VALUE TRANSFER PRIVACY REVIEW THROUGH CRYPTOCURRENCIES

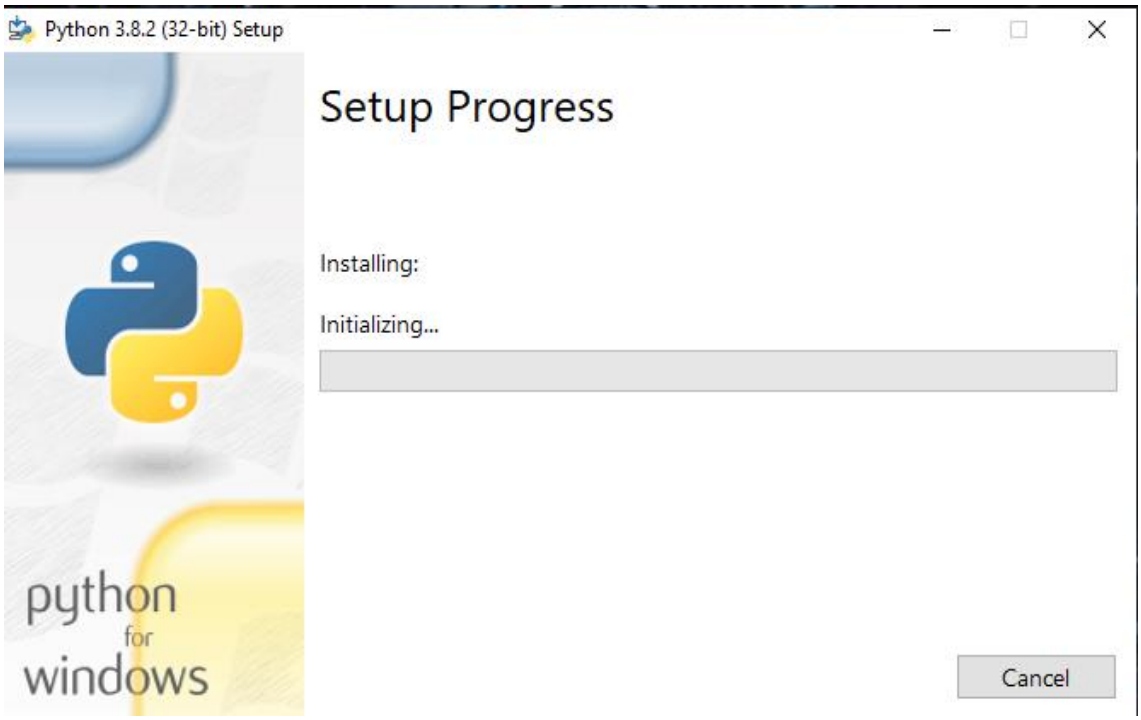
Technical Manual

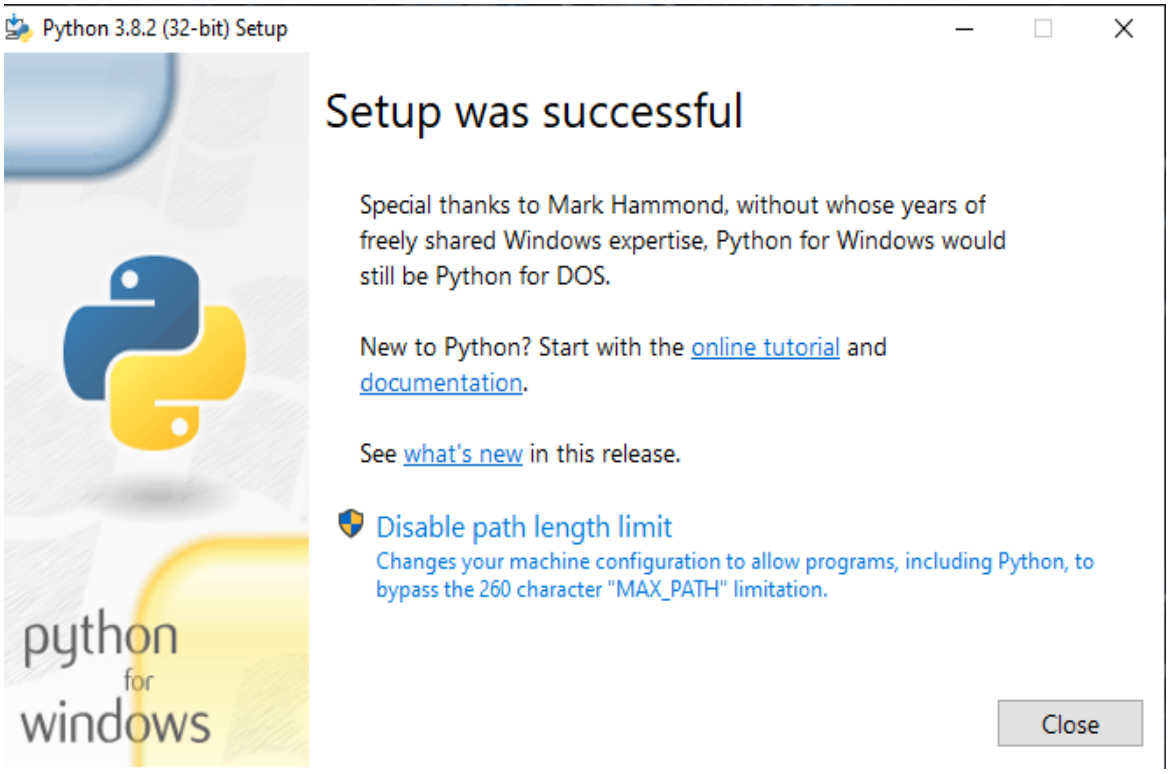
by
Tomas Coleman

Student ID: C00218923
Institute of Technology Carlow
Supervised by: Richard Butler
20th April 2020

Introduction 3
Installation and Setup Requirements 4
Installing Python 4
Installing PyQT 7
Installing Visual Studio 7
Installing SQL 9

D. Install and wait for the setup to be completed.





Installing PyQT

A. Ensure that pip and its necessary packages are up-to-date.

Commands:

- python -m pip install –upgrade pip
- pip install –upgrade setuptools

B. Use pip (The Python Package Installer) to install PyQT5 using a command line, which is necessary for the GUI of the application.

Command:

- pip install PyQT5

Installing Visual Studio

A. Download the appropriate visual studio file for your processor from the following source;

https://support.microsoft.com/en-us/help/2977003/the-latest-supported-visual-c-downloads

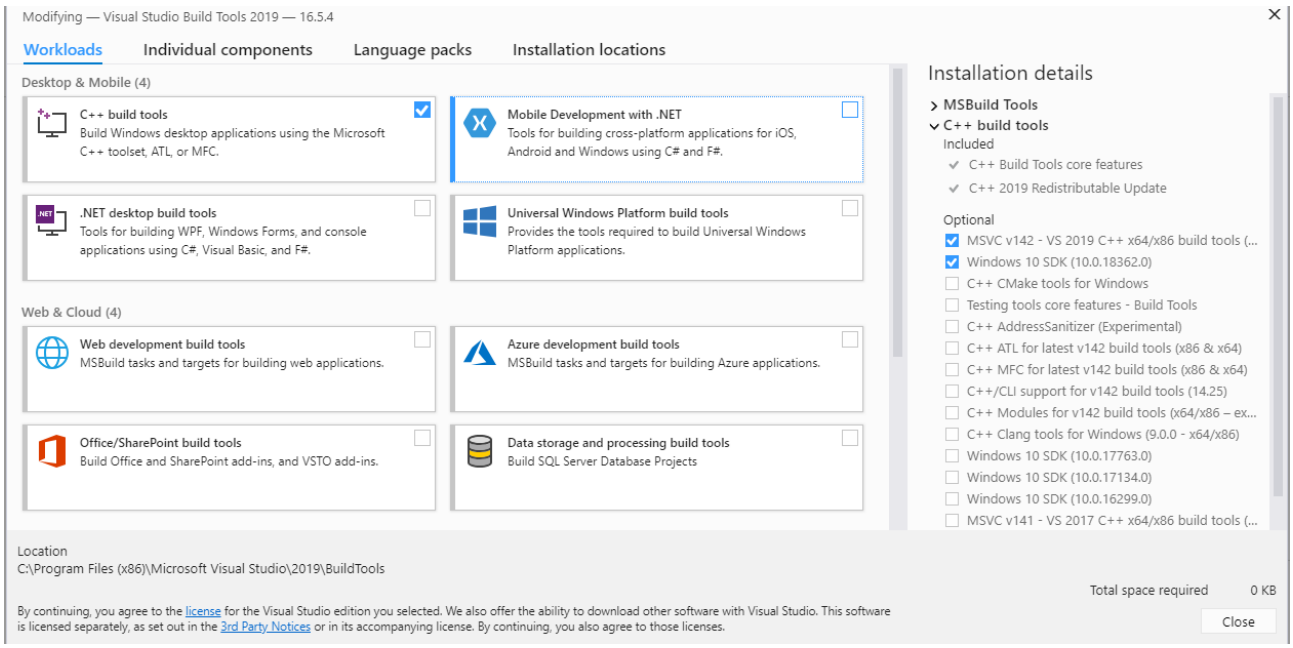
Visual Studio 2015, 2017 and 2019

Download the Microsoft Visual C++ Redistributable for Visual Studio 2015, 2017 and 2019. The following updates are the latest supported Visual C++ redistributable packages for Visual Studio 2015, 2017 and 2019. Included is a baseline version of the Universal C Runtime see MSDN for details.

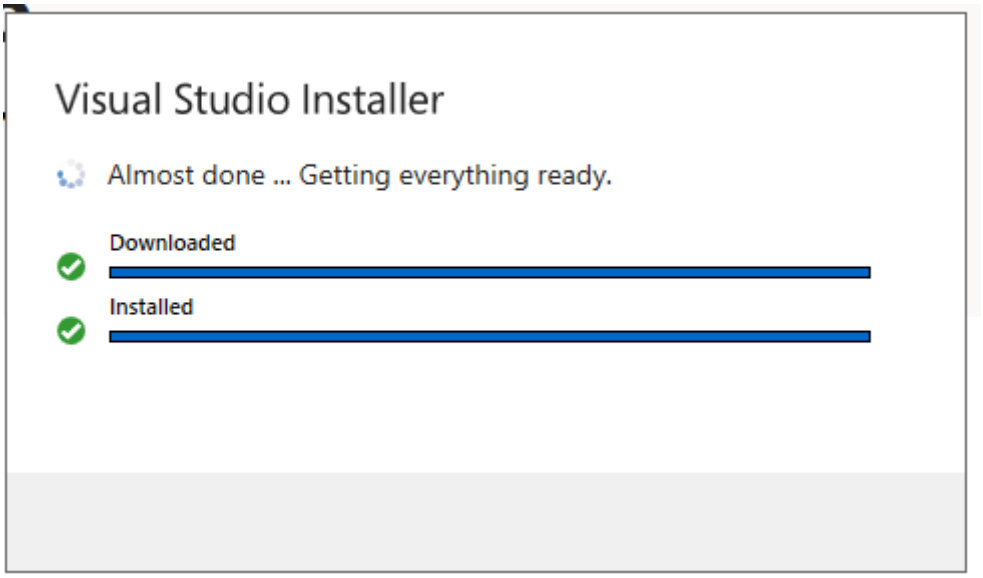
- x86: vc_redist.x86.exe
- x64: vc_redist.x64.exe
- ARM64: vc_redist.arm64.exe

B. Run the installer once downloaded. Select the following options from the list provided;

- C++ Build Tools
- Windows 10 SDK



C. Wait for the download and installation process to be completed.



Installing SQL

A. Install MySQL connector. This is necessary to be able to access the database.

Command:

- pip install mysql-connector

The back end statement used by the application to add new users to the database is;

```
by CREATE USER 'root'@'ip_address' IDENTIFIED BY 'some_pass';
GRANT ALL PRIVILEGES ON *.* TO 'root'@'ip_address';
```

Run the Monero testnet

A. Launch the executable file called monero-wallet-cli.exe for windows or monero-wallet-CLI for Linux. In you CLI you can use the command provided to run the Monero testnet ahead of time as this will several hours.. The Linux version is able to fork to background so that you don't need to leave the shell open. Please ensure that you user has the right to run the following file.

Command:

Linux: ./monerod --testnet --detach

Windows: monerod.exe –testnet

Installing Other Dependencies

A. Install pysha3 module, as this is necessary for password hashing within the application.

Command:

- pip install pysha3

B. Install qrcode module.

Command:

- pip install qrcode

C. Install requests.

Command:

- pip install requests

Once users have this setup completed, they will be able to run the program by right clicking “main.py” and run using a python interpreter, for example IDLE.

Code

I will go through the code in the classes that they are accessed in. The Main.py class is the launcher for the program. The commented text at the start of this file is placed here to show that there is copyrighted code in this program. This is for the money python code which are the python bindings that translate python calls into JSON format so that the monero code can understand it.

Main.py

```
#BSD 3-Clause License
#
  & R S \ U L J K W    F            0 L F K D á  6 D á D E D Q
#Copyright (c) 2016 The MoneroPy Developers
#All rights reserved.
import sys

from PyQt5.QtWidgets import QApplication, QMainWindow, QPushButton, QWidget
from Layer1.MainWindow import *
from Layer1.Database import *
from Layer1.BeforeWallet import *
from threading import Thread

if __name__ == '__main__':
    app = QApplication(sys.argv)
    before = beforeWallet()
    before.startUp()
    CurrentWallet = Wallet(JSONRPCWallet(port=28088))
    Data = Database()
    DbConnection = Data.startUpConnection()
    MainWindow = QtWidgets.QMainWindow()
    MainWindowUi = Ui_MainWindow()
    MainWindowUi.setupUi(MainWindow, CurrentWallet)
    MainWindow.show()
    sys.exit(app.exec_())
```

BeforeWallet.py

The BeforeWallet class is called first. As the name suggests, this class runs setup code that is required to be launched before people use the wallet. This includes updating the monero blockchain, running the requirements of monero-python, creating an instance of a monero wallet and moving it into where the code can access it. Finally, it runs the monero rpc.

```
import subprocess
import time
import os
import shutil
from threading import Thread
class beforeWallet:
    def startUp(self):
        try:
            subprocess.run(['./Layer1/monero/monerod', '--testnet', '--detach'])
        except:
            print("error starting monero")
        try:
            subprocess.run(['python3', '-m', 'pip', 'install', '-r', 'Layer1/moneroPython/requirements.txt', '-
r', 'Layer1/moneroPython/test_requirements.txt'])
        except:
            print("error starting pyhton3 pip")
        try:
            CreateWallet = subprocess.Popen(['./Layer1/monero/monero-wallet-cli', '--testnet', '--
generate-new-wallet', 'test90003', '--password', '*redacted*', '--mnemonic-language', 'English'],
stdin=subprocess.PIPE, stdout=subprocess.PIPE)
            CreateWallet.stdin.write(b'1')
            CreateWallet.stdin.write(b'N')
            CreateWallet.stdin.write(b'^z')
        except:
            print("error Creating wallet")
        try:
            moving1 =
"/home/c00218923/Documents/College/Project/MoneroWallet/Code/test90003.address.txt"
            moving2 =
"/home/c00218923/Documents/College/Project/MoneroWallet/Code/Layer1/monero/test90003.addr
ess.txt"
            os.rename(moving1, moving2)
            shutil.move(moving1, moving2)
            os.replace(moving1, moving2)
        except:
            print("error moving wallet")
        try:
            moving3 =
"/home/c00218923/Documents/College/Project/MoneroWallet/Code/test90003.keys"
            moving4 =
```

```
"/home/c00218923/Documents/College/Project/MoneroWallet/Code/Layer1/monero/test90003.keys
"
            os.rename(moving3, moving4)
            shutil.move(moving3, moving4)
            os.replace(moving3, moving4)
        except:
            print("error moving wallet")
        try:
            moving5 = "/home/c00218923/Documents/College/Project/MoneroWallet/Code/test90003"
            moving6 =
"/home/c00218923/Documents/College/Project/MoneroWallet/Code/Layer1/monero/test90003"
            os.rename(moving5, moving6)
            shutil.move(moving5, moving6)
            os.replace(moving5, moving6)
        except:
            print("error moving wallet")
        time.sleep(5)
        try:
            subprocess.run(['./Layer1/monero/monero-wallet-rpc', '--testnet', '--wallet-file', 'test90003', '-
-password', '"*redacted*"', '--rpc-bind-port 28088', '--disable-rpc-login'])
        except:
            print("error starting monero rpc")
```

Database.py

The Database class handles all of the authentication required for users of the program. This starts with the class connecting to the server. It will then add users in for demo use only. This would not be left in if I were to allow other users to use this. The class will also add in users when they register if they have a strong enough password and the username chosen is not already being used by another user. It will then be used to allow a user to login.

```
import mysql.connector
import sys
import os
import time
import random
import bcrypt
import string
import re
class Database:
    mydb =  mysql.connector.connect(host="localhost",user="root")
    def startUpConnection(self):
        try:
            mycursor = Database.mydb.cursor()
            mycursor.execute("CREATE DATABASE Wallet")
        except:
            e = sys.exc_info()[0]
            print("Error: %s" % e )
        try:
            mycursor = Database.mydb.cursor()
            Database.mydb = mysql.connector.connect(host="localhost",user="root",database="Wallet")
        except:
            e = sys.exc_info()[0]
            print("Error: %s" % e )
        try:
            mycursor = Database.mydb.cursor()
            mycursor.execute("CREATE TABLE Users (Id INT(11) KEY AUTO_INCREMENT,
Username VARCHAR(255) UNIQUE, Password VARCHAR(255), SessionId VARCHAR(255))")
        except:
            e = sys.exc_info()[0]
            print("Error: %s" % e )
        try:
            PasswordInBytes = bytes("admin", encoding='utf8')
            Salt = bcrypt.gensalt()
            HashedPassword = bcrypt.hashpw(PasswordInBytes, Salt)
            SessionId = "temp"
            sql = "INSERT INTO Users(Username, Password, SessionId) VALUES (%s, %s, %s)"
            val = ("admin", HashedPassword, SessionId)
            mycursor = Database.mydb.cursor()
            mycursor.execute(sql, val)
```

```
        Database.mydb.commit()
    except:
        e = sys.exc_info()[0]
        print("Error: %s" % e )
def createUsername(username, Password, ConPassword):
    varForReturn = None
    if Password == ConPassword:
        if Database.checkPasswordStrength(Password):
            try:
                PasswordInBytes = bytes(Password, encoding='utf8')
                Salt = bcrypt.gensalt()
                HashedPassword = bcrypt.hashpw(PasswordInBytes, Salt)
                SessionId = "temp"
                sql = "INSERT INTO Users(Username, Password, SessionId) VALUES (%s, %s, %s)"
                val = (username, HashedPassword, SessionId)
                mycursor = Database.mydb.cursor()
                mycursor.execute(sql, val)
                Database.mydb.commit()
                return "PasswordRight"
            except:
                return "UsernameTaken"
        else:
            return "PasswordNotStrongAnuf"
    else:
        return "ConPasswordWrong"

def checkPasswordStrength(password):
    if len(password) > 15:
        if re.search(r"\d", password):
            if re.search(r"[a-z]", password):
                if re.search(r"[A-Z]", password):
                    if re.search(r"[ !#$%&'()*+,-./[\\\]^_`{|}~"+r"']", password):
                        return True
                    else:
                        return False
                else:
                    return False
            else:
                return False
        else:
            return False
    else:
        return False
def checkLogin(username, password):
    varForReturn = None
    mycursor = Database.mydb.cursor()
```

```
    sql = "SELECT Username FROM Users WHERE Username = %s"
    adr = (username, )
    mycursor.execute(sql, adr)
    try:
        result = mycursor.fetchone()
        for x in result:
            username = x
        if Database.checkPassword(username, password):
            return True
        else:
            return False
    except:
        return False
def checkPassword(username, password):
    passwordInBytes = bytes(password, encoding='utf8')
    hashedPasswordServer = bytes(Database.getPassword(username), encoding='utf8')
    if bcrypt.checkpw(passwordInBytes, hashedPasswordServer):
        return True
    else:
        return False
def getPassword(username):
    var = None
    mycursor = Database.mydb.cursor()
    sql = "SELECT Password FROM Users WHERE Username = %s"
    adr = (username,)
    mycursor.execute(sql, adr)
    varForReturn = mycursor.fetchone()
    for x in varForReturn:
        varForReturn = x
    return varForReturn
```

QrCode.py

QrCode is the class that launches a window that translates your address into a QR code so that people can scan into another monero wallet in a timely manner. This also ensures that there is no misspelling so that value is transferred correctly.

```
from PyQt5 import QtWidgets, QtCore, QtGui
import qrcode

class Image(qrcode.image.base.BaseImage):
    def __init__(self, border, width, box_size):
        self.border = border
        self.width = width
        self.box_size = box_size
        size = (width + border * 2) * box_size
        self._image = QtGui.QImage(
            size, size, QtGui.QImage.Format_RGB16)
        self._image.fill(QtCore.Qt.white)

    def pixmap(self):
        return QtGui.QPixmap.fromImage(self._image)

    def drawrect(self, row, col):
        painter = QtGui.QPainter(self._image)
        painter.fillRect(
            (col + self.border) * self.box_size,
            (row + self.border) * self.box_size,
            self.box_size, self.box_size,
            QtCore.Qt.black)

    def save(self, stream, kind=None):
        pass

class Window(QtWidgets.QWidget):
    def __init__(self, address):
        print(address)
        QtWidgets.QWidget.__init__(self)
        _translate = QtCore.QCoreApplication.translate
        self.setWindowTitle(_translate("self", "QRCode"))
        self.label = QtWidgets.QLabel(self)
        self.edit = QtWidgets.QLineEdit(self)
        self.edit.setText(address)
        self.handleTextEntered(address)
        layout = QtWidgets.QVBoxLayout(self)
        layout.addWidget(self.label)
        layout.addWidget(self.edit)
```

```
    def handleTextEntered(self, address):
        text = address
        self.label.setPixmap(
            qrcode.make(text, image_factory=Image).pixmap())

class QrCode(QtWidgets.QWidget):
    def __init__(self, MainWindow, address):
        super(QrCode, self)
        import sys
        MainWindow.window = Window(address)
        MainWindow.window.setGeometry(500, 300, 200, 200)
        MainWindow.window.show()
```

MainWindow.p y

The MainWindow class handles the complete GUI. It handle all the signals and slots for the program which are the event handling, like the buttons and calls for all of the other functions.

```
from PyQt5 import QtCore, QtGui, QtWidgets
from PyQt5.Qt import QDialog
from PyQt5.QtGui import QImage, QPalette, QBrush
from PyQt5.QtCore import QSize
from .BeforeWallet import *
from .Database import *
from Layer1.moneroPython.monero.backends.jsonrpc import JSONRPCWallet
from Layer1.moneroPython.monero.wallet import *
from Layer1.moneroPython.monero.base58 import *
from Layer1.moneroPython.monero.numbers import *
from .QrCode import *
from threading import Thread, Timer
from functools import partial

class Ui_MainWindow(object):
    def setupUi(self, MainWindow, CurrentWallet):
        MainWindow.setObjectName("MainWindow")
        MainWindow.resize(1021, 442)
        self.centralwidget = QtWidgets.QWidget(MainWindow)
        self.centralwidget.setObjectName("centralwidget")
        self.gridLayout_2 = QtWidgets.QGridLayout(self.centralwidget)
        self.gridLayout_2.setObjectName("gridLayout_2")
        self.stackedWidget = QtWidgets.QStackedWidget(self.centralwidget)
        self.stackedWidget.setObjectName("stackedWidget")
        self.page = QtWidgets.QWidget()
        self.page.setObjectName("page")
        self.gridLayout_5 = QtWidgets.QGridLayout(self.page)
        self.gridLayout_5.setObjectName("gridLayout_5")
```

```
self.labelBalance = QtWidgets.QLabel(self.page)
self.labelBalance.setObjectName("labelBalance")
self.gridLayout_5.addWidget(self.labelBalance, 4, 0, 1, 1)
self.labelAmount = QtWidgets.QLabel(self.page)
self.labelAmount.setObjectName("labelAmount")
self.gridLayout_5.addWidget(self.labelAmount, 6, 0, 1, 1)
self.labelAmountInfo = QtWidgets.QLabel(self.page)
self.labelAmountInfo.setObjectName("labelAmountInfo")
self.gridLayout_5.addWidget(self.labelAmountInfo, 6, 1, 1, 1)
self.labelMainAddress_2 = QtWidgets.QLabel(self.page)
self.labelMainAddress_2.setObjectName("labelMainAddress_2")
self.gridLayout_5.addWidget(self.labelMainAddress_2, 3, 0, 1, 1)
self.lineEditAmmount = QtWidgets.QLineEdit(self.page)
self.lineEditAmmount.setObjectName("lineEditAmmount")
self.gridLayout_5.addWidget(self.lineEditAmmount, 6, 2, 1, 1)
self.lineFee = QtWidgets.QLineEdit(self.page)
self.lineFee.setObjectName("lineFee")
self.gridLayout_5.addWidget(self.lineFee, 7, 2, 1, 1)
self.pushButton_8 = QtWidgets.QPushButton(self.page)
self.pushButton_8.setObjectName("pushButton_8")
self.gridLayout_5.addWidget(self.pushButton_8, 0, 0, 1, 1)
self.labelBalanceInfo = QtWidgets.QLabel(self.page)
self.labelBalanceInfo.setObjectName("labelBalanceInfo")
self.gridLayout_5.addWidget(self.labelBalanceInfo, 4, 1, 1, 1)
self.pushButton_9 = QtWidgets.QPushButton(self.page)
self.pushButton_9.setObjectName("pushButton_9")
self.gridLayout_5.addWidget(self.pushButton_9, 1, 1, 1, 1)
self.labelReceivingAddress = QtWidgets.QLabel(self.page)
self.labelReceivingAddress.setObjectName("labelReceivingAddress")
self.gridLayout_5.addWidget(self.labelReceivingAddress, 5, 0, 1, 1)
self.labelReceivingAddressInfo = QtWidgets.QLabel(self.page)
self.labelReceivingAddressInfo.setObjectName("labelReceivingAddressInfo")
self.gridLayout_5.addWidget(self.labelReceivingAddressInfo, 5, 1, 1, 1)
self.lineEditAddress = QtWidgets.QLineEdit(self.page)
self.lineEditAddress.setObjectName("lineEditAddress")
self.gridLayout_5.addWidget(self.lineEditAddress, 5, 2, 1, 1)
self.pushButton_7 = QtWidgets.QPushButton(self.page)
self.pushButton_7.setObjectName("pushButton_7")
self.gridLayout_5.addWidget(self.pushButton_7, 1, 0, 1, 1)
self.labelHeading = QtWidgets.QLabel(self.page)
self.labelHeading.setAlignment(QtCore.Qt.AlignCenter)
self.labelHeading.setObjectName("labelHeading")
self.gridLayout_5.addWidget(self.labelHeading, 0, 1, 1, 2)
self.labelFeeInfo = QtWidgets.QLabel(self.page)
self.labelFeeInfo.setObjectName("labelFeeInfo")
self.gridLayout_5.addWidget(self.labelFeeInfo, 7, 1, 2, 1)
```

```
        self.labelBalance_2 = QtWidgets.QLabel(self.page)
        self.labelBalance_2.setObjectName("labelBalance_2")
        self.gridLayout_5.addWidget(self.labelBalance_2, 4, 2, 1, 1)
        self.labelMainAddress = QtWidgets.QLabel(self.page)
        self.labelMainAddress.setObjectName("labelMainAddress")
        self.gridLayout_5.addWidget(self.labelMainAddress, 3, 2, 1, 1)
        self.labelFee = QtWidgets.QLabel(self.page)
        self.labelFee.setObjectName("labelFee")
        self.gridLayout_5.addWidget(self.labelFee, 7, 0, 1, 1)
        self.pushButton_4 = QtWidgets.QPushButton(self.page)
        self.pushButton_4.setObjectName("pushButton_4")
        self.gridLayout_5.addWidget(self.pushButton_4, 8, 2, 1, 1)
        self.labeMainAddressInfo = QtWidgets.QLabel(self.page)
        self.labeMainAddressInfo.setObjectName("labeMainAddressInfo")
        self.gridLayout_5.addWidget(self.labeMainAddressInfo, 3, 1, 1, 1)
        self.stackedWidget.addWidget(self.page)
        self.page_2 = QtWidgets.QWidget()
        self.page_2.setObjectName("page_2")
        self.gridLayout_4 = QtWidgets.QGridLayout(self.page_2)
        self.gridLayout_4.setObjectName("gridLayout_4")
        self.label = QtWidgets.QLabel(self.page_2)
        self.label.setAlignment(QtCore.Qt.AlignCenter)
        self.label.setObjectName("label")
        self.gridLayout_4.addWidget(self.label, 0, 0, 1, 1)
        self.pushButton_2 = QtWidgets.QPushButton(self.page_2)
        self.pushButton_2.setObjectName("pushButton_2")
        self.gridLayout_4.addWidget(self.pushButton_2, 1, 0, 1, 1)
        self.pushButton = QtWidgets.QPushButton(self.page_2)
        self.pushButton.setObjectName("pushButton")
        self.gridLayout_4.addWidget(self.pushButton, 2, 0, 1, 1)
        self.stackedWidget.addWidget(self.page_2)
        self.page_3 = QtWidgets.QWidget()
        self.page_3.setObjectName("page_3")
        self.gridLayout_3 = QtWidgets.QGridLayout(self.page_3)
        self.gridLayout_3.setObjectName("gridLayout_3")
        self.labelUsername = QtWidgets.QLabel(self.page_3)
        self.labelUsername.setObjectName("labelUsername")
        self.gridLayout_3.addWidget(self.labelUsername, 1, 0, 1, 1)
        self.buttonBox = QtWidgets.QDialogButtonBox(self.page_3)
        self.buttonBox.setOrientation(QtCore.Qt.Horizontal)

self.buttonBox.setStandardButtons(QtWidgets.QDialogButtonBox.Cancel|QtWidgets.QDialogButtonBox.Ok)
        self.buttonBox.setObjectName("buttonBox")
        self.gridLayout_3.addWidget(self.buttonBox, 7, 0, 1, 1)
        self.lineEditUsername_3 = QtWidgets.QLineEdit(self.page_3)
```

```
        self.lineEditUsername_3.setObjectName("lineEditUsername_3")
        self.gridLayout_3.addWidget(self.lineEditUsername_3, 2, 0, 1, 1)
        self.lineEditPassword_2 = QtWidgets.QLineEdit(self.page_3)
        self.lineEditPassword_2.setObjectName("lineEditPassword_2")
        self.gridLayout_3.addWidget(self.lineEditPassword_2, 5, 0, 1, 1)
        self.labelHeading_2 = QtWidgets.QLabel(self.page_3)
        self.labelHeading_2.setObjectName("labelHeading_2")
        self.gridLayout_3.addWidget(self.labelHeading_2, 0, 0, 1, 1)
        self.passwordLabel = QtWidgets.QLabel(self.page_3)
        self.passwordLabel.setObjectName("passwordLabel")
        self.gridLayout_3.addWidget(self.passwordLabel, 3, 0, 1, 1)
        self.stackedWidget.addWidget(self.page_3)
        self.page_7 = QtWidgets.QWidget()
        self.page_7.setObjectName("page_7")
        self.verticalLayout_2 = QtWidgets.QVBoxLayout(self.page_7)
        self.verticalLayout_2.setObjectName("verticalLayout_2")
        self.label_3 = QtWidgets.QLabel(self.page_7)
        self.label_3.setObjectName("label_3")
        self.verticalLayout_2.addWidget(self.label_3)
        self.pushButton_5 = QtWidgets.QPushButton(self.page_7)
        self.pushButton_5.setObjectName("pushButton_5")
        self.verticalLayout_2.addWidget(self.pushButton_5)
        self.stackedWidget.addWidget(self.page_7)
        self.page_4 = QtWidgets.QWidget()
        self.page_4.setObjectName("page_4")
        self.gridLayout = QtWidgets.QGridLayout(self.page_4)
        self.gridLayout.setObjectName("gridLayout")
        self.buttonBox_2 = QtWidgets.QDialogButtonBox(self.page_4)
        self.buttonBox_2.setOrientation(QtCore.Qt.Horizontal)

self.buttonBox_2.setStandardButtons(QtWidgets.QDialogButtonBox.Cancel|QtWidgets.QDialogBu
ttonBox.Ok)
        self.buttonBox_2.setObjectName("buttonBox_2")
        self.gridLayout.addWidget(self.buttonBox_2, 10, 0, 1, 1)
        self.lineEditUsername_4 = QtWidgets.QLineEdit(self.page_4)
        self.lineEditUsername_4.setObjectName("lineEditUsername_4")
        self.gridLayout.addWidget(self.lineEditUsername_4, 3, 0, 1, 1)
        self.lineEditPassword_3 = QtWidgets.QLineEdit(self.page_4)
        self.lineEditPassword_3.setObjectName("lineEditPassword_3")
        self.gridLayout.addWidget(self.lineEditPassword_3, 6, 0, 1, 1)
        self.labelConfirmPassword = QtWidgets.QLabel(self.page_4)
        self.labelConfirmPassword.setObjectName("labelConfirmPassword")
        self.gridLayout.addWidget(self.labelConfirmPassword, 8, 0, 1, 1)
        self.lineEditConfirmPassword = QtWidgets.QLineEdit(self.page_4)
        self.lineEditConfirmPassword.setObjectName("lineEditConfirmPassword")
        self.gridLayout.addWidget(self.lineEditConfirmPassword, 9, 0, 1, 1)
```

```
        self.labelHeading_3 = QtWidgets.QLabel(self.page_4)
        self.labelHeading_3.setObjectName("labelHeading_3")
        self.gridLayout.addWidget(self.labelHeading_3, 1, 0, 1, 1)
        self.labelPassword = QtWidgets.QLabel(self.page_4)
        self.labelPassword.setObjectName("labelPassword")
        self.gridLayout.addWidget(self.labelPassword,
5, 0, 1, 1)
        self.labelUsername_2 = QtWidgets.QLabel(self.page_4)
        self.labelUsername_2.setObjectName("labelUsername_2")
        self.gridLayout.addWidget(self.labelUsername_2, 2, 0, 1, 1)
        self.stackedWidget.addWidget(self.page_4)
        self.page_6 = QtWidgets.QWidget()
        self.page_6.setObjectName("page_6")
        self.verticalLayout = QtWidgets.QVBoxLayout(self.page_6)
        self.verticalLayout.setObjectName("verticalLayout")
        self.label_2 = QtWidgets.QLabel(self.page_6)
        self.label_2.setMidLineWidth(0)
        self.label_2.setObjectName("label_2")
        self.verticalLayout.addWidget(self.label_2)
        self.pushButton_6 = QtWidgets.QPushButton(self.page_6)
        self.pushButton_6.setObjectName("pushButton_6")
        self.verticalLayout.addWidget(self.pushButton_6)
        self.stackedWidget.addWidget(self.page_6)
        self.page_5 = QtWidgets.QWidget()
        self.page_5.setObjectName("page_5")
        self.gridLayout_6 = QtWidgets.QGridLayout(self.page_5)
        self.gridLayout_6.setObjectName("gridLayout_6")
        self.labelHeading_4 = QtWidgets.QLabel(self.page_5)
        self.labelHeading_4.setObjectName("labelHeading_4")
        self.gridLayout_6.addWidget(self.labelHeading_4, 0, 0, 1, 1)
        self.pushButton_3 = QtWidgets.QPushButton(self.page_5)
        self.pushButton_3.setObjectName("pushButton_3")
        self.gridLayout_6.addWidget(self.pushButton_3, 1, 0, 1, 1)
        self.stackedWidget.addWidget(self.page_5)
        self.page_8 = QtWidgets.QWidget()
        self.page_8.setObjectName("page_8")
        self.gridLayout_12 = QtWidgets.QGridLayout(self.page_8)
        self.gridLayout_12.setObjectName("gridLayout_12")
        self.label_7 = QtWidgets.QLabel(self.page_8)
        self.label_7.setMidLineWidth(0)
        self.label_7.setObjectName("label_7")
        self.gridLayout_12.addWidget(self.label_7, 0, 0, 1, 1)
        self.pushButton_15 = QtWidgets.QPushButton(self.page_8)
        self.pushButton_15.setObjectName("pushButton_15")
        self.gridLayout_12.addWidget(self.pushButton_15, 1, 0, 1, 1)
```

```
        self.stackedWidget.addWidget(self.page_8)
        self.page_9 = QtWidgets.QWidget()
        self.page_9.setObjectName("page_9")
        self.label_4 = QtWidgets.QLabel(self.page_9)
        self.label_4.setGeometry(QtCore.QRect(20, 30, 985, 328))
        self.label_4.setObjectName("label_4")
        self.stackedWidget.addWidget(self.page_9)
        self.gridLayout_2.addWidget(self.stackedWidget, 0, 0, 1, 1)
        MainWindow.setCentralWidget(self.centralwidget)
        self.statusbar = QtWidgets.QStatusBar(MainWindow)
        self.statusbar.setObjectName("statusbar")
        MainWindow.setStatusBar(self.statusbar)
        self.actionCreate_MainAddress = QtWidgets.QAction(MainWindow)
        self.actionCreate_MainAddress.setObjectName("actionCreate_MainAddress")
        self.actionCreate_SubAddress = QtWidgets.QAction(MainWindow)
        self.actionCreate_SubAddress.setObjectName("actionCreate_SubAddress")
        self.actionView_Keys = QtWidgets.QAction(MainWindow)
        self.actionView_Keys.setObjectName("actionView_Keys")
        self.actionAdd_Keys = QtWidgets.QAction(MainWindow)
        self.actionAdd_Keys.setObjectName("actionAdd_Keys")

        MainWindow.setStyleSheet("background-image: url(Layer1/Media/backgound.png);")
        oImage = QImage("Layer1/Media/background.png")
        sImage = oImage.scaled(QSize(1021, 442))
        palette = QPalette()
        palette.setBrush(QPalette.Window, QBrush(sImage))
        MainWindow.setPalette(palette)
        MainWindow.setFixedWidth(1021)
        MainWindow.setFixedHeight(442)
        self.retranslateUi(MainWindow)
        self.stackedWidget.setCurrentIndex(6)
        QtCore.QMetaObject.connectSlotsByName(MainWindow)

        #code for buttons to change screen
        self.pushButton_3.clicked.connect(self.buttonAction3)
        self.pushButton_2.clicked.connect(self.buttonAction2)
        self.pushButton.clicked.connect(self.buttonAction)

self.buttonBox.button(QtWidgets.QDialogButtonBox.Cancel).clicked.connect(self.buttonAction5)

self.buttonBox_2.button(QtWidgets.QDialogButtonBox.Cancel).clicked.connect(self.buttonAction5
)

self.buttonBox.button(QtWidgets.QDialogButtonBox.Ok).clicked.connect(partial(self.buttonAction
7, CurrentWallet))
```

```
self.buttonBox_2.button(QtWidgets.QDialogButtonBox.Ok).clicked.connect(self.buttonAction4)
        self.pushButton_4.clicked.connect(partial(self.buttonAction6, CurrentWallet))
        self.pushButton_5.clicked.connect(self.buttonAction8)
        self.pushButton_6.clicked.connect(self.buttonAction9)
        self.pushButton_7.clicked.connect(partial(self.buttonAction10, CurrentWallet))
        self.pushButton_8.clicked.connect(partial(self.buttonAction12, MainWindow, CurrentWallet))
        self.pushButton_9.clicked.connect(self.buttonAction13)
        self.pushButton_15.clicked.connect(self.buttonAction11)

        #password fleids **
        self.lineEditPassword_2.setEchoMode(QtWidgets.QLineEdit.NoEcho)
        self.lineEditPassword_3.setEchoMode(QtWidgets.QLineEdit.NoEcho)
        self.lineEditConfirmPassword.setEchoMode(QtWidgets.QLineEdit.NoEcho)

    def retranslateUi(self, MainWindow):
        _translate = QtCore.QCoreApplication.translate
        MainWindow.setWindowTitle(_translate("MainWindow", "My Monero Wallet"))
        self.labelBalance.setText(_translate("MainWindow", "Balance :"))
        self.labelAmount.setText(_translate("MainWindow", "Amount:"))
        self.labelAmountInfo.setText(_translate("MainWindow", "This is the amount\n"
" you want to send"))
        self.labelMainAddress_2.setText(_translate("MainWindow", "Main address:"))
        self.pushButton_8.setText(_translate("MainWindow", "QRCode"))
        self.labelBalanceInfo.setText(_translate("MainWindow", "This is the value at\n"
"this address"))
        self.pushButton_9.setText(_translate("MainWindow", "Log Out"))
        self.labelReceivingAddress.setText(_translate("MainWindow", "Receiving\n"
" address:"))
        self.labelReceivingAddressInfo.setText(_translate("MainWindow", "This is the address\n"
" you want to pay"))
        self.pushButton_7.setText(_translate("MainWindow", "Refresh"))
        self.labelHeading.setText(_translate("MainWindow", "Welcome to my Monero Wallet\n"
"To send Monero please enter the amount and the receiving address below"))
        self.labelFeeInfo.setText(_translate("MainWindow", "This is the fee you will\n"
"pay for the transaction.\n"
"This goes from 1 to 4"))
        self.labelBalance_2.setText(_translate("MainWindow", "TextLabel"))
        self.labelMainAddress.setText(_translate("MainWindow", "TextLabel"))
        self.labelFee.setText(_translate("MainWindow", "Fee:"))
        self.pushButton_4.setText(_translate("MainWindow", "Send"))
        self.labeMainAddressInfo.setText(_translate("MainWindow", "This is your address"))
        self.label.setText(_translate("MainWindow", "Are you new or current user?"))
        self.pushButton_2.setText(_translate("MainWindow", "Current"))
        self.pushButton.setText(_translate("MainWindow", "New"))
        self.labelUsername.setText(_translate("MainWindow", "Username "))
        self.labelHeading_2.setText(_translate("MainWindow", "Welcome to my Monero wallet.
```

```
Please login."))
        self.passwordLabel.setText(_translate("MainWindow", "Password"))
        self.label_3.setText(_translate("MainWindow", "You have entered an incorrect login info."))
        self.pushButton_5.setText(_translate("MainWindow", "Back"))
        self.labelConfirmPassword.setText(_translate("MainWindow", "Confirm Password"))
        self.labelHeading_3.setText(_translate("MainWindow", "Welcome to my Monero wallet.
Please register with the system."))
        self.labelPassword.setText(_translate("MainWindow", "Password"))
        self.labelUsername_2.setText(_translate("MainWindow", "Username "))
        self.label_2.setText(_translate("MainWindow", "The two passwords you entered were not the
same."))
        self.pushButton_6.setText(_translate("MainWindow", "Back"))
        self.labelHeading_4.setText(_translate("MainWindow", "Welcome to my Monero wallet. \n"
"\n"
"Please wait for your copy of the blockchain to be updated."))
        self.pushButton_3.setText(_translate("MainWindow", "ForTestingUseOnly"))
        self.label_7.setText(_translate("MainWindow", "Please check that the address is a usable
address, you have the ammount to send\n"
" (it might take a few mins to unlock funds that have been sent to you) and the fee is between 1 and
4 inclusive"))
        self.pushButton_15.setText(_translate("MainWindow", "Back"))
        self.label_4.setText(_translate("MainWindow", "You have now been logged out. Please close
this window."))
        self.actionCreate_MainAddress.setText(_translate("MainWindow", "Create MainAddress"))
        self.actionCreate_SubAddress.setText(_translate("MainWindow", "Create SubAddress"))
        self.actionView_Keys.setText(_translate("MainWindow", "View Keys"))
        self.actionAdd_Keys.setText(_translate("MainWindow", "Add Keys"))

        #function for buttons to change screen and to update node and createWallet
    def buttonAction(self):
        self.stackedWidget.setCurrentIndex(4)
    def buttonAction2(self):
        self.stackedWidget.setCurrentIndex(2)
    def buttonAction3(self):
        self.stackedWidget.setCurrentIndex(1)
    def buttonAction4(self):
        username = self.lineEditUsername_4.text()
        password = self.lineEditPassword_3.text()
        conPassword = self.lineEditConfirmPassword.text()
        varForReturn = Database.createUsername(username, password, conPassword)
        if "PasswordRight" == varForReturn:
            print("PasswordRight")
            self.stackedWidget.setCurrentIndex(2)
```

```
        elif "ConPasswordWrong" == varForReturn:
            print("ConPasswordWrong")
            _translate = QtCore.QCoreApplication.translate
            self.label_2.setText(_translate("MainWindow", "Please check that the two passwords that you entered are the same in every way"))
            self.stackedWidget.setCurrentIndex(5)
        elif "PasswordNotStrongAnuf" == varForReturn:
            print("PasswordNotStrongAnuf")
            _translate = QtCore.QCoreApplication.translate
            self.label_2.setText(_translate("MainWindow", "Please enter a password that has a minimum of 15 chars, has at least one lowercase, one upercase and one special characters in it"))
            self.stackedWidget.setCurrentIndex(5)
        elif "UsernameTaken" == varForReturn:
            print("UsernameTaken")
            _translate = QtCore.QCoreApplication.translate
            self.label_2.setText(_translate("MainWindow", "The Username you entered is taken"))
            self.stackedWidget.setCurrentIndex(5)
        else:
            print("UsernameIsNone")

    def buttonAction5(self):
        self.stackedWidget.setCurrentIndex(1)
    def buttonAction6(self, CurrentWallet): ## send vaule
        try:
            print(CurrentWallet)
            addresStr = self.lineEditAddress.text()
            addressHex = decode(addresStr)
            addressBase58 = encode(addressHex)
            print(addressBase58)
            ammountStr = self.lineEditAmmount.text()
            ammountInt = as_monero(ammountStr)
            print(ammountInt)
            feeStr = self.lineFee.text()## add in an if to catch <1 or >4
            feeInt = int(feeStr)
            print(feeInt)
            CurrentWallet.transfer(addressBase58, ammountInt, feeInt)
        except:
            print("Please check that the address is a usable address, you have the ammount to send(it might take a few mins to unlock funds that have been sent to you) and the fee is between 1 and 4 inclusive")
            self.stackedWidget.setCurrentIndex(7)
    def buttonAction7(self, CurrentWallet):
        username = self.lineEditUsername_3.text()
        password = self.lineEditPassword_2.text()
        if True == Database.checkLogin(username, password):
            self.stackedWidget.setCurrentIndex(0)
```

```
            self.buttonAction10(CurrentWallet)
        else:
            _translate = QtCore.QCoreApplication.translate
            self.label_2.setText(_translate("MainWindow", "You have entered an incorrect login info"))
            self.stackedWidget.setCurrentIndex(3)
    def buttonAction8(self):
        self.stackedWidget.setCurrentIndex(2)
    def buttonAction9(self):
        self.stackedWidget.setCurrentIndex(4)
    def buttonAction10(self, CurrentWallet):
        _translate = QtCore.QCoreApplication.translate
        test = CurrentWallet.balance()
        test2 = CurrentWallet.address()
        self.labelBalance_2.setText(_translate("MainWindow", str(test)))
        self.labelMainAddress.setText(_translate("MainWindow", str(test2)))
        print("test")
    def buttonAction11(self):
        self.stackedWidget.setCurrentIndex(0)
    def buttonAction12(self, MainWindow, CurrentWallet):
        address = str(CurrentWallet.address())
        super(Ui_MainWindow, self)
        a = QrCode(MainWindow, address)
    def buttonAction13(self):
        self.stackedWidget.setCurrentIndex(8)
```

I also spent time to understand how a monero address is created. The below methods were going to be used to create monero just like the monero blockchain does, but due to time constraints I was not able to finish this portion of this project.

Other

```
##    def stringBase58ToString(self, address):
##        outputString = 0
##        for i in range(0,94):
##            x = address[i]
##            print(x)
##            outputString = outputString * 58 + self.stringBase58ToChar(x)
##            print(outputString)
##        return outputString

    #def intBase10ToInt58(self, address):
    #    outputString = 0
        #make this function turn int 10 into int 58
    #    return outputString

##    def Base58SwapOrder(self, oldOrder):
```

```
##        stringForReturn = ''
##        for i in range(0,49):
##            x = oldOrder[i]
##            stringForReturn += self.getNewChar(x)
##        return stringForReturn


##    def getNewChar(self, x):
##        return {abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ
##            'a': 'A',
##            'b': 'B',
##            'c': 'C',
##            'd': 'D',
##            'e': 'E',
##            'f': 'F',
##            'g': 'G',
##            'h': 'H',
##            'j': 'J',
##            'k': 'K',
##            'l': 'L',
##            'm': 'M',
##            'n': 'N',
##            'o': 'P',
##            'p': 'Q',
##            'q': 'R',
##            'r': 'S',
##            's': 'T',
##            't': 'U',
##            'u': 'V',
##            'v': 'W',
##            'w': 'X',
##            'x': 'Y',
##            'y': 'Z',
##            'z': 'a',
##            'a': 'b',
##            'B': 'c',
##            'C': 'd',
##            'D': 'e',
##            'E': 'f',
##            'F': 'g',
##            'G': 'h',
##            'H': 'i',
##            'J': 'j',
##            'K': 'k',
##            'L': 'm',
##            'M': 'n',
```

```
##          'N': 'o',
##          'P': 'p',
##          'Q': 'q',
##          'R': 'r',
##          'S': 's',
##          'T': 't',
##          'U': 'u',
##          'V': 'v',
##          'W': 'w',
##          'X': 'x',
##          'Y': 'y',
##          'Z': 'z',
##      }[x]
##
##  def int2base(self, x, base):
##      import string
##      digs = '123456789ABCDEFGHJKLMNPQRSTUVWXYZabcdefghijkmnopqrstuvwxyz'
##      if x < 0:
##          sign = -1
##      elif x == 0:
##          return digs[0]
##      else:
##          sign = 1
##
##      x *= sign
##      digits = []
##
##      while x:
##          digits.append(digs[int(x % base)])
##          x = int(x / base)
##
##      if sign < 0:
##          digits.append('-')
##
##      digits.reverse()
##
##      return ''.join(digits)
##  def stringBase58ToChar(self, x):
##      return {
##          '1': 1,
##          '2': 2,
##          '3': 3,
##          '4': 4,
##          '5': 5,
##          '6': 6,
##          '7': 7,
```

```
##         '8': 8,
##         '9': 9,
##         'A': 10,
##         'B': 11,
##         'C': 12,
##         'D': 13,
##         'E': 14,
##         'F': 15,
##         'G': 16,
##         'H': 17,
##         'J': 18,
##         'K': 19,
##         'L': 20,
##         'M': 21,
##         'N': 22,
##         'P': 23,
##         'Q': 24,
##         'R': 25,
##         'S': 26,
##         'T': 27,
##         'U': 28,
##         'V': 29,
##         'W': 30,
##         'X': 31,
##         'Y': 32,
##         'Z': 33,
##         'a': 34,
##         'b': 35,
##         'c': 36,
##         'd': 37,
##         'e': 38,
##         'f': 39,
##         'g': 40,
##         'h': 41,
##         'j': 42,
##         'k': 43,
##         'l': 44,
##         'm': 45,
##         'n': 46,
##         'o': 47,
##         'p': 48,
##         'q': 49,
##         'r': 50,
##         's': 51,
##         't': 52,
##         'u': 53,
```

```
##            'v': 54,
##            'w': 55,
##            'x': 56,
##            'y': 57,
##            'z': 58,
##        }[x]
##
##    def stringDecIntToBase58(self, num, vers):
##        alphabet =
"123456789ABCDEFGHJKLMNPQRSTUVWXYZabcdefghijkmnopqrstuvwxyz"
##        baseCount = 58
##        encoded = None
##        div = None
##        mod = None
##        while num >= baseCount:
##            div = num / baseCount
##            mod = (num - (baseCount * div))
##            print(mod)
##            #mod = int(mod)
##            encoded = alphabet[int(mod,58)] + encoded
##            num = div
##            #num = int(num)
##        encoded = vers + alphabet[int(num,58)] + encoded
##        return encoded
##
```