Lesson 1: Week 7

Created: 2024-05-27T01:25:48.687891+10:00

Exercise 1: A circle class

Write a program in which you define a class Circle, to facilitate working with circles, and then illustrate its use.Your Circle class should:Have an __init__ special method that allows you to create a Circle object by supplying a number which is the radius of the circle. This should be stored in an attribute radius. The method should check that the supplied value is a valid radius (i.e. a non-negative float), and deal with invalid values appropriately.Have a __str__ special method that allows you to print a Circle object in an informative way (for example, if the radius is 5 it returns Circle of radius 5).Have an instance method called circumference that returns the circumference of a Circle object.Have an instance method called area that returns the area of a Circle object.Have __eq__, __lt__, __le__, __gt__, and __ge__ special methods that allow you to compare two Circle objects. You should compare them by radius, so that two Circle objects are equal when they have the same radius; one Circle object is less than another Circle object when it has a smaller radius; and so on.Make appropriate use of docstrings.To illustrate the use of your class, here is the kind of code you might add to your program, after your class definition:# Inspect the documentation print(Circle.__doc__) # Output: A class to facilitate working with circles print(Circle.circumference.__doc__) # Output: Returns the circumference of the circle print(Circle.area.__doc__) # Output: Returns the area of the circle # Create some Circle objects circle_1 = Circle(4) circle_2 = Circle(5.3) circle_3 = Circle(-2) # Output: an appropriate error message circle_4 = Circle('a') # Output: an appropriate error message # Print them print(circle_1) # Output: Circle of radius 4 print(circle_2) # Output: Circle of radius 5.3 # Get their circumference and area print(circle_1.circumference()) # Output: 25.13 print(circle_1.area()) # Output: 50.27 print(circle_2.circumference()) # Output: 33.3 print(circle_2.area()) # Output: 88.25 # Compare them print(circle_1 = circle_2) # Output: False print(circle_1 > circle_2) # Output: False As a guide, use the Length class that was defined in Week 5 content as an illustrative example.

Exercise 2: A text class

Write a program in which you define a class Text, to facilitate working with pieces of text.Define your class such that:Each instance of Text has an attribute words which holds the words of the text in an array, a method num_words which returns the number of words in the text, a method num_chars which returns the total number of characters in the words, and a method word_length which returns the average number of characters per word (as a float rounded to one decimal place).Get your program to ask the user for a piece of text, and then, using the class you have defined, tell the user the number of words in the text, the number of characters in the text, and the average number of characters per word in the text.Example:Please enter your text: The quick brown fox jumped over the lazy dog. Number of words: 9 Number of characters: 36 Average word length: 4.0 Note: you might find it helpful to have your class clean the text before splitting it into words, in the way that you did in the Week 3 Exercise 1: Word frequency practice exercise.

Exercise 3: A location class

Sometimes we work with data about things that have a location, given as a latitude and a longitude. Define a class to help us work with locations.Give a location two data attributes: lat, and lon. Give also location the following three method attributes:hemisphere(), which returns which hemisphere the location is in:Northern Hemisphere if lat > 0Southern Hemisphere if lat Equator if lat = 0zone(), which returns which zone the location is in, defined by the tropics and the arctic/antarctic circles:North Frigid Zone if lat >= 66.57North Temperate Zone if lat >= 23.43Tropical Zone if lat >= -23.43South Temperate Zone if lat >= -66.57South Frigid Zone if lat direction_to(), which returns the direction to another location:North if lat South if lat > another latEast if lon West if lon > another lonAdd some error checking. For instance, latitudes must be between -90 and 90, and longitudes must be between -180 and 180.To illustrate the use of your class, here is the kind of code you might add to your program, after your class definition:sydney = Location(-33.87, 151.21) wellington = Location(-41.28, 174.77) hobart = Location(-42.88, 147.33) stockholm = Location(59.33, 18.07) print(f"Sydney is in the {sydney.zone()}") # Sydney is in the South Temperate Zone print(f"Stockholm is in the {stockholm.zone()}") # Stockholm is in the North Temperate Zone print(f"To get from Sydney to Stockholm you need to travel {sydney.direction_to(stockholm)}") # To get from Sydney to Stockholm you need to travel North West print(f"To get from Hobart to Wellington you

need to travel {hobart.direction_to(wellington)}") # To get from Hobart to Wellington you need to travel North East nowhere = Location(-200, 151.21) # The output is shown below # Traceback (most recent call last): # File "/home/main.py", line 41, in # nowhere = Location(-200, 151.21) # ^^^^^^^^^^^^^^^^^^^^^ # File "/home/main.py", line 7, in __init__ # raise Exception("Invalid coordinates") # Exception: Invalid coordinates

Exercise 4: A temperature module

There are many different units in which temperature can be measured. Three of the most common are Celsius, Fahrenheit, and Kelvin (used a lot in science).Your task is to define a Temperature class, which we can use to more easily convert temperatures from one scale to another, and to compare temperatures that are on different scales.The conversions between the scales go as follows:Converting to Celsius: Celsius = (Fahrenheit - 32) × 5/9 Celsius = Kelvin - 273.15 Converting from Celsius: Fahrenheit = (9/5 x Celsius) + 32 Kelvin = Celsius + 273.15 Your Temperature class should:Have an __init__ special method that allows you to create a Temperature object by supplying a number and a unit: either Celsius (C), Fahrenheit (F), or Kelvin (K).Have a __str__ special method that allows you to print a Temperature object in an informative way.Have an instance method called to that has a parameter for a unit, an optional parameter for a number of decimal places, and returns the temperature of the instance in the given unit, rounded to the given number of decimal places, if any were given, otherwise not rounded.Have __eq__, __lt__, __le__, __gt__, and __ge__ special methods that allow you to compare two Temperature objects.Make appropriate use of docstrings.Save your class in a module called temperature.py. Import this module into code.py, and add some code to code.py that illustrates the use of your Temperature class.Example:Here is the kind of code you might use in code.py to illustrate your class:# Create Temperature objects temp_1 = Temperature(32, 'C') temp_2 = Temperature(100, 'F') temp_3 = Temperature(324, 'K') # Print them print(temp_1) # Outputs Temperature: 32C print(temp_2) # Outputs Temperature: 100F print(temp_3) # Outputs Temperature: 324K # Convert them print(temp_1.to('F')) # Outputs 89.6 print(temp_2.to('K', 3)) # Outputs 310.928 print(temp_3.to('C', 1)) # Outputs 50.9 # Compare them print(temp_1 == temp_2) # Outputs False print(temp_1 temp_2) # Outputs False print(temp_1 >= temp_2) # Outputs False As a guide, use the Length class that was defined in Week 5 content as an illustrative example.

Exercise 5: Mediants

Let two distinct reduced positive fractions $F_1=\frac{p_1}{q_1}$ and $F_2=\frac{p_2}{q_2}$ be given, with the denominator set to 1 in case the fraction is 0. The mediant of F_1 and F_2 is defined as $\frac{p_1+p_2}{q_1+q_2}$; it is also in reduced form, and sits between F_1 and F_2. Let a reduced fraction $F=\frac{p}{q}$ in $(0,1)$ be given. It can be shown that starting with $\frac{0}{1}$ and $\frac{1}{1}$, one can compute a finite number of mediants and eventually generate F. More precisely, there exists $n\in\mathbf N$ and a sequence of pairs of fractions $(F_1^i,F_2^i)_{i\leq n}$ such that:$F_1^0=\frac{0}{1}$ and $F_2^0=\frac{1}{1}$;for all iF$ is the mediant of F_1^n and F_2^n.The program mediants.py defines a function mediants_to() that given as arguments two strictly positive integers p and q with $pmediants_to() with your code, possibly defining other functions.