Budget App

You will be working on this project with our Replit starter code (https://replit.com/github/freeCodeCamp/boilerplate-budget-app).

Complete the `Category` class in `budget.py`. It should be able to instantiate objects based on different budget categories like *food*, *clothing*, and *entertainment*. When objects are created, they are passed in the name of the category. The class should have an instance variable called `ledger` that is a list. The class should also contain the following methods:

- A `deposit` method that accepts an amount and description. If no description is given, it should default to an empty string. The method should append an object to the ledger list in the form of `{"amount": amount, "description": description}`.
- A `withdraw` method that is similar to the `deposit` method, but the amount passed in should be stored in the ledger as a negative number. If there are not enough funds, nothing should be added to the ledger. This method should return `True` if the withdrawal took place, and `False` otherwise.
- A `get_balance` method that returns the current balance of the budget category based on the deposits and withdrawals that have occurred.
- A `transfer` method that accepts an amount and another budget category as arguments. The method should add a withdrawal with the amount and the description "Transfer to [Destination Budget Category]". The method should then add a deposit to the other budget category with the amount and the description "Transfer from [Source Budget Category]". If there are not enough funds, nothing should be added to either ledgers. This method should return `True` if the transfer took place, and `False` otherwise.
- A `check_funds` method that accepts an amount as an argument. It returns `False` if the amount is greater than the balance of the budget category and returns `True` otherwise. This method should be used by both the `withdraw` method and `transfer` method.

When the budget object is printed it should display:

- A title line of 30 characters where the name of the category is centered in a line of `*` characters.
- A list of the items in the ledger. Each line should show the description and amount. The first 23 characters of the description should be displayed, then the amount. The amount should be right aligned, contain two decimal places, and display a maximum of 7 characters.
- A line displaying the category total.

Here is an example of the output:

```
*************Food*************
initial deposit        1000.00
groceries               -10.15
restaurant and more foo -15.89
Transfer to Clothing    -50.00
Total: 923.96
```

Besides the `Category` class, create a function (outside of the class) called `create_spend_chart` that takes a list of categories as an argument. It should return a string that is a bar chart.

The chart should show the percentage spent in each category passed in to the function. The percentage spent should be calculated only with withdrawals and not with deposits. Down the left side of the chart should be labels 0 - 100. The "bars" in the bar chart should be made out of the "o" character. The height of each bar should be rounded down to the nearest 10. The horizontal line below the bars should go two spaces past the final bar. Each category name should be written vertically below the bar. There should be a title at the top that says "Percentage spent by category".

This function will be tested with up to four categories.

Look at the example output below very closely and make sure the spacing of the output matches the example exactly.

```
Percentage spent by category
100|
 90|
 80|
 70|
 60| o
 50| o
 40| o
 30| o
 20| o  o
 10| o  o  o
  0| o  o  o
    ----------
     F  C  A
     o  l  u
     o  o  t
     d  t  o
        h
        i
        n
        g
```

The unit tests for this project are in `test_module.py`.

Development

Write your code in `budget.py`. For development, you can use `main.py` to test your `Category` class. Click the "run" button and `main.py` will run.

Testing

We imported the tests from `test_module.py` to `main.py` for your convenience. The tests will run automatically whenever you hit the "run" button.

Submitting

Copy your project's URL and submit it to freeCodeCamp.

Solution Link (/learn) Menu

ex: https://replit.com/@camperbot/hello

I've completed this challenge

Get a Hint (https://forum.freecodecamp.org/t/462361)

Ask for Help