Object Oriented Programming

In this exercise you will primarily be exploring a Pokemon dataset. Pokemon are fictional creatures from the Nintendo franchise of the same name.

Some Pokemon facts that might be useful:

- The word "pokemon" is both singular and plural. You may refer to "one pokemon" or "many pokemon".
- Pokemon have attributes such as a name, weight, and height.
- Pokemon have one or multiple "types". A type is something like "electric", "water", "ghost", or "normal" that indicates the abilities that pokemon may possess.
- The humans who collect pokemon are called "trainers".

As a pokemon trainer we want to make sure our pokemon are performing at their peak. To measure this, we want to calculate a pokemon's Body Mass Index (or BMI). This is a statistic calculated using the pokemon's height and weight.

To help with this task we we will create Pokemon objects that methods can be called on.

You'll be working with following dictionaries to create the `Pokemon` objects

```
# Run this cell without changes
bulbasaur_data = {
    "name": 'bulbasaur',
    "weight": 69,
    "height": 7,
    "base_experience": 64,
    "types": ["grass", "poison"]
}
charmander_data = {
    "name": 'charmander',
    "weight": 85,
    "height": 6,
    "base_experience": 62,
    "types": ["fire"]
}
squirtle_data = {
    "name": 'squirtle',
    "weight": 90,
    "height": 5,
    "base_experience": 63,
    "types": ["water"]
}
```

1. Creating a Class

Create a class called `Pokemon` with an `__init__` method.

Along with the necessary `self` parameter, the `__init__` method should take in `data` as a parameter.

With the idea that one of the dictionaries above will be passed in as `data`, assign these specific attributes within the `__init__` method:

- `name` : value from the 'name' key of the dictionary passed in `data`
- `weight` : value from the 'weight' key of the dictionary passed in `data`
- `height` : value from the 'height' key of the dictionary passed in `data`

```
# Create your class below with the correct syntax, including an __init__ method.

# YOUR CODE HERE
class Pokemon:
    def __init__(self, data):
        self.name = data["name"]
        self.weight = data["weight"]
        self.height = data["height"]


# PUT ALL WORK FOR THE ABOVE QUESTION ABOVE THIS CELL
# THIS UNALTERABLE CELL CONTAINS HIDDEN TESTS

# Pokemon should be a class that exists in this namespace
assert type(Pokemon) == type
```

2. Instantiating Objects

Using the `bulbasaur_data`, `charmander_data` and `squirtle_data` variables, create the corresponding pokemon objects.

```
# Replace None with appropriate code

bulbasaur = Pokemon(bulbasaur_data)
charmander = Pokemon(charmander_data)
squirtle = Pokemon(squirtle_data)

# YOUR CODE HERE
bulbasaur = Pokemon(bulbasaur_data)
charmander = Pokemon(charmander_data)
squirtle = Pokemon(squirtle_data)

# This code will test implementation.
def print_pokeinfo(pkmn):
    print('Name: ' + pkmn.name)
    print('Weight: ' + str(pkmn.weight))
    print('Height: ' + str(pkmn.height))
    print('\n')

print_pokeinfo(bulbasaur)
print_pokeinfo(charmander)
print_pokeinfo(squirtle)
```

```
Name: bulbasaur
Weight: 69
Height: 7


Name: charmander
Weight: 85
Height: 6


Name: squirtle
Weight: 90
Height: 5
```

```
# PUT ALL WORK FOR THE ABOVE QUESTION ABOVE THIS CELL
# THIS UNALTERABLE CELL CONTAINS HIDDEN TESTS

assert type(bulbasaur) == Pokemon
assert type(charmander) == Pokemon
assert type(squirtle) == Pokemon
```

3. Update an instance attribute

Using the charmander instance, increase the weight attribute by 5.

```
# YOUR CODE HERE
charmander.weight += 5
print_pokeinfo(charmander)
```

```
Name: charmander
Weight: 90
Height: 6
```

```
# PUT ALL WORK FOR THE ABOVE QUESTION ABOVE THIS CELL
# THIS UNALTERABLE CELL CONTAINS HIDDEN TESTS
```

4. Instance Methods

Re-write the class `Pokemon` below, so that it has an instance method called `bmi` that calculates the BMI of a Pokemon.

BMI is defined by the formula: $\frac{weight}{height^2}$

The BMI should be calculated with weight in **kilograms** and height in **meters**.

The height and weight data of Pokemon from the dictionaries above is in **decimeters** and **hectograms**, respectively.

You will have to convert the given values of height and weight to kilograms and meters to make the BMI calculations correct.

For your convenience, here are the conversions:

```
1 decimeter = 0.1 meters
1 hectogram = 0.1 kilograms
```

Don't forget: since you are changing the `Pokemon` class, you will have to create **new objects** of this **new class**.

If you use the objects created by the first class you wrote, they will not have the `bmi` instance!

You can assign these new objects the same names as you assigned the old ones: `bulbasaur`, `charmander` and `squirtle`

```
# Define the new Pokemon class here
class Pokemon:
    def __init__(self, data):
        self.name = data["name"]
        self.weight = data["weight"]
        self.height = data["height"]

    def bmi(self):
```

```
            # Convert weight to kilograms and height to meters
            weight_kg = self.weight * 0.1  # 1 hectogram = 0.1 kilograms
            height_m = self.height * 0.1  # 1 decimeter = 0.1 meters

            # Calculate BMI using the formula: weight / height^2
            bmi_value = weight_kg / (height_m ** 2)
            return bmi_value

    # Replace None with appropriate code
    bulbasaur = Pokemon(bulbasaur_data)
    charmander = Pokemon(charmander_data)
    squirtle = Pokemon(squirtle_data)

    # This code will test your implementation
    print(bulbasaur.bmi())
    print(charmander.bmi())
    print(squirtle.bmi())
```

```
     14.081632653061222
     23.611111111111104
     36.0
```

```
    # PUT ALL WORK FOR THE ABOVE QUESTION ABOVE THIS CELL
    # THIS UNALTERABLE CELL CONTAINS HIDDEN TESTS

    # bulbasaur.bmi should be a method on an instance of type Pokemon
    import types
    assert type(bulbasaur.bmi) == types.MethodType
    # bulbasaur.bmi() should return a floating point number
    assert type(bulbasaur.bmi()) == float
```