

Cyber 100 Section 001

Group Project

Group 5

Members:

Henley Ramoodit, Daniyal Chaudhry, & Christian Garza

Table of Contents

Program 1

The Password Generator project

Pages **3-4**

Program 2

The Timer Project

Pages **5-6**

Program 3

The Dice roll game

Pages **7-8**

Program 4

Rock, Paper, Scissors vs. Computer

Pages **9-10**

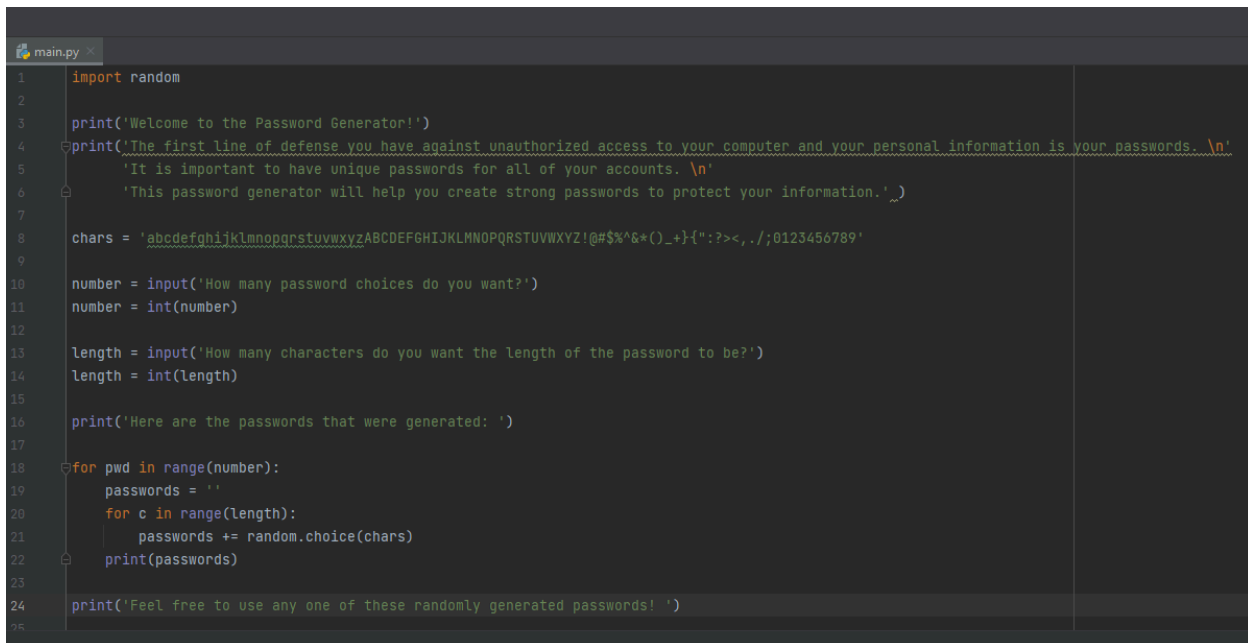
Program 5

ATM Bank

Pages **11-12**

Program 1: The Password Generator

The Password Generator was made to help those struggling with developing strong passwords for their accounts. In this day and age where passwords are used for every account you create and manage online, it is essential not to use the same password repetitively. As this can make it easy for hackers to access not just one of your accounts, but all of them by discovering your one password. The Password Generator will ask you how long you want your password to be, and how many passwords you are looking to create. This will allow you to make strong passwords for every account that you want to. Just don't forget to document these passwords somewhere like a password manager for future use.



```

1  import random
2
3  print('Welcome to the Password Generator!')
4  print('The first line of defense you have against unauthorized access to your computer and your personal information is your passwords. \n')
5  print('It is important to have unique passwords for all of your accounts. \n')
6  print('This password generator will help you create strong passwords to protect your information.')
7
8  chars = 'abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ!@#$%^&*()_+{}~:;<.,/;0123456789'
9
10 number = input('How many password choices do you want?')
11 number = int(number)
12
13 length = input('How many characters do you want the length of the password to be?')
14 length = int(length)
15
16 print('Here are the passwords that were generated: ')
17
18 for pwd in range(number):
19     passwords = ''
20     for c in range(length):
21         passwords += random.choice(chars)
22     print(passwords)
23
24 print('Feel free to use any one of these randomly generated passwords! ')

```

Figure 1.1: The figure above shows the source code for The Password Generator

The first step in creating The Password Generator was using the `import random` command. This imports the random module for you to use in your code, this module contains a variety of commands you can use for random number generation. The print statements that follow are just the messages the user sees when they first open The Password Generator, these messages can be anything a user wants. Next, is the `chars`

command, this is where our random variables for our passwords are gonna be selected from, so here I inputted the entire alphabet both lowercase and uppercase, as well as special characters and numbers. Using all of these characters will ensure that the passwords that are generated are always unique. The next two steps in the code ask the user how many password choices they want and how long they want the passwords to be. Using the input command we can prompt the questions to show up for them and they have to respond with an integer for both questions. So if they want two password choices they input 2, and if they want the length of the passwords to be 4 characters they would input 4. Next, we print a message to show them the passwords that were generated. The last couple of steps of this code is the most important to the actual password generator portion. The code shown in lines 18 through 21 uses a ranged for loop to take the input of the number of passwords and the length the user inputted earlier. In line 21 we use the `random.choice(chars)` command to select random values from the `chars` statement we set up earlier. This will create the passwords and populate them with random variables. We simply type `print(passwords)` and the code will respond with the passwords that were generated. If you take a look at Figure 1.2 below this is an example of The Password Generator in use, the user asks for 4 password choices with a length of 12 characters each. You can see the code executes and generates 4 options for the user to use if they please.

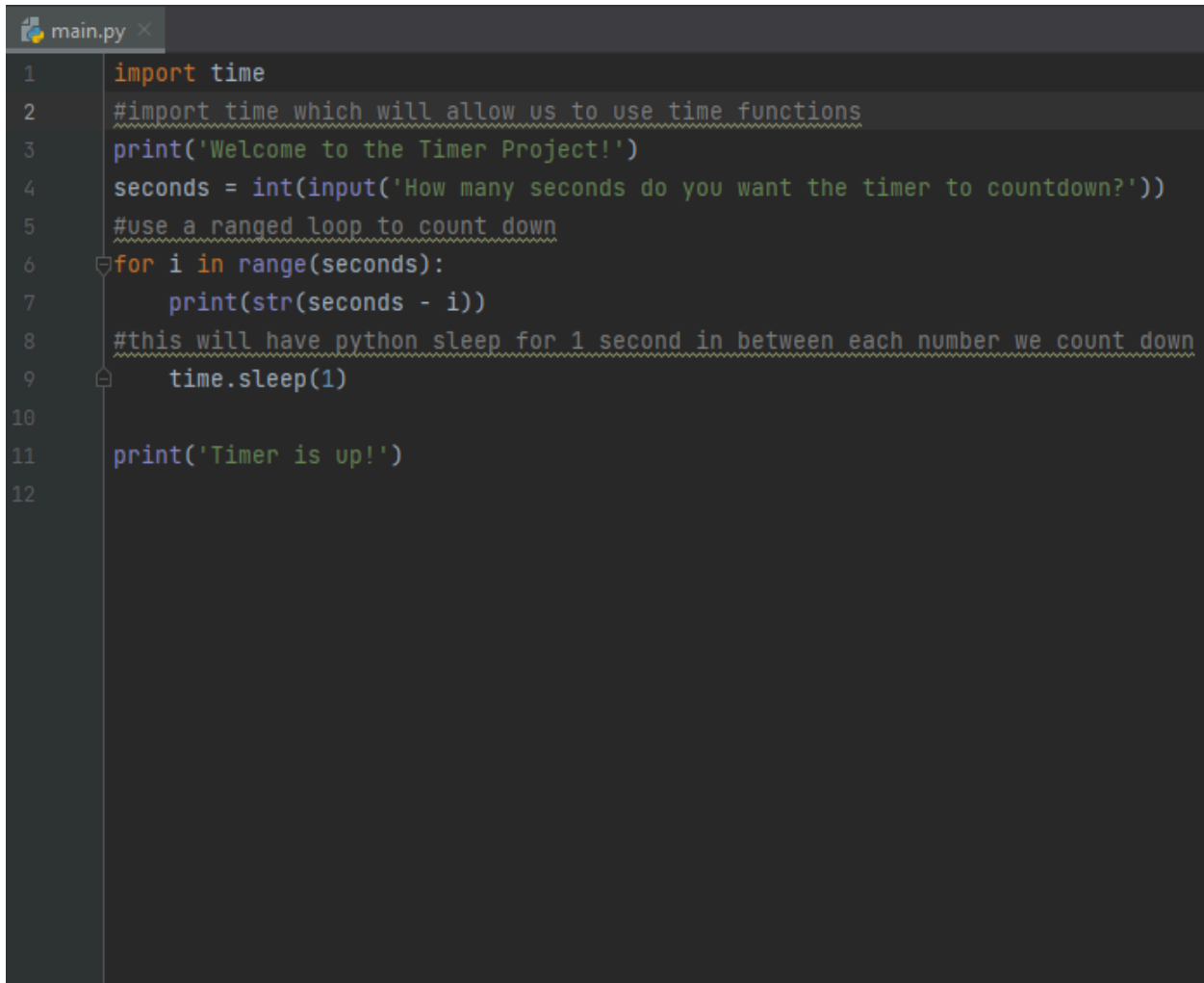
```
The first line of defense you have against unauthorized access to your computer and your personal information is your passwords.
It is important to have unique passwords for all of your accounts.
This password generator will help you create strong passwords to protect your information.
How many password choices do you want? 4
How many characters do you want the length of the password to be? 12
Here are the passwords that were generated:
{NG#s3bhb8A>
kioY&}0&,"Z7
*jrTg01P@eEU
i$1U35KxPK+)
Feel free to use any one of these randomly generated passwords!

Process finished with exit code 0
```

Figure 1.2: The screenshot shows the course code running where the user selects 4 password choices with the length of 12 characters each.

Program 2: The Timer Project

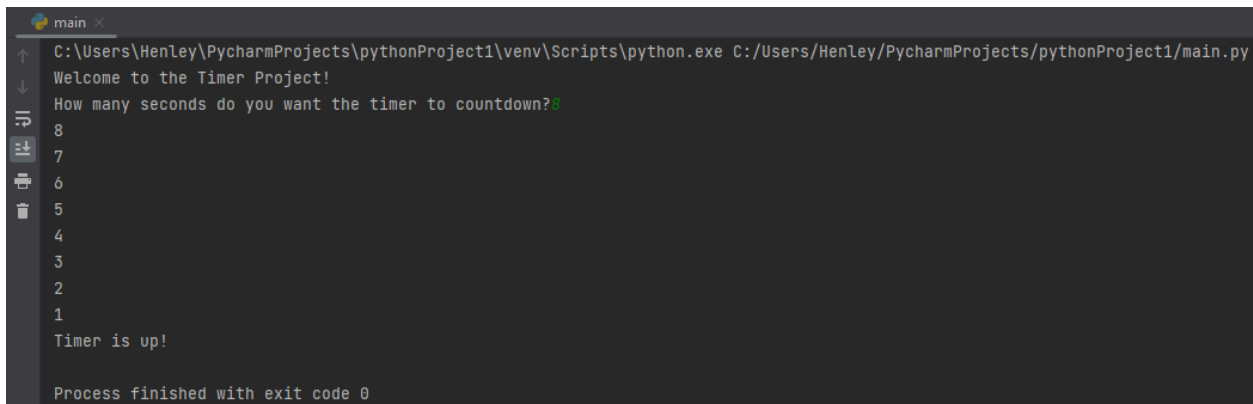
The Timer Project was created for those who like to pace themselves and stay on track. This simple python project will have the user create a timer that will count down from any amount of seconds down to 0 until the timer is complete. This is a simple project with few steps, but it is very satisfying to watch when the code is complete.

A screenshot of a code editor window titled 'main.py'. The code is written in Python and implements a countdown timer. It starts with an import statement for the 'time' module, followed by a welcome message. The user is prompted to enter the number of seconds for the timer. A 'for' loop is used to iterate from the entered seconds down to 0, printing the current count and pausing for one second between each iteration. Finally, a message is printed when the timer is complete.

```
1 import time
2 #import time which will allow us to use time functions
3 print('Welcome to the Timer Project!')
4 seconds = int(input('How many seconds do you want the timer to countdown?'))
5 #use a ranged loop to count down
6 for i in range(seconds):
7     print(str(seconds - i))
8     #this will have python sleep for 1 second in between each number we count down
9     time.sleep(1)
10
11 print('Timer is up!')
```

Figure 2.1: The figure shown above shows the source code for The Timer Project

To create a timer that will count down by seconds we first have to input the time module. This module allows you to use time in your code, and it will allow you to use time commands. After we input the time module, we will create a print statement introducing the user to our Timer, this can be anything you want it to be. Next, we ask the user for input by asking the user how many seconds they want the timer to countdown. The user will input their answer as an integer, so if they want it to countdown for 10 seconds they will input 10. After that, we will use a for ranged loop to count downwards by the number of seconds that the user chose. We will also need python to sleep for 1 second between each number we count, this is possible because we inputted the time module. So we use the `time.sleep(1)` command, now python will sleep for 1 second before it counts to the next second. Finally, to finish this code off we leave a print statement for the user to see when the timer runs out, this can be anything you want it to be so be creative. If you take a look at Figure 2.2 below the user chose 8 seconds to count down from. After it was done counting down, it gave the Timer is up! message to let the user know it was finished.



```
main
C:\Users\Henley\PycharmProjects\pythonProject1\venv\Scripts\python.exe C:/Users/Henley/PycharmProjects/pythonProject1/main.py
Welcome to the Timer Project!
How many seconds do you want the timer to countdown?
8
7
6
5
4
3
2
1
Timer is up!
Process finished with exit code 0
```

Figure 2.2: The screenshot above shows the code counting down from the input the user chose as 8 seconds.

Program 3: The dice roll game

This dice roll code is a game that can be used as just a pure game of chance, in a text based game, or even a game of Dungeons & Dragons. This Python code can have an infinite amount of players. When using the code, the dice can have as many faces as desired, and set as many rounds as needed.

```

1 import random
2
3
4 def main():
5     player1 = 0
6     player1wins = 0
7     player2 = 0
8     player2wins = 0
9     rounds = 1
10
11     while rounds != 8:
12         print("Round " + str(rounds))
13         player1 = dice_roll()
14         player2 = dice_roll()
15         print("Player 1 Roll: " + str(player1))
16         print("Player 2 Roll: " + str(player2))
17
18         if player1 == player2:
19             print("Draw!\n")
20         elif player1 > player2:
21             print("Player1 wins!\n")
22         else:
23             print("Player 2 wins!\n")
24
25         rounds = rounds + 1
26
27     if player1wins == player2wins:
28         print("Draw!")
29     elif player1wins > player2wins:
30         print("Player 1 Wins - Rounds Won: " + str(player1wins))
31     else:
32         print("Player 2 Wins - Rounds Won: " + str(player2wins))
33
34 def dice_roll():
35     diceRoll = random.randint(1, 6)
36     return diceRoll
37
38 main()

```

Figure 3.1: The screenshot shown, is the source of code for the dice roll game.

To start off the code, using the `import random` command is what allows the game to use a random assortment of numbers to be used together. Setting “`def main`” correlates with “`def dice_roll`” at line 34. This is the point of execution for the game. Using the command “`Random.randint`” will set a die with 6 sides that can be rolled at random. This die can be changed to any size dice, for example a 20 sided dice for D&D. Starting at line 5, this shows that there will be two players and they will both start at zero score and zero rounds won. At line 11 with the command “`While`”, this is used to repeat a loop. But instead of the rounds looping continuously, the number has been set to 8 rounds of play. This number can be changed to any amount. Moving below to the “`if`”, “`elif`”, and “`else`” commands. This sets the rules of “`if`” the players land the same numbers, it is a draw. “`Elif`” is if player 1 lands a higher number than player 2, player 1 wins. “`Else`” is when player 2 rolls a higher number, player 2 wins. The second set of “`if`”, “`elif`”, and “`else`” commands, is for overall games won. If both players win the same amount of rounds,

it's a draw and will print "Draw". If player 1 wins more rounds, then it will say "Player 1 wins". Then if player 2 wins more overall rounds, it will say "Player 2 wins". This will all appear at the bottom of each game.

```
Python 3.10.5 (v3.10.5:f377153967, Jun 6 2022, 12:36:10) [Clang 13.0.0 (clang-1300.0.29.30)] on darwin
Type "help", "copyright", "credits" or "license()" for more information.
>>>
=== RESTART: /Users/christiangularza/Desktop/Group 5 project/diceRollpython2.py ==
Round 1
Player 1 Roll: 5
Player 2 Roll: 3
Player1 wins!

Round 2
Player 1 Roll: 5
Player 2 Roll: 6
Player 2 wins!

Round 3
Player 1 Roll: 4
Player 2 Roll: 4
Draw!

Round 4
Player 1 Roll: 4
Player 2 Roll: 4
Draw!

Round 5
Player 1 Roll: 3
Player 2 Roll: 2
Player1 wins!

Round 6
Player 1 Roll: 2
Player 2 Roll: 3
Player 2 wins!

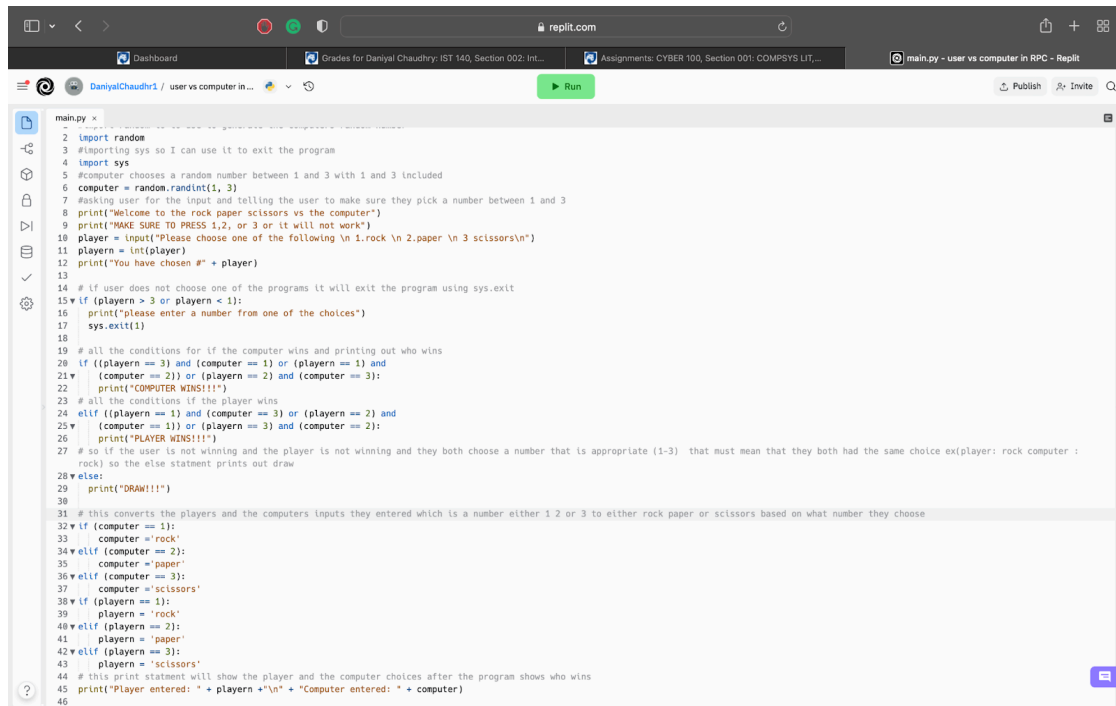
Round 7
Player 1 Roll: 4
Player 2 Roll: 1
Player1 wins!

Draw!
>>>
```

Figure 2.2: Screenshot of code being run with 7 rounds of play.

In the figure above, There were 7 rounds played. We can see that overall the game was a tie.

Program 4: Rock Paper Scissors VS Computer

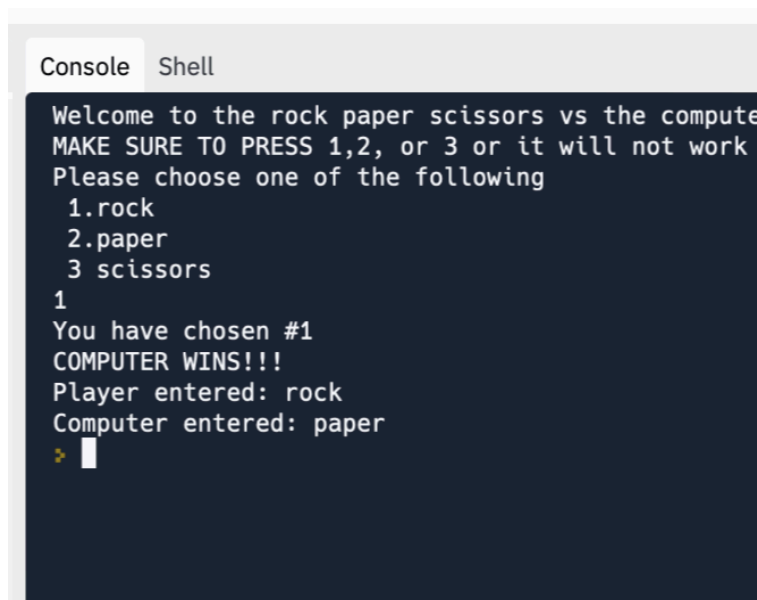


```

1 import random
2 #importing sys so I can use it to exit the program
3 import sys
4 #computer chooses a random number between 1 and 3 with 1 and 3 included
5 computer = random.randint(1, 3)
6 #asking user for the input and telling the user to make sure they pick a number between 1 and 3
7 print("Welcome to the rock paper scissors vs the computer")
8 print("MAKE SURE TO PRESS 1,2, or 3 or it will not work")
9 player = input("Please choose one of the following \n 1.rock \n 2.paper \n 3.scissors\n")
10 player = int(player)
11 print("You have chosen #" + player)
12
13
14 # if user does not choose one of the programs (it will exit the program using sys.exit)
15 if (player > 3 or player < 1):
16     print("please enter a number from one of the choices")
17     sys.exit(1)
18
19 # all the conditions for if the computer wins and printing out who wins
20 if ((player == 3) and (computer == 1) or (player == 1) and
21     (computer == 2)) or (player == 2) and (computer == 3):
22     print("COMPUTER WINS!!!")
23 # all the conditions if the player wins
24 elif ((player == 1) and (computer == 3) or (player == 2) and
25     (computer == 1)) or (player == 3) and (computer == 2):
26     print("PLAYER WINS!!!")
27 # so if the user is not winning and the player is not winning and they both choose a number that is appropriate (1-3) that must mean that they both had the same choice ex(player: rock computer :
28 # rock) so the else statement prints out draw
29 else:
30     print("DRAW!!!")
31 # this converts the players and the computers inputs they entered which is a number either 1 2 or 3 to either rock paper or scissors based on what number they choose
32 if (computer == 1):
33     computer = 'rock'
34 elif (computer == 2):
35     computer = 'paper'
36 elif (computer == 3):
37     computer = 'scissors'
38 if (player == 1):
39     player = 'rock'
40 elif (player == 2):
41     player = 'paper'
42 elif (player == 3):
43     player = 'scissors'
44 # this print statement will show the player and the computer choices after the program shows who wins
45 print("Player entered: " + player + "\n" + "Computer entered: " + computer)
46

```

Figure 4.1: code for the rock paper scissors game



```

Welcome to the rock paper scissors vs the compute
MAKE SURE TO PRESS 1,2, or 3 or it will not work
Please choose one of the following
1.rock
2.paper
3.scissors
1
You have chosen #1
COMPUTER WINS!!!
Player entered: rock
Computer entered: paper
>

```

Figure 4.2: Output if the user enters a number that is between 1 and 3:

User choose rock so they press 1 and the program will output the following:

You have chosen #1

COMPUTER WINS!!!

Player entered: rock

The computer entered: paper

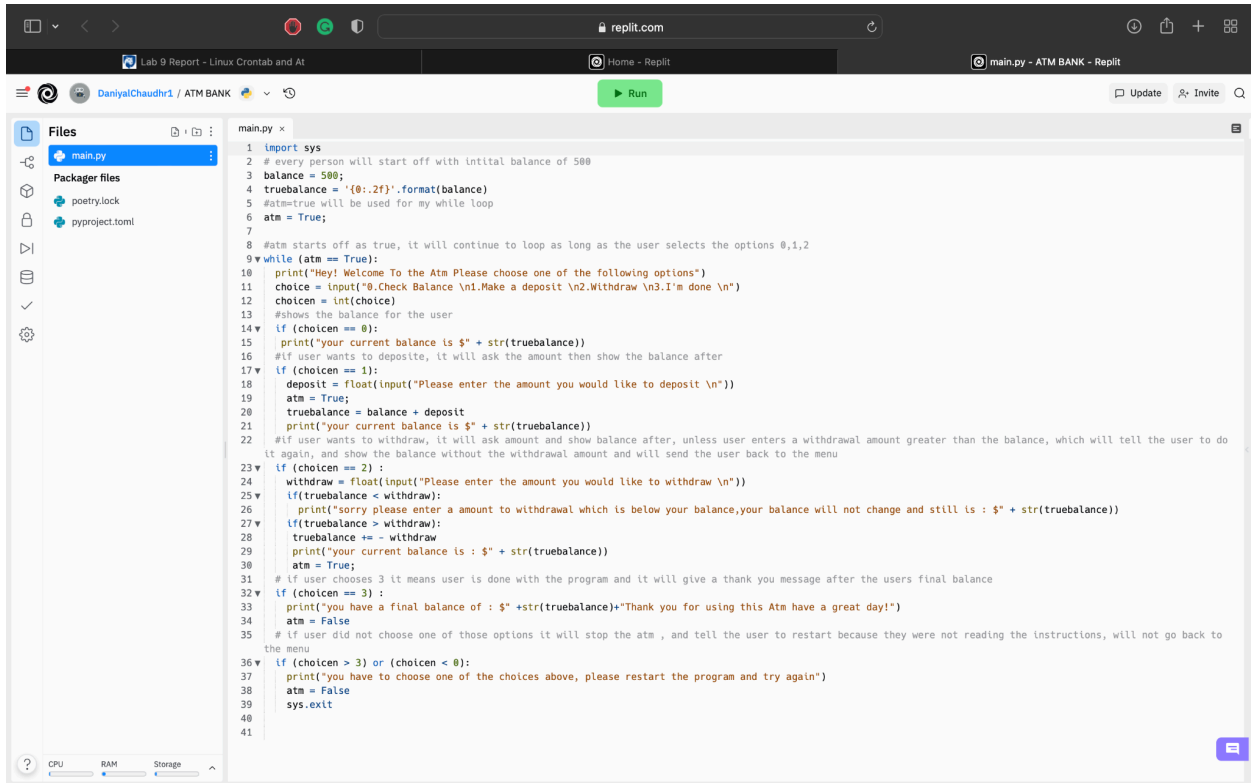
```
Welcome to the rock paper scissors vs the computer
MAKE SURE TO PRESS 1,2, or 3 or it will not work
Please choose one of the following
  1.rock
  2.paper
  3.scissors
4
You have chosen #4
please enter a number from one of the choices
repl process died unexpectedly: exit status 1
> |
```

Figure 4.3: Output if the user enters a number that is not between 1 and 3 : **lets say the user enters 4 program will output the following:**

```
You have chosen #4
please enter a number from one of the choices
repl process died unexpectedly: exit status 1
```

The code above in python runs a rock paper scissors game vs the computer game. The user chooses one of the following options(Press 1: Rock, Press 2: Paper, Press 3: Scissors.). If the user decides not to choose one of the options they will get the output of an error message, telling them to enter a number from one of the choices. The computer will also choose a number between 1 and 3, randomly utilizing `import random` and using the `randint ()` method. Afterward, the program has received the user's input and generates a random number for the computer, it will decide who wins using the `if` statements, and will declare the winner. Then the program will convert the number that the user and computer choose and change it to rock, paper, or scissors. Finally, it will print out what both choose, so if both choose #1 then it will print out player-entered rock and computer-entered rock.

Program 5: ATM BANK



```

1 import sys
2 # every person will start off with initial balance of 500
3 balance = 500;
4 truebalance = '{0:.2f}'.format(balance)
5 #atmtrue will be used for my while loop
6 atm = True;
7
8 #atm starts off as true, it will continue to loop as long as the user selects the options 0,1,2
9 while (atm == True):
10     print("Hey! Welcome To the Atm Please choose one of the following options")
11     choice = input("0.Check Balance \n1.Make a deposit \n2.Withdraw \n3.I'm done \n")
12     choicen = int(choice)
13     #shows the balance for the user
14     if (choicen == 0):
15         print("your current balance is $" + str(truebalance))
16         #if user wants to deposit, it will ask the amount then show the balance after
17         if (choicen == 1):
18             deposit = float(input("Please enter the amount you would like to deposit \n"))
19             atm = True;
20             truebalance = balance + deposit
21             print("your current balance is $" + str(truebalance))
22         #if user wants to withdraw, it will ask amount and show balance after, unless user enters a withdrawal amount greater than the balance, which will tell the user to do
23         #it again, and show the balance without the withdrawal amount and will send the user back to the menu
24         if (choicen == 2):
25             withdraw = float(input("Please enter the amount you would like to withdraw \n"))
26             if(truebalance < withdraw):
27                 print("sorry please enter a amount to withdrawal which is below your balance,your balance will not change and still is : $" + str(truebalance))
28             if(truebalance > withdraw):
29                 truebalance -= withdraw
30                 print("your current balance is : $" + str(truebalance))
31             atm = True;
32         # if user chooses 3 it means user is done with the program and it will give a thank you message after the users final balance
33         if (choicen == 3):
34             print("you have a final balance of : $" +str(truebalance)+"Thank you for using this Atm have a great day!")
35             atm = False
36             # if user did not choose one of those options it will stop the atm , and tell the user to restart because they were not reading the instructions, will not go back to
37             # the menu
38             if (choicen > 3) or (choicen < 0):
39                 print("you have to choose one of the choices above, please restart the program and try again")
40                 atm = False
41                 sys.exit

```

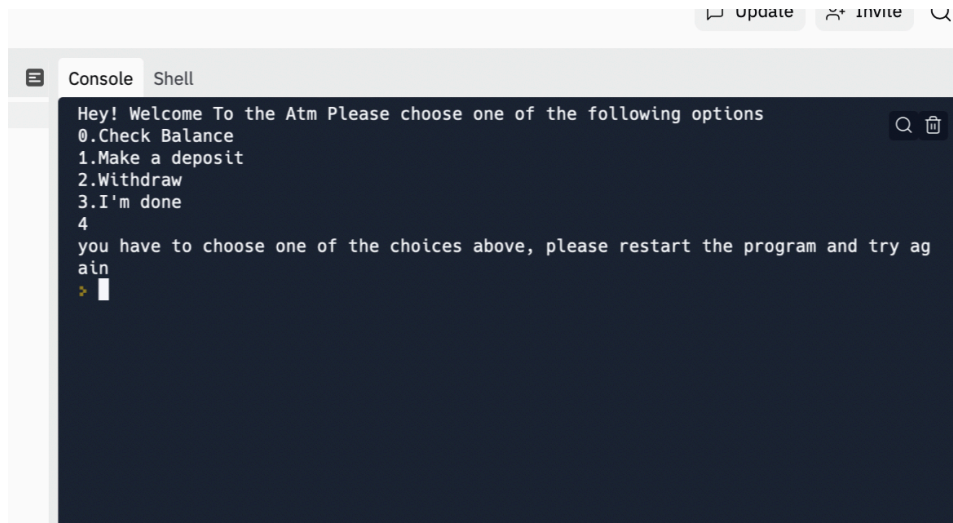
Figure 5.1: code for the ATM BANK program

```

Hey! Welcome To the Atm Please choose one of the following options
0.Check Balance
1.Make a deposit
2.Withdraw
3.I'm done
0
your current balance is $500.00
Hey! Welcome To the Atm Please choose one of the following options
0.Check Balance
1.Make a deposit
2.Withdraw
3.I'm done
1
Please enter the amount you would like to deposit
200.99
your current balance is $700.99
Hey! Welcome To the Atm Please choose one of the following options
0.Check Balance
1.Make a deposit
2.Withdraw
3.I'm done
2
Please enter the amount you would like to withdraw
701
sorry please enter a amount to withdrawal which is below your balance,your balance will not change and still is : $700.99
Hey! Welcome To the Atm Please choose one of the following options
0.Check Balance
1.Make a deposit
2.Withdraw
3.I'm done
2
Please enter the amount you would like to withdraw
100.97
your current balance is : $600.02
Hey! Welcome To the Atm Please choose one of the following options
0.Check Balance
1.Make a deposit
2.Withdraw
3.I'm done
3
you have a final balance of : $600.02Thank you for using this Atm have a great day!
>

```

Figure 5.2: Output, if the user does this series of steps, check their balance, make a deposit, make a withdrawal > true balance makes a true balance > withdrawal, and finally, they are done. It will keep the user updated about their account balance as shown in the screen shot above.



```

Hey! Welcome To the Atm Please choose one of the following options
0.Check Balance
1.Make a deposit
2.Withdraw
3.I'm done
4
you have to choose one of the choices above, please restart the program and try again
>

```

Figure 5.3: If the user does not enter one of the input choices given, it will stop the loop and end the code telling the user to choose one of the choices given and try again.

The two screenshots shown above relate to the ATM BANK project, to let the user continuously withdraw, check balance, or deposit money like a real atm. Everyone will start with 500 dollars. The while loop was the reason this was all possible. Before the while loop began, I set `atm == true`. The loop only runs when `atm` is true, if the user presses three or an invalid number it will say `atm` equals false. Hence, they exit the program with a message. Even if the loop starts as always `atm` equals true, it will not repeat itself if the user enters one of these choices (3 or an invalid number). Moreover, this program has `balance` and `true balance`, because `true balance` is utilized to format the decimal in two decimal places which I've set equal to `balance` at the beginning of the program. The if statements are used to display, add, and subtract based on the number the user entered followed by if it asks for another input from the user.