**HAUB SCHOOL OF BUSINESS**

**SAINT JOSEPH’S UNIVERSITY**

**DSS 615: Python Programming**

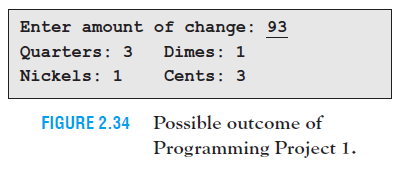
**Instructor: Michael Ghen**

**Assignment 3**

By:

Vinayak Suresh Tayshetye (10673718)

1. **Make Change** Write a program to make change for an amount of money from0 through 99 cents input by the user. The output of the program should show the number of coins from each denomination used to make the change. See Fig. 2.34.



[Code]

amt = 101

# checking if the amount is between 0-99;

while not int(amt) in range(0,99):

amt = int(input("Enter amount of change (0 - 99) : "))

# Formula to calculate each coin

quarters = ( amt // 25)

amt = amt % 25

dimes = ( amt // 10)

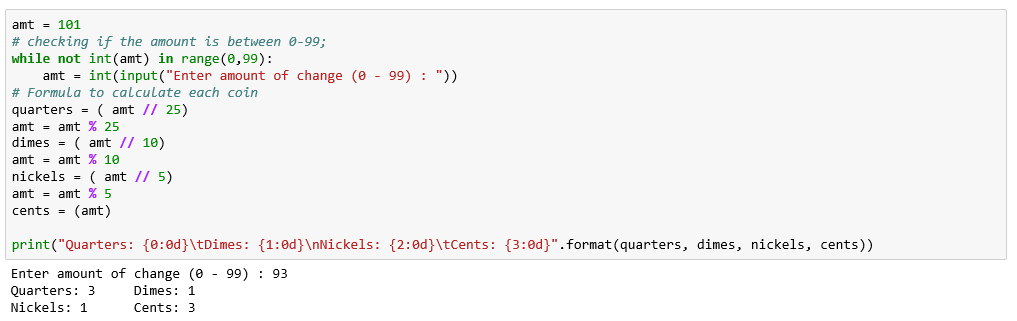
amt = amt % 10

nickels = ( amt // 5)

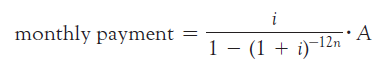
amt = amt % 5

cents = (amt)

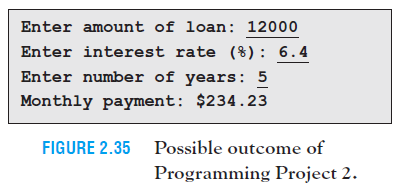
print("Quarters: {0:0d}\tDimes: {1:0d}\nNickels: {2:0d}\tCents: {3:0d}".format(quarters, dimes, nickels, cents))



2. **Car Loan** If *A* dollars is borrowed at *r*% interest compounded monthly to purchase a car with monthly payments for *n* years, then the monthly payment is given by the formula



where *i* = *r/*1200. Write a program that calculates the monthly payment after the user gives the amount of the loan, the interest rate, and the number of years. See Fig. 2.35.



[Code]

##2

amount = eval(input("Enter amount of loan: "))

intRate = eval(input("Enter interest rate (%) : "))

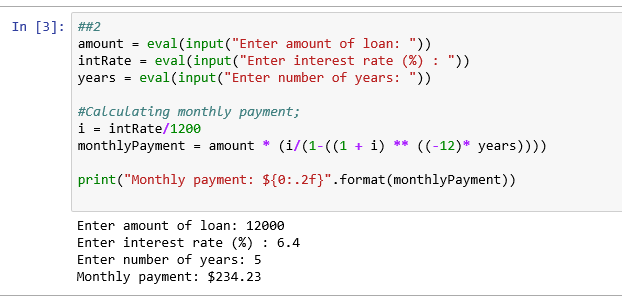
years = eval(input("Enter number of years: "))

#Calculating monthly payment;

i = intRate/1200

monthlyPayment = amount \* (i/(1-((1 + i) \*\* ((-12)\* years))))

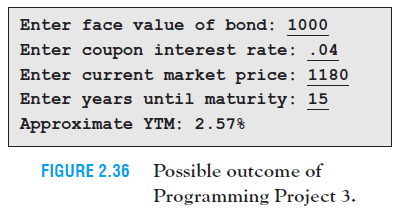
print("Monthly payment: ${0:.2f}".format(monthlyPayment))



3.  **Bond Yield** One measure of a bond’s performance is its *Yield To Maturity* (YTM). YTM values for government bonds are complex to calculate and are published in tables. However, they can be approximated with the simple formula YTM = (*intr* + *a*)/*b* , where *intr* is the interest earned per year,

*a* = (face value - current market price)/years until maturity , and

*b* = (face value + current market price)/2 . For instance, suppose a bond has a face value of $1,000, a coupon interest rate of 4%, matures in 15 years, and currently sells for $1,180. Then *intr* = .04 \* 1,000 = 40, *a* = (1000 – 1180)/15 = -12, *b* = (1000 + 1180)/2 = 1090, and *YTM* = (40 – 12)/1090 = 2.57%. ***Note:*** The *face value* of the bond is the amount it will be redeemed for when it matures, and the *coupon interest rate* is the interest rate stated on the bond. If a bond is purchased when it is first issued, then the YTM is the same as the coupon interest rate. Write a program that requests the face value, coupon interest rate, current market price, and years until maturity for a bond, and then calculates the bond’s YTM. See Fig. 2.36.



[Code]

##3

faceValueBond = eval( input("Enter face value of bond: "))

intRate = eval( input("Enter coupon interest rate: "))

currentMarketPrice = eval( input("Enter current market price: "))

maturity = eval( input("Enter years until maturity: "))

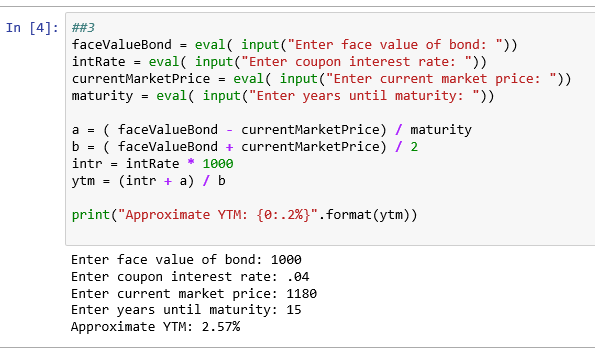
a = ( faceValueBond - currentMarketPrice) / maturity

b = ( faceValueBond + currentMarketPrice) / 2

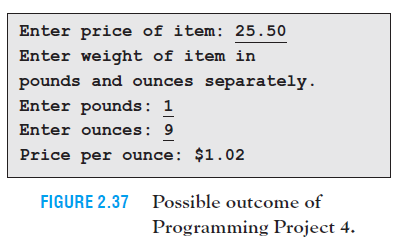
intr = intRate \* 1000

ytm = (intr + a) / b

print("Approximate YTM: {0:.2%}".format(ytm))



4. **Unit Price** Write a program that requests the price and weight of an item in pounds and ounces, and then determines the price per ounce. See Fig. 2.37.



[Code]

##4

miles = eval(input("Enter price item: "))

print("Enter weight of item in pounds and ounces separately.")

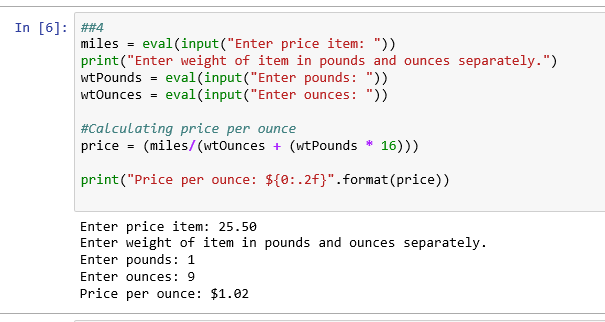
wtPounds = eval(input("Enter pounds: "))

wtOunces = eval(input("Enter ounces: "))

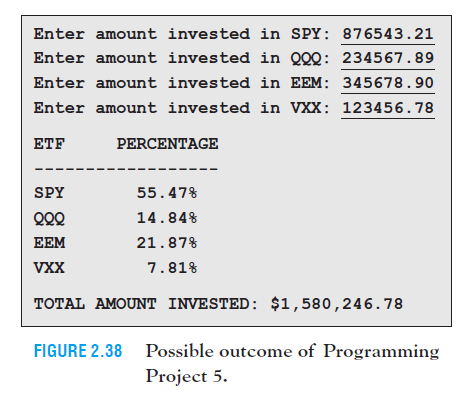
#Calculating price per ounce

price = (miles/(wtOunces + (wtPounds \* 16)))

print("Price per ounce: ${0:.2f}".format(price))



5. **Stock Portfolio** An investor’s stock portfolio consists of four Exchange Traded Funds (SPY, QQQ, EEM, and VXX). Write a program that requests the amount invested in each fund as input and then displays the total amount invested and each fund’s percentage of the total amount invested. See Fig. 2.38.



[Code]

##5

spy = eval(input("Enter amount invested in SPY: "))

qqq = eval(input("Enter amount invested in QQQ: "))

eem = eval(input("Enter amount invested in EEM: "))

vxx = eval(input("Enter amount invested in VXX: "))

#Calculating total investment;

totalInvestment = spy + qqq + eem + vxx

print("\n{0:10s}{1:10s}".format("ETF", "PERCENTAGE"))

print("-" \* 20)

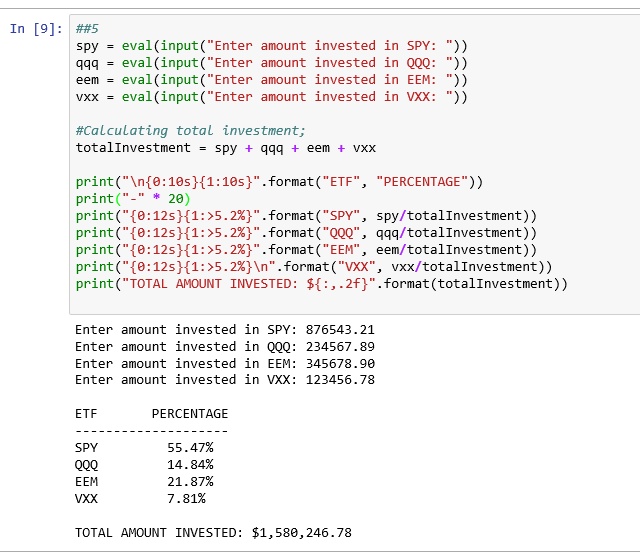
print("{0:12s}{1:>5.2%}".format("SPY", spy/totalInvestment))

print("{0:12s}{1:>5.2%}".format("QQQ", qqq/totalInvestment))

print("{0:12s}{1:>5.2%}".format("EEM", eem/totalInvestment))

print("{0:12s}{1:>5.2%}\n".format("VXX", vxx/totalInvestment))

print("TOTAL AMOUNT INVESTED: ${:,.2f}".format(totalInvestment))

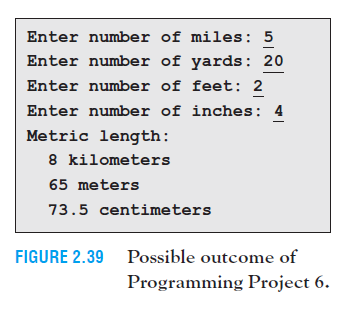


6. **Length Conversion** Write a program to convert a U.S. Customary System length in miles, yards, feet, and inches to a Metric System length in kilometers, meters, and centimeters. A sample run is shown in Fig. 2.39. After the numbers of miles, yards, feet, and inches are entered, the length should be converted entirely to inches and then divided by 39.37 to obtain the value in meters. The **int** function should be used to break the total number of meters into a whole number of kilometers and meters. The number of centimeters should be displayed to one decimal place. The needed formulas are as follows:

total inches = 63,360 \* miles + 36 \* yards + 12 \* feet + inches

total meters = total inches/39.37

kilometers = int(meters/1000)



[Code]

##6

miles = eval(input("Enter number of miles: "))

yards = eval(input("Enter number of yards: "))

feet = eval(input("Enter number of feet: "))

inches = eval(input("Enter number of inches: "))

# Calculating length conversion;

totalinches = (63360 \* miles) + (36 \* yards) + (12 \* feet) + inches

totalmeters = totalinches/39.37

kilometers = int (totalmeters/1000)

# Calculating meters;

meters = int(totalmeters - (kilometers \* 1000))

# Calculating meters;

centimeters = float(totalmeters - (meters + (kilometers \* 1000))) \* 100

# Writing a string statement to reduce print statements;

conversion = "Metric length: \n {0:} kilometers \n {1:} meters \n {2:.1f} centimeters"

print(conversion.format(kilometers, meters, centimeters)).

