OPO/OLA: Programming 1 CODE: MBI02J

LECTURERS: Aimée Backiel, Serhat Erdogan

DATE: 11/27/2023

Tools: Your own laptop, course materials, secured network, xToledo

Maximum duration: 2u

Last name and first name: ……..

Studentnumber: ……

This practice exam consists of 5 questions. It is a good idea to read the full assignment first. Work step by step. If you get stuck somewhere, continue with what can be done.

It is allowed to take inspiration from already made exercises and summaries. We suggest that you turn off your WIFI while taking this practice exam. This way you will experience what information on your laptop will and will not be available on the real exam.

Remember that duplicate code is rarely a good idea.

We provide a file *test_student.py* with some very simple tests to check superficial details such as names of functions and parameters. This does not test the functionality of your program.

Check thoroughly that you have covered all sub-tasks.

Good luck!

1. **C-som**

The c-som (c-sum) of a number is determined by calculating the sum of the digits of this number, and repeating this procedure until the obtained number consists of a single digit. Thus, the c-som of a number ultimately results in a single digit c ($1 \leq c \leq 9$).

Write a function **csom** with parameter n that accepts a positive integer. The function must return the c-som of n.

Example:

c-som(8) = 8

c-som(377096267) = c-som(47) = c-som(11) = 2

2. **Palindrome**

We call a tuple a palindrome when the elements are arranged symmetrically. This means that the order of the elements remains the same whether it's read front to back or back to front.

Write a function **isPalindrome** with a tuple **t** as a parameter. The function should return True or False.

Example:

isPalindrome(1,2) → *False*

isPalindrome (1,1) → *True*

isPalindrome (1,2,3,2,1) → *True*

isPalindrome (1,"1") → *False*

isPalindrome (()) → *True*

isPalindrome (1,) → *True*

3. Attraction

Create a class **Attraction** with the following properties:

- **name** → cannot be empty
- **height** → minimal height in centimeters
  → cannot be empty, cannot be negative
- **visitors** → has a value of zero for every new attraction

For name and height, provide a getter and a setter for each via properties.

When the parameters do not meet the conditions, it should generate a ValueError with an appropriate message.

Provide a method **visit** with **height** as parameter that specifies the height of the visitor in cm. If the height is acceptable, the visitor number is incremented by 1. If the height does not meet the attraction's requirements, a relevant error message is added to the file 'log.txt'. If 'log.txt' does not already exist, it should be created.

Create 3 attractions:

- name "De Pagode" – minimal height 0 cm
- name "Max & Moritz" – minimal height 100 cm
- name "De Baron" – minimal height 132 cm

Visit Max & Moritz 2x with a valid height.

4. Theme Park

Create a class **ThemePark** with the following properties:

- **name** → cannot be empty
  → has to start with a capital letter
- **attractions** → always empty when creating a new theme park
  → you are obliged to use a list, not a set!

Provide a getter and a setter for **name** via properties.

If the parameters do not meet the conditions, generate a ValueError with an appropriate message.

Provide a method **addAttraction** with **attraction** as parameter that passes an object of the class Attraction. When adding an attraction that already exists in the amusement park, this attraction is not added again. To determine if 2 attractions are the same, we only check the name. It should not be a problem if the name of the attraction is written with an incorrect number of capital letters or spaces, your program is flexible enough to catch these typos.

Example: 'Max & Moritz' is considered the same name as 'Max&moritz'

Provide a method **printOverview** that displays a message in the terminal as a result, taking into account the formatting in the following examples.

Create an amusement park named "The Efteling." Print the overview.

Desired output:

De Efteling is an amusement park with 0 attractions visited a total of 0 times.

Add The Pagoda from exercise 3. Print the overview.

Desired output:

De Efteling is an amusement park with 1 attraction visited a total of 0 times.

Add Max & Moritz and De Baron from exercise 3. Print the overview.

Desired output:

De Efteling is an amusement park with 3 attractions that were visited a total of 2 times.

5. Extra: security

For security reasons, you will be asked to add additional functionality to your program. It should be possible to adjust the minimum height associated with an attraction. To accomplish this, add a new method to the class **ThemePark** called **securityupdate** which takes a file as a parameter. The file contains an overview of the necessary adjustments.

You may assume that the new data will be delivered in a text file whose lines have the following structure:

110 cm : De Pagode, Max&Moritz

140 cm : De Baron

It is important that the visitor numbers of the attractions are preserved after the update!

It should not be a problem if the name of an attraction is written with an incorrect number of capital letters or spaces, your program is flexible enough to catch these typos.

Example: 'Max & Moritz' is considered the same name as 'Max&moritz'

If the text file contains an attraction that does not exist in the amusement park, a relevant error message is added to the file 'log.txt'. If 'log.txt' does not already exist, it should be created.

After the update, a list of attractions with corresponding maximum size should be displayed in the terminal as follows:

Overview of the attractions after security update:

- *De Pagode: 110 cm*
- *Max & Moritz: 110 cm*
- *De Baron: 140 cm*