Published

Week 11: Conversions, Lists 8 of 8

11 Week 11 Assignment

Computer SCIENCE

This week's homework is perhaps the most difficult of the semester. However, many have said this one isn't that bad because of what was learned in the Blobber assignment. I hope you find that to be the case. The task is comprised of a number of smaller problems, each associated with a menu item. Try to break things down into pieces and create a good design plan for each piece.

Objectives

1. Lists

Task 1 - Orbian Family

You are charged with creating and managing an Orbian family. An Orbian is an interesting human-like species. The name comes from their perfectly spherical heads. Their bodies are shaped like perfect cylinder. You will manage the family by creating an initial group of Orbians, and performing various actions such as creating Orbian babies and sending Orbians to pasture either one at a time, or Thanos style (If this needs to be explained, then I feel very badly for you. In any case, read the second sentence of this article for clarification: https://www.mentalfloss.com/article/566894/avengers-infinity-war-thanos-snap-official-name (https://www.mentalfloss.com/article/566894/avengers-infinity-war-thanos-snap-official-name)).

You will create three files for this assignment: plan1.txt, orbian.py and task1.py.

plan1.txt: The software design for the Orbian class. Include UML as well. A plan for the task1.py is not necessary

orbian.py: Contains the class definition for Orbian. See the rest of the assignment description for all details. Use the orbian_starter.py file to get started.

task1.py: This file runs the program and uses the orbian.py file. Use the starter code for this file. Only modify the sections indicated. Use the existing code to help you know how to build orbian.py

Watch the demo video first. It'll help the requirements make sense.

DEMO VIDEO

Start UP

Starter Code: **unit6_task4_starter.py** (https://usu.instructure.com/courses/681553/files/81729291/download?download_frd=1) (this will become your task1.py file), **orbian_starter.py** (https://usu.instructure.com/courses/681553/files/81729289/download?download_frd=1)

Make sure to use the starter code. It will give you insight into how your program should be built. Pay attention to the sections indicated so you know what to modify and what not to change.

The program starts by creating four Orbians. The user selects a name for each Orbian, but its physical characteristics are determined randomly.

Menu

The menu has been created for you. Each menu item selection makes a single function call, except for the *Quit* option, as will be explained. You will define and/or complete the functions. Each of the following items explains what the user should experience for each menu item.

Meet Orbian Family

This selection should display a message stating each of the Orbian's names. The print function should take an Orbian object as a parameter. This means you should not call a method on the orbian object to determine what prints. Your print statement should look similar to: print(orbian), where orbian is an Orbian Object.

Compare Orbians

Prompt the user with a message and a menu to select from the members of the Orbian family. After making one selection the selection menu is presented again, indicating which Orbian was already selected. You may assume that the user will select an appropriate Orbian the second time. The selection menu (but not the selection menu title) and the prompt for selection must all be handled in a function called *selectOrbian().* Note that the *selectOrbian()* function returns an Orbian object.

Notice that the Orbian objects are compared with the == and > operators. This should help you understand how to build the Orbian class. The actual comparison is made by comparing the total volume of the Orbians, meaning the volume of their head and their body.

Orbian Info

This function should not be modified at all. Use it to help you understand how to build the Orbian class.

The message includes the following information for the selected Orbian:

Name
Age - Age is measured in zungs. A zung is defined as 5 seconds
Volume - Volume is measure in zogs. A zog is equal to volume of the head and body of the Orbian.
Height - Height is measured in zings. A zing is equal to the total height of the Orbian. That is the body height plus the head height.

Your output should only include values with up to two decimal places. However, changing everything to an integer representation is fine as well. You will not lose points for results that are off-by-one (rounding errors). Note that the only way to access the height is by passing the Orbian object to the len() function. Ex. print("The height is " + str(len(orbian)). The *len()* function only returns integers.

Create Orbian Baby

Any two Orbians may create an Orbian baby. Present the user with a menu to select the first Orbian, followed by a menu to select the second. It should follow the same pattern as above, where the Orbian selected first should be indicated while selecting the second. After selecting both Orbians, you create a new Orbian by using the + operator on the two

Orbians. The result should be a new baby Orbian. This means a baby is created like this: babyOrbian = parentOne + parentTwo. A baby Orbian's vitals and name are determined automatically as follows.

Name - A random combination of the letters from the parent's names, with the length being the average length of the parents' names. The random.shuffle() function may come in handy for you. Make sure that only the first letter of the baby's name is capitalized!

Head Radius: 25% of the sum of the parents' head radius

Body Radius: 25 % of the sum of the parents' body radius

Body Height: 12.5% of the sum of the parents' body height

Send Orbian to Pasture

Sending an Orbian to pasture is something that is celebrated, not mourned. So do not think of this as a bad thing. It means they lived a full and happy life and going to pasture is a reward.

Present the user with the selection menu to choose the Orbian to send to pasture. Then print a farewell message that includes the Orbian's name

Orbian Thanos

Unlike sending an Orbian to pasture, Orbian Thanos is a terrible terrible thing. Orbian Thanos snaps his fingers (not sure how since he has no appendages) and 1/2 of all Orbians vanish. Most likely they end up somewhere terrible. Who knows? I made this world up, and I don't know. The process is completely random (Think about how you're making the baby names for insight on how to do this). For an odd number of initial Orbian's, you decide if you keep the extra Orbian, or they disappear. This will give the grader insight into your personality.

Quit

This menu item is completed for you.

Orbian Class

Almost all details of the Orbian class is described above. Here are few extra items you need to know.

Starter Code

You do not need to modify any of the starter code. Make note of the class data. All but the *__adult* data are constants. Three other methods are provided for you, and are intended to

help. Notice the methods associated with volume do not use the radius and height values directly. There is a reason for this. You can take the existence of the *__ageCheck()* method as a hint as to why this is.

Growing Old

Orbians age in an interesting way. Their age is measured in zungs. A zung is defined as 5 seconds. So 5 seconds after birth an Orbian is 1 zung old. Then at 10 seconds is 2 zungs old. When an Orbian reaches the age of 2 zungs, they instantly turn into an adult. As an adult they grow as follows:

Head Radius: increases by 100%

Body Radius: increases by 100%

Body Height: increases by 300%

Note that once reaching adulthood, an Orbian does not grow anymore. This is a one-time event.

Private data/methods

Everything that should not be accessible outside of the class should be made private.

Dunders

The appropriate dunder methods should be used to overload the operators appropriately. See the Menu description and starter code for details on how certain things must be handled.

NOTE

This is the most difficult task of the entire semester. It will take a lot of time and thought. The program is really a bunch of small problems, each one a menu item. Make a good plan and break the problems down into small pieces that are simple to understand and implement.

Points 100

Submitting a text entry box or a file upload

Due	For	Available from	Until
Nov 19, 2021	Everyone	-	-

Unit 6 (1)

Criteria	Ratings		Pts
Task 1: UML and Software Design for Orbian class Design only required for Orbian class. Include software design for all methods not included in starter code.	**9 pts Full Marks**	**0 pts No Marks**	9 pts
Task 1: Orbian class implemented properly Starter code is not modified. Only additional methods are defined. All methods that should be private are private. Appropriate public getter and setter methods (if necessary) are included. Other functionality is part of the grade associated with each menu item.	**9 pts Full Marks**	**0 pts No Marks**	9 pts
Task 1: Compare Orbians Menu Item Use the selectOrbian() function properly. Use appropriate comparison operators on objects. Orbian class defined properly to allow for comparison operators to be used on the object. Prints proper message.	**14 pts Full Marks**	**0 pts No Marks**	14 pts
Task 1: Orbian Info Menu Item Do not modify this at all.	**2 pts Full Marks**	**0 pts No Marks**	2 pts
Task 1: Create Baby Orbian Menu Item Use the selectOrbian() function properly. Creates new orbian using the + operator. Orbian class has proper implementation to make the + operator work properly. Initial values of baby, including name, are created as described above. Baby added to list properly. Proper messaging as described.	**18 pts Full Marks**	**0 pts No Marks**	18 pts
Task 1: To Pasture Menu Item Use the selectOrbian() function properly. Proper messaging and management of the orbian object list.	**14 pts Full Marks**	**0 pts No Marks**	14 pts
Task 1: Orbian Thanos Menu Item Proper messaging as demonstrated. Proper randomization and elimination of orbian objects. Proper management of the Orbian object list.	**18 pts Full Marks**	**0 pts No Marks**	18 pts
Task 1: selectOrbian() function Get points for not modifying this function at all!	**2 pts Full Marks**	**0 pts No Marks**	2 pts

Criteria	Ratings		Pts
Task 1: Orbian Age Age and orbian measurements are managed properly as described above. All code to manage this is in the Orbian Class. All output should use the appropriate age if applicable.	**14 pts Full Marks**	**0 pts No Marks**	14 pts
			Total Points: 100