Published

Week 9: Classes, Objects 8 of 8

9 Week 9 Assignment

Computer SCIENCE

IMPORTANT: Please review the statement on Academic Integrity. Submitting code that is not your own may result in an automatic F for the class.
Statement on Academic Integrity

Objectives

1. Classes
2. Objects
3. Methods
4. UML
5. Private data
6. Constructors

Task 1 - Dr. Moody

Starter code: **unit5_task1_starter.py** (https://usu.instructure.com/courses/681553/files/81729315/download?download_frd=1) (make sure to change the name to task1.py).

You will not make a UML diagram or Software Design Plan for this task.

See the demo video for this assignment. Use the provided code file to complete the task to match the video. Use the starter code provided. Note that the class is defined in the same file as the main function. This is for convenience and you may leave it that way. No import statement is needed. You must **fill in the blanks (<Fill-In>), as well as add any other methods or data** that is needed. All blanks should be completed with a single line of code (or partial line). You should NOT modify any other code that exists. I repeat, you should NOT modify any other code. If you have any questions about this, the answer is, "No, you may not modify any existing code." Only remove the blanks and add other missing methods.

Use a radius of 100 for the head. The mouth is a thicker line. The face is centered at 0, 0.

Video Demo

Common Errors

The #1 problem on this assignment is not using the information that has been bolded above. Make sure to review **conditional expressions**, and add any methods. You may not modify the code other than the **<Fill-In>** sections, but there are some methods that you have to add. Read the starter code and it should be obvious what they are.

Task 2 - Blobber

Call your files task2.py, blobber.py, and plan2.txt.

NOTE: This is a challenging assignment. The key is to break it down into small pieces. Follow the instructions closely. It's almost a checklist of things to do. Figuring out how feeding the Blobber works takes some thought. You can save that one for last. Make sure to take time to understand the starter code too.

You are going to complete a program called Blobber. A Blobber is an imaginary being who is shaped like a cylinder. Your program will allow a user to create a Virtual Blobber they can take care of, similar to virtual pet like a Tamagotchi. Taking care of a Blobber is not an easy task. It takes constant vigilance and dedication. By the time you're done testing your program you will never want to own an actual Blobber.

Starter Code

unit5_task2_starter.py (https://usu.instructure.com/courses/681553/files/81729328/download?download_frd=1)

Change the name of this file to task2.py. Only change the file where marked **<Fill-In>**. If **<Fill-In>** is only a line by itself, then you must complete the entire line. If it is part of a line, then you just add code to replace the **<Fill-In>** marker.

How a Blobber works

A Blobber is a funny kind of being. It is known by a combination of a name and color. It is shaped like a cylinder and likes to maintain a certain level of happiness. Its happiness is determined solely by how close its current volume is compared to its volume at birth. Immediately following birth, its radius begins to decrease. The Blobber caretaker can feed the Blobber regularly in order to maintain its volume. If the volume is ever too high or too low, the Blobber turns to dust.

Create a class called Blobber inside of a file called blobber.py. Your Blobber class should have the following properties:

- A private float data field called radius
- A private float data field called height
- A private string data field called color. The color should always be lowercase.
- A private string data field called name. The name should always be capitalized.
- A constructor is called with a name, color, radius, and height.
- Accessor (getter) and mutator (setter) methods for name and color
- A method called feedBlobber()
  - Takes a float value and increases the radius by the given amount
  - This method takes some thought!
- A method called blobberSpeak() that returns a string that can be printed.
  - The string should be a complete two line statement that includes the name, color, and happiness level of the Blobber.
  - Ex:
    - My name is Alice, and I am green.
    - My current happiness level is 93.25%
  - The happiness level should be shown as a percentage with two decimal places
  - Note: this method returns a string, it does not print a string
- A method called vitalsOK().
  - See the Blobber Vitality section for more details
  - Returns two values:
    - The current happiness level
    - A Boolean value
      - True if the Blobber is alive.
      - False if the Blobber has turned to dust.
- Any other private data or methods you need.

Blobber Vitality

A Blobber's happiness level is a measure of how close its current volume is compared to its original volume. It is 100% when the volumes are exactly the same. It cannot survive when the current volume is +/- 10% of its original volume. This means a Blobber can only exist if its happiness is from 90% to 110%.

A Blobber loses 0.2% of its **original** radius every second. If its volume is ever greater or less than 10% of its **original** volume, then it turns to dust. The vitality should be checked and displayed right after the menu. It should also be checked right after a menu selection is made. In either case, if the Blobber has turned to dust, an appropriate message should be displayed and the program should end. See the starter code for details.

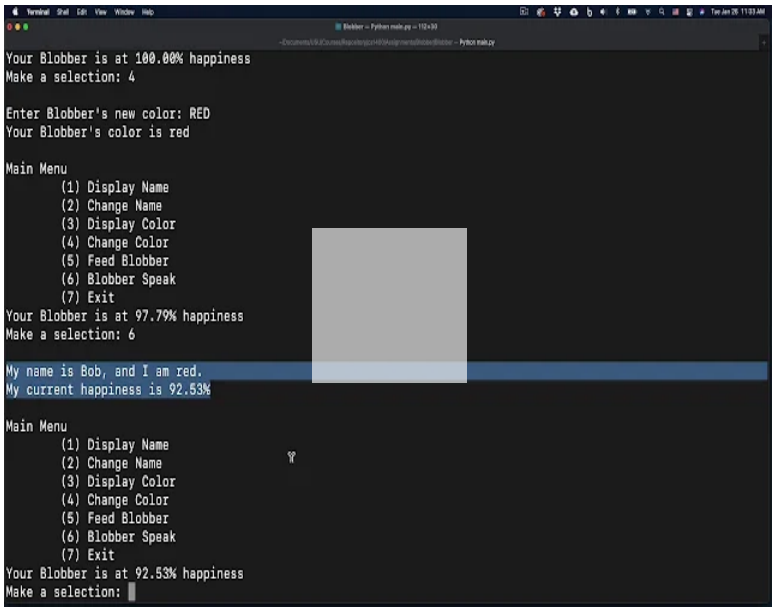
Main Menu

(1): Display Name
(2): Change Name
(3): Display Color
(4): Change Color
(5): Feed Blobber
(6): Blobber Speak
(7): Exit

You will need to decide on some submenus for some of the choices. Use blank lines in your output so it's easy for the user to see what is happening.

Video Demo

Sorry, there is a bug in my demo. The calculation of the happiness level is off. I believe I had the radius and height variables backwards. Double check your formula and go with your calculation.



Timer Help

This is not really part of the assignment, but may help you figure out how to manage the changing of the Blobber's vitality.

FakeTimer.zip (https://usu.instructure.com/courses/681553/files/81729320/download?download_frd=1)

Design Plan

Your Software Design is getting more and more complicated. Make your plan for blobber.py, including UML. Within the plan create a design for each function/method. As you break down the problem into individual pieces, calling a function/method you design counts as an individual piece since it will be designed as well. This means the design of one function/method can include a function call to another function/method. Make sure to include UML for the Blobber class.

Common Errors

This is the most difficult task to this point of this semester. It is meant to be very challenging. The most common error is not breaking the program down into small pieces. Complete one menu item at a time.

Points	100
Submitting	a text entry box or a file upload
File Types	zip

Due	For	Available from	Until
Nov 5, 2021	Everyone	-	-

Unit 5 (1)

Criteria	Ratings		Pts
Task1: main() function Each fill-in section properly completed with a single line or partial line of code	**5 pts** **Full Marks**	**0 pts** **No Marks**	5 pts
Task1: Face Class fill in sections Each fill in sections properly completed with a single line of code	**7 pts** **Full Marks**	**0 pts** **No Marks**	7 pts
Task1: Face Class methods Appropriate methods added to Face class	**6 pts** **Full Marks**	**0 pts** **No Marks**	6 pts
Task1: Initial face Initial face draws properly	**3 pts** **Full Marks**	**0 pts** **No Marks**	3 pts
Task1: Modify face Face is modified properly for each menu item	**8 pts** **Full Marks**	**0 pts** **No Marks**	8 pts
Task2: Software Design Design plan for Blobber class. Include UML. Each method should have a design. No plan is needed for other files associated with the task.	**5 pts** **Full Marks**	**0 pts** **No Marks**	5 pts
Task2: starter code starter code not modified other than appropriately adding <Fill-In> portions	**6 pts** **Full Marks**	**0 pts** **No Marks**	6 pts
Task2: Blobber Class Blobber class in its own file named blobber.py. Located in the appropriate location according to the starter file.	**5 pts** **Full Marks**	**0 pts** **No Marks**	5 pts
Task2: Blobber Constructor Use parameters appropriately. Properly handle required casing of name and color.	**7 pts** **Full Marks**	**0 pts** **No Marks**	7 pts
Task2: Blobber vitalsOk Method properly implemented and returns required data	**15 pts** **Full Marks**	**0 pts** **No Marks**	15 pts
Task2: Name and Color Name and color can be displayed and modified properly. Proper casing is maintained	**9 pts** **Full Marks**	**0 pts** **No Marks**	9 pts

Criteria	Ratings		Pts
Task2: Feed Blobber Feeding Blobber works properly. The new radius value must be accurate taking into consideration the periodic reduction of the radius.	**16 pts** **Full Marks**	**0 pts** **No Marks**	16 pts
Task2: Blobber Speak This method should only return a string, it should not print anything. Must be a complete two-line statement and include name, color, and happiness level in output.	**8 pts** **Full Marks**	**0 pts** **No Marks**	8 pts
			Total Points: 100