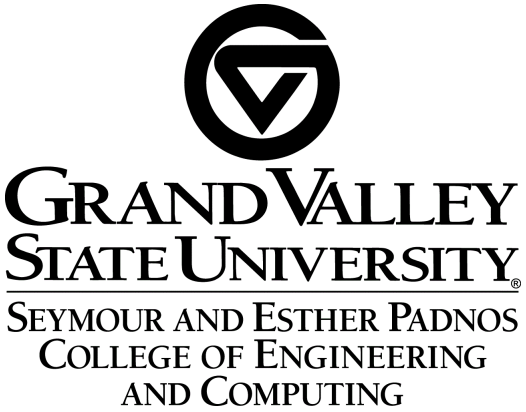

EGR680 High Level Implementation on FPGA

Laboratory 07

Python Functions, File I/O, and Object-Oriented Programming

Professor: Dr. C. Parikh

Student: Dimitri Häring

October 24, 2018

Contents

1 Introduction

The goal of laboratory seven is to familiarize the student with functions, file I/O, and objective oriented programming in python.

2 Design

In this section the design and decisions that where made to achieve the laboratory are discussed.

2.1 ATM machine Part I

The software of the previous lab has been split up into two files so that the main logic and main program flow is in the main file and the functions are separated in an atm_func file. Furthermore a file I/O for login errors was implemented to aid assistance by unusual behavior. The two code listening are shown in Listening 8 nad 9. A single function is shown in Listening 1.

```
def mainMenue ( ) :
  print ( " Main Menu" )
  print ( "----------------------- \n" )
  print ( " 1 . CHECK BALANCE" )
  print ( " 2 . WITHDRAW CASH" )
  print ( " 3 . DEPOSIT CASH" )
  print ( " 4 . CHANGE PIN" )
  print ( " 5 . EXIT " )
  return 1 # no error 1 , error 0
```

Listing 1: Python function mainMenu .

The following Listening shows the function call in the main program. That shows how bigger programs in python can be structured sequentially.

```
mainMenue ()
```

Listing 2: Python function call mainMenu .

The following listening shows the error log output file content. This can be easily used to check which errors has been tested and has a message generated to it. The error log is continuously appended into a text file and has to be managed manually so far this can be automated with a script as example.

Listing 3: Error log Part II.

```
Thu Oct 25 09:15:45 2018ATM program starts
Thu Oct 25 09:15:54 2018 User logged in
Thu Oct 25 09:16:06 2018 Withdraw error
Thu Oct 25 09:16:19 2018 Deposit error
Thu Oct 25 09:16:34 2018 User PIN error
Thu Oct 25 09:27:05 2018 Program Closed
```

2.2 ATM machine Part II

In this part the program was expended with an receipt function that would log all the transactions of a ongoing section and let the user decide to print it out the end of the session. Figure 1 shows how the user can decide after exiting the program if he wishes to print the receipt or not. The python code himself for file I/O used can be examen in closer detail in Listening 10 and 11. First a file has to be open this can be done with open(<filename>, mode) and then be accessed with the functions .read() and .write().

```
 Main Menu
-----------------------

 1. CHECK BALANCE
 2. WITHDRAW CASH
 3. DEPOSIT CASH
 4. CHANGE PIN
 5. EXIT

PLEASE SELECT FROM THE ABOVE OTPIONS: 5

Do you wish to print the receipt, press y or n? y

-------------------Bank Me-------------------
---------------Transaction Receipt-----------
*********************************************

Your checking account balance is----> $0

Your withdraw amounte is------------> $0

Your deposit amounte is-------------> $3

Your checking account balance is----> $3

You have changed your PIN number

Your withdraw amounte is------------> $1

Your checking account balance is----> $2

THANK YOU FOR CHOOSING BANK ME!

In [49]:
```

printed receipt

Figure 1: Printed receipt.

Listening 4 shows how a text file can be opened either with append mode or write mode. It shows also how errors can be handled with try and except commands. The lowest line show how to close a file.

```
# Open a file
try:
  fo = open("07_lab_error_log.txt", "a")
  fr = open(atm.ReceiptFile, "w")
  fo.write(asctime()+ ' ATM program starts '+ '\n')
  fr.write('\n----------------------Bank Me----------------------\n')
  fr.write('-------------------Transaction Receipt-------------\n')
  fr.write('*********************************************\n\n')
except IOError as e:
  print('File '+e.filename+' could not be opened or writen to!')

fo.write(asctime()+ ' Program Closed\n')
fo.close() # to ensure file is closed
fr.close() # to ensure file is closed
```

Listing 4: Python file I/O example.

2.3 ATM machine Part III

This section shall demonstrated the objective programming that allows two crate a class an instance this class as instances or an instance. A class contains usually data objects and methods. Listening 5 shows the class named Atm with class variables followed by the __init__ function which is first call after instantiation of a class and serves to initializes specific variables. next the normal statements or so called methods are

listed which can be used in the main program. Notice, the word *self* which is used in a method which defines a method and separate it from a normal static function.

```
import re
class Atm:
  AMOUNT_MIN = 0
  AMOUNT_MAX = 1000

  Pin  = "1234"
  Select = 0
  AccountValue = 0
  ReceiptFile = "atm_receipt.txt"

  def __init__(self, AccountValue, Pin):
    self.AccountValue = AccountValue
    self.Pin = Pin

  # Print Receipt
  def receipt(self, filename, file):
    file.write('THANK YOU FOR CHOOSING BANK ME!')
    file.close()
    file = open(filename, "r")
    for line in file:
    print(line, end='')
    file.close()
    return 1
```

Listing 5: Python class example with a method.

Listening 6 shows how a class can be imported with from import command and how it can be instantiated with initial variables as an object called atm. It also shows how methods can be used as example atm.welcome(). Class variables can be accessed like atm.Pin as example.

```
from atm_class import Atm
atm = Atm(100,'2345') # instantiate class Atm to atm object

# Star program
if atm.welcome(): # first method used from the atm object
  error = 0
else:
  error = 1
  fo.write(asctime()+ ' welcome() error\n')

# PIN validation
while True:
  ret = 0
  if atm.pin( input("Please enter your PIN: "), atm.Pin):
    error = 0
    fo.write(asctime()+ ' User logged in \n')
    break
  else:
    error = 1
    fo.write(asctime()+ ' User login error\n')
```

Listing 6: Python object example with a method.

The following listening shows the error log output file content. This can be easily used to check which errors has been tested and has a message generated to it. The error log is continuously appended into a text file and has to be managed manually so far this can be automated with a script as example.

Listing 7: Error log Part III.

```
Thu Oct 25 15:53:33 2018 ATM program starts
Thu Oct 25 15:53:37 2018 User login error
Thu Oct 25 15:53:40 2018 User login error
Thu Oct 25 15:53:44 2018 User login error
```

```
Thu Oct 25 15:53:46 2018 User login error
Thu Oct 25 15:53:48 2018 User logged in
Thu Oct 25 15:54:13 2018 Deposit error
Thu Oct 25 15:54:26 2018 Deposit error
Thu Oct 25 15:54:54 2018 Withdraw error
Thu Oct 25 15:55:00 2018 Withdraw error
Thu Oct 25 15:56:47 2018 Program Closed
```

Figure 2 shows the receipt output generated by the program written in part three where a class of an ATM was instantiated instead of functions methods where used.This together with the error log confirms the correct functioning of the ATM.

```
 Main Menu
-----------------------

 1. CHECK BALANCE
 2. WITHDRAW CASH
 3. DEPOSIT CASH
 4. CHANGE PIN
 5. EXIT

PLEASE SELECT FROM THE ABOVE OTPIONS: 5

Do you wish to print the receipt, press y or n? y

--------------------Bank Me--------------------
---------------Transaction Receipt-----------
*********************************************

Your checking account balance is----> $100

Your deposit amounte is-------------> $0

Your deposit amounte is-------------> $45

Your checking account balance is----> $145

Your withdraw amounte is------------> $0

You have changed your PIN number

THANK YOU FOR CHOOSING BANK ME!
```

printed receipt

Figure 2: Printed receipt Part III.

Further work would be to write the file I/O access in better methods which makes it nicer to look and easier to handle as well as the error log handling is still plumb and needs a better strategy if the program would be advanced.

3 Conclusion

The lab demonstrates the use of the python as simple and fast scripting language that allows access to vast number of packages that allows an decreased development time. The syntax is easy to learn but it is possible to lose the overview by having too many continuations of statements. Furthermore, it allows file I/O to save data on a disk and objective oriented programming with classes.

4 Appendix

The appendix contains code listening and other large information parts that contain partial or complete relevance to the reports topic.

4.1 Python code Part I

```
# -*- coding: utf-8 -*-
"""
Spyder Editor

This is a temporary script file.
"""
from time import asctime

from atm_func_partI import welcome, pin, \
selMainMenue, newPin, deposit, withdraw, balance

AMOUNT_MIN = 0
AMOUNT_MAX = 1000

gPin  = "1234"
gSelect = 0
gAccountValue = 0

# Open a file
try:
    fo = open("07_lab_error_log.txt", "a")
    fo.write(asctime()+ 'ATM program starts '+ '\n')
except IOError as e:
    print('File '+e.filename+' could not be opened!')

# Star program
if welcome():
    error = 0
else:
    error = 1
    fo.write(asctime()+ ' welcome() error\n')

# PIN validation
while True:
    ret = 0
    if pin( input("Please enter your PIN: "), gPin):
        error = 0
        fo.write(asctime()+ ' User logged in \n')
        break
    else:
        error = 1
        fo.write(asctime()+ ' User login error\n')

# Main menu
while  gSelect != "5":
    ret = 0
    ret = selMainMenue(input("PLEASE SELECT FROM THE ABOVE OTPIONS: ") )
#    print(ret) # debug only
    if (ret[0]):
        error = 0
        gSelect = ret[1]
#        print(ret[1] ) # debug only
    else: # error
        error = 1
        gSelect = 0
        fo.write(asctime()+ ' User slection error\n')
        print(ret[0])

    if gSelect == '5': # exit
```

```
        exit
    elif gSelect == '4': # change PIN
        while True:
            ret = 0
            ret = newPin(input("ENTER YOUR NEW PIN: "))
#             print(ret) # debug only
            if (ret[0]):
                error = 0
                gPin = ret[1]
        #         print(ret[1] ) # debug only
                break
            else: # error
                error = 1
                gSelect = 0
                fo.write(asctime()+ ' User PIN error\n')
                break

    elif gSelect == '3': # Deposit
        while True:
            ret = 0
            ret = deposit(input("ENTER YOUR DEPOSIT AMOUNT IN $: ")\
                          , AMOUNT_MIN, AMOUNT_MAX, gAccountValue)
#             print(ret) # debug only
            if (ret[0]):
                error = 0
                gAccountValue = ret[1]
        #         print(ret[1] ) # debug only
                break
            else: # error
                error = 1
                gSelect = 0
                fo.write(asctime()+ ' Deposit error\n')

    elif gSelect == '2': # Withdraw
        while True:
            ret = 0
            ret = withdraw(input("ENTER YOUR WITHDRAW AMOUNT IN $: ")\
                           , AMOUNT_MIN, AMOUNT_MAX, gAccountValue)
#             print(ret) # debug only
            if (ret[0]):
                error = 0
                gAccountValue = ret[1]
        #         print(ret[1] ) # debug only
                break
            else: # error
                error = 1
                gSelect = 0
                fo.write(asctime()+ ' Withdraw error\n')

    elif gSelect == '1': # Balance
        balance(gAccountValue)
    else:
        print("\nIncorrect Selecion!")

print("\nThank you for chosing BANK ME!")
fo.write(asctime()+ ' Program Closed\n')
fo.close()
fr.close()
```

Listing 8: Python code for an ATM Part I.

```
# -*- coding: utf-8 -*-
"""
Created on Thu Oct 18 18:34:49 2018

@author: schwa
"""
import re
```

```
# Balance
def balance( account_value , file):
    print("\nYOUR BALANCE IS $%.2f \n \n\r" % account_value)
    file.write('Your checking account balance is----> $'\
               +str(account_value)+'\n')
    delay(3000)
    mainMenue( )
    return 1
# Withdraw money
def withdraw( inVal, AMOUNT_MIN, AMOUNT_MAX, account_value, file ):
    ret = [0, 0]; # no error 1, error 0
    amount = 0
    if not re.match("^[0-9]*$", inVal):
        amount = 0
        print("\nInvalid Input only nmbers are alowed.\n")
        delay(3000)
        mainMenue( )
        ret=[0, inVal]
#        break
    else:
        amount = int(inVal)
        if amount >= AMOUNT_MIN and amount <= account_value :
            account_value = account_value - amount
            print("\nWITHDRAW AMOUNT $%.2f\n" % amount)
            file.write('Your withdraw amounte is----------> $'\
                       +str(amount)+'\n')
            amount = 0
            delay(3000)
            mainMenue( )
            ret=[1, account_value]
#            break
        else:
            ret=[0, inVal]
            print("\nWITHDRAW AMOUNT $%.2f TOO BIG OR TOO SMALL\n" % amount)
    return ret
# Deposit money
def deposit( inVal, AMOUNT_MIN, AMOUNT_MAX, account_value, file ):
    ret = [0, 0]; # no error 1, error 0
    amount = 0
    if not re.match("^[0-9]*$", inVal):
        amount = 0
        print("\nInvalid Input only numbers are alowed.\n")
        delay(3000)
        mainMenue( )
        ret=[0, inVal]
    else:
        amount = int(inVal)
        if amount >= AMOUNT_MIN and amount <= AMOUNT_MAX :
            account_value = account_value + amount
            print("\nDEPOSIT AMOUNT $%.2f\n" % amount)
            file.write('Your deposit amounte is----------> $'\
                       +str(amount)+'\n')
            amount = 0
            delay(3000)
            mainMenue( )
            ret=[1, account_value]
        else:
            ret=[0, inVal]
            print("\nDEPOSIT AMOUNT $%.2f TOO BIG OR TOO SMALL\n" % amount)
    return ret
# set new pin
def newPin( inVal, file ):
    ret = [0, 0]; # no error 1, error 0
    if not re.match("^[0-9]{4}$", inVal):
        print ("Error! Make sure you only use numbers from 0-9 in PIN")
        inVal = 'Z'
        ret=[0, inVal]
    else:
```

```
        ret =[1, inVal]
#         print(ret) # debug only
        print("\nYOUR NEW PIN IS", inVal, "\n")
        file.write('You have changed your PIN number\n')
        delay(3000)
        mainMenue()
    return ret
# selection main menue
def selMainMenue( inVal ):
    ret = 0; # no error 1, error 0
    # inVal = input("PLEASE SELECT FROM THE ABOVE OTPIONS: ")
    if not re.match("^[1-5]*$", inVal):
        print ("Error! Make sure you only use numbers from 1-5 in selecion")
        ret=[0, inVal]
    else:
        ret =[1, inVal]
    return ret
# PIN validation
def pin( inVal, PIN ):
    ret = 0; # no error 1, error 0
#    while True:
       #inVal = input("Please enter your PIN: ") # regex
    if not re.match("^[0-9]{4}$", inVal):
        print ("Error! Make sure you only use numbers from 0-9 in PIN")
        inVal = 'Z'
        ret = 0
    else:
        if inVal == PIN :
            #print("\nCorect PIN") # debug only
            print("\n")
            mainMenue()
            inVal = 'Z'
            ret = 1
#                break
        else:
            print("\nInvalid PIN!")
            inVal = 'Z'
            ret = 0
    return ret

def delay( msec ):
    cnt = 0
    while cnt < msec:
        cnt += 0.0001
    return 1  # no error 1, error 0

def mainMenue( ):
    print(" Main Menu")
    print("---------------------- \n")
    print(" 1. CHECK BALANCE")
    print(" 2. WITHDRAW CASH")
    print(" 3. DEPOSIT CASH")
    print(" 4. CHANGE PIN")
    print(" 5. EXIT ")
    return 1  # no error 1, error 0

def welcome( ):
    print("\n    $$$$$           $$          $$$           $$   $$     $  ")
    print("    $     $         $$$$         $$ $          $$    $$     $   ")
    print("    $      $        $   $        $$  $         $$    $$    $    ")
    print("    $      $       $$   $$       $$   $        $$    $$   $     ")
    print("    $     $       $$     $$      $$    $       $$    $$ $       ")
    print("    $$$$$        $$      $$      $$     $      $$    $$$        ")
    print("    $$$$$        $$$$$$$$$      $$      $     $$    $$$        ")
    print("    $     $      $$       $$     $$       $    $$    $$ $       ")
    print("    $      $    $$         $$    $$        $   $$    $$  $      ")
    print("    $      $   $$          $$    $$         $ $$    $$   $     ")
    print("    $     $   $$           $$   $$          $$$   $$     $    ")
```

```
    print("     $$$$$     $$            $$    $$            $$    $$      $  \n")
    print("               $$$           $$    $$$$$$$$$$$$$$  ")
    print("               $$ $         $$$    $$$$$$$$$$$$$$   ")
    print("               $$  $      $ $$    $$       ")
    print("               $$   $     $  $$    $$      ")
    print("               $$    $  $    $$    $$      ")
    print("               $$      $$     $$    $$$$$$$$$$$$$$        ")
    print("               $$             $$    $$$$$$$$$$$$$$        ")
    print("               $$             $$    $$       ")
    print("               $$             $$    $$       ")
    print("               $$             $$    $$       ")
    print("               $$             $$    $$$$$$$$$$$$$$   ")
    print("               $$             $$    $$$$$$$$$$$$$$  ")
    print("-------------------------------------------------------------")
    print("                    Welcome to Bank Me  ")
    print("=============================================================")
    return 1  # no error 1, error 0
```

Listing 9: Python functions for an ATM Part I.

4.2 Python code Part II

```
# -*- coding: utf-8 -*-
"""
Spyder Editor

This is a temporary script file.
"""
from time import asctime
import re
from atm_func_partII import welcome, pin, \
selMainMenue, newPin, deposit, withdraw, balance, receipt, delay

AMOUNT_MIN = 0
AMOUNT_MAX = 1000

gPin  = "1234"
gSelect = 0
gAccountValue = 0
gReceiptFile = "atm_receipt.txt"

# Open a file
try:
    fo = open("07_lab_error_log.txt", "a")
    fr = open(gReceiptFile, "w")
    fo.write(asctime()+ 'ATM program starts '+ '\n')
    fr.write('\n----------------------Bank Me----------------------\n')
    fr.write('-----------------Transaction Receipt-----------\n')
    fr.write('*********************************************\n\n')
except IOError as e:
    print('File '+e.filename+' could not be opened!')

# Star program
if welcome():
    error = 0
else:
    error = 1
    fo.write(asctime()+ ' welcome() error\n')

# PIN validation
while True:
    ret = 0
    if pin( input("Please enter your PIN: "), gPin):
        error = 0
        fo.write(asctime()+ ' User logged in \n')
        break
    else:
        error = 1
```

```
        fo.write(asctime()+ ' User login error\n')

# Main menu
while  gSelect != "5":
    ret = 0
    ret = selMainMenue(input("PLEASE SELECT FROM THE ABOVE OTPIONS: ") )
#    print(ret) # debug only
    if (ret[0]):
        error = 0
        gSelect = ret[1]
#        print(ret[1] ) # debug only
    else: # error
        error = 1
        gSelect = 0
        fo.write(asctime()+ ' User slection error\n')
        print(ret[0])

    if gSelect == '5': # exit
        while True:
            if not re.match("^[y]$", input('Do you wish to print the receipt, press y or n?
    ')):
                print("\nThank you for chosing BANK ME!")
                break
            else:
                receipt(gReceiptFile, fr)
                break
        exit
    elif gSelect == '4': # change PIN
        while True:
            ret = 0
            ret = newPin(input("ENTER YOUR NEW PIN: "), fr)
#            print(ret) # debug only
            if (ret[0]):
                error = 0
                gPin = ret[1]
        #        print(ret[1] ) # debug only
                break
            else: # error
                error = 1
                gSelect = 0
                fo.write(asctime()+ ' User PIN error\n')
                break

    elif gSelect == '3': # Deposit
        while True:
            ret = 0
            ret = deposit(input("ENTER YOUR DEPOSIT AMOUNT IN $: ")\
                          , AMOUNT_MIN, AMOUNT_MAX, gAccountValue, fr)
#            print(ret) # debug only
            if (ret[0]):
                error = 0
                gAccountValue = ret[1]
        #        print(ret[1] ) # debug only
                break
            else: # error
                error = 1
                gSelect = 0
                fo.write(asctime()+ ' Deposit error\n')

    elif gSelect == '2': # Withdraw
        while True:
            ret = 0
            ret = withdraw(input("ENTER YOUR WITHDRAW AMOUNT IN $: ")\
                           , AMOUNT_MIN, AMOUNT_MAX, gAccountValue, fr)
#            print(ret) # debug only
            if (ret[0]):
                error = 0
                gAccountValue = ret[1]
```

```
        #         print(ret[1] ) # debug only
                  break
              else: # error
                  error = 1
                  gSelect = 0
                  fo.write(asctime()+ ' Withdraw error\n')

    elif gSelect == '1': # Balance
        balance(gAccountValue, fr)
    else:
        print("\nIncorrect Selecion!")

fo.write(asctime()+ ' Program Closed\n')
fo.close() # to ensure file is closed
fr.close() # to ensure file is closed
delay(3000)
```

Listing 10: Python code for an ATM Part II.

```
# -*- coding: utf-8 -*-
"""
Created on Thu Oct 18 18:34:49 2018

@author: schwa
"""
import re
# Print Receipt
def receipt(filename, file):
    file.write('THANK YOU FOR CHOOSING BANK ME!')
    file.close()
    file = open(filename, "r")
    for line in file:
        print(line, end='')
    file.close()
    return 1
# Balance
def balance( account_value , file):
    print("\nYOUR BALANCE IS $%.2f \n \n\r" % account_value)
    file.write('Your checking account balance is----> $'\
               +str(account_value)+'\n\n')
    delay(3000)
    mainMenue( )
    return 1
# Withdraw money
def withdraw( inVal, AMOUNT_MIN, AMOUNT_MAX, account_value, file ):
    ret = [0, 0]; # no error 1, error 0
    amount = 0
    if not re.match("^[0-9]*$", inVal):
        amount = 0
        print("\nInvalid Input only nmbers are alowed.\n")
        ret=[0, inVal]
    else:
        amount = int(inVal)
        if amount >= AMOUNT_MIN and amount <= account_value :
            account_value = account_value - amount
            print("\nWITHDRAW AMOUNT $%.2f\n" % amount)
            file.write('Your withdraw amounte is----------> $'\
                       +str(amount)+'\n\n')
            amount = 0
            delay(3000)
            mainMenue( )
            ret=[1, account_value]
        else:
            ret=[0, inVal]
            print("\nWITHDRAW AMOUNT $%.2f TOO BIG OR TOO SMALL\n" % amount)
    return ret
# Deposit money
def deposit( inVal, AMOUNT_MIN, AMOUNT_MAX, account_value, file ):
```

```
    ret = [0, 0]; # no error 1, error 0
    amount = 0
    if not re.match("^[0-9]*$", inVal):
        amount = 0
        print("\nInvalid Input only numbers are alowed.\n")
        ret=[0, inVal]
    else:
        amount = int(inVal)
        if amount >= AMOUNT_MIN and amount <= AMOUNT_MAX :
            account_value = account_value + amount
            print("\nDEPOSIT AMOUNT $%.2f\n" % amount)
            file.write('Your deposit amounte is----------> $'\
                       +str(amount)+'\n\n')
            amount = 0
            delay(3000)
            mainMenue( )
            ret=[1, account_value]
        else:
            ret=[0, inVal]
            print("\nDEPOSIT AMOUNT $%.2f TOO BIG OR TOO SMALL\n" % amount)
    return ret
# set new pin
def newPin( inVal, file ):
    ret = [0, 0]; # no error 1, error 0
    if not re.match("^[0-9]{4}$", inVal):
        print ("Error! Make sure you only use numbers from 0-9 in PIN")
        delay(3000)
        mainMenue( )
        inVal = 'Z'
        ret=[0, inVal]
    else:
        ret =[1, inVal]
#        print(ret) # debug only
        print("\nYOUR NEW PIN IS", inVal, "\n")
        file.write('You have changed your PIN number\n\n')
        delay(3000)
        mainMenue()
    return ret
# selection main menue
def selMainMenue( inVal ):
    ret = 0; # no error 1, error 0
    if not re.match("^[1-5]*$", inVal):
        print ("Error! Make sure you only use numbers from 1-5 in selecion")
        ret=[0, inVal]
    else:
        ret =[1, inVal]
    return ret
# PIN validation
def pin( inVal, PIN ):
    ret = 0; # no error 1, error 0
#    while True:
    if not re.match("^[0-9]{4}$", inVal):
        print ("Error! Make sure you only use numbers from 0-9 in PIN")
        inVal = 'Z'
        ret = 0
    else:
        if inVal == PIN :
            #print("\nCorect PIN") # debug only
            print("\n")
            mainMenue()
            inVal = 'Z'
            ret = 1
#                break
        else:
            print("\nInvalid PIN!")
            inVal = 'Z'
            ret = 0
    return ret
```

```
118
119 def delay( msec ):
120     cnt = 0
121     while cnt < msec:
122         cnt += 0.0001
123     return 1  # no error 1, error 0
124
125 def mainMenue( ):
126     print(" Main Menu")
127     print("----------------------- \n")
128     print(" 1. CHECK BALANCE")
129     print(" 2. WITHDRAW CASH")
130     print(" 3. DEPOSIT CASH")
131     print(" 4. CHANGE PIN")
132     print(" 5. EXIT ")
133     return 1  # no error 1, error 0
134
135 def welcome( ):
136     print("\n     $$$$$            $$           $$$            $$    $$      $   ")
137     print("     $     $          $$$$          $$ $           $$    $$     $    ")
138     print("     $      $        $    $         $$  $          $$    $$    $     ")
139     print("     $      $       $$    $$        $$   $         $$    $$   $      ")
140     print("     $     $       $$      $$       $$    $        $$    $$ $       ")
141     print("     $$$$$         $$      $$       $$     $       $$    $$$        ")
142     print("     $$$$$         $$$$$$$$$        $$      $      $$    $$$        ")
143     print("     $     $      $$        $$      $$       $     $$    $$ $       ")
144     print("     $      $    $$          $$     $$        $    $$    $$  $      ")
145     print("     $      $    $$          $$     $$          $ $$    $$   $     ")
146     print("     $     $    $$            $$    $$           $$$    $$    $    ")
147     print("     $$$$$     $$            $$    $$             $$    $$     $   \n")
148     print("               $$$           $$    $$$$$$$$$$$$$$  ")
149     print("               $$ $         $$$    $$$$$$$$$$$$$$   ")
150     print("               $$  $       $ $$    $$       ")
151     print("               $$   $     $  $$    $$      ")
152     print("               $$    $   $   $$    $$      ")
153     print("               $$     $$     $$    $$$$$$$$$$$$$$        ")
154     print("               $$            $$    $$$$$$$$$$$$$$        ")
155     print("               $$            $$    $$       ")
156     print("               $$            $$    $$       ")
157     print("               $$            $$    $$       ")
158     print("               $$            $$    $$$$$$$$$$$$$$   ")
159     print("               $$            $$    $$$$$$$$$$$$$$  ")
160     print("-------------------------------------------------------------------")
161     print("                    Welcome to Bank Me  ")
162     print("===================================================================")
163     return 1  # no error 1, error 0
```

Listing 11: Python code for an ATM Part II functions.

4.3 Python code Part III

```
1 # -*- coding: utf-8 -*-
2 """
3 Spyder Editor
4
5 This is a temporary script file.
6 """
7 from time import asctime
8 import re
9 #from atm_class import welcome, pin, \
10 #selMainMenue, newPin, deposit, withdraw, balance, receipt, delay
11 from atm_class import Atm
12
13 atm = Atm(100,'2345') # instantiate class Atm to atm object
14
15 # Open a file
16 try:
17     fo = open("07_lab_error_log.txt", "a")
```

```
    fr = open(atm.ReceiptFile, "w")
    fo.write(asctime()+ ' ATM program starts '+ '\n')
    fr.write('\n--------------------Bank Me-------------------\n')
    fr.write('----------------Transaction Receipt-------------\n')
    fr.write('*********************************************\n\n')
except IOError as e:
    print('File '+e.filename+' could not be opened or writen to!')

# Star program
if atm.welcome(): # first method used from the atm object
    error = 0
else:
    error = 1
    fo.write(asctime()+ ' welcome() error\n')

# PIN validation
while True:
    ret = 0
    if atm.pin( input("Please enter your PIN: "), atm.Pin):
        error = 0
        fo.write(asctime()+ ' User logged in \n')
        break
    else:
        error = 1
        fo.write(asctime()+ ' User login error\n')

# Main menu
while  atm.Select != "5":
    ret = 0
    ret = atm.selMainMenue(input("PLEASE SELECT FROM THE ABOVE OTPIONS: ") )
#    print(ret) # debug only
    if (ret[0]):
        error = 0
        atm.Select = ret[1]
#        print(ret[1] ) # debug only
    else: # error
        error = 1
        atm.Select = 0
        fo.write(asctime()+ ' User slection error\n')
        print(ret[0])

    if atm.Select == '5': # exit
        while True:
            if not re.match("^[y]$", input('Do you wish to print the receipt, press y or n?
    ')):
                print("\nThank you for chosing BANK ME!")
                break
            else:
                atm.receipt(atm.ReceiptFile, fr)
                break
        exit
    elif atm.Select == '4': # change PIN
        while True:
            ret = 0
            ret = atm.newPin(input("ENTER YOUR NEW PIN: "), fr)
#            print(ret) # debug only
            if (ret[0]):
                error = 0
                atm.Pin = ret[1]
        #        print(ret[1] ) # debug only
                break
            else: # error
                error = 1
                atm.Select = 0
                fo.write(asctime()+ ' User PIN error\n')
                break

    elif atm.Select == '3': # Deposit
```

```
        while True:
            ret = 0
            ret = atm.deposit(input("ENTER YOUR DEPOSIT AMOUNT IN $: ")\
                              , atm.AMOUNT_MIN, atm.AMOUNT_MAX, atm.AccountValue, fr)
#            print(ret) # debug only
            if (ret[0]):
                error = 0
                atm.AccountValue = ret[1]
        #        print(ret[1] ) # debug only
                break
            else: # error
                error = 1
                atm.Select = 0
                fo.write(asctime()+ ' Deposit error\n')

    elif atm.Select == '2': # Withdraw
        while True:
            ret = 0
            ret = atm.withdraw(input("ENTER YOUR WITHDRAW AMOUNT IN $: ")\
                               , atm.AMOUNT_MIN, atm.AMOUNT_MAX, atm.AccountValue, fr)
#            print(ret) # debug only
            if (ret[0]):
                error = 0
                atm.AccountValue = ret[1]
        #        print(ret[1] ) # debug only
                break
            else: # error
                error = 1
                atm.Select = 0
                fo.write(asctime()+ ' Withdraw error\n')

    elif atm.Select == '1': # Balance
        atm.balance(atm.AccountValue, fr)
    else:
        print("\nIncorrect Selecion!")

fo.write(asctime()+ ' Program Closed\n')
fo.close() # to ensure file is closed
fr.close() # to ensure file is closed
atm.delay(3000)
```

Listing 12: Python instance for an ATM Part III.

```
# -*- coding: utf-8 -*-
"""
Created on Thu Oct 18 18:34:49 2018

@author: schwa
"""
import re
class Atm:
    AMOUNT_MIN = 0
    AMOUNT_MAX = 1000

    Pin  = "1234"
    Select = 0
    AccountValue = 0
    ReceiptFile = "atm_receipt.txt"

    def __init__(self, AccountValue, Pin):
        self.AccountValue = AccountValue
        self.Pin = Pin

    # Print Receipt
    def receipt(self, filename, file):
        file.write('THANK YOU FOR CHOOSING BANK ME!')
        file.close()
        file = open(filename, "r")
```

```
        for line in file:
            print(line, end='')
        file.close()
        return 1
    # Balance
    def balance(self, account_value , file):
        print("\nYOUR BALANCE IS $%.2f \n \n\r" % account_value)
        file.write('Your checking account balance is----> $'\
                   +str(account_value)+'\n\n')
        self.delay(3000)
        self.mainMenue( )
        return 1
    # Withdraw money
    def withdraw(self,  inVal, AMOUNT_MIN, AMOUNT_MAX, account_value, file ):
        ret = [0, 0]; # no error 1, error 0
        amount = 0
        if not re.match("^[0-9]*$", inVal):
            amount = 0
            print("\nInvalid Input only nmbers are alowed.\n")
            ret=[0, inVal]
        else:
            amount = int(inVal)
            if amount >= AMOUNT_MIN and amount <= account_value :
                account_value = account_value - amount
                print("\nWITHDRAW AMOUNT $%.2f\n" % amount)
                file.write('Your withdraw amounte is----------> $'\
                           +str(amount)+'\n\n')
                amount = 0
                self.delay(3000)
                self.mainMenue( )
                ret=[1, account_value]
            else:
                ret=[0, inVal]
                print("\nWITHDRAW AMOUNT $%.2f TOO BIG OR TOO SMALL\n" % amount)
        return ret
    # Deposit money
    def deposit( self, inVal, AMOUNT_MIN, AMOUNT_MAX, account_value, file ):
        ret = [0, 0]; # no error 1, error 0
        amount = 0
        if not re.match("^[0-9]*$", inVal):
            amount = 0
            print("\nInvalid Input only numbers are alowed.\n")
            ret=[0, inVal]
        else:
            amount = int(inVal)
            if amount >= AMOUNT_MIN and amount <= AMOUNT_MAX :
                account_value = account_value + amount
                print("\nDEPOSIT AMOUNT $%.2f\n" % amount)
                file.write('Your deposit amounte is----------> $'\
                           +str(amount)+'\n\n')
                amount = 0
                self.delay(3000)
                self.mainMenue( )
                ret=[1, account_value]
            else:
                ret=[0, inVal]
                print("\nDEPOSIT AMOUNT $%.2f TOO BIG OR TOO SMALL\n" % amount)
        return ret
    # set new pin
    def newPin( self, inVal, file ):
        ret = [0, 0]; # no error 1, error 0
        if not re.match("^[0-9]{4}$", inVal):
            print ("Error! Make sure you only use numbers from 0-9 in PIN")
            self.delay(3000)
            self.mainMenue( )
            inVal = 'Z'
            ret=[0, inVal]
        else:
```

```
            ret =[1, inVal]
    #        print(ret) # debug only
            print("\nYOUR NEW PIN IS", inVal, "\n")
            file.write('You have changed your PIN number\n\n')
            self.delay(3000)
            self.mainMenue()
        return ret
    # selection main menue
    def selMainMenue(self, inVal ):
        ret = 0; # no error 1, error 0
        if not re.match("^[1-5]*$", inVal):
            print ("Error! Make sure you only use numbers from 1-5 in selecion")
            ret=[0, inVal]
        else:
            ret =[1, inVal]
        return ret
    # PIN validation
    def pin( self, inVal, PIN ):
        ret = 0; # no error 1, error 0
    #    while True:
        if not re.match("^[0-9]{4}$", inVal):
            print ("Error! Make sure you only use numbers from 0-9 in PIN")
            inVal = 'Z'
            ret = 0
        else:
            if inVal == PIN :
                #print("\nCorect PIN") # debug only
                print("\n")
                self.mainMenue()
                inVal = 'Z'
                ret = 1
    #               break
            else:
                print("\nInvalid PIN!")
                inVal = 'Z'
                ret = 0
        return ret

    def delay(self, msec ):
        cnt = 0
        while cnt < msec:
            cnt += 0.0001
        return 1  # no error 1, error 0

    def mainMenue(self):
        print(" Main Menu")
        print("---------------------- \n")
        print(" 1. CHECK BALANCE")
        print(" 2. WITHDRAW CASH")
        print(" 3. DEPOSIT CASH")
        print(" 4. CHANGE PIN")
        print(" 5. EXIT ")
        return 1  # no error 1, error 0

    def welcome(self):
        print("\n    $$$$$          $$          $$$          $$   $$     $  ")
        print("    $    $         $$$$         $$ $         $$   $$    $   ")
        print("    $     $        $   $        $$  $        $$   $$   $    ")
        print("    $     $       $$   $$       $$   $       $$   $$  $     ")
        print("    $    $       $$     $$      $$    $      $$   $$ $      ")
        print("    $$$$$        $$     $$      $$     $     $$   $$$       ")
        print("    $$$$$        $$$$$$$$$      $$      $    $$   $$$       ")
        print("    $    $      $$       $$     $$       $   $$   $$ $      ")
        print("    $     $    $$         $$    $$        $  $$   $$  $     ")
        print("    $     $    $$         $$    $$         $ $$   $$   $    ")
        print("    $    $    $$           $$   $$          $$$   $$    $   ")
        print("    $$$$$     $$           $$   $$          $$   $$     $  \n")
        print("              $$$          $$   $$$$$$$$$$$$$$  ")
```

```
162         print("               $$ $         $$$    $$$$$$$$$$$$$$    ")
163         print("               $$  $       $ $$    $$       ")
164         print("               $$   $     $  $$    $$      ")
165         print("               $$    $   $   $$    $$      ")
166         print("               $$      $$     $$    $$$$$$$$$$$$$$$        ")
167         print("               $$             $$    $$$$$$$$$$$$$$$        ")
168         print("               $$             $$    $$       ")
169         print("               $$             $$    $$       ")
170         print("               $$             $$    $$       ")
171         print("               $$             $$    $$$$$$$$$$$$$$$    ")
172         print("               $$             $$    $$$$$$$$$$$$$$$  ")
173         print("_____________________________________________________________________")
174         print("                    Welcome to Bank Me  ")
175         print("=====================================================================")
176         return 1  # no error 1, error 0
```

Listing 13: Python class for an ATM Part III.