Zhao FangLue, Axel

Student ID: 14036554

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  
OUTPUT\_FILE = **"Capitalist\_conrad.txt"**out\_file = open(OUTPUT\_FILE, **'w'**)  
  
MAX\_INCREASE = 0.175  
MAX\_DECREASE = 0.05  
MIN\_PRICE = 1  
MAX\_PRICE = 100  
INITIAL\_PRICE = 10.0  
  
day = 1  
price = INITIAL\_PRICE  
print(**"${:,.2f}"**.format(price))  
  
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 = day + 1  
 else:  
 *# generate a random floating-point number  
 # between negative MAX\_DECREASE and 0* price\_change = random.uniform(-MAX\_DECREASE, 0)  
 day = day + 1  
 price \*= (1 + price\_change)  
 print(**"On day {} price is ${:,.2f}"**.format(day, price), file=out\_file)  
  
out\_file.close()

exceptions\_demo

*"""  
CP1404/CP5632 - Practical  
Answer the following questions:  
1. When will a ValueError occur?  
 ANS: When the item is not an integer  
2. When will a ZeroDivisionError occur?  
ANS: When the denominator is zero  
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: "**))  
 *# if the denominator is 0 it will prompt thus not triggered the ZeroDivisionError* if denominator == 0:  
 print(**'Denominator cannot be 0'**)  
 else:  
 fraction = numerator / denominator  
 print(fraction)  
except ValueError:  
 print(**"Numerator and denominator must be valid numbers!"**)  
except ZeroDivisionError:  
 print(**"Cannot divide by zero!"**)  
print(**"Finished."**)

exceptions\_to\_complete

*"""  
CP1404/CP5632 - Practical  
Fill in the TODOs to complete the task  
"""  
  
# This is a try,except,else way to do it doesn't make use of boolean so "not" and "false" is pointless*finished = False  
result = 0  
while not finished:  
 try:  
 result = int(input(**"Enter an integer : "**))  
 except ValueError:  
 print(**"Please enter a valid integer."**)  
 else:  
 print(**"Valid result is:"**, result)  
 exit()  
  
*# This is what is expected*finished = False  
result = 0  
while not finished:  
 try:  
 result = int(input(**"Enter an integer: "**))  
 finished = True  
 except ValueError:  
 print(**"Please enter a valid integer."**)  
print(**"Valid result is:"**, result)

Files.py

*# Question 1*output\_file = **"name.txt"**out\_file = open(output\_file, **'w'**)  
  
name = str(input(**"Enter your name: "**))  
  
print(name, file=out\_file)  
  
out\_file.close()  
  
*# Question 2*file = open(**'name.txt'**, **'r'**)  
print(**"Your name is "** + file.readline())  
  
*# Question 3*file = open(**'numbers.txt'**, **'r'**)  
first\_number = file.readline()  
second\_number = file.readline()  
  
answer = int(first\_number) + int(second\_number)  
print(answer)  
  
*# Question 4*file = open(**'numbers.txt'**, **'r'**)  
  
count = 0  
total = 0  
  
*# For loop to read total number of lines in it by adding +1 to count every line read and save to count*for line in open(**'numbers.txt'**).readlines():  
 count += 1  
  
*# Loop the amount of times there is lines due to count*for i in range(count):  
 number = file.readline()  
 total += int(number)  
  
print(total)

password\_checker

import re  
  
MIN\_LENGTH = 5  
MAX\_LENGTH = 15  
SPECIAL\_CHARS\_REQUIRED = False  
SPECIAL\_CHARACTERS = **r"[!@#$%^&\*()\_-=+`~,./'[]<>?{}|\\]"**def main():  
 print(**"Please enter a valid password"**)  
 print(**"Your password must be between"**, MIN\_LENGTH, **"and"**, MAX\_LENGTH,  
 **"characters, and contain:"**)  
 print(**"**\t**1 or more uppercase characters"**)  
 print(**"**\t**1 or more lowercase characters"**)  
 print(**"**\t**1 or more numbers"**)  
 print(**"**\t**and 1 or more special characters: "**, SPECIAL\_CHARACTERS)  
 password = input(**"> "**)  
 while not is\_valid\_password(password):  
 print(**"Invalid password!"**)  
 password = input(**"> "**)  
 print(**"Your "** + str(  
 len(password)) + **" character password is valid: "** + password)  
  
  
def is\_valid\_password(password):  
 *# Make sure the length requirement is met* if len(password) < MIN\_LENGTH or len(password) > MAX\_LENGTH:  
 return False  
 *# Check for if a all the way to z is not inside, if its not it will be false triggering line 19* elif not re.search(**"[a-z]"**, password):  
 return False  
 *# Checking for capital letters from A to Z* elif not re.search(**"[A-Z]"**, password):  
 return False  
 *# Checking for numbers 0 to 9* elif not re.search(**"[0-9]"**, password):  
 return False  
 *# If predefined special characters is inside, it will return true and skip the statement in line 18 due to boolean* for character in SPECIAL\_CHARACTERS:  
 if character in password:  
 return True  
 *# When it return false it will stay in while loop and execute the codes in it, if true it will skip the while loop* return False  
  
  
main()

randoms.py

*# What did you see on line 1?  
# ANS: A single integer  
# What was the smallest number you could have seen, what was the largest?  
# ANS: Smallest was 5 and largest was 20  
  
# What did you see on line 2?  
# ANS: A single integer that's an odd number  
# What was the smallest number you could have seen, what was the largest?  
# ANS: Smallest was 3, largest was 9  
# Could line 2 have produced a 4?  
# ANS: No, it is all odd numbers  
  
# What did you see on line 3?  
# ANS: A single float  
# What was the smallest number you could have seen, what was the largest?  
# ANS: Smallest was 2.5, largest was 5.5  
  
  
# Write code, not a comment, to produce a random number between 1 and 100 inclusive.*import random  
  
print(random.randint(1, 100))

string\_formatting\_examples

*"""  
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:  
# Both method works*print(**"1992 {0} for about ${1:,}"**.format(name, int(cost)))  
print(**"1992 {0} for about ${1:,.0f}"**.format(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*numbers = [0, 50, 100, 150]  
for number in numbers:  
 print(**"{:>3}"**.format(number))