`Restaurant` and `Food`

Your objective is to create a set of classes that represent restaurants and their food items.

Build the `Food` class - 5 marks

This class represents a "food item" in a restaurant (= a dish you can order).

The constructor receives 1 required argument, and 2 optional arguments:

- the `name` must be provided and must be a non-empty string.
- the "type of cuisine" (or `style`) is optional. When provided, it must be a non-empty string. If not provided or provided with an invalid value, set the attribute to `None`.
- whether the food is vegetarian or not (boolean). If not provided or provided with anything else than `True`, set the attribute to `False`.

The class must implement string conversion, so that food items can easily be printed:

```
>>> pasta = Food("Cacio y Pepe", "italian", vegetarian=True)
>>> print(pasta)
Cacio y Pepe [V]
```

All the tests in the `test_food.py` file must pass.

Build the `Restaurant` class - 3 marks

This class represents a restaurant. Some code for the `Restaurant` class is provided in `restaurant.py`.

The constructor has 3 required arguments, and 1 optional argument:

- the `name` must be provided, and must be a non-empty string.
- the opening time must be provided, and it must be an integer between 0 and 24
- the closing time must be provided, and it must be an integer between 0 and 24. It must also be strictly greater than the opening time.
- the "type of cuisine", or `style`, is a string and is optional. If not provided or something else than a non-empty string is provided, set the attribute to `None`.

The restaurant must implement string conversion, so that its information can easily be printed.

```
>>> restaurant = Restaurant("Tim's restaurant", opening_time=10, closing_time=22)
>>> print(restaurant)
Tim's restaurant [10:00 - 22:00]
```

Add the `is_open_at` method. It receives an argument, and must return `True` if the restaurant is open at the time provided. For example, if `restaurant` is open from 10.00 to 22.00:

- `restaurant.is_open_at(1)` is `False`
- `restaurant.is_open_at(12)` is `True`

If the argument received is not an integer, return `False`.

Make sure all the tests in the `test_restaurant.py` file pass.

Add `Food` items to your `Restaurant` - 5 marks

Each instance of a `Restaurant` will contain `Food` items. Make sure you define an instance attribute in the constructor for your *list* of items. Then, add the following methods and any code required to your `Restaurant` class:

- `get_foods()`: returns the list of all food items available in the restaurant
- `add_food(food_item)`:
  - takes an argument. The argument must be an instance of `Food` or a child class of `Food`.
  - it adds the food item to the list of food items available in the restaurant, **ONLY IF** the food item has the same "style" as the restaurant (you cannot serve Mexican food in an Italian restaurant).
- `has_food(food_name)`:
  - takes an argument: a **STRING**
  - it returns `True` if the restaurant has any food items whose name contain the string provided.
  - otherwise, it returns `False`.
- `has_vegetarian_options()`:
  - returns `True` if there are any vegetarian food items available in the restaurant
  - returns `False` otherwise (including when there are no food items available at all)

Make sure all the tests in the `test_restaurant_food.py` file pass.

Build the `JunkFood` and `FastFood` classes - 4 marks

Use inheritance to build the `JunkFood` (inherits from `Food`) and `FastFood` classes.

A `JunkFood` instance:

- is never vegetarian
- has the "style" `junk`

A `FastFood` instance:

- is always open (from 0.00 to 24.00)
- has the "style" `junk`

Make sure all the tests in the `test_fastfood.py` file pass.

> Note: all you need to do is to write custom constructors. Keep it simple and don't rewrite code!