Prac 2 python code

*"""  
CP1404/CP5632 - Practical  
Capitalist Conrad wants a stock price simulator for a volatile stock.  
The price starts off at $10.00, and, at the end of every day there is  
a 50% chance it increases by 0 to 10%, and  
a 50% chance that it decreases by 0 to 5%.  
If the price rises above $1000, or falls below $0.01, the program should end.  
The price should be displayed to the nearest cent (e.g. $33.59, not $33.5918232901)  
"""*import random  
  
OUTPUT\_FILE = "pricelog.txt"  
MAX\_INCREASE = 0.175 # 17.5%  
MAX\_DECREASE = 0.05 # 5%  
MIN\_PRICE = 1.0  
MAX\_PRICE = 100.0  
INITIAL\_PRICE = 10.0  
  
out\_file = open(OUTPUT\_FILE, 'w')  
price = INITIAL\_PRICE  
print("${:,.2f}".format(price))  
day\_counter = 0  
while price >= MIN\_PRICE and price <= MAX\_PRICE:  
 price\_change = 0  
 # generate a random integer of 1 or 2  
 # if it's 1, the price increases, otherwise it decreases  
 if random.randint(1, 2) == 1:  
 # generate a random floating-point number  
 # between 0 and MAX\_INCREASE  
 price\_change = random.uniform(0, MAX\_INCREASE)  
 day\_counter += 1  
 else:  
 # generate a random floating-point number  
 # between negative MAX\_DECREASE and 0  
 price\_change = random.uniform(-MAX\_DECREASE, 0)  
 day\_counter += 1  
  
 price \*= (1 + price\_change)  
 print("On day {} price is: ${:,.2f}".format(day\_counter, price), file=out\_file)  
out\_file.close()  
  
print\_log = open(OUTPUT\_FILE, 'r')  
print(print\_log.read())

finished = False  
result = 0  
while not finished:  
 try:  
 result = int(input("Please enter a number: "))  
 finished = True  
 except ValueError:  
 print("Please enter a valid integer.")  
print("Valid result is:", result)

denominator = 0  
while denominator == 0:  
 try:  
 numerator = int(input("Enter the numerator: "))  
 denominator = int(input("Enter the denominator: "))  
 except ValueError:  
 print("Numerator and denominator must be valid numbers!")  
 continue  
 if denominator == 0:  
 print("Denominator cannot be zero!")  
 else:  
 break  
fraction = numerator / denominator  
print(fraction)  
  
print("Finished.")  
  
"""  
1. When will a ValueError occur?  
-- a ValueError will occur when an invalid value is provided when a   
 specific value type is asked for  
 (for example, in this case, if a str or float is   
 entered when asked for a numerator/denominator,  
 a ValueError will be returned)  
2. When will a ZeroDivisionError occur?  
-- when a float or integer is attempted to be divided by '0'   
 a ZeroDivisionError will occur  
  
3. Could you change the code to avoid the possibility of a ZeroDivisionError  
-- to accomplish this we must make it so that the division   
 does not take place when '0' is entered as the denominator   
"""

# Question one  
NAME\_FILE = "name\_store.txt"  
name = '0'  
name\_out = open(NAME\_FILE, 'w')  
while name.isnumeric() is True:  
 name = str(input("please enter your name: "))  
 if name.isnumeric() is True:  
 print("name cannot include numbers!")  
  
print(name, file=name\_out)  
name\_out.close()  
  
# Question two  
  
name\_in = open(NAME\_FILE, 'r')  
  
print("Your name is {}".format(name\_in.read()))  
name\_in.close()  
  
# Question three  
NUM\_IN = "numbers.txt"  
num\_out = open(NUM\_IN, 'r')  
num1 = int(num\_out.readline())  
num2 = int(num\_out.readline())  
num\_total = num1 + num2  
print('the result is: {}'.format(num\_total))  
num\_out.close()  
  
# Question four  
num\_read = open(NUM\_IN, 'r')  
total = 0  
for line in num\_read:  
 num\_hold = int(line)  
 total += num\_hold  
num\_read.close()  
print(total)

*"""  
CP1404/CP5632 - Practical  
Password checker "skeleton" code to help you get started  
"""*MIN\_LENGTH = 2  
MAX\_LENGTH = 6  
SPECIAL\_CHARS\_REQUIRED = True  
SPECIAL\_CHARACTERS = "!@#$%^&\*()\_-=+`~,./'[]<>?{}|\\"  
  
  
def main():  
 *"""Program to get and check a user's password."""* print("Please enter a valid password")  
 print("Your password must be between", MIN\_LENGTH, "and", MAX\_LENGTH,  
 "characters, and contain:")  
 print("\t1 or more uppercase characters")  
 print("\t1 or more lowercase characters")  
 print("\t1 or more numbers")  
 if SPECIAL\_CHARS\_REQUIRED:  
 print("\tand 1 or more special characters: ", SPECIAL\_CHARACTERS)  
 password = input("> ")  
 while not is\_valid\_password(password):  
 print("Invalid password!")  
 password = input("> ")  
 print("Your {}-character password is valid: {}".format(len(password),  
 password))  
  
  
def is\_valid\_password(password):  
 *"""Determine if the provided password is valid."""* # *TODO: if length is wrong, return False* if MIN\_LENGTH < len(password) > MAX\_LENGTH:  
 return False  
  
 count\_lower = 0  
 count\_upper = 0  
 count\_digit = 0  
 count\_special = 0  
 for char in password:  
 # *TODO: count each kind of character (use str methods like isdigit)* if char.islower():  
 count\_lower += 1  
 elif char.isupper():  
 count\_upper += 1  
 elif char.isdigit():  
 count\_digit += 1  
 pass  
  
 # *TODO: if any of the 'normal' counts are zero, return False* if count\_lower == 0 or count\_digit == 0 or count\_upper == 0:  
 return False  
 else:  
 pass  
 # *TODO: if special characters are required, then check the count of those* if SPECIAL\_CHARS\_REQUIRED:  
 for char in password:  
 if char in SPECIAL\_CHARACTERS:  
 count\_special += 1  
 if count\_special == 0:  
 return False  
 else:  
 pass  
 # and return False if it's zero  
  
 # if we get here (without returning False), then the password must be valid  
 return True  
  
  
main()

*"""  
CP1404/CP5632 - Practical  
Various examples of using Python string formatting with the str.format() method  
Want to read more about it? https://docs.python.org/3/library/string.html#formatstrings  
"""*name = "Gibson L-5 CES"  
year = 1922  
cost = 16035.4  
  
# The ‘old’ manual way to format text with string concatenation:  
print("My guitar: " + name + ", first made in " + str(year))  
  
# A better way - using str.format():  
print("My guitar: {}, first made in {}".format(name, year))  
print("My guitar: {0}, first made in {1}".format(name, year))  
print("My {0} was first made in {1} (that's right, {1}!)".format(name, year))  
  
# Formatting currency (grouping with comma, 2 decimal places):  
print("My {} would cost ${:,.2f}".format(name, cost))  
  
# Aligning columns:  
numbers = [1, 19, 123, 456, -25]  
for number in numbers:  
 print("Number is {:>5}".format(number))  
  
# A version of the above loop using the enumerate function, useful when you want the index and value  
for i, number in enumerate(numbers):  
 print("Number {0} is {1:>5}".format(i + 1, number))  
  
# *TODO: Use string formatting to produce the output:*# 1922 Gibson L-5 CES for about $16,035!  
print("{0} {1} for about ${2:.0f}!".format(year, name, cost))  
  
# *TODO: Using a for loop with the range function and string formatting,*# produce the following right-aligned output (do not use a list):  
# 0  
# 50  
# 100  
# 150  
number = 0  
for i in range(4):  
 print("{:>5}".format(number))  
 number += 50

import random  
print(random.randint(5, 20)) # line 1  
print(random.randrange(3, 10, 2)) # line 2  
print(random.uniform(2.5, 5.5)) # line 3  
  
"""  
  
what did you see on line 1  
- The smallest number I could have seen is a 5, and the largest is a 20  
what did you see on line 2?  
- The smallest was a 3 and the largest is a 9  
 I would not have been able to see a 4 on line 2 because the step is 2 and the range starts off on an odd number  
what did you see on line 3?  
-The smallest number I could have see is a 2.5 and the largest would be a 5.4999999999999...  
  
"""  
  
print(random.randint(1, 100))