Python Coding Skills Assessment

Thank you for your interest in joining Sizzle. As part of the interview and assessment process, we'd like to assess your Python skills with the following questions. We expect the test should take no longer than an hour of your time. Feel free to send us a python file with the code you write, or just copy/paste your code into an email and send it back to us. You may answer the questions in any way you feel is appropriate, whether that's full, runnable code; or pseudo-code explaining the algorithm; or a simple discussion of what your approach would be along with the pro's/con's of alternative approaches.

1. Object-Oriented Programming Principles

For these questions, we realize there are multiple approaches and solutions that will give the results we're asking for. Feel free to choose whichever approach you think is the best in terms of code structure, readability, and maintainability. If you wish, you may include your reasons for choosing your approach over the alternative approaches.

1.1

Please write a set of classes to model vehicles. Each vehicle has the following properties:

starting_position -> (where does the vehicle start its journey) (in miles)

speed -> (how fast is it moving) (in miles per hour)

time -> (how much time it has been moving) (in hours)

Additionally, vehicles have the following methods:

Vehicle.current_position -> current position based on the above properties (starting_position + speed * time)

vehicle1.distance(vehicle2) -> how much distance there is between the two vehicles (in miles)

Please write classes for Car(), Truck(), and Bicycle(). You may use inheritance, mixins, etc. as needed. You may structure the parameters / methods / variables any way you wish to enable the following code to run:

```
car = Car()
print(car.starting_position)
--> 10
car = Car(starting_position=0, time=10, speed=50)
print(car.starting_position)
--> 0
```

```
truck = Truck()
print(truck.starting_position)
-->100
truck.starting_position = 50
truck.time = 10
truck.speed = 40

print(car.distance(truck))
--> 50
Please show us your code for these class definitions.
```

1.2

Next, instead of all classes using mph (miles per hour) as the unit for speed, we'd like to use the following units:

Cars: miles per hour

Trucks: kilometers per hour

Bicycles: feet per minute

The units for position is still miles for all classes, and the units for time is still hours

So for example, if car.speed = 50, that would be 50 mph, whereas if bicycle.speed = 50, that would be 50 feet per minute.

Please refactor your code so that the distance() method returns the distance between 2 vehicles regardless of what type of vehicles the 2 vehicles are. Please return the distance in miles.

1.3

Let's define a new class Airplane that has the properties speed and time (and nothing else). And a function:

```
def distance_traveled(vehicle: Car):
    return vehicle.current_position - vehicle.starting_position
```

If we pass an instance of the class Car, Truck, Bicycle, Airplane, which instance(s) would return a value, and which instance(s) would throw an error? Why? What type system does this demonstrate (e.g. static, dynamic, duck, polymorphic, union, intersection, etc)?

2. Data Structures

Python has many ways of structuring data, including dictionaries, lists, tuples, named tuples, sets, custom classes, etc. For the following questions, please create at least 2 different ways of structuring the data, and then explain which structure you'd choose and why. You can create multiple objects (e.g. multiple classes / subclasses / dicts / tuples, etc.) but they eventually should all be combined into one object that contains all of the data.

2.1

A car has the following data associated with it:

- Color: can be red, blue, green, or yellow (no other values are allowed)
- Weight (kg): float number
- Parts: one or more of various subcomponents, all of which are instances of the class Component or one of its subclasses (order doesn't matter, but there can be duplicates)
- Accident History: a sorted list of one or more dates on which an accident occurred (order matters, and no duplicate dates are allowed)
- License: license plate information including:
    - -State
    - -License plate number
    - -Expiration Date

Please write out the data structure you would use to represent this information.

2.2

Please convert the 'Parts' list into a linked list which stores all of the subcomponents along with a quantity that indicates how many of that subcomponent the car has. So now, there will be no duplicates, but rather a count for each subcomponent listed.

3. Functional programming

In geometry, the straight-line distance between two points on a circle is given as:

$$\sqrt{2(r^2) - 2rCos(\theta)}$$

As a function this can be written as:

```
import math
def straight_length(r,theta):
    return math.isqrt((2 * r**2) - (2 * r * math.cos(theta)))
```

Calculating the arc-length between two points on a circle can be done with this function:

```
def arc_length(r, theta):
    return (r * theta)
```

3.1

Please write a function length(l_function, r, theta) which takes 3 arguments and returns the length:

l_function is the function to use to calculate the length (either straight_length or arc_length from above)

r is radius

theta is the angle (in radians)

So for example:

```
x = straight_length
y = arc_length
r = 5
theta = 0.5
length(x, r, theta) -> will return a string with the shortest
straight distance in the format "straight length = <result>"
length(y, r, theta) -> will return a string with the shortest arc
length in the format "arc length = <result>"
```

3.2

Please convert the length function from above into a lambda function.

4. Error Handling

4.1

Please rewrite the straight_length() function from section 3 to raise a custom exception when the value of theta is > 2 * pi, and a different custom exception when the value of theta is < 0

4.2

Please rewrite the length() function from section 3.1 to handle these 2 errors:

4.2.1 When the first error occurs (theta > 2 * pi): subtract 2*pi from theta and re-try the function; keep doing this until either there is no error, or theta < 0

4.2.2 When the second error occurs (theta < 0): add 2*pi to theta, and re-try the function. If it doesn't work the second time, raise an exception to the calling function