Numerology Inheritance – Properties and Decorators

Since you wrote the Numerology class code some things have transpired…

You have learned how to change traditional **"getters"** and **"setters"** into Python properties using the **property()** function.

Then you have learned how to change traditional **"getters"** and **"setters"** into Python properties using the @**property()** function as a decorator.

And…

Your client was thrilled with the Numerology calculator you created. You collected your consulting fee while learning a lot about Python and a little bit about numerology. It was a win-win for everyone!

A few months later the client contacts you and wishes to add a new feature to your class. But you decided you don't want to change the existing class because it has been marketed and sold around the world. You conclude the best way to implement the change is to inherit from the original object and add new logic.

So Now….
You will now have to remake your class to use properties and inheritance!

You will have to code TWO Python source files:

1. Copy the **Numerology.py** from the previous assignment into a file called **NumerologyLifePathDetails.py.** Add in these requirements:

    1. Rewrite Numerology's calls so ALL the "getters" to become properties using Python's decorators:
        - getName() – returns the subjects name should now be called **Name**
        - getBirthdate() – returns the subjects Date of Birth should now be called **Birthdate**
        - getAttitude() – returns the computed attitude number should now be called **Attitude**
        - getBirthDay() – returns the computed birthday number should now be called **BirthDay**
        - getLifePath() – returns the computed life path number should now be called **LifePath**
        - getPersonality() – returns the computed personality number should now be called **Personality**
        - getPowerName () – returns the computed power name number should now be called **PowerName**
        - getSoul() – returns the computed soul number should now be called **Soul**
    2. This class should inherit from the Numerology.py class. You will need to put both classes in the same directory for this to work.
    3. Create a __init__ that accepts two String objects: Name and Date of Birth. You must do one very important thing in this new constructor?
    4. Add a new function called **getLifePathDescription()** that accepts nothing but will return a String object. And then decorator it to become a **LifePathDescription** property. In this function you will invoke the **getLifePath()** function that is coded in the parent class. Evaluate the Life Path return integer and return one of the strings below.

        Note: To get maximum points instead of using Python if/elif/else statements code of the other Python's language elements we learned about this semester.
        - Is a 1 return: The Independent: Wants to work/think for themselves
        - Is a 2 return: The Mediator: Avoids conflict and wants love and harmony
        - Is a 3 return: The Performer: Likes music, art and to perform or get attention
        - Is a 4 return: The Teacher/Truth Seeker: Is meant to be a teacher or mentor and is truthful
        - Is a 5 return: The Adventurer: Likes to travel and meet others, often a extrovert
        - Is a 6 return: The Inner Child: Is meant to be a parent and/or one that is young at heart
        - Is a 7 return: The Naturalist: Enjoy nature and water and alternative life paths, open to spirituality

- Is a 8 return: The Executive: Gravitates to money and power
- Is a 9 return: The Humanitarian: Helps others and/or experiences pain and learns the hard way

2. Copy the previous assignment's **Use Numerology** to a new file called **Use NumerologyInherit.py** that will prompt for the user name and birthday and then use the **NumerologyLifePathDetails** class to get the computed numbers:
   - Revise all references to the **"getters"** to use the newly decorated properties done above.
   - The two things that are needed to perform a reading is a person's birth date in mm-dd-yyyy format and birth name. The inputted date should be tested to verify that it is entered as a full 8 digit date with dashes or slashes (- or /) as separators.
   - The inputted name should be populated and not empty.
   - Call the **NumerologyLifePathDetails**' __init__ and each function to get the calculated results
   - Output each of the findings to the screen using the Properties (NOT the "getters").

Grading Rubric

Criteria	Meets (100%)	Somewhat (50%)	Not Present (0%)
Two source files are present: **NumerologyLifePathDetails.py with the class coded as per the instructions.** **Use NumerologyInherit.py that prompts for input and uses the NumerologyLifePathDetails class.** 20 points	Two source files are present and coded to match the instructions.	Two source files are present and somewhat coded to match the instructions.	One source file is present with no class specifications.
Numerology Calculations are correct and outputted 20 points	All calculations are correct.	Most of the calculation are correct.	None of the calculations are correct.
Numerology class is coded efficiently and concise with public and private functions (encapsulation) and property decorations. 30 points	The class is coded with no redundancy and private and public functions are correctly coded. Decorated Properties are present.	The class is coded with some redundancy and/or private and public functions are incorrectly coded. Decorated Properties are attempted.	The class is coded with redundancy and no private and public functions differentiation. No Decorated Properties are present.

Coding of the Life Path return value: 30 points	Coded something other than Python's if/elif/else statements to return the string value.	Coded Python if/elif/else statements to return the string value.	No logic to support the return of the Life Path Details.