**CSE1100 – Programming Concepts with Python**

**Programming Concepts Lab Report**

**Fall 2021**

*By*

Cael Shoop

*Computer Science, B.S.*

*cshoop2018@my.fit.edu*

September 22, 2021

Homework #1

Teaching Instructor:

Stefan Joe-Yen, Ph.D.

# 1. Problem Statement

**Problem 1:**

The program must simulate a soda dispenser machine. It will execute in a loop while waiting for users to enter coins. The only acceptable values are $0.05, $0.10, and $0.25. Other values will be rejected by the machine. The cost of a soda is $1, so the program must continue accepting coins until the value reaches at least $1. The system repeats when the soda is dispensed, returning any change to the user.

**Problem 2:**

The program must draw shapes on the terminal. It will execute in a loop to display the menu to the user. Users can select four different shapes: a filled/empty rectangle and a filled/empty triangle. The program must request the width and height if the user selects a rectangle, and it will use these to draw the rectangle. If triangles are selected, the program will request a height from the user and create a right triangle. Both rectangles and triangles have the option of being filled or empty.

**Problem 3:**

The program must read in a five-digit integer and determine whether or not it is a palindrome. This must be done using the division and modulus operators.

**Problem 4:**

The program must display a checkerboard pattern of asterisks dynamically. It may only use 3 output statements: “print(“\* “),” “print(“ “),” and “print(“\n”).”

**Problem 5:**

The program must take input as a number *n* and return the maximum number of possible point-to-point connections *Kn* among these points.

# 2. Requirements

**Problem 1**

1. The software shall loop to receive the user’s coin value input.
   1. The program will only accept $0.05, $0.10, and $0.25.
   2. The program will wait for coin input until the value reaches $1.00.
2. The software will dispense the soda.
3. The software will return any change above the $1.00.

**Problem 2**

1. The software shall display a menu to receive user input.
   1. The user menu shall contain 5 options.
      1. Filled rectangle.
      2. Empty rectangle.
      3. Filled triangle.
      4. Empty triangle.
      5. Quit.
2. The software shall require a height and width to be input for rectangles.
3. The software shall require a height to be input for triangles.

**Problem 3**

1. The software shall receive a 5 digit integer.
   1. The program will not accept non 5 digit integers.
2. The software will determine whether or not a 5 digit integer is a palindrome.
3. The software must utilize division and modulus to accomplish this determination.

**Problem 4**

1. The software shall dynamically display a checkerboard pattern of asterisks.
2. The program must use a nested for loop.
3. The software shall only have 3 output statements.

3.1 print(“\* “)

3.2 print(“ “)

3.3 print(“\n”)

**Problem 5**

1. The software shall take a value *n* from the user.
2. The software shall use the choose formula to calculate the maximum possible number of point-to-point connections for that value.
3. The program must output that info for the user to view.

# 3. Software Construction (Annotated Python Code)

Problem 1:

'''Written by Cael Shoop. Soda Dispenser program.'''

def soda\_check(soda):

# Checks if the input value exists, also allowing lowercase

if soda != '1A' and soda != '1B' and soda != '2A' and soda != '2B' and soda != '2C' and soda != '3A' and soda != '4A' and soda != '5A' and soda != '6A' and soda != '7A' and soda != '1a' and soda != '1b' and soda != '2a' and soda != '2b' and soda != '2c' and soda != '3a' and soda != '4a' and soda != '5a' and soda != '6a' and soda != '7a':

return False

return True

def soda\_select(soda):

# Converts soda to the soda name from the input value

if soda == '1A' or soda == '1a':

return 'Pepsi'

elif soda == '1B' or soda == '1b':

return 'Diet Pepsi'

elif soda == '2A' or soda == '2a':

return 'Coca Cola'

elif soda == '2B' or soda == '2b':

return 'Diet Coke'

elif soda == '2C' or soda == '2c':

return 'Coke Zero'

elif soda == '3A' or soda == '3a':

return 'Dr. Pepper'

elif soda == '4A' or soda == '4a':

return 'Diet Dr. Pepper'

elif soda == '5A' or soda == '5a':

return 'Sprite'

elif soda == '6A' or soda == '6a':

return '7-Up'

elif soda == '7A' or soda == '7a':

return 'Ginger Ale'

def main():

# The program creates coins and soda to utilize user input

coins = 0.00

soda = ''

# Accepts user's soda choice

while not soda\_check(soda):

print('Every soda is $1.00. What soda would you like?')

print('\t1A. Pepsi')

print('\t1B. Diet Pepsi')

print('\t2A. Coca Cola')

print('\t2B. Diet Coke')

print('\t2C. Coke Zero')

print('\t3A. Dr. Pepper')

print('\t4A. Diet Dr. Pepper')

print('\t5A. Sprite')

print('\t6A. 7-Up')

print('\t7A. Ginger Ale')

soda = input('Selection: ')

# Prints an error if user makes an invalid selection

if not soda\_check(soda):

print('Selection invalid.\n')

soda = soda\_select(soda)

# A while loop keeps the program accepting coins

while (coins < 1):

# An input check to confirm input value is one of the designated values

new\_coins = 0

while (new\_coins != 0.05 and new\_coins != 0.10 and new\_coins != 0.25):

new\_coins = round(float(input('Insert coins (0.05, 0.10, 0.25): ')), 2)

# If the value is not accepted, print an error message

if new\_coins != 0.05 and new\_coins != 0.10 and new\_coins != 0.25:

print('Value not accepted. Please enter 0.05, 0.10, or 0.25.')

# If the value is accepted, add it to the count

coins = coins + new\_coins

coins = round(coins, 2)

print('Accepted: $' + str(coins))

# If the value is overpaid, this gives the user their change

if coins > 1:

coins = coins - 1.00

coins = round(coins, 2)

print('Change: $' + str(coins))

# The program remembers what soda the user selected and dispenses it

print(f'Here is your {soda}!')

if \_\_name\_\_ == '\_\_main\_\_':

try:

main()

except:

print('Main failed.')

else:

print('Program failed. Please run independently.')

Problem 2:

'''ASCII Shapes, Problem 2. Written by Cael Shoop.'''

import sys

# Function to print rectanges. Fill is a boolean for whether or not the shape is filled.

def print\_rect(height, width, fill):

for ii in range(height):

for jj in range(width):

# Filled rectangles get \* everywhere

if fill:

sys.stdout.write('\*')

# Non-filled rectangles get specific \* placement

else:

# Top and bottom rows get \*

if ii == 0 or ii == height - 1:

sys.stdout.write('\*')

# Insides get filled with spaces

elif jj > 0 and jj < width - 1:

sys.stdout.write(' ')

# Edges get \*

else:

sys.stdout.write('\*')

sys.stdout.write('\n')

sys.stdout.write('\n')

# Function to print triangles. Fill is a bool for whether or not it is filled.

def print\_tri(height, fill):

width = height \* 2 - 1

count = 0

for ii in range(height):

for jj in range(width):

# If the triangle is to be filled

if fill:

if jj <= count:

sys.stdout.write('\*')

else:

break

# If the triangle is to be empty

else:

# Compares current height to count.

if jj <= count:

# Checks if current location is not last, first, or bottom.

if jj > 0 and jj < count and ii != height - 1:

sys.stdout.write(' ')

# Alternates bottom line with \* and space

elif ii == height - 1 and jj % 2 == 1:

sys.stdout.write(' ')

else:

sys.stdout.write('\*')

else:

break

count = count + 2

sys.stdout.write('\n')

sys.stdout.write('\n')

def main():

print('Welcome to ASCII Shapes!')

# Menu loop

while True:

choice = 0

# Menu selection check

while choice < 1 or choice > 5:

print('Please select an option from the menu.')

print('\t1. Filled rectangle')

print('\t2. Empty rectangle')

print('\t3. Filled triangle')

print('\t4. Empty triangle')

print('\t5. Quit')

# Check if input is an integer and within parameters

try:

choice = int(input('Selection: '))

if choice < 1 or choice > 5:

print('Please enter a choice between 0 and 5.')

except:

print('Please enter an integer.')

# If quit is chosen, say goodbye and break

if choice == 5:

print('Goodbye!')

break

# Take shape height info from user

height = 0

while height < 1:

# Check if input is integer and within parameters

try:

if choice < 3:

height = int(input('Please enter the rectangle\'s height: '))

else:

height = int(input('Please enter the triangle\'s height: '))

if height < 1:

print('Please enter a height larger than 0.')

except:

print('Please enter an integer.')

# Take shape width info from user if shape is rectangle

if choice == 1 or choice == 2:

width = 0

while width < 1:

# Check if input is integer and within parameters

try:

width = int(input('Please enter the rectangle\'s width: '))

if width < 1:

print('Please enter a width larger than 0.')

except:

print('Please enter an integer.')

# Formatting newline

sys.stdout.write('\n')

# Call shape print function based on menu selection

if choice == 1:

print\_rect(height, width, True)

elif choice == 2:

print\_rect(height, width, False)

elif choice == 3:

print\_tri(height, True)

elif choice == 4:

print\_tri(height, False)

if \_\_name\_\_ == '\_\_main\_\_':

try:

main()

except:

print('Main failed.')

else:

print('Program failed. Please run independently.')

Problem 3:

'''Written by Cael Shoop. Third problem, identifies palindromes.'''

import math

def is\_palindrome(num):

# Compares first number to fifth number

if math.floor(num/10000)%10 != num%10:

return False

# Compares second number to fourths number

elif math.floor(num/1000)%10 != math.floor(num/10)%10:

return False

return True

def main():

num = 0

while num != -1:

# Checks if input number matches parameters

while (num < 10000 or num > 99999) and num != -1:

# Makes sure input is an integer, correct length, and not a quit condition

try:

num = int(input('Enter a five-digit integer (or -1 to quit): '))

if (num < 10000 or num > 99999) and num != -1:

print(f'The number {num} is not a five-digit number.\n')

elif num == -1:

break

except:

print('Please enter an integer.')

# If the user is not quitting, check if input is a palindrome

if num != -1:

if is\_palindrome(num):

print(f'The number {num} is a palindrome!\n')

else:

print(f'The number {num} is not a palindrome.\n')

# Reset input for next loop. Without this, it will infinitely loop with

# the given input. Don't ask me how I was reminded of this.

num = 0

print('Good bye!')

if \_\_name\_\_ == '\_\_main\_\_':

try:

main()

except:

print('Main failed.')

else:

print('Program failed. Please run independently.')

Problem 4:

'''Written by Cael Shoop. Checkerboard pattern for problem 4.'''

def main():

for ii in range(7):

# Odd rows get a space at the beginning and are shortened by one

if ii % 2 == 1:

# Using 'end=""' to eradicate the automatic newline in print

print(" ", end="")

jj = jj + 1

# Printing out rows

for jj in range(8):

print("\* ", end="")

# I could use 'print("")' but this is too cursed not to be used

print("\n", end="")

if \_\_name\_\_ == '\_\_main\_\_':

try:

main()

except:

print('Main failed.')

else:

print('Program failed. Please run independently.')

Problem 5:

'''Written by Cael Shoop. Problem 5, Connectivity Calculator.'''

def main():

num = 0

# Loop to check input

while num < 1:

# Makes sure input is a positive integer

try:

num = int(input('Enter an integer n: '))

if num < 1:

print('Please enter a positive value for n.')

else:

break

except:

print('Please enter an integer.')

# Safely counts the number of connections using a choose 2 formula

try:

conns = int((num \* (num - 1)) / 2)

# Corrected output

if conns == 0:

print(f'n={num} has no possible point-to-point connections.')

elif conns == 1:

print(f'n={num} has K{num}=1 possible point-to-point connection.')

else:

print(f'n={num} has K{num}={conns} possible point-to-point connections.')

except:

print('Connection formula failed.')

if \_\_name\_\_ == '\_\_main\_\_':

try:

main()

except:

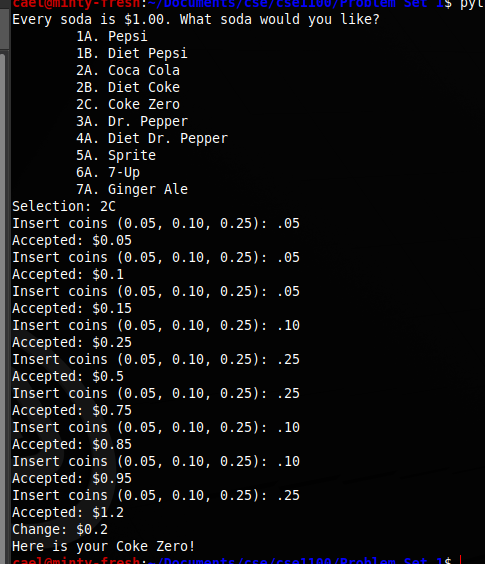
print('Main failed.')

else:

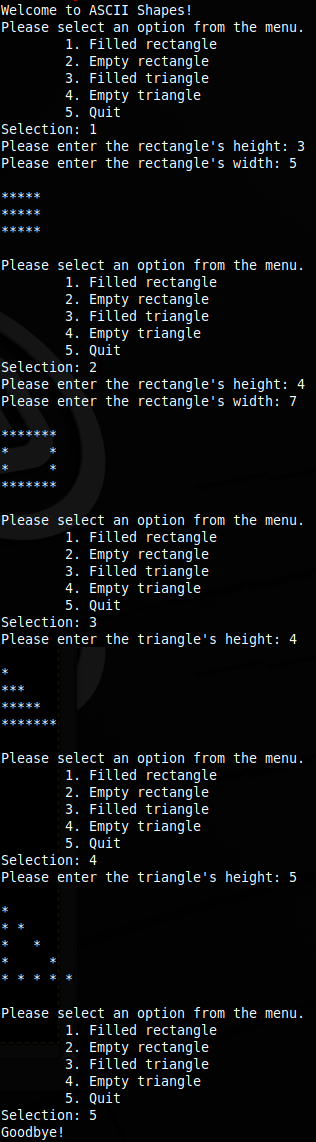
print('Program failed. Please run independently.')

# 4. Software Testing

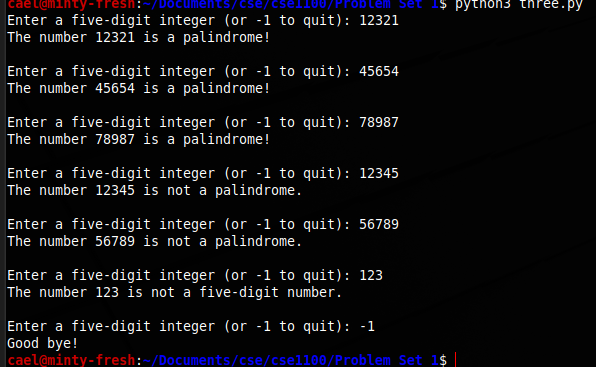
Problem 1:



Problem 2:



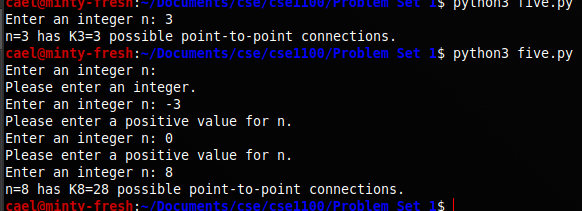
Problem 3:



Problem 4:



Problem 5:



# 5. Self-Reflection

I ran into no issues with problem 1, and had some fun adding different soda choices and a menu for the user. The code verbosely counts the coins added as a real vending machine does.

I failed to notice that, in problem 2, the triangles only require a height input. This led to me unnecessarily creating a system that would change the slope of the hypotenuse depending on the height/width provided by the user, which was functional for widths equal to or greater than the height by the time I noticed my mistake.

In problem 3, I had done some calculations that made the case 12321 work, but when I went to take a screenshot of the testing I found that that was the only case it happened to work for. I fixed my calculations and it worked. My modulus operations took place before the division operations and I used rounding instead of the floor operation, victims of pre-coffee coding.

In problem 4, I was confused by how the instructions would work. Going strictly by what was written, the task was impossible due to print() automatically adding an endline character. With a quick Google search, I found that you can modify this by adding “, end=””” to make the end character null instead of newline. I then realized that you could make a print statement of a newline and then set the automatic newline to null, which hurts me emotionally (just kidding).

In problem 5, I attempted to use numpy choose, but had to create a list for that and felt that it was much more work than just doing the choose calculation directly.