Recruitment Task

Virtual columns in pandas dataframe

You have a panda DataFrame with existing data and want to create a new DataFrame that includes the original data along with an additional column calculated based on specified operations. To achieve this, implement the add_virtual_column function.

Inputs:

- df: Any pandas DataFrame.
- role: A mathematical expression defining how to compute the values for the virtual column. For example, first_column - second_column.
- new_column: The name of the new virtual column to be added.

Examples:

```
>>> print(fruits_sales)
   name   quantity  price
0  banana    10      10
1  apple      3       1
```

```
>>> sales_total=add_virtual_column(fruits_sales, "quantity * price", "total")
>>> print(sales_total)
   name   quantity  price  price_total
0  banana   10        10        100
1  apple     3         1          3
```

Function Signature:

```
import pandas

def add_virtual_column(df: pandas.DataFrame, role: str, new_column: str) ->
pandas.DataFrame:
   return pandas.DataFrame([])
```

Validations:

- Column labels must consist only of letters and underscores (_).
- The function must support basic operations: addition (+), subtraction (-), and multiplication (*).
- If the `role` or any column label is incorrect, the function should return an empty DataFrame.

Sample Unit Tests (Passing them doesn't necessarily mean the solution is correct):

```python
import pandas as pd
from solution import add_virtual_column


def test_sum_of_two_columns():
    df = pd.DataFrame([[1, 1]] * 2, columns = ["label_one", "label_two"])
    df_expected = pd.DataFrame([[1, 1, 2]] * 2, columns = ["label_one",
"label_two", "label_three"])
    df_result = add_virtual_column(df, "label_one+label_two", "label_three")
    assert df_result.equals(df_expected), f"The function should sum the
columns: label_one and
label_two.\n\nResult:\n\n{df_result}\n\nExpected:\n\n{df_expected}"


def test_multiplication_of_two_columns():
    df = pd.DataFrame([[1, 1]] * 2, columns = ["label_one", "label_two"])
    df_expected = pd.DataFrame([[1, 1, 1]] * 2, columns = ["label_one",
"label_two", "label_three"])
    df_result = add_virtual_column(df, "label_one * label_two", "label_three")
    assert df_result.equals(df_expected), f"The function should multiply the
columns: label_one and
label_two.\n\nResult:\n\n{df_result}\n\nExpected:\n\n{df_expected}"


def test_subtraction_of_two_columns():
    df = pd.DataFrame([[1, 1]] * 2, columns = ["label_one", "label_two"])
    df_expected = pd.DataFrame([[1, 1, 0]] * 2, columns = ["label_one",
"label_two", "label_three"])
    df_result = add_virtual_column(df, "label_one - label_two", "label_three")
    assert df_result.equals(df_expected), f"The function should subtract the
columns: label_one and
label_two.\n\nResult:\n\n{df_result}\n\nExpected:\n\n{df_expected}"


def test_empty_result_when_invalid_labels():
    df = pd.DataFrame([[1, 2]] * 3, columns = ["label_one", "label_two"])
    df_result = add_virtual_column(df, "label_one + label_two", "label3")
    assert df_result.empty, f"Should return an empty df when the
\"new_column\" is invalid.\n\nResult:\n\n{df_result}\n\nExpected:\n\nEmpty df"
    df = pd.DataFrame([[1, 2]] * 3, columns = ["label-one", "label_two"])
    df_result = add_virtual_column(df, "label-one + label_two", "label")
    assert df_result.empty, f"Should return an empty df when both df columns
and roles are invalid.\n\nResult:\n\n{df_result}\n\nExpected:\n\nEmpty df"
    df = pd.DataFrame([[1, 2]] * 3, columns = ["label-one", "label_two"])
    df_result = add_virtual_column(df, "label_one + label_two", "label")
    assert df_result.empty, f"Should return an empty df when a df column is
invalid.\n\nResult:\n\n{df_result}\n\nExpected:\n\nEmpty df"
```

```python
def test_empty_result_when_invalid_rules():
    df = pd.DataFrame([[1, 1]] * 2, columns = ["label_one", "label_two"])
    df_result = add_virtual_column(df, "label_one \ label_two", "label_three")
    assert df_result.empty, f"Should return an empty df when the role have
invalid character: '\\'.\n\nResult:\n\n{df_result}\n\nExpected:\n\nEmpty df"
    df_result = add_virtual_column(df, "label&one + label_two", "label_three")
    assert df_result.empty, f"Should return an empty df when the role have
invalid character: '&'.\n\nResult:\n\n{df_result}\n\nExpected:\n\nEmpty df"
    df_result = add_virtual_column(df, "label_five + label_two",
"label_three")
    assert df_result.empty, f"Should return an empty df when the role have a
column which isn't in the df:
'label_five'.\n\nResult:\n\n{df_result}\n\nExpected:\n\nEmpty df"


def test_when_extra_spaces_in_rules():
    df = pd.DataFrame([[1, 1]] * 2, columns = ["label_one", "label_two"])
    df_expected = pd.DataFrame([[1, 1, 2]] * 2, columns = ["label_one",
"label_two", "label_three"])
    df_result = add_virtual_column(df, "label_one+label_two", "label_three")
    assert df_result.equals(df_expected), f"Should work when the role haven't
spaces between the operation and the
column.\n\nResult:\n\n{df_result}\n\nExpected:\n\n{df_expected}"
    df_result = add_virtual_column(df, "label_one + label_two ",
"label_three")
    assert df_result.equals(df_expected), f"Should work when the role have
spaces between the operation and the
column.\n\nResult:\n\n{df_result}\n\nExpected:\n\n{df_expected}"
    df_result = add_virtual_column(df, "  label_one + label_two ",
"label_three")
    assert df_result.equals(df_expected), f"Should work when the role have
extra spaces in the
start/end.\n\nResult:\n\n{df_result}\n\nExpected:\n\n{df_expected}"
```