UNIVERSITY OF CANBERRA
INTRODUCTION TO INFORMATION TECHNOLOGY (4478)

Alex Greenfield u3242795 Due date: 19/4/2022 Julio Romero

I Instructions

The sequence of assessments will test your knowledge and skills in writing an application software for a task, understanding the business rules of a particular problem, coding these in a computer program, developing a graphical user interface, and reading data from a text file from a disk. As such, the assignments will require you to integrate and synthesise what you will be learning each week in this unit, in order to design and create a proper working solution.

This fourth assignment finally completes your journey in learning how to code. Once again, you will be using your topic and code developed in assessments 2, and 3. The challenge involves handle exceptions in your Assignment 3 code, as well as creating a Graphical user interface to replace the shell menu created in the same Assignment 3.

This fourth assessment requires you to:

1. Implement in your code exception handling. You need to do this adding extra code into your already completed Assignment 3 submission. Marks will be allocated according to Section III of this document. Bear in mind that you must use the error handler according to the exceptions to be captured. **Using the construct Try-Except only will not attract any marks.**
2. Produce a Graphical user Interface following the process shown in Part II of this document
3. **You must preserve all the elements previously developed in your code** (see marks distribution.)

I) Background:

Python can be very useful to create calculators, especially in finance where you can create programs to automate particular problems, instead of manually calculating, in our case 3 banks and finding their APY and balance. We can create a program/module so that the user does not have to manually calculate these problems and shorten the process for the user by using the program, after all inputs are entered into the module, the answer will be output.

The Module in Python language should compare interest rates offered by three different banks. Determine the most favorable interest rate and the balance in that account after one year. Create a calculator in python to do this for the user, printing the solution as an output in various ways. Such as a shell-based menu to choose from, and writing the output to a text file/files. The list should be read from a text file. In our case, this is a list of the banks. This is displayed above the menu as a list, of the available options to choose from.

II) Business Rules (based on the 5-steps solving - problems process)

1. Identify your inputs.

Input 1 = **r** (interest in decimal form (2dp)) = *0.04, 0.05, 0.03*
Input 2 = **m** (compound interval in a year, in months) = *8, 4, 12*
Input 3 = **P** *(deposit ($)) = $2000*

2. Identify the goal or objective (related to the output)

We need to assess the problem, and create a solution in the form of a calculator that outputs our balance and APY percentage.

We need to use the APY formula to compare the banks and find the highest APY percentage among the 3 banks. Outputting the results. Different compounding periods cannot be compared directly that is why we must use APY. The following formula is to be implemented into the calculator below.

$$APY = \left(1 + \frac{r}{m}\right)^{m} - 1$$

$$P \cdot \left(1 + \frac{r}{m}\right)^{m}$$

Details: (*Bank - 1*, **P** = $2000, **r** = 0.04%, **m** = 8) (*Bank - 2*, **P** = $2000, **r** = 0.05%, **m** = 4) *(Bank - 3* **P** = $2000, **r** = 0.03%, **m** = 12)

Compare banks, find the most favorable bank and it's interest rate using the APY formula. Output the bank, balance and APY % in that account after one year.

3. Create a list of tasks to achieve your objective

1. Create a flow chart, develop pseudocode from the flowchart, identify inputs and outputs, variables, constants decisions, loops, booleans, strings, integers, etc. (see rubric for more details)

2. Code the solution into Python IDLE, using the pseudocode as a reference. Use your flowchart as a reference to code your solution into python more easily. This process helps in the designing and planning of your code when it comes time to write it. It is important to eliminate easy mistakes and problems.

   3. Code inputs in (**r(1,2,3), m(1,2,3)**), floats for interest, interest and principal. Code in elements such as lists, booleans, decision, while and for loops, etc. Added a shell-based menu and at least one write/read text file to write the outputs into.

   4. Make sure code follows the Python Style Guide for easy writing/reading and interpretation of the code. This makes sure the code is neat and tidy. This will help, and sort your code into sections, overall this helps the creator of the code and the reader.

4. Develop a hand-written solution

Using the inputs interest, principle, and compound interval. Solve the best bank balance and APY % between the 3 banks.

P = Principal
r(1,2,3) = Interest
m(1,2,3) = Compound Interval

Commonwealth Bank: $2000, 4%, 8
Beyond Bank: $2000, 5%, 4
St George Bank: $2000, 3%, 12

Formula for Balance ($):

$$P \cdot \left(1 + \frac{r}{m}\right)^{m}$$

Commonwealth Bank (1) = $2081.41 = $2081
Beyond Bank (2) = $2101.89 = $2102
St George Bank (3) = $2060.83 = $2061

Formula for APY (%):

$$APY = \left(1 + \frac{r}{m}\right)^{m} - 1$$

Commonwealth Bank (1) = 4%
Beyond Bank (2) = 5%
St George Bank (3) = 3%

Compare banks with the APY formula, not with the balance. Output the best bank option, balance, and APY.

Solution: Bank 2 had the best APY, 5% interest, compounded quarterly. $2000 dollars principal, $2101.89 after one year.

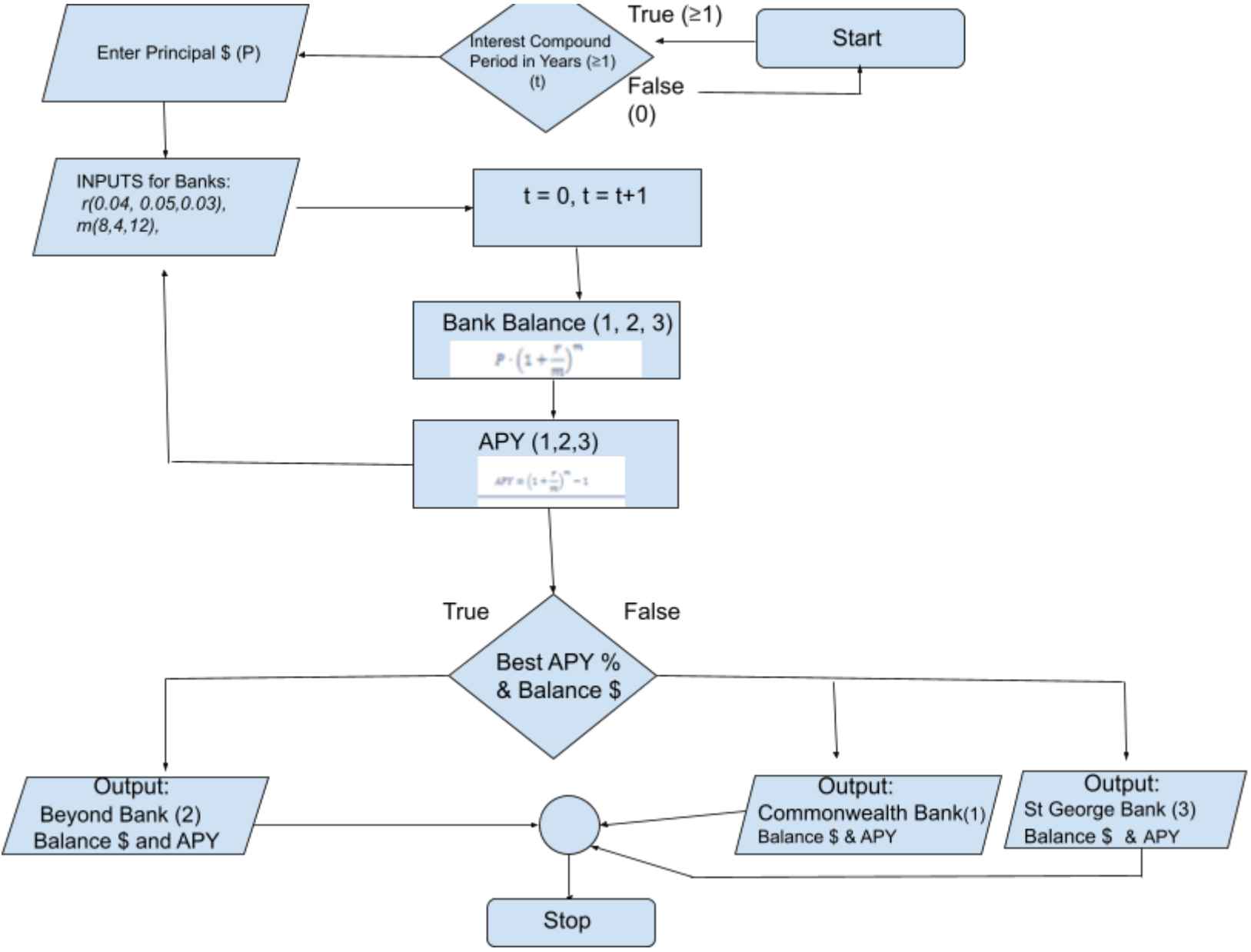
5. Assess (test) your solution

1. Handwritten solution tested with a scientific calculator, handwritten results checked.
2. Use a Balance & APY calculator to test the results, tested and confirmed.
3. Inputs and output assessed and tested, correct.
4. Test our solution, in our own calculator and compare results. Results match the handwritten solution.
5. Tests conducted. Results match. Python calculator outputs correctly (TUI).
6. Shell-based menu outputs a correct answer, along with the best bank & banks in their own individual text files (TUI).

III) The process to Start Coding your Solution

6. Produce a flowchart to describe the flow of instructions

[MARKS: 8 marks for a properly designed flowchart]



7. Derive pseudocode

Pseudocode:

```
## Date last changed:25/04/2022

## Author: Alex Greenfield

## Date created: 15/03/2022

## This program is an APY calculator, comparing 3 different banks.

## Inputs: r, m, P, Output: APY % & Balance $

## Balance ($) & APY (%) Calculator: Commonwealth Bank (1), Beyond Bank (2), St George(3)

## Read Input: banklist.txt, Write Outputs: bestbank.txt, banks.txt

##Error Handling

#Time Input & Decision

TRY:
    SET T TO int(INPUT("How many years are you compounding interest?(must be ≥ 1):"))
    SET T0 TO T * 1 / T #Do not use zero (This is here so 0 shows invalid)
except ValueError:    #ValueError IF fails
    OUTPUT("\nYou did not respond with an integer value. ")
except ZeroDivisionError: #When divided by 0 it will fail
    OUTPUT("Invalid. Please do not use zero")


#Principal Input

TRY:
    SET P TO int(INPUT("\nEnter Principal Here: $"))
    SET P0 TO P * 1 / P #Do not use zero (This is here so 0 shows invalid)
except ValueError: #ValueError IF fails
    OUTPUT("\nYou did not respond with an integer value. ")
except ZeroDivisionError: #When divided by 0 it will fail
    OUTPUT("Invalid. Please do not use zero")
```

```
#Bank 1

TRY:
    SET r1 TO float(INPUT('\nEnter the interest IN decimal form (%): '))
    SET m1 TO int(INPUT('Enter the Compound Interval:'))
    SET B1 TO P *( 1 +  r1 / m1 )** m1
    SET APY1 TO ( 1 +  r1 /  m1 )**  m1 - 1
    OUTPUT('\nBalance, Bank 1: $', B1)
    OUTPUT('APY, Bank 1: ', 100*APY1,'%')
except ValueError: #ValueError IF fails
    OUTPUT("Error, please INPUT an actual number.")
except ZeroDivisionError: #When divided by 0 it will fail
    OUTPUT("Invalid. Please do not use zero")

SET t TO 0

WHILE t<= T:
        SET B1 TO P*(1+r1/m1)**m1
        SET APY1 TO (1+r1/m1)**m1-1
        SET t TO t+ 1
        OUTPUT("Years Compounded =", t)
        break

#Bank 2

TRY:
    SET r2 TO float(INPUT('\nEnter the interest IN decimal form (%): '))
    SET m2 TO int(INPUT('Enter the Compound Interval:'))
    SET B2 TO P *( 1 + r2 / m2 )** m2
    SET APY2 TO ( 1 + r2 / m2 )** m2 - 1
```

```
    OUTPUT('\nBalance, Bank 2: $', B2)
    OUTPUT('APY, Bank 2: ', 100*APY2,'%')
except ValueError: #ValueError IF fails
    OUTPUT("Error, please INPUT an actual number.")
except ZeroDivisionError: #When divided by 0 it will fail
    OUTPUT("Invalid. Please do not use zero")

SET t TO 0

WHILE t<= T:
        SET B2 TO P *(1+r2/m2)**m2
        SET APY2 TO (1+r2/m2)**m2-1
        SET t TO t+ 1
        OUTPUT("Years Compounded =", t)
        break

SET str TO 'Beyond Bank'

#Bank 3

TRY:
    SET r3 TO float(INPUT('\nEnter the interest IN decimal form (%): '))
    SET m3 TO int(INPUT('Enter the Compound Interval:'))
    SET B3 TO P *( 1 + r3 /  m3 )** m3
    SET APY3 TO ( 1 + r3 / m3 )** m3 - 1
    OUTPUT('\nBalance, Bank 3: $', B3)
    OUTPUT('APY, Bank 3: ', 100*APY3,'%')
except ValueError: #ValueError IF fails
    OUTPUT("Error, please INPUT an actual number.")
except ZeroDivisionError: #When divided by 0 it will fail
    OUTPUT("Invalid. Please do not use zero")
```

```
SET t TO 0

WHILE t<= T:
        SET B3 TO P*(1+r3/m3)**m3
        SET APY3 TO (1+r3/m3)**m3-1
        SET t TO t+ 1
        OUTPUT("Years Compounded =", t)
        break

#APY Output

SET list1 TO [APY1,APY2,APY3]

SET top_APY TO None

FOR num IN list1:
     if(top_APY is None or num > top_APY):
        SET top_APY TO num
OUTPUT('\nBank with the Highest APY & Balance:', str)
OUTPUT("\nHighest APY:", round(100*top_APY,6),"%")

#Balance & Bank Output

SET list2 TO [B1,B2,B3]
SET best_bank TO None

FOR num IN list2:
    if(best_bank is None or num> best_bank):
       SET best_bank TO num
```

```
OUTPUT("Highest Balance: $", round(best_bank,2))



#Write Bank Options ($,%) & Best Bank ($,%) to text files


DEFINE FUNCTION main():
    SET outfile TO open ("banks.txt",'w')
    createWithWritelines(outfile)
    SET outfile TO open("bestbank.txt",'w')
    createWithWrite(outfile)
    outfile.close()


DEFINE FUNCTION createWithWritelines(outfile):
    outfile.writelines("Commonwealth Bank")
    outfile.writelines(":$2081, 4%\n")
    outfile.writelines("Beyond Bank")
    outfile.writelines(":$2101, 5%\n")
    outfile.writelines("St George Bank")
    outfile.writelines(":$2060, 3%\n")
    outfile.close()


TRY:
    SET infile TO open ("banks.txt","r")
    SET banks TO int(infile.readline())
    OUTPUT("All Banks:", banks)
except FileNotFoundError: # FileNotFound IF fails
    OUTPUT("File banks.txt not found.")
except ValueError: #ValueError IF fails
    OUTPUT("\nbanks.txt contains all bank options, and their Balance and APY.")
    infile.close()
ELSE:
```

```
    infile.close()


DEFINE FUNCTION createWithWrite(outfile):
    SET outfile TO open("bestbank.txt", 'a')
    outfile.write("Best Bank Option\n")
    outfile.write("Beyond Bank:$2101, 5%")


    outfile.close()

TRY:
    SET infile TO open ("bestbank.txt","r")
    SET bestbank TO int(infile.readline())
    OUTPUT("Best Bank:", bestbank)
except FileNotFoundError: # FileNotFound IF fails
    OUTPUT("File bestbank.txt not found.")
except ValueError: #ValueError IF fails
    OUTPUT("\nbestbank.txt contains the best bank option.\n")
    infile.close()
ELSE:
    infile.close()

main()


#Reading Bank / Bank List from an External File


DEFINE FUNCTION main():
    SET file TO "banklist.txt"
    displayWithListComprehension(file)


DEFINE FUNCTION displayWithListComprehension(file):
    TRY:
```

```
        SET infile TO open(file, 'r')
        SET listPres TO [line.rstrip() FOR line IN infile]
        infile.close()
        OUTPUT("banklist.txt found, displayed below:\n")
        OUTPUT(listPres,"\n")
    except FileNotFoundError: # FileNotFound IF fails
        OUTPUT("File banklist.txt not found.")
    finally:
        OUTPUT("Thank you FOR using our program\n.")
        infile.close()


main()
```

```
## Date last changed:25/04/2022
## Author: Alex Greenfield
## Date created: 15/03/2022
## This program is an APY calculator, comparing 3 different banks.
## Inputs: r, m, P, Output: APY % & Balance $
## Balance ($) & APY (%) Calculator: Commonwealth Bank (1), Beyond Bank (2), St George(3)
## Read Input: banklist.txt, Write Outputs: bestbank.txt, banks.txt


#Import the tkinter library


IMPORT tkinter as tk
from tkinter IMPORT *

#GUI Main


SET BTN_ENT_WIDTH TO 25


DEFINE FUNCTION Banks():
```

```
    SET bank TO ""
    IF chkVarC.get() EQUALS 1:
        bank+= "$2081, 4%"
    IF chkVarB.get() EQUALS 1:
        bank+= " $2102, 5.1%"

    IF chkVarG.get() EQUALS 1:
        bank+= "$2061, 3%"


    txtEnt. set(bank)


SET window TO Tk ()


window.title("Bank APY & Balance List Output")


window.geometry("700x300")


window.configure(bg='blue')


# create a menu bar options
SET menubar TO Menu (window)
SET Menu TO Menu (menubar, tearoff=0)


SET Menu.add_command(label="Info", command TO OUTPUT('Made by Alex Greenfield, 2022 UC'))


Menu.add_separator() # add a line separating the menu items
Menu.add_command(label="Exit", command=window.destroy)



# create a menu
menubar.add_cascade(label="Details", menu=Menu)
window.config(menu=menubar)
```

```
SET lbl TO Label (window, text="Choose Bank Options", font="Calibri 16")

lbl.grid(row=3, column=0, padx=10, sticky=W)


SET chkVarC TO IntVar()#

#chkVarC. set (1) # check the checkbox

SET chkC TO Checkbutton(window, text TO "Commonwealth Bank", variable=chkVarC, font="Calibri 14")

SET chkC.grid(row=4, column TO 0,padx=10, sticky=W)


SET chkVarB TO IntVar()

SET chkB TO Checkbutton(window, text TO "Beyond Bank", variable=chkVarB, font="Calibri 14")

chkB.grid(row=5, column=0, padx=10, sticky=W)


SET chkVarG TO IntVar()

SET chkG TO Checkbutton(window, text TO "St George Bank", variable=chkVarG, font="Calibri 14")

SET chkG.grid(row TO 6,column TO 0,padx=10, sticky=W)


txtEnt= StringVar()

txtEnt. set (" ")

ent= Entry(window, width=BTN_ENT_WIDTH, font= ('Comic Sans M' ,15, "bold"),textvariable=txtEnt)

ent.grid(row=8,column=0, columnspan=1, padx=20,pady=10)


btn= Button(window, text="Display Bank Balance & APY", width=BTN_ENT_WIDTH, command= Banks,
font="Calibri 14")

SET btn.grid(row=9, column TO 0, columnspan TO 1, padx TO 20, pady TO 10)


btnQuit= Button(window, text="QUIT",width=BTN_ENT_WIDTH, command=window.destroy, font="Calibri 14")

SET btnQuit.grid(row=9, column TO 1, padx=20, pady TO 10)


#GUI Calculator Fields
```

```
SET fields TO ('Principal','Interest','Compound Interval', 'FINAL_BALANCE','APY')
#APY


DEFINE FUNCTION APY(entries):
    TRY:
        # Interest:
        SET r TO (float(entries['Interest'].get()) / 100)
        # Principal
        SET P TO float(entries['Principal'].get())
        # Compound Interval
        SET n TO  float(entries['Compound Interval'].get())
        SET APY TO (1 + r / n) ** n - 1
        entries['APY'].delete(0, tk.END)
        entries['APY'].insert(0, 100*APY )
        OUTPUT("APY: %", 100*APY)
    except ValueError:  #ValueError IF fails
        OUTPUT("Error, please INPUT an actual number.")
    except ZeroDivisionError: #When divided by 0 it will fail
        OUTPUT("Invalid. Please do not use zero")


#FINAL_BALANCE


DEFINE FUNCTION FINAL_BALANCE(entries):
    TRY:
        # Interest:
        SET r TO (float(entries['Interest'].get()) / 100)
        # Principal
        SET P TO float(entries['Principal'].get())
        # Compound Interval
        SET n TO  float(entries['Compound Interval'].get())
        SET FINAL_BALANCE TO P * (1 + r / n) ** n
```

```
        entries['FINAL_BALANCE'].delete(0, tk.END)
        entries['FINAL_BALANCE'].insert(0, FINAL_BALANCE)
        OUTPUT("Balance: $", FINAL_BALANCE)
    except ValueError: #ValueError IF fails
        OUTPUT("Error, please INPUT an actual number.")
    except ZeroDivisionError: #When divided by 0 it will fail
        OUTPUT("Invalid. Please do not use zero")


#makeform (labels, entries, rows)

DEFINE FUNCTION makeform(root, fields):
    SET entries TO {}
    FOR field IN fields:
        SET row TO tk.Frame(root)
        SET lab TO tk.Label(row, width=22, text=field+": ", anchor='w')
        SET ent TO tk.Entry(row,)
        ent.insert(0, "0")
        row.pack(side=tk.TOP, fill=tk.X,  padx=5, pady=5)
        lab.pack(side=tk.LEFT)
        ent.pack(side=tk.RIGHT, expand=tk.YES, fill=tk.X)
        SET entries[field] TO ent
    RETURN entries


# GUI Main (bg, buttons, size)


IF __name__ EQUALS '__main__':
    SET root TO tk.Tk()
    root.geometry("400x300")
    root.configure(bg='blue')
    root.title("Bank APY & Balance Calculator")
    SET ents TO makeform(root, fields)
```

```
    SET b1 TO tk.Button(root, text='APY',command=(lambda e=ents: APY (e)))
    b1.pack(side=tk.LEFT, padx=5, pady=5)
    SET b2 TO tk.Button(root, text='Final Balance',command=(lambda e=ents: FINAL_BALANCE (e)))
    b2.pack(side=tk.LEFT, padx=5, pady=5)
    SET b3 TO tk.Button(root, text="QUIT", command=root.destroy)
    b3.pack(side=tk.LEFT, padx=5, pady=5)


root.mainloop()
```

8. Code your solution using Python-based on step 7 above.

[MARKS: 55marks for including below elements]

Open py files in zip.”error handling.py” & “assignment_4_GUI.py” .