the variable \"num2\"").grid(row=12, column=0, columnspan=2, sticky=W)
    p7 = Entry(beginner_w, width=55)
    Label(beginner_w, text="Your answer:").grid(row=13, column=0)
    p7.grid(row=13, column=1, sticky=W)
    p7.insert(0, "---Your answers here---")
    Label(beginner_w, text="Problem 2F: The variable \"a\" must be declared as a \"global variable\"").grid(row=14, column=0, columnspan=2, sticky=W)
    p8 = Entry(beginner_w, width=55)
    Label(beginner_w, text="Your answer:").grid(row=15, column=0)
    p8.grid(row=15, column=1, sticky=W)
    p8.insert(0, "---Your answers here---")
    Label(beginner_w, text="Problem 3: Return the palindrome of the given string. The string is \"message\"").grid(row=16, column=0, columnspan=2, sticky=W)
    Label(beginner_w, text="*Palindrome means the reverse of a word or string. Ex: Palindrome of \"hello\" is \"olleh\".").grid(row=17, column=0, columnspan=2, sticky=W)
    p9 = Entry(beginner_w, width=55)
    Label(beginner_w, text="Your answer:").grid(row=18, column=0)
    p9.grid(row=18, column=1, sticky=W)
    p9.insert(0, "---Your answers here---")
    Button(beginner_w, text="Tips", command=tips).grid(row=19, column=0, sticky=W)
    Button(beginner_w, text="Check the Answer", command=chk).grid(row=19, column=1, sticky=E)


def beginner_set3_1():
    def chk():
        try:
            from ProblemSets import ProblemSet3
            reload(ProblemSet3)
        except:
            messagebox.showerror("Result", "Cannot find the file.")
        try:
            ProblemSet3.problem1()
            assert ProblemSet3.problem1() == "AAAAA"
            assert ProblemSet3.problem1("K ", 3) == "K K K "
            correct1 = 1
        except:
            correct1 = 0
            messagebox.showerror("Result", "Wrong answer in Problem 1")
        try:
            ProblemSet3.modulo(25, 2)
            assert ProblemSet3.modulo(25, 2) == 25 % 2
            assert ProblemSet3.modulo(65, 6) == 65 % 6
            assert ProblemSet3.modulo(4, 2) == 4 % 2
            correct2 = 1
        except:
            correct2 = 0
            messagebox.showerror("Result", "There was an error in Problem 2")
        try:
            ProblemSet3.importing(6)
            assert ProblemSet3.importing(6) == np.sqrt(6)
            assert ProblemSet3.importing(25) == np.sqrt(25)
            correct3 = 1
        except:
            correct3 = 0
            messagebox.showerror("Result", "There was an error in Problem 3")
        try:
            ProblemSet3.Paragraph()
            assert ProblemSet3.Paragraph() == 3
            correct4 = 1
        except:
```

```
        correct4 = 0
        messagebox.showerror("Result", "There was an error in Problem 4")
    try:
        ProblemSet3.creating()
        with open("creating.txt", "r") as file:
            file.read()
        correct5 = 1
    except:
        correct5 = 0
        messagebox.showerror("Result", "There was an error in Problem 5A")
    try:
        ProblemSet3.accessing()
        assert ProblemSet3.accessing() == txt
        correct6 = 1
    except:
        correct6 = 0
        messagebox.showerror("Result", "There was an error in Problem 5B")
    if correct1 == 1 and correct2 == 1 and correct3 == 1 and correct4 == 1 and correct5 == 1 and correct6 == 1:
        messagebox.showinfo("Result", "Correct Answers, Problem Set 3 Complete")
        beginner_b.config(state=NORMAL)
        intermediate_b.config(state=NORMAL)
        advanced_b.config(state=NORMAL)
        interview_b.config(state=NORMAL)
        sets()
        beginner_w.destroy()

beginner_w = Toplevel()
beginner_w.title("Problem Set 3 - Beginner")
Label(beginner_w, text="In this problem set, lets use a python interpreter, ex. PyCharm, Jupyter Notebook, etc.").grid(row=0, column=0, columnspan=2, sticky=W)
Label(beginner_w, text="Please put your program (.py file) in the \"ProblemSets\" folder, and name your file as \"ProblemSet3\"").grid(row=1, column=0, columnspan=2, sticky=W)
Label(beginner_w, text="Problem 1:").grid(row=2, column=0, columnspan=2, sticky=W)
Label(beginner_w, text="    Write a program that returns a character or string that is repeated by a specific value").grid(row=3, column=0, columnspan=2, sticky=W)
Label(beginner_w, text="    It must be on a module with arguments. Arguments must be in this order: character, number of times repeated").grid(row=4, column=0, columnspan=2, sticky=W)
Label(beginner_w, text="    Default value for the character is “A” and default number of times is 5").grid(row=5, column=0, columnspan=2, sticky=W)
Label(beginner_w, text="    This problem must be in a module named “problem1”").grid(row=6, column=0, columnspan=2, sticky=W)
Label(beginner_w, text="Problem 2:").grid(row=7, column=0, columnspan=2, sticky=W)
Label(beginner_w, text="    Write a program that modulus a number, only by using addition, subtraction, multiplication, or division.").grid(row=8, column=0, columnspan=2, sticky=W)
Label(beginner_w, text="    While loops/statements can only be used.").grid(row=9, column=0, columnspan=2, sticky=W)
Label(beginner_w, text="    This must be on a module with arguments, and the module named \"modulo\", and if \"a modulo b\" the arguments should be \"(a,b)\"").grid(row=10, column=0, columnspan=2, sticky=W)
Label(beginner_w, text="Problem 3:").grid(row=11, column=0, columnspan=2, sticky=W)
Label(beginner_w, text="    Import \"numpy\"").grid(row=12, column=0, columnspan=2, sticky=W)
Label(beginner_w, text="    Write a program that square roots the given numbers using Numpy").grid(row=13, column=0, columnspan=2, sticky=W)
Label(beginner_w, text="    This must be in a module with arguments, and the module named as \"importing\"").grid(row=14, column=0, columnspan=2, sticky=W)
Label(beginner_w, text="Problem 4:").grid(row=15, column=0, columnspan=2, sticky=W)
Label(beginner_w, text="    Write a program that finds and counts the word “right” from the given paragraph.").grid(row=16, column=0, sticky=W)
Button(beginner_w, text="Paragraph in here", command=paragraphs_f).grid(row=17, column=0, columnspan=2)
Label(beginner_w, text="    The program should find and count the word regardless of case or capitalization.").grid(row=18, column=0, columnspan=2, sticky=W)
Label(beginner_w, text="    Use If statements, for statements, and membership operators only").grid(row=19, column=0, columnspan=2, sticky=W)
Label(beginner_w, text="    It must be on a module named \"Paragraph\"").grid(row=20, column=0, columnspan=2, sticky=W)
Label(beginner_w, text="Problem 5A:").grid(row=21, column=0, columnspan=2, sticky=W)
Label(beginner_w, text="    Write a program that creates a text (.txt) file. File name should be \"creating\"").grid(row=22, column=0, columnspan=2, sticky=W)
Label(beginner_w, text="    The file must contain the given paragraph").grid(row=23, column=0, columnspan=2, sticky=W)
Button(beginner_w, text="Paragraph in here", command=paragraphs_f).grid(row=24, column=0, columnspan=2)
Label(beginner_w, text="    It must be in a module named \"creating\"").grid(row=25, column=0, columnspan=2, sticky=W)
Label(beginner_w, text="Problem 5B:").grid(row=26, column=0, columnspan=2, sticky=W)
Label(beginner_w, text="    Write a program that reads the text (.txt) file created from the previous problem and return the contents of the file.").grid(row=27, column=0, columnspan=2, sticky=W)
Label(beginner_w, text="    It must be in a module named \"accessing\"").grid(row=28, column=0, columnspan=2, sticky=W)
```

```
    Button(beginner_w, text="Tips",
command=tips).grid(row=29, column=0, sticky=W)
    Button(beginner_w, text="Check the Answer",
command=chk).grid(row=29, column=1, sticky=E)


def uni_set1():
    def chk():
        try:
            from ProblemSets import ProbSet1
            reload(ProbSet1)
        except ModuleNotFoundError:
            messagebox.showerror("Result", "Cannot find the
file.")
            return 0
        except:
            messagebox.showerror("Result", "There is an error,
cannot run your program.")
        try:
            ProbSet1.X()
            assert ProbSet1.X() == "Hello World"
            correct1 = 1
        except:
            correct1 = 0
            messagebox.showerror("Result", "There was an error
in Problem 1")
        try:
            ProbSet1.program()
            assert ProbSet1.program() == "Python!!!"
            correct2 = 1
        except:
            correct2 = 0
            messagebox.showerror("Result", "There was an error
in Problem 2")
        try:
            ProbSet1.add(1,2)
            assert ProbSet1.add(1,2) == 1+2
            assert ProbSet1.add(56,21) == 56+21
            correct3 = 1
        except:
            correct3 = 0
            messagebox.showerror("Result", "There was an error
in Problem 3 - add")
        try:
            ProbSet1.subtract(2,1)
            assert ProbSet1.subtract(2,1) == 2-1
            assert ProbSet1.subtract(89,65) == 89-65
            correct4 = 1
        except:
            correct4 = 0
            messagebox.showerror("Result", "There was an error
in Problem 3 - subtract")
        try:
            ProbSet1.multiply(5,2)
            assert ProbSet1.multiply(5,2) == 5*2
            assert ProbSet1.multiply(12,11) == 12*11
            correct5 = 1
        except:
            correct5 = 0
            messagebox.showerror("Result", "There was an error
in Problem 3 - multiply")
        try:
            ProbSet1.divide(8,5)
            assert ProbSet1.divide(8,5) == 8/5
            assert ProbSet1.divide(104,4) == 104/4
            correct6 = 1
        except:
            correct6 = 0
            messagebox.showerror("Result", "There was an error
in Problem 3 - divide")
        if correct1 == 1 and correct2 == 1 and correct3 == 1
and correct4 == 1 and correct5 == 1 and correct6 == 1:
            messagebox.showinfo("Result", "Correct Answers,
Problem Set 1 Complete")
            prob_set01_l.config(state=NORMAL)
            prob_set02_l.config(state=NORMAL)
            prob_set03_l.config(state=NORMAL)
            interview_b.config(state=NORMAL)
            sets()
            uni_set1_w.destroy()

    if diff == "Advanced":
        prob_set01_l.config(state=DISABLED)
        prob_set02_l.config(state=DISABLED)
        prob_set03_l.config(state=DISABLED)
    elif diff == "Intermediate":
        prob_set01_l.config(state=NORMAL)
        prob_set02_l.config(state=NORMAL)
        prob_set03_l.config(state=NORMAL)

    title_win = "Problem Set 1 - " + diff
    uni_set1_w = Toplevel()
    uni_set1_w.title(title_win)
    Label(uni_set1_w, text="Problem 1:").grid(row=0,
column=0, columnspan=2, sticky=W)
    Label(uni_set1_w, text="            Write a program that
returns \"Hello World\"").grid(row=1, column=0,
columnspan=2, sticky=W)
    Label(uni_set1_w, text="            It must be in a module
named \"X\"").grid(row=2, column=0, columnspan=2,
sticky=W)
    Label(uni_set1_w, text="Problem 2:").grid(row=3,
column=0, columnspan=2, sticky=W)
    Label(uni_set1_w, text="            Write a program that
prints and return \"Python!!!\"").grid(row=4, column=0,
columnspan=2, sticky=W)
    Label(uni_set1_w, text="            It must be in a module
named \"program\"").grid(row=5, column=0,
columnspan=2, sticky=W)
    Label(uni_set1_w, text="Problem 3:").grid(row=6,
column=0, columnspan=2, sticky=W)
    Label(uni_set1_w, text="            Write a program that
can add the given values, and return it. [module name:
add]").grid(row=7, column=0, columnspan=2, sticky=W)
    Label(uni_set1_w, text="            Write a program that
can subtract the given values, and return it. [module name:
subtract]").grid(row=8, column=0, columnspan=2,
sticky=W)
    Label(uni_set1_w, text="            Write a program that
can multiply the given values, and return it. [module name:
multiply]").grid(row=9, column=0, columnspan=2,
sticky=W)
    Label(uni_set1_w, text="            Write a program that
can divide the given values, and return it. [module name:
```

```
divide]").grid(row=10, column=0, columnspan=2,
sticky=W)
    Label(uni_set1_w, text="                All of the above must
be in their own module with arguments. Ex. \"a+b\" -->
\"add(a,b)\"").grid(row=11, column=0, columnspan=2,
sticky=W)
    fnt_reminder = tkFont.Font(size=14)
    Label(uni_set1_w, text="Please name your program (.py
file) as \"ProbSet1\"", font=fnt_reminder).grid(row=12,
column=0, columnspan=2)
    Tips = Button(uni_set1_w, text="Tips", command=tips)
    Tips.grid(row=13, column=0, sticky=W)
    Button(uni_set1_w, text="Check the Answer",
command=chk).grid(row=13, column=1, sticky=E)
    if diff == "Advanced":
        Tips.config(state=DISABLED)
    elif diff == "Intermediate":
        Tips.config(state=NORMAL)


def uni_set2():
    def chk():
        try:
            from ProblemSets import ProbSet2
            reload(ProbSet2)
        except ModuleNotFoundError:
            messagebox.showerror("Result", "Cannot find the
file.")
            return 0
        except:
            messagebox.showerror("Result", "There is an error,
cannot run your program.")
        try:
            ProbSet2.modulo(8,5)
            assert ProbSet2.modulo(8,5) == 8 % 5
            assert ProbSet2.modulo(256,2) == 256 % 2
            correct1 = 1
        except:
            correct1 = 0
            messagebox.showerror("Result", "There was an
Error in Problem 1 - [modulo]")
        try:
            ProbSet2.squared(5)
            assert ProbSet2.squared(5) == 5**2
            assert ProbSet2.squared(11) == 11**2
            correct2 = 1
        except:
            correct2 = 0
            messagebox.showerror("Result", "There was an error
in Problem 1 - [squared]" )
        lsts = [2,4,6,8,10]
        try:
            ProbSet2.lists()
            assert ProbSet2.lists() == lsts
            correct3 = 1
        except:
            correct3 = 0
            messagebox.showerror("Result", "There was an error
in Problem 2 - [lists]")
        try:
            ProbSet2.y()
            assert ProbSet2.y() == lsts[0]
            correct4 = 1
        except:
            correct4 = 0
            messagebox.showerror("Result", "There was an error
in Problem 2 - [y]")
        try:
            ProbSet2.lengths()
            assert ProbSet2.lengths() == len(lsts)
            correct5 = 1
        except:
            correct5 = 0
            messagebox.showerror("Result", "There was an error
in Problem 2 - [lengths]")
        try:
            ProbSet2.num1()
            assert type(ProbSet2.num1()) == int
            correct6 = 1
        except:
            correct6 = 0
            messagebox.showerror("Result", "There was an error
in Problem 2 - [num1]")
        try:
            ProbSet2.num2()
            assert type(ProbSet2.num2()) == float
            correct7 = 1
        except:
            correct7 = 0
            messagebox.showerror("Result", "There was an error
in Problem 2 - [num2]")
        try:
            ProbSet2.palindrome("from")
            assert ProbSet2.palindrome("from") == "from"[::-1]
            assert ProbSet2.palindrome("Hello") == "Hello"[::-
1]
            correct8 = 1
        except:
            correct8 = 0
            messagebox.showerror("Result", "There was an error
in Problem 3")
        if correct1 == 1 and correct2 == 1 and correct3 == 1
and correct4 == 1 and correct5 == 1 and correct6 == 1 and
correct7 == 1 and correct8 == 1:
            messagebox.showinfo("Result", "Correct Answers,
Problem Set 2 Complete")
            prob_set01_l.config(state=NORMAL)
            prob_set02_l.config(state=NORMAL)
            prob_set03_l.config(state=NORMAL)
            interview_b.config(state=NORMAL)
            sets()
            uni_set2_w.destroy()

    if diff == "Advanced":
        prob_set01_l.config(state=DISABLED)
        prob_set02_l.config(state=DISABLED)
        prob_set03_l.config(state=DISABLED)
    elif diff == "Intermediate":
        prob_set01_l.config(state=NORMAL)
        prob_set02_l.config(state=NORMAL)
        prob_set03_l.config(state=NORMAL)

    title_win = "Problem Set 2 - " + diff
    uni_set2_w = Toplevel()
```

```
    uni_set2_w.title(title_win)
    Label(uni_set2_w, text="Problem 1:").grid(row=0,
column=0, columnspan=2, sticky=W)
    Label(uni_set2_w, text="            Create a module with
arguments that can modulo the given values, and return it.
[module name: modulo]").grid(row=1, column=0,
columnspan=2, sticky=W)
    Label(uni_set2_w, text="            Create a module with
arguments that squares the given value, and return it.
[module name: squared]").grid(row=2, column=0,
columnspan=2, sticky=W)
    Label(uni_set2_w, text="Problem 2:").grid(row=3,
column=0, columnspan=2, sticky=W)
    Label(uni_set2_w, text="            Create a module
named \"lists\"; it must return a list with 5 integers, the
integers are: 2, 4, 6, 8, and 10, repectively. The list must be
declared as a global variable.").grid(row=4, column=0,
columnspan=2, sticky=W)
    Label(uni_set2_w, text="            Create a module
named \"y\", then call the 1st element from the list you made
in previous item.").grid(row=5, column=0, columnspan=2,
sticky=W)
    Label(uni_set2_w, text="            Create a module
named \"lengths\"; use the \"len\" function return the
number of elements from the list you made in previous
item.").grid(row=6, column=0, columnspan=2, sticky=W)
    Label(uni_set2_w, text="            Create a module
named \"num1\", it must return an integer data
type.").grid(row=7, column=0, columnspan=2, sticky=W)
    Label(uni_set2_w, text="            Create a module
named \"num2\", it must return a float data
type.").grid(row=8, column=0, columnspan=2, sticky=W)
    Label(uni_set2_w, text="Problem 3:").grid(row=9,
column=0, columnspan=2, sticky=W)
    Label(uni_set2_w, text="            Write a Python
program that returns the palindrome of the given
string.").grid(row=10, column=0, columnspan=2, sticky=W)
    Label(uni_set2_w, text="            It must be in a module
with arguments, module must be named
\"palindrome\"").grid(row=11, column=0, columnspan=2,
sticky=W)
    Label(uni_set2_w, text="            Palindrome means the
reverse of a word or string. Ex: Palindrome of \"hello\" is
\"olleh\".").grid(row=12, column=0, columnspan=2,
sticky=W)
    fnt_reminder = tkFont.Font(size=14)
    Label(uni_set2_w, text="Please name your program (.py
file) as \"ProbSet2\"", font=fnt_reminder).grid(row=13,
column=0, columnspan=2)
    Tips = Button(uni_set2_w, text="Tips", command=tips)
    Tips.grid(row=14, column=0, sticky=W)
    Button(uni_set2_w, text="Check the Answer",
command=chk).grid(row=14, column=1, sticky=E)
    if diff == "Advanced":
        Tips.config(state=DISABLED)
    elif diff == "Intermediate":
        Tips.config(state=NORMAL)


def uni_set3():
    def chk():
        try:
            from ProblemSets import ProbSet3
            reload(ProbSet3)
        except ModuleNotFoundError:
            messagebox.showerror("Result", "Cannot find the
file.")
            return 0
        except:
            messagebox.showerror("Result", "There is an error,
cannot run your program.")
        try:
            ProbSet3.problem1()
            assert ProbSet3.problem1() == "AAAAA"
            assert ProbSet3.problem1("K ", 3) == "K K K "
            """with open("ProblemSets/ProbSet3.py", "r") as
file:
                P1_test = file.read()
            if "for" not in P1_test:
                assert 1 == 0"""
            correct1 = 1
        except:
            correct1 = 0
            messagebox.showerror("Result", "There was an error
in Problem 1")
        try:
            ProbSet3.modulo(25, 2)
            assert ProbSet3.modulo(25,2) == 25 % 2
            assert ProbSet3.modulo(65,6) == 65 % 6
            assert ProbSet3.modulo(4,2) == 4 % 2
            correct2 = 1
        except:
            correct2 = 0
            messagebox.showerror("Result", "There was an error
in Problem 2")
        try:
            ProbSet3.importing(6)
            assert ProbSet3.importing(6) == np.sqrt(6)
            assert ProbSet3.importing(25) == np.sqrt(25)
            correct3 = 1
        except:
            correct3 = 0
            messagebox.showerror("Result", "There was an error
in Problem 3")
        try:
            ProbSet3.Paragraph()
            """with open("ProblemSets/ProbSet3.py", "r") as
file:
                P3_test = file.read()
            if "for" not in P3_test:
                assert 1 == 0
            if "if" not in P3_test:
                assert 1 == 0"""
            assert ProbSet3.Paragraph() == 3
            correct4 = 1
        except:
            correct4 = 0
            messagebox.showerror("Result", "There was an error
in Problem 4")
        try:
            ProbSet3.creating()
            with open("creating.txt", "r") as file:
                file.read()
            correct5 = 1
```

```
        except:
            correct5 = 0
            messagebox.showerror("Result", "There was an error in Problem 5A")
        try:
            ProbSet3.accessing()
            assert ProbSet3.accessing() == txt
            correct6 = 1
        except:
            correct6 = 0
            messagebox.showerror("Result", "There was an error in Problem 5B")
        if correct1 == 1 and correct2 == 1 and correct3 == 1 and correct4 == 1 and correct5 == 1 and correct6 == 1:
            messagebox.showinfo("Result", "Correct Answers, Problem Set 3 Complete")
            prob_set01_l.config(state=NORMAL)
            prob_set02_l.config(state=NORMAL)
            prob_set03_l.config(state=NORMAL)
            interview_b.config(state=NORMAL)
            beginner_b.config(state=NORMAL)
            intermediate_b.config(state=NORMAL)
            advanced_b.config(state=NORMAL)
            sets()
            uni_set3_w.destroy()

    if diff == "Advanced":
        prob_set01_l.config(state=DISABLED)
        prob_set02_l.config(state=DISABLED)
        prob_set03_l.config(state=DISABLED)
    elif diff == "Intermediate":
        prob_set01_l.config(state=NORMAL)
        prob_set02_l.config(state=NORMAL)
        prob_set03_l.config(state=NORMAL)

    title_win = "Problem Set 3 - " + diff
    uni_set3_w = Toplevel()
    uni_set3_w.title(title_win)
    Label(uni_set3_w, text="Problem 1:").grid(row=0, column=0, columnspan=2, sticky=W)
    Label(uni_set3_w, text="        Create a module that returns a character or string that is repeated by a specific value").grid(row=1, column=0, columnspan=2, sticky=W)
    Label(uni_set3_w, text="        It must be on a module with arguments.").grid(row=2, column=0, columnspan=2, sticky=W)
    Label(uni_set3_w, text="        Arguments must be in this order: character, number of times repeated, respectively.").grid(row=3, column=0, columnspan=2, sticky=W)
    Label(uni_set3_w, text="        Default value for the character is “A” and default number of times is 5").grid(row=4, column=0, columnspan=2, sticky=W)
    Label(uni_set3_w, text="        This problem must be in a module named “problem1”").grid(row=5, column=0, columnspan=2, sticky=W)
    Label(uni_set3_w, text="        Use for statements only").grid(row=6, column=0, columnspan=2, sticky=W)
    Label(uni_set3_w, text="Problem 2:").grid(row=7, column=0, columnspan=2, sticky=W)
    Label(uni_set3_w, text="        Write a program that modulus a number, only by using addition, subtraction, multiplication, or division.").grid(row=8, column=0, columnspan=2, sticky=W)
    Label(uni_set3_w, text="        While loops/statements can only be used.").grid(row=9, column=0, columnspan=2, sticky=W)
    Label(uni_set3_w, text="        This must be on a module with arguments, and named “modulo”").grid(row=10, column=0, columnspan=2, sticky=W)
    Label(uni_set3_w, text="        Arguments must be in this order: dividend, divisor.").grid(row=11, column=0, columnspan=2, sticky=W)
    Label(uni_set3_w, text="Problem 3:").grid(row=12, column=0, columnspan=2, sticky=W)
    Label(uni_set3_w, text="        Create a module that returns square roots the given numbers using Numpy").grid(row=13, column=0, columnspan=2, sticky=W)
    Label(uni_set3_w, text="        This must be in a module with arguments, and the module named as \"importing\"").grid(row=14, column=0, columnspan=2, sticky=W)
    Label(uni_set3_w, text="Problem 4:").grid(row=15, column=0, columnspan=2, sticky=W)
    Label(uni_set3_w, text="        Create a module that finds and counts the word \"right\" in the given paragraph, and return the result.").grid(row=16, column=0, columnspan=2, sticky=W)
    Button(uni_set3_w, text="Paragraph in here", command=paragraphs_f).grid(row=17, column=0, columnspan=2)
    Label(uni_set3_w, text="        Use \"If\" statements & \"for\" statements only").grid(row=18, column=0, columnspan=2, sticky=W)
    Label(uni_set3_w, text="        The program should also find and count the word regardless of case or capitalization.").grid(row=19, column=0, columnspan=2, sticky=W)
    Label(uni_set3_w, text="        It must be on a module named \"Paragraph\"").grid(row=20, column=0, columnspan=2, sticky=W)
    Label(uni_set3_w, text="Problem 5A:").grid(row=21, column=0, columnspan=2, sticky=W)
    Label(uni_set3_w, text="        Create a module that creates a text (.txt) file.").grid(row=22, column=0, columnspan=2, sticky=W)
    Label(uni_set3_w, text="        The contents of the file is a paragraph.").grid(row=23, column=0, columnspan=2, sticky=W)
    Button(uni_set3_w, text="Paragraph in here", command=paragraphs_f).grid(row=24, column=0, columnspan=2)
    Label(uni_set3_w, text="        It must me in a module named \"creating\"").grid(row=25, column=0, columnspan=2, sticky=W)
    Label(uni_set3_w, text="Problem 5B:").grid(row=26, column=0, columnspan=2, sticky=W)
    Label(uni_set3_w, text="        Create a module that reads the text (.txt)  file created from the previous item and return the contents of the file.").grid(row=27, column=0, columnspan=2, sticky=W)
```

```
    Label(uni_set3_w, text="                It must be in a module
named \"accessing\"").grid(row=28, column=0,
columnspan=2, sticky=W)
    fnt_reminder = tkFont.Font(size=14)
    Label(uni_set3_w, text="Please name your program (.py
file) as \"ProbSet3\"", font=fnt_reminder).grid(row=29,
column=0, columnspan=2)
    Tips = Button(uni_set3_w, text="Tips", command=tips)
    Tips.grid(row=30, column=0, sticky=W)
    Button(uni_set3_w, text="Check the Answer",
command=chk).grid(row=30, column=1, sticky=E)
    if diff == "Advanced":
        Tips.config(state=DISABLED)
    elif diff == "Intermediate":
        Tips.config(state=NORMAL)


def url_browser(url):
    webbrowser.open(url, new=2)


def intro():
    def mainmenu_manual():
        global my_img_b
        mainmenu_intro_w = Toplevel()
        mainmenu_intro_w.title("Manual")
        Label(mainmenu_intro_w, text="Main Menu",
font=fontHeaders).pack()
        my_img_b =
ImageTk.PhotoImage(Image.open("Images/Introduction/But
tons.png"))
        Label(mainmenu_intro_w, image=my_img_b).pack()

    def interview_manual():
        global my_img_i
        interview_intro_w = Toplevel()
        interview_intro_w.title("Manual")
        Label(interview_intro_w, text="Interview",
font=fontHeaders).pack()
        my_img_i =
ImageTk.PhotoImage(Image.open("Images/Introduction/Int
erview.png"))
        Label(interview_intro_w, image=my_img_i).pack()

    def aq_manual():
        global my_img_s1
        aq_w = Toplevel()
        aq_w.title("Manual")
        Label(aq_w, text="Answering Questions",
font=fontHeaders).pack()
        my_img_s1 =
ImageTk.PhotoImage(Image.open("Images/Introduction/Su
bmission 01.png"))
        Label(aq_w, image=my_img_s1).pack()

    def add_files_manual():
        global my_img_s2
        add_files_w = Toplevel()
        add_files_w.title("Manual")
        Label(add_files_w, text="Adding Files to Submit",
font=fontHeaders).pack()
        my_img_s2 =
ImageTk.PhotoImage(Image.open("Images/Introduction/Su
bmission 02.png"))
        Label(add_files_w, image=my_img_s2).pack()

    intro_w = Toplevel()
    intro_w.title("Introduction")
    fontHeaders = tkFont.Font(size=15)
    fontSubtitle = tkFont.Font(size=10)
    Label(intro_w, text="Python Interpreters:",
font=fontHeaders).pack(anchor=W)
    Label(intro_w, text="Interpreters are converting your
codes to make an output, or program.",
font=fontSubtitle).pack()
    Button(intro_w, text="Anaconda", command=lambda:
url_browser("https://www.anaconda.com/")).pack()
    Button(intro_w, text="PyCharm", command=lambda:
url_browser("https://www.jetbrains.com/pycharm/")).pack()
    Button(intro_w, text="Online Interpreters",
state=DISABLED).pack()
    Label(intro_w, text="   \n   ").pack()
    Label(intro_w, text="Installation Instructions:",
font=fontHeaders).pack(anchor=W)
    Label(intro_w, text="Anaconda").pack()
    Button(intro_w, text="Documentation",
command=lambda:
url_browser("https://docs.anaconda.com/")).pack()
    Button(intro_w, text="Installation", command=lambda:
url_browser("https://docs.anaconda.com/anaconda/install/"))
.pack()
    Label(intro_w, text="   ").pack()
    Label(intro_w, text="PyCharm").pack()
    Button(intro_w,text="Guides", command=lambda:
url_browser("https://www.jetbrains.com/pycharm/guide/")).
pack()
    Button(intro_w, text="Installation", command=lambda:
url_browser("https://www.jetbrains.com/help/pycharm/insta
llation-guide.html")).pack()
    Label(intro_w, text="   ").pack()
    Label(intro_w, text="Online Interpreter - Geeks For
Geeks").pack()
    Button(intro_w, text="Installation",
command=noinstall).pack()
    Button(intro_w, text="Program", command=lambda:
url_browser("https://ide.geeksforgeeks.org/")).pack()
    Label(intro_w, text="   ").pack()
    Label(intro_w, text="How to Use This Program:
[Manual]", font=fontHeaders).pack(anchor=W)
    Button(intro_w, text="Main Menu",
command=mainmenu_manual).pack()
    Button(intro_w, text="Interview",
command=interview_manual).pack()
    Button(intro_w, text="Answering Questions",
command=aq_manual).pack()
    Button(intro_w, text="Adding files to submit",
command=add_files_manual).pack()


def noinstall():
    messagebox.showinfo("No installation needed!\nYou
only need an internet connection and a browser.")
```

```
def lec_prob_set01():
    def lesson1():
        lesson1_w = Toplevel()
        lesson1_w.title("Lesson 1")
        Label(lesson1_w, text="Lesson 1: Return Function").pack(anchor=W)
        Label(lesson1_w, text="Definition: Ends a function and returns the result of the expression").pack(anchor=W)
        Label(lesson1_w, text="Usage: return <your function>").pack(anchor=W)
        Label(lesson1_w, text="Examples: return \"Hello Juan!\" ---> returns \"Hello Juan!\"").pack(anchor=W)
        Label(lesson1_w, text="                return 0 ---> returns the value 0").pack(anchor=W)
        Label(lesson1_w, text="                return x ---> returns the value of x").pack(anchor=W)

    def lesson3():
        lesson3_w = Toplevel()
        lesson3_w.title("Lesson 3")
        Label(lesson3_w, text="Lesson 3: Print").pack(anchor=W)
        Label(lesson3_w, text="Definition: It basically prints the message or output the message to the screen.").pack(anchor=W)
        Label(lesson3_w, text="Usage: print(<things to print>)").pack(anchor=W)
        Label(lesson3_w, text="Examples: print(\"Hello World!!!!\") ---> displays \"Hello World!!!!\"").pack(anchor=W)
        Label(lesson3_w, text="                print(\"12345\") ---> displays \"12345\"").pack(anchor=W)
        Label(lesson3_w, text="                print(x) ---> displays the value of x").pack(anchor=W)

    def lesson4():
        lesson4_w = Toplevel()
        lesson4_w.title("Lesson 4")
        Label(lesson4_w, text="Lesson 4: Variables").pack(anchor=W)
        Label(lesson4_w, text="Definition: It is a name you assign to a value").pack(anchor=W)
        Label(lesson4_w, text="Rules: Variable names can only start with an underscore or a letter; it can only contain alphanumeric characters and underscores; it is case sensitive; it should also not shadow any Python functions.").pack(anchor=W)
        Label(lesson4_w, text="Examples:").pack(anchor=W)
        Label(lesson4_w, text="        Correct variable naming: x my_variable _names num2 xyz").pack(anchor=W)
        Label(lesson4_w, text="        Wrong variable naming: 12_variable print my+variable *xy").pack(anchor=W)

    lec_prob_set01_w = Toplevel()
    lec_prob_set01_w.title("Lecture - Problem Set 1")
    Label(lec_prob_set01_w, text="Lesson 1:").grid(row=0, column=0)
    Button(lec_prob_set01_w, text="Return Function", command=lesson1).grid(row=0, column=1, sticky=W)
    Label(lec_prob_set01_w, text="Lesson 2:").grid(row=1, column=0)
    Button(lec_prob_set01_w, text="Modules", command=lambda: url_browser("https://www.codementor.io/@kaushikpal/user-defined-functions-in-python-8s7wyc8k2")).grid(row=1, column=1, sticky=W)
    Label(lec_prob_set01_w, text="Lesson 3:").grid(row=2, column=0)
    Button(lec_prob_set01_w, text="Print", command=lesson3).grid(row=2, column=1, sticky=W)
    Label(lec_prob_set01_w, text="Lesson 4:").grid(row=3, column=0)
    Button(lec_prob_set01_w, text="Variables", command=lesson4).grid(row=3, column=1, sticky=W)
    Label(lec_prob_set01_w, text="Lesson 5:").grid(row=4, column=0)
    Button(lec_prob_set01_w, text="Basic Arithmetic Operations", command=lambda: url_browser("https://www.geeksforgeeks.org/basic-operators-python/")).grid(row=4, column=1, sticky=W)


def lec_prob_set02():
    def lesson9():
        lesson9_w = Toplevel()
        lesson9_w.title("Lesson 9")
        Label(lesson9_w, text="Lesson 9: \"len\" Function")
        Label(lesson9_w, text="Definition: It counts the length of the item, i.e. number of elements in a list, length of a string, etc.")
        Label(lesson9_w, text="Usage: len(<things to count>)")
        Label(lesson9_w, text="Example: len(x) ---> counts the length in variable \"x\"")

    def lesson10():
        lesson10_w = Toplevel()
        lesson10_w.title("Lesson 10")
        Label(lesson10_w, text="Lesson 10: \"global\" Function")
        Label(lesson10_w, text="Definition: If you declared a variable inside a module, you cannot use it outside the module. If you need to use the variable outside the module, you have to use this function.")
        Label(lesson10_w, text="Usage: global <variable>")
        Label(lesson10_w, text="Example: global x ---> declared the variable \"x\" globally")

    lec_prob_set02_w = Toplevel()
    lec_prob_set02_w.title("Lecture - Problem Set 2")
    Label(lec_prob_set02_w, text="Lesson 6:").grid(row=0, column=0)
    Button(lec_prob_set02_w, text="Data Types", command=lambda: url_browser("https://www.programiz.com/python-programming/variables-datatypes")).grid(row=0, column=1, sticky=W)
    Label(lec_prob_set02_w, text="Lesson 7:").grid(row=1, column=0)
    Button(lec_prob_set02_w, text="Operators", command=lambda: url_browser("https://www.programiz.com/python-
```

```
programming/operators")).grid(row=1, column=1,
sticky=W)
    Label(lec_prob_set02_w, text="Lesson 8:").grid(row=2,
column=0)
    Button(lec_prob_set02_w, text="Palindrome Program",
command=lambda:
url_browser("https://www.geeksforgeeks.org/python-
program-check-string-palindrome-not/")).grid(row=2,
column=1, sticky=W)
    Label(lec_prob_set02_w, text="Lesson 9:").grid(row=3,
column=0)
    Button(lec_prob_set02_w, text="\"len\" Function",
command=lesson9).grid(row=3, column=1, sticky=W)
    Label(lec_prob_set02_w, text="Lesson 10:").grid(row=4,
column=0)
    Button(lec_prob_set02_w, text="\"global\" Function",
command=lesson10).grid(row=4, column=1, sticky=W)


def lec_prob_set03():
    lec_prob_set03_w = Toplevel()
    lec_prob_set03_w.title("Lecture - Problem Set 3")
    Label(lec_prob_set03_w, text="Lesson 11:").grid(row=0,
column=0)
    Button(lec_prob_set03_w, text="Imports [Numpy]",
command=lambda:
url_browser("https://www.geeksforgeeks.org/python-
numpy/")).grid(row=0, column=1, sticky=W)
    Label(lec_prob_set03_w, text="Lesson 12:").grid(row=1,
column=0)
    Button(lec_prob_set03_w, text="Accessing & Creating
Files", command=lambda:
url_browser("https://www.geeksforgeeks.org/reading-
writing-text-files-python/")).grid(row=1, column=1,
sticky=W)
    Label(lec_prob_set03_w, text="Lesson 13:").grid(row=2,
column=0)
    Button(lec_prob_set03_w, text="Conditional [If]
Statements", command=lambda:
url_browser("https://www.tutorialspoint.com/python/python
_if_else.htm")).grid(row=2, column=1, sticky=W)
    Label(lec_prob_set03_w, text="Lesson 14:").grid(row=3,
column=0)
    Button(lec_prob_set03_w, text="Loopings [\"For\" &
\"While\" Statements]", command=lambda:
url_browser("https://www.geeksforgeeks.org/loops-in-
python/")).grid(row=3, column=1, sticky=W)
    Label(lec_prob_set03_w, text="Lesson 15:").grid(row=4,
column=0)
    Button(lec_prob_set03_w, text="\"split\" Functions",
command=lambda:
url_browser("https://www.geeksforgeeks.org/python-
extract-words-from-given-string/")).grid(row=4, column=1,
sticky=W)
    Label(lec_prob_set03_w, text="Lesson 16:").grid(row=5,
column=0)
    Button(lec_prob_set03_w, text="Text Formatting",
command=lambda:
url_browser("https://www.digitalocean.com/community/tuto
rials/how-to-format-text-in-python-3")).grid(row=5,
column=1, sticky=W)
```

```
def tips():
    tips_w = Toplevel()
    tips_w.title("Tips")
    Label(tips_w, text="Check your indents.\nCheck your
variable name.\nCheck your module name.\nCheck your
capitalizations, spelling, and symbols.\nCheck your file
names\nDon’t forget to return\nFor more information about
submitting your work, please check the introduction
page.").pack(anchor=W)
    Button(tips_w, text="Introduction", command=intro)


txt = """Right has multiple meanings:\nIt can mean without
error.\nExample: Juan is right, there are no classes
tomorrow.\nIt can also be used to tell directions.\nExample:
Look at your right side first before crossing the road in
Hong Kong."""


def paragraphs_f():
    paragraphs_w = Toplevel()
    paragraphs_w.title("Paragraph")
    Label(paragraphs_w, text=txt, anchor=W,
justify=LEFT).pack()


first_run()
root = Tk()
root.title("PL4Python")
fontStyle1 = tkFont.Font(family="Consolas", size=20)
fontStyle2 = tkFont.Font(family="Consolas", size=15)
Header = Label(root, text="Welcome to PL4Python",
font=fontStyle1)
Subtitle = Label(root, text="Learn Python Easily",
font=fontStyle2)
Blank = Label(root, text="")

frame1 = LabelFrame(root, text="Recommend Difficulty",
pady=36)
interview_b = Button(frame1, text="Interview",
command=interview01)

frame2 = LabelFrame(root, text="Difficulty of Problem
Sets", pady=10)
beginner_b = Button(frame2, text="Beginner",
command=beginner)
intermediate_b = Button(frame2, text="Intermediate",
command=intermediate)
advanced_b = Button(frame2, text="Advanced",
command=advanced)

frame3 = LabelFrame(root, text="Lectures", pady=20,
padx=30)
intro_l = Button(frame3, text="Introduction",
command=intro, padx=15)
prob_set01_l = Button(frame3, text="For Problem Set 1",
command=lec_prob_set01)
prob_set02_l = Button(frame3, text="For Problem Set 2",
command=lec_prob_set02)
prob_set03_l = Button(frame3, text="For Problem Set 3",
command=lec_prob_set03)
```

```
space_l = Label(frame3, text="     ")
space_l1 = Label(frame3, text=" ")

Header.grid(row=0, column=0, columnspan=2)
Subtitle.grid(row=1, column=0, columnspan=2)
Blank.grid(row=2, column=0)
frame1.grid(row=3, column=0)
interview_b.pack()
frame2.grid(row=3, column=1)
beginner_b.pack()
intermediate_b.pack()
advanced_b.pack()
frame3.grid(row=4, column=0, columnspan=2)
intro_l.grid(row=0, column=0)
space_l.grid(row=0, column=1)
prob_set01_l.grid(row=0, column=2)
space_l1.grid(row=1, column=0, columnspan=3)
prob_set02_l.grid(row=2, column=0)
space_l.grid(row=2, column=1)
prob_set03_l.grid(row=2, column=2)

root.mainloop()
```

VIII. Discussion of Results

Nowadays, Personalized learning is becoming more useful and also a fundamental option to be used in teaching the topics in their subjects. Due to these exceptional measures, everybody is experiencing at the moment, personalized learning as the cruel of examining appears to be the leading choice for students to continue their studies and in this project. that's the objective of the group. To make running a running program that particularly prepare to their needs and could be able to advance progress a student's information on a indicated subjects, in this program it will instruct and teach python programming language.

The program was done using PyCharm and finished last May 1,2020. The program gives lectures in introduction of python and in problem sets 1-3. After the user studying the lectures, it will display the three levels which is beginner, intermediate and expert. The user is free to choose on what level the learner may answer. The features of the program are all running well.

The programmers of this program have found no future errors that can affect the user's development in learning the course. It assumes that the project was successfully programmed and achieve the objectives to create a program that unified personalized learning to this course. Furthermore, as seen in chapter VIII (8), the program ran well, as seen on the screenshots.

IX. Conclusion & Recommendations

PL4Python, a program created using Tkinter was able to contain personalized learning in teaching Python Programming language depending on the user's level (beginner, intermediate, and expert). The users are offered different sets of topics depending on the student's level. After the topics were discussed in each problem sets, it will be followed by a question to test if the discussions were absorbed by the students, since, the programmers are promoting personalized learning which means the user can learn the topics at own pace.

As found in chapter V (5) of this paper, the programmers were able to create problem sets for the users to answer. The problem sets are used to check whether the user had absorbed the lessons taught in the program. In this case, the programmers had achieved one of their objectives, which is to create a problem set for the users to answer. As seen in part VIII, the code for the program are for python and it had made use of Python programming language and "Tkinter". As seen on chapter II (2), the programmers were also able to create an interview for the user to check their ability in programming, as stated in the objectives in chapter I (1).

The programmers suggested that to design and add features similar to a game in order the user may feel motivated to learn. To evaluate the program's effectiveness, the programmers recommended that to add a posttest to analyze the improvement that the user was able to achieve after using the PL4Python program.

References

[1] D. Z., 5 Key Elements of Personalized Learning,Jul. 18, 2019 Accessed on: Mar. 04, 2020. [Online]. Available: https://blog.neolms.com/5-key-elements-of-personalized-learning/

[2] Nandigam, David & Sremath Tirumala, Sreenivas & Baghaei, Nilufar. (2014). Personalized Learning: Current status and Potential. IC3e 2014 - 2014 IEEE Conference on e-Learning, e-Management, and e-Services. 10.1109/IC3e.2014.7081251

[3] A. Morin, Personalized Learning: What You Need to Know, Accessed on: Mar. 04, 2020. [Online]. Available: https://www.understood.org/en/school-learning/partnering-with-childs-school/instructional-strategies/personalized-learning-what-you-need-to-know

[4] Professional & Continuing Education University of Washington, Success at Your Own Speed: Self-Paced Certificate Enables Student to Fast-Forward His Education— and His Career, Aug. 29, 2018 Accessed on: Mar. 04, 2020. [Online]. Available:https://www.pce.uw.edu/news-features/articles/success-at-your-own-speed

[5] N. Morris, How To Test Developers' Coding Skills Before Hiring, Jul. 23, 2018. Accessed on: Mar. 04, 2020. [Online]. Available: https://www.codingame.com/work/blog/tech-recruiting/test-developers-skills-before-hiring/

[6] Advantages and Disadvantages of Personalized Learning, Accessed on: Mar. 04, 2020. [Online]. Available:http://www.jcccc.org/School/personallearning.html

[7] J. Boyers, Online Done Right: The Importance of Human Interaction for Student Success, Sep. 2013. Accessed on: Mar. 04, 2020. [Online]. Available: https://elearnmag.acm.org/archive.cfm?aid=2524201

[8] UNESCO. (2017). Personalized Learning. Retrieved from https://unesdoc.unesco.org/ark:/48223/pf0000250057

[9] Saint Mary's University of Minnesota. The Benefits of Personalized Learning Through Technology. Retrieved from https://onlineprograms.smumn.edu/meldt/masters-of-education-in-learning-design-and-technology

[10] Kamenetz, A. (2018, November 16). The Future Of Learning? Well, It's Personal. Retrieved from https://www.npr.org/2018/11/16/657895964/the-future-of-learning-well-it-s-personal