*# 1.*

*"""  
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(**"{} {} for about ${:,.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***for** number **in** range(0, 151, 50):  
 print(**"{:3}"**.format(number))

*# 2. Randoms  
  
# On line 1 i saw the number 9. The smallest number i could have seen was 5 and the largest was 20  
  
# On line 2 i saw the number 7. The smallest number i could have seen was 3 and the largest was 9. Line 2 could not have produced a 4  
  
# On line 3 i saw the number 5.349298472950316. The smallest number i could have seen was 2.5 and the largest was 5.5.***import** random  
  
print(random.randint(0,100))

*# 3.  
"""  
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 = 0.175 *# 17.5%*MAX\_DECREASE = 0.05 *# 5%*MIN\_PRICE = 1  
MAX\_PRICE = 100  
INITIAL\_PRICE = 10.0  
OUTPUT\_FILE = **"output.txt"**out\_file = open(OUTPUT\_FILE, **"w"**)  
current\_day = 0  
price = INITIAL\_PRICE  
print(**"Starting price: ${:,.2f}"**.format(price), file=out\_file)  
  
**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)  
 **else**:  
 *# generate a random floating-point number  
 # between negative MAX\_DECREASE and 0* price\_change = random.uniform(-MAX\_DECREASE, 0)  
  
 current\_day += 1  
 price \*= (1 + price\_change)  
 print(**"On day {} price is : ${:,.2f}"**.format(current\_day,price), file=out\_file)  
out\_file.close()

*# 4.  
"""  
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: "**))  
 **while** denominator == 0:  
 print(**"Cannot divide by zero!"**)  
 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."**)  
  
*# ValueError will occur when a valid value is not entered as one of the numbers. e.g. if no numbers are entered  
# ZeroDivisionError will occur when the user try's to divide by 0  
# The code could be changed to avoid a ZeroDivisionError by making it impossible to input a zero.*

*# 5.  
"""  
CP1404/CP5632 - Practical  
Fill in the TODOs to complete the task  
"""*finished = **False**result = 0  
**while not** finished:  
 **try**:  
 result = int(input(**"Enter a valid integer: "**))  
 finished = **True  
 except** ValueError:  
 print(**"Please enter a valid integer."**)  
print(**"Valid result is:"**, result)

*# 6.*out\_file = open(**"name.txt"**, **"w"**)  
name = input(**"What is your name?: "**)  
print(name, file=out\_file)  
out\_file.close()

in\_file = open(**"name.txt"**, **"r"**)  
name = in\_file.read()  
in\_file.close()  
print(**"Your name is {}"**.format(name))

in\_file = open(**"numbers.txt"**, **"r"**)  
number\_one = int(in\_file.readline())  
number\_two = int(in\_file.readline())  
in\_file.close()  
result = number\_one + number\_two  
print(result)  
  
total = 0  
in\_file = open(**"numbers.txt"**, **"r"**)  
**for** line **in** in\_file:  
 number = int(line)  
 total += number  
in\_file.close()  
print(total)

*# 7.  
"""  
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."""* **if** len(password) < MIN\_LENGTH **or** len(password) > MAX\_LENGTH:  
 **return False** count\_lower = 0  
 count\_upper = 0  
 count\_digit = 0  
 count\_special = 0  
 **for** char **in** password:  
 **if** char.isdigit():  
 count\_digit += 1  
 **elif** char.islower():  
 count\_lower += 1  
 **elif** char.isupper():  
 count\_upper += 1  
 **elif** char **in** SPECIAL\_CHARACTERS:  
 count\_special += 1  
  
 **if** count\_lower == 0 **or** count\_upper == 0 **or** count\_digit == 0:  
 **return False  
  
 if** SPECIAL\_CHARS\_REQUIRED:  
 **if** count\_special == 0:  
 **return False  
  
 return True**main()