Capitalist Conrad:

*"""  
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  
  
MAX\_INCREASE = 0.10 # 17.5%  
MAX\_DECREASE = 0.05 # 5%  
MIN\_PRICE = 0.01  
MAX\_PRICE = 1000.0  
INITIAL\_PRICE = 10.0  
OUTPUT\_FILE = "capitalistconradoutput.txt"  
price = INITIAL\_PRICE  
days = 0  
out\_file = open(OUTPUT\_FILE, 'w')  
print("The starting price is: ${:,.2f}".format(price), file=out\_file)  
  
while MIN\_PRICE <= price <= MAX\_PRICE:  
 price\_change = 0  
 days = days + 1  
 # 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)  
 else:  
 # generate a random floating-point number  
 # between negative MAX\_DECREASE and 0  
 price\_change = random.uniform(-MAX\_DECREASE, 0)  
  
 price \*= (1 + price\_change)  
 print("On day", days, "price is ${:,.2f}".format(price))  
  
out\_file.close()

Exceptions Demo:

*"""  
CP1404/CP5632 - Practical  
Answer the following questions:  
1. When will a ValueError occur?  
2. When will a ZeroDivisionError occur?  
3. Could you change the code to avoid the possibility of a ZeroDivisionError?  
"""*try:  
 numerator = int(input("Enter the numerator: "))  
 denominator = int(input("Enter the denominator: "))  
 fraction = numerator / denominator  
 print(fraction)  
except ValueError:  
 print("Numerator and denominator must be valid numbers!")  
except ZeroDivisionError:  
 print("Cannot divide by zero!")  
print("Finished.")  
  
"""  
1. A ValueError will occur when the numerator and denominator are not integers.  
2. ZeroDivisionError will occur when the user inputs 0 to the denominator.  
  
"""

Exceptions to complete:

*"""  
CP1404/CP5632 - Practical  
Fill in the TODOs to complete the task  
"""*finished = False  
result = 0  
while not finished:  
 try:  
 result = int(input("Enter an integer: "))  
 finished = True  
 pass  
 except ValueError:  
 print("Please enter a valid integer.")  
print("Valid result is:", result)

Files:

# question 1  
question1 = open("name.txt", "w")  
name = input("Enter your name: ")  
print(name, file=question1)  
question1.close()  
  
# question 2  
question2 = open("name.txt", "r")  
name = question2.read().strip()  
print("The name you have entered is:", name)  
question2.close()  
  
# question 3  
question3 = open("numbers.txt", "r")  
num1 = int(question3.readline())  
num2 = int(question3.readline())  
print(num1 + num2)  
question3.close()  
  
# question 4  
question4 = open("numbers.txt", "r")  
total = 0  
for i in question4:  
 q4num = int(i)  
 total += q4num  
print(total)  
question4.close()

Password Checker:

*"""  
CP1404/CP5632 - Practical  
Password checker "skeleton" code to help you get started  
"""*MIN\_LENGTH = 2  
MAX\_LENGTH = 6  
SPECIAL\_CHARS\_REQUIRED = False  
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* 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  
 elif char in SPECIAL\_CHARACTERS:  
 count\_special += 1  
 pass  
  
 # *TODO: if any of the 'normal' counts are zero, return False* if count\_lower == 0 or count\_upper == 0 or count\_digit == 0:  
 return False  
  
 # *TODO: if special characters are required, then check the count of those* # and return False if it's zero  
 if SPECIAL\_CHARS\_REQUIRED:  
 if count\_special == 0:  
 return False  
 # if we get here (without returning False), then the password must be valid  
 return True  
  
  
main()

Randoms:

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