3) [35 points] Stock management is one of the intense business areas that require swift action from many aspects. Due to the rapid pace of the field, errors are likely to occur during inventory counting related to the storage, transportation, consolidation, and segregation of products. In this task, you are required to write a *Stock* class that can be used to prevent these errors.

- The *Stock* class is responsible for representing a stock of a specific type with the given amount. It should have the following attributes::
  - *type*: The type of the stock.
  - *amount*: The amount of the stock. If the amount is not given during the construction of the *Stock* object it should be 0 by default.
- During the initialization of the *Stock* object,
  - it should check if the given type is a string value and if it is not empty. Otherwise, it should raise an exception "`Stock type must be a string and cannot be empty.`".
  - it should check if the given amount is an integer or a float value and if it is not a negative value. Otherwise, it should raise an exception "`Stock amount must be an integer or a float and it should not be negative.`".
- The *Stock* class should overwrite the following operators:
  - **Addition operator (+):** Calculates the addition in the following manner and returns the calculated *Stock* object.
    - If the other value is another *Stock* object, it should first check if their type attribute is the same. If their type is the same, the calculated object's amount should be the sum of the amounts. Otherwise, it should raise an exception with the message "`Stock types cannot be different during the addition of two` **`class_name`** `objects.`".
    - If the other value is an integer or a float, it should first check if the other value is not negative. If the other value is not negative, the calculated object's amount should be the sum of the amount and the other value. Otherwise, it should raise an exception with the message "`Addition operator for` **`class_name`** `cannot accept negative values.`".
    - If the other value is any other type, it should raise an exception with the message "`Addition operator for` **`class_name`** `can only support` **`class_name`**`, integer, or float.`".

- **Subtraction operator (-):** Calculates the subtraction in the following manner and returns the calculated *Stock* object.
  - If the other value is another *Stock* object, it should first check if their type attribute is the same. If their type is not the same, it should raise an exception with the message "`Stock types cannot be different during the subtraction of two` **`class_name`** `objects.`". Then, it should check if this object is not less than other object. If it is less, it should raise an exception with the message "**`class_name`** `objects cannot subtract` **`class_name`** `objects with amounts greater than its amount.`". Otherwise, the calculated object's amount should be the other object's amount subtracted from this object's amount.
  - If the other value is an integer or a float, it should first check if the other value is not negative. If the other value is negative, it should raise an exception with the message "`Subtraction operator for` **`class_name`** `cannot accept negative values.`". Then, it should check if this object is not less than other value. If it is less, it should raise an exception with the message "**`class_name`** `objects cannot subtract values greater than its amount.`". Otherwise, the calculated object's amount should be the other value subtracted from this object's amount.
  - If the other value is any other type, it should raise an exception with the message "`Subtraction operator for` **`class_name`** `can only support` **`class_name`**`, integer, or float.`".
- **Multiplication operator (*):** Calculates the multiplication in the following manner and returns the calculated *Stock* object.
  - If the other value is an integer or a float, it should first check if the other value is not negative. If the other value is not negative, the calculated object's amount should be this object's amount multiplied by the other value. Otherwise, it should raise an exception with the message "`Multiplication operator for` **`class_name`** `cannot accept negative values.`".
  - If the other value is any other type, it should raise an exception with the message "`Addition operator for` **`class_name`** `can only support integer or float.`".

- **Less than operator (<):** Checks if the *Stock* object is less than the other value and returns the boolean result.
  - If the other value is another *Stock* object, it should first check if their type attribute is the same. If their type is the same, the result is a boolean value determined by if this object's amount is less than the other object's amount. Otherwise, it should raise an exception with the message "`Stock types cannot be different while comparing two class_name objects.`".
  - If the other value is an integer or a float, the result is a boolean value determined by if this object's amount is less than the other value.
  - If the other value is any other type, it should raise an exception with the message "`Less than operator for class_name can only support class_name, integer, or float.`".

- The *Stock* object should also overwrite its string value. The format of the string should be like:
  `A stock of stock_type with the amount of stock_amout`

WARNING

Please use the template *stock_management.py* provided with the exam package and modify this file to construct your solution. Note that you should NOT change the lines below
`if __name__ == "__main__":` line in the template.

WARNING

When printing the stock amounts, make sure to **print with 2 floating points** to ensure consistency with the test cases.
[i.e. `150 as 150.00` `12400.12345 as 12400.12`]

HINT

You can use the `.__class__` attribute of an object to get its class name.