*Problem 1*

user\_name = input("Please enter your full name: ")

birth\_year = int(input("To find your generation, please enter your birth year: "))

generation = ""

# determines an invalid birth year

if birth\_year < 1901 or birth\_year > 2023:

print("Error: there are no recorded people in your birth year")

# determines the generation

else:

if 1901 <= birth\_year <= 1924:

generation = "from the Greatest Generation"

elif 1925 <= birth\_year <= 1945:

generation = "from the Silent Generation"

elif 1946 <= birth\_year <= 1964:

generation = "a Baby Boomer"

elif 1965 <= birth\_year <= 1980:

generation = "from Generation X"

elif 1981 <= birth\_year <= 1996:

generation = "a millenial"

elif 1997 <= birth\_year <= 2012:

generation = "from Generation Z"

else:

generation = "from Generation Alpha"

# prints the output message

print(f"Hello {user\_name.title()}, you are {generation}")

*Problem 2*

num = input("Enter a number, and I will check if it is an armstrong number: " )

# number of digits in num string = exponent

exp = len(num)

# Initialize the variable to store the "Armstrong number"

arm\_output = 0

for digit in num:

# Raise each digit to the power of the number of digits, storing additions in arm\_output

arm\_output += pow(int(digit), exp)

# Check if the input number is equal to the obtained Armstrong number

if num == str(arm\_output):

print(f"{num} is an Armstrong number !!")

else:

print(f"{num} is not an Armstrong number ")

*Problem 3*

# number of guesses available

N = 3

print(f"Welcome to Pointless! A game where the lowest scorers are the biggest winners! You will have {N} guesses to get the correct answer.")

# declaring answer variables and their respective points

answers = ["norway", "newzealand", "netherlands", "nigeria", "north korea", "niger", "nicaragua", "namibia", "nepal", "nauru"]

points = [48, 45, 41, 29, 13, 11, 9, 2, 1, 0]

while N >= 1: # exits loop when number of guesses are less than 1

user\_answer = input("Name a country beginning with N: ")

i = 0

for answer in answers:

if user\_answer.lower() == answer:

score = points[i]

print(f"Correct answer, +{score} points!")

if score == 0:

print("Well done!! You have guessed a pointless answer")

break # exits loop if the answer is correct

else:

i = i + 1

continue # moves to the next iteration of the loop

if user\_answer.lower() == answer:

break # exits while loop if the answer is correct

N = N - 1 # for each incorrect answer, num of guesses are reduced by one

# messages for incorrect outcomes

if N == 1:

print(f"Incorrect! You have {N} guess left")

else:

print(f"Incorrect! You have {N} guesses left")

if N == 0: # when there are no guesses left

print("Better luck next time!")

*Problem 4*

x= int(input("x= "))

y= int(input("y= "))

hcf\_list=[] # Creating an empty list to store the remainders

hcf= 0

# makes sure that x always has the largest value, and x, y are not equal.

if y > x:

x, y = y, x

elif y == x:

print("Error: the integers must not be equal")

a= x % y

b= x - a

hcf\_list.append(a) # Adding the first remainder to the list of remainders

if a== 0:

hcf= y

# Loop to calculate the HCF

while a!= 0:

a= b % a

b= b - a

hcf\_list.append(a)

hcf= hcf\_list[-2] # The second last item in the list is the HCF

lcm = int(abs(x\*y)/hcf) # Calculating the LCM

print(f"HCF= {hcf}")

print(f"LCM= {lcm}")

*Problem 5*

N = int(input("Enter the amount of numbers in the Stern-Brocot sequence you want to generate: "))

# Starting point of the sequence

stern\_brocot = [1,1]

# Variable to keep track of the index in the sequence

i= 0

# Generating the sequence until it has N numbers

while len(stern\_brocot) < N:

a = stern\_brocot[i]

b = stern\_brocot[i + 1]

c = a + b # Calculate the sum of the current and next number

stern\_brocot.append(c)

stern\_brocot.append(b)

i +=1

# Trimming the sequence to have only the desired amount of numbers

stern\_brocot= stern\_brocot[:N]

# Copying the sequence for further processing

sb\_copy = stern\_brocot.copy()

# Removing the last number if the sequence length is odd

if len(sb\_copy)%2 != 0:

sb\_copy.pop(-1)

r\_list = []

i= 0

# Converting the numbers into fractions

while i < N-1:

a= sb\_copy[i]

b= sb\_copy[i + 1]

c= f"{a}/{b}" # Create a string representation of the fraction

r\_list.append(c) # Add the fraction to the list

i +=1

if len(stern\_brocot) == 1:

print(f"The sequence is: {stern\_brocot}")

print("There are no rational numbers for this sequence.")

else:

print(f"The sequence is: {stern\_brocot}")

print(f"The rational numbers are: {r\_list}")

*Problem 6*

# Ask the user to input the coefficients of the quintic equation

print("input the coefficients of your quintic equation: ")

a= int(input("a= "))

b= int(input("b= "))

c= int(input("c= "))

d= int(input("d= "))

e= int(input("e= "))

f= int(input("f= "))

# Ask the user to input an interval

print("input an interval [l,h] where f(l) and f(h) yield opposing signs: ")

l= int(input("l= "))

h= int(input("h= "))

soln= False

NMAX = 5000

TOL = 0.000001

# Loop through a certain number of iterations, specified by NMAX

for N in range(1, NMAX, 1):

# Calculate the midpoint of the interval

x= (l+h)/2

# Calculate the value of the equation at the midpoint

fx= a\*(pow(x,5)) + b\*(pow(x,4)) + c\*(pow(x,3)) + d\*(pow(x,2)) + e\*x + f

# Calculate the value of the equation at the lower end of the interval

fl= a\*(pow(l,5)) + b\*(pow(l,4)) + c\*(pow(l,3)) + d\*(pow(l,2)) + e\*l + f

# Check if the value of the equation is zero or if the interval is very small

if fx == 0 or (h-l)/2 < TOL:

soln = True

break

# Check if the value of the equation at the midpoint and lower end have the same sign

if fx\*fl > 0:

# If they have the same sign, update the lower end of the interval to the midpoint

l= x

else:

# Otherwise, update the upper end of the interval to the midpoint

h= x

if soln:

# If a solution is found, print the value of the solution

print(f"x= {x}")

else:

print("Method failed.")

*Problem 7*

hidden\_list\_a = [0,0,0,0,0,0,0,0]

hidden\_list\_b = [0,0,0,0,0,0,0,0]

# setting the values for list a and b which the user will slowly reveal

list\_a = [6,8,2,1,9,3,5,7]

list\_b = [5,3,1,9,7,6,8,2]

print(hidden\_list\_a)

print(hidden\_list\_b)

while hidden\_list\_a.count(0) > 0 and hidden\_list\_b.count(0) > 0:

choice1 = int(input("Which position do you want to check in the first row?: ")) - 1

chosen\_num1 = list\_a[choice1]

a= hidden\_list\_a.copy()

a.pop(choice1)

a.insert(choice1, chosen\_num1)

print(a)

print(hidden\_list\_b)

choice2 = int(input("Now guess where that number is in the second row: ")) -1

chosen\_num2 = list\_b[choice2]

b= hidden\_list\_b.copy()

b.pop(choice2)

b.insert(choice2, chosen\_num2)

print(a)

print(b)

if chosen\_num1 == chosen\_num2:

hidden\_list\_a = a

hidden\_list\_b = b

else:

print("Try again!")

print(hidden\_list\_a)

print(hidden\_list\_b)

continue

print("Congratulations! You have completed the game ")