Assignment 3.2 Practice Problem 2 (Split the Bill)

```
class Splitwise():

    def __init__(self, n, m):
        self.n = n # declaring the value of n to be assigned at self.n
        self.m = m # declaring the value of n to be assigned at self.m
        self.balance = [0 for i in range(n)] # creating a list that contains 0s depending on the number of members(n)
        self.transaction_bal_list = [] # creating an empty list for the list of balances during the transaction
        self.j = 0 # a variable to be used at indexing later

    def balance_compute(self): # a 5function so that we can identify who pays and who will split
        for transact in range(self.m): # starting a loop that will process the code below depending on the value of m
            print("- - - - - - - - - - - - - - - ")
            transaction_id = input("Enter Transaction id: ") # naming the transaction
            no_pay,no_split = map(int,input("\nEnter how many will pay and how many will split\n").split()) # allowing the user to input how r
                                                                                                           # will split for the payment
            print("\nEnter the Person ID and the amount to pay separated by whitespace")
            for i in range(no_pay): # running the function inside the loop depending on the value of no_pay
                person_id,amount = map(int,input().split()) # entering the id of the person and the amount that they will pay
                self.balance[person_id - 1] -= amount # setting the new value of balance by subtracting the amount input by the user

            print("\nEnter the Person ID and the amount to be split separated by whitespace")
            for i in range(no_split): # running the function inside the loop depending on the value of no_pay
                person_id, amount = map(int,input().split()) # entering the id of the person who will split the money
                self.balance[person_id - 1] += amount # adding it to their current balance

    def balance_identify(self): # a function to compute the balance of the user and who needs to pay who and how much
        for i in range(self.n): # running the code inside the loop up until the last value of n
            if(self.balance[i] > 0): # condition that if the value inside the balance[i] is greater than 0, the code will run
                current_balance = self.balance[i] # creating a variable cur_bal and assigning it with the value the same as the balance[i]
                while current_balance > 0 and self.j < self.n: # conditional loop that if the cur_bal is not negative it will run the followi
                    if(self.balance[self.j] >= 0): # conditional statement that if the cur_bal is not negative it will run the following code
                        self.j = self.j+1; # adding 1 to j so that it can check the next index of the balance
                        continue; # skipping the loop in the current running iteration
                    minimum_balance = min(current_balance, abs(self.balance[self.j])) # comparing the value between cur_bal and abs(balance[j
                    current_balance -= minimum_balance # changing the value of cur_balance by subtraticting the minimum_balance to it
                    self.balance[self.j] += minimum_balance # chaging the value of the balance[j] by adding minimum_bal to it
                    self.transaction_bal_list.append([[i+1, self.j+1], minimum_balance]) # appending the value of the index inside the answer
                                                                                        # adding 1 to it since our index starts with 0 and ou

    def print_status(self): # function so that we can display the balance of the members of the group
        print("- - - - - - - - - - - - - - - ")
        for i in range(len(self.transaction_bal_list)):
            print("Person", self.transaction_bal_list[i][0][0], "needs to give", "Person", self.transaction_bal_list[i][0][1], self.transacti



n,m = map(int,input("Enter the size of members and number of transaction separated by white space: ").split())
SW = Splitwise(n,m)
SW.balance_compute()
SW.balance_identify()
SW.print_status()
```

```
Enter the size of members and number of transaction separated by white space: 4 1
- - - - - - - - - - - - - - - 
Enter Transaction id: transaction1

Enter how many will pay and how many will split
1 4

Enter the Person ID and the amount to pay separated by whitespace
1 100

Enter the Person ID and the amount to be split separated by whitespace
2 25
3 25
4 25
1 25
- - - - - - - - - - - - - - - 
Person 2 needs to give Person 1 25 PHP
Person 3 needs to give Person 1 25 PHP
Person 4 needs to give Person 1 25 PHP
```

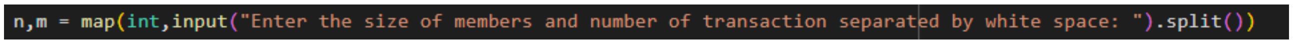
```
class Splitwise():

    def __init__(self, n, m):
        self.n = n # declaring the value of n to be assigned at self.n
        self.m = m # declaring the value of n to be assigned at self.m
        self.balance = [0 for i in range(n)] # creating a list that contains 0s depending on the number of members(n)
        self.transaction_bal_list = [] # creating an empty list for the list of balances during the transaction
        self.j = 0 # a variable to be used at indexing later

    def balance_compute(self): # a 5function so that we can identify who pays and who will split
        for transact in range(self.m): # starting a loop that will process the code below depending on the value of m
            print("- - - - - - - - - - - - - - - ")
            transaction_id = input("Enter Transaction id: ") # naming the transaction
            no_pay,no_split = map(int,input("\nEnter how many will pay and how many will split\n").split()) # allowing the user to input how
                                                                                                       # will split for the payment
            print("\nEnter the Person ID and the amount to pay separated by whitespace")
            for i in range(no_pay): # running the function inside the loop depending on the value of no_pay
                person_id,amount = map(int,input().split()) # entering the id of the person and the amount that they will pay
                self.balance[person_id - 1] -= amount # setting the new value of balance by subtracting the amount input by the user

            print("\nEnter the Person ID and the amount to be split separated by whitespace")
            for i in range(no_split): # running the function inside the loop depending on the value of no_pay
                person_id, amount = map(int,input().split()) # entering the id of the person who will split the money
                self.balance[person_id - 1] += amount # adding it to their current balance

    def balance_identify(self): # a function to compute the balance of the user and who needs to pay who and how much
        for i in range(self.n): # running the code inside the loop up until the last value of n
            if(self.balance[i] > 0): # condition that if the value inside the balance[i] is greater than 0, the code will run
                current_balance = self.balance[i] # creating a variable cur_bal and assigning it with the value the same as the balance[i]
                while current_balance > 0 and self.j < self.n: # conditional loop that if the cur_bal is not negative it will run the follow
                    if(self.balance[self.j] >= 0): # conditional statement that if the cur_bal is not negative it will run the following cod
                        self.j = self.j+1; # adding 1 to j so that it can check the next index of the balance
                        continue; # skipping the loop in the current running iteration
                    minimum_balance = min(current_balance, abs(self.balance[self.j])) # comparing the value between cur_bal and abs(balance[
                    current_balance -= minimum_balance # changing the value of cur_balance by subtraticting the minimum_balance to it
                    self.balance[self.j] += minimum_balance # chaging the value of the balance[j] by adding minimum_bal to it
                    self.transaction_bal_list.append([[i+1, self.j+1], minimum_balance]) # appending the value of the index inside the answe
                                                                                          # adding 1 to it since our index starts with 0 and

    def print_status(self): # function so that we can display the balance of the members of the group
        print("- - - - - - - - - - - - - - - ")
        for i in range(len(self.transaction_bal_list)):
            print("Person", self.transaction_bal_list[i][0][0], "needs to give", "Person", self.transaction_bal_list[i][0][1], self.transact



n,m = map(int,input("Enter the size of members and number of transaction separated by white space: ").split())
SW = Splitwise(n,m)
SW.balance_compute()
SW.balance_identify()
SW.print_status()
```

```
Enter Transaction id: transaction2

Enter how many will pay and how many will split
1 4

Enter the Person ID and the amount to pay separated by whitespace
4 100

Enter the Person ID and the amount to be split separated by whitespace
1 25
2 25
```

```
Enter Transaction id: transaction4

Enter how many will pay and how many will split
1 3

Enter the Person ID and the amount to pay separated by whitespace
2 150

Enter the Person ID and the amount to be split separated by whitespace
1 50
2 50
3 50
- - - - - - - - - - - - - - -
Enter Transaction id: transaction5

Enter how many will pay and how many will split
2 2

Enter the Person ID and the amount to pay separated by whitespace
5 13
6 25

Enter the Person ID and the amount to be split separated by whitespace
4 25
1 13
- - - - - - - - - - - - - - -
Person 1 needs to give Person 2 75 PHP
Person 1 needs to give Person 4 13 PHP
Person 3 needs to give Person 4 12 PHP
Person 3 needs to give Person 5 18 PHP
Person 3 needs to give Person 6 20 PHP
```

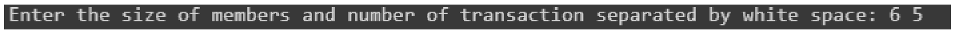
CODE BREAKDOWN

This line of the code, allows us to input 2 different variables and use it in a function. The n variable is the number of members inside the group and m is the number of transactions to be made.



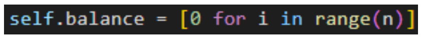
```
n,m = map(int,input("Enter the size of members and number of transaction separated by white space: ").split())
```

If we run the code and there two different integers (for example 6 and 5), it means that the group have 6 members and 5 transactions are to be made.



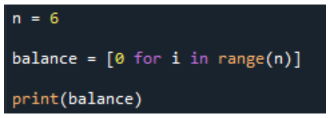
```
Enter the size of members and number of transaction separated by white space: 6 5
```

This line of code will create a list containing 0s depending on the value of n

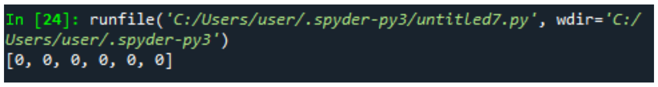


```
self.balance = [0 for i in range(n)]
```

For example:

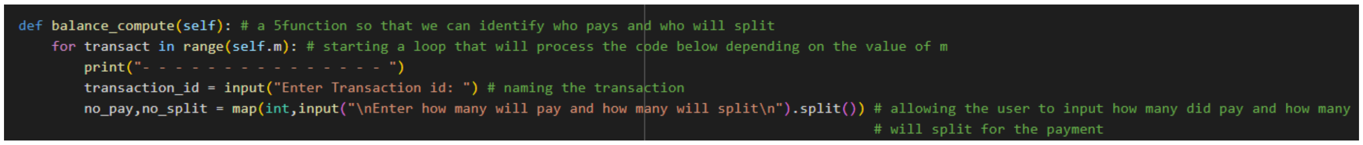


```
n = 6

balance = [0 for i in range(n)]

print(balance)
```

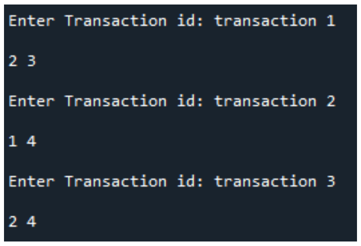


```
In [24]: runfile('C:/Users/user/.spyder-py3/untitled7.py', wdir='C:/
Users/user/.spyder-py3')
[0, 0, 0, 0, 0, 0]
```

This block of code allows us to run the code inside the loop depending on the value of our m. For example, the value of our m is 3, it means that we will be able to input 3 transactions, and we can input the no_pay and no_splits 3 times



```
def balance_compute(self): # a 5function so that we can identify who pays and who will split
    for transact in range(self.m): # starting a loop that will process the code below depending on the value of m
        print("- - - - - - - - - - - - - - - - -")
        transaction_id = input("Enter Transaction id: ") # naming the transaction
        no_pay,no_split = map(int,input("\nEnter how many will pay and how many will split\n").split()) # allowing the user to input how many did pay and how many
                                                                                                        # will split for the payment
```



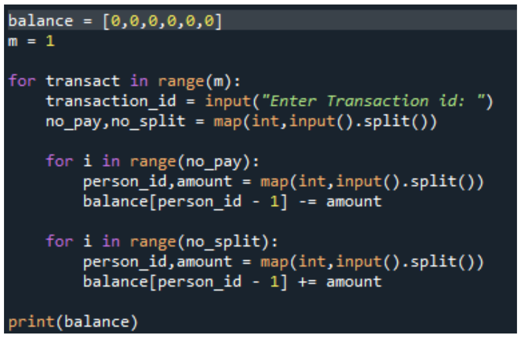
```
Enter Transaction id: transaction 1

2 3

Enter Transaction id: transaction 2

1 4

Enter Transaction id: transaction 3

2 4
```

• Transaction 1 means 2 will pay and 3 will split

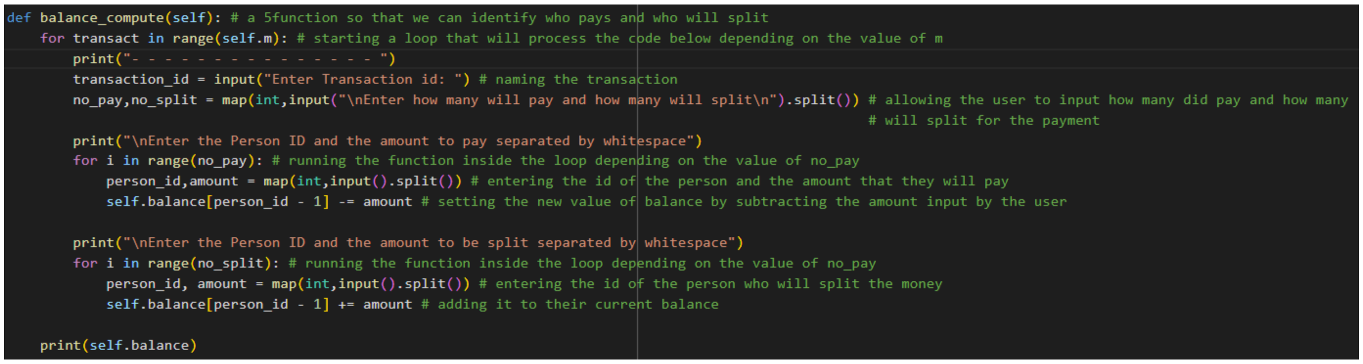
• Transaction 2 means 1 will pay and 4 will split

• Transaction 3 means 2 will pay and 4 will split

In this block of code, we entered another for loop inside the the for loop. The for loops inside will continue to process or transact transactions up until the last value of m.

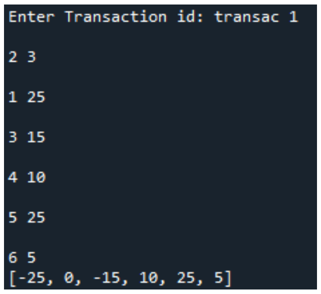


```
balance = [0,0,0,0,0,0]
m = 1

for transact in range(m):
    transaction_id = input("Enter Transaction id: ")
    no_pay,no_split = map(int,input().split())

    for i in range(no_pay):
        person_id,amount = map(int,input().split())
        balance[person_id - 1] -= amount

    for i in range(no_split):
        person_id,amount = map(int,input().split())
        balance[person_id - 1] += amount

print(balance)
```



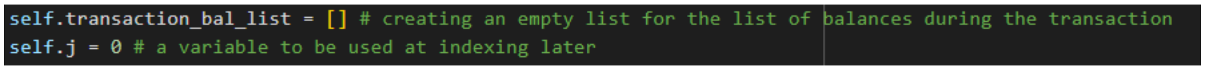
```
def balance_compute(self): # a 5function so that we can identify who pays and who will split
    for transact in range(self.m): # starting a loop that will process the code below depending on the value of m
        print("- - - - - - - - - - - - - - - - ")
        transaction_id = input("Enter Transaction id: ") # naming the transaction
        no_pay,no_split = map(int,input("\nEnter how many will pay and how many will split\n").split()) # allowing the user to input how many did pay and how many
                                                                                                      # will split for the payment
        print("\nEnter the Person ID and the amount to pay separated by whitespace")
        for i in range(no_pay): # running the function inside the loop depending on the value of no_pay
            person_id,amount = map(int,input().split()) # entering the id of the person and the amount that they will pay
            self.balance[person_id - 1] -= amount # setting the new value of balance by subtracting the amount input by the user

        print("\nEnter the Person ID and the amount to be split separated by whitespace")
        for i in range(no_split): # running the function inside the loop depending on the value of no_pay
            person_id, amount = map(int,input().split()) # entering the id of the person who will split the money
            self.balance[person_id - 1] += amount # adding it to their current balance

    print(self.balance)
```

We have labeled the transaction id as transac 1 and entered 2 for the number of people that will pay and 3 for the number of people that will split the payment. By printing the balance list, we can see that there are negative values for the payer because they paid, there money will get reduced, as for the splits, those 3 persons will split for the total amount of paid by the first inputs.

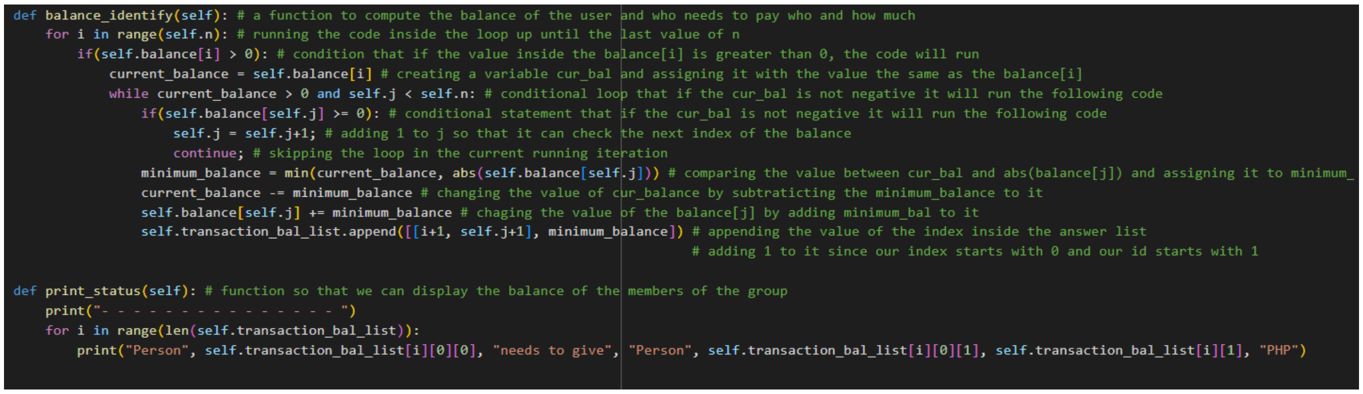


```
Enter Transaction id: transac 1

2 3

1 25

3 15

4 10

5 25

6 5
[-25, 0, -15, 10, 25, 5]
```

We created an empty list named transaction_bal_list so that we can append computed values insied and print it. We also created a variable j and assigned 0 to it so that we can use it inside the conditional statements.



```
self.transaction_bal_list = [] # creating an empty list for the list of balances during the transaction
self.j = 0 # a variable to be used at indexing later
```



```python
def balance_identify(self): # a function to compute the balance of the user and who needs to pay who and how much
    for i in range(self.n): # running the code inside the loop up until the last value of n
        if(self.balance[i] > 0): # condition that if the value inside the balance[i] is greater than 0, the code will run
            current_balance = self.balance[i] # creating a variable cur_bal and assigning it with the value the same as the balance[i]
            while current_balance > 0 and self.j < self.n: # conditional loop that if the cur_bal is not negative it will run the following code
                if(self.balance[self.j] >= 0): # conditional statement that if the cur_bal is not negative it will run the following code
                    self.j = self.j+1; # adding 1 to j so that it can check the next index of the balance
                    continue; # skipping the loop in the current running iteration
                minimum_balance = min(current_balance, abs(self.balance[self.j])) # comparing the value between cur_bal and abs(balance[j]) and assigning it to minimum_
                current_balance -= minimum_balance # changing the value of cur_balance by subtraticting the minimum_balance to it
                self.balance[self.j] += minimum_balance # chaging the value of the balance[j] by adding minimum_bal to it
                self.transaction_bal_list.append([[i+1, self.j+1], minimum_balance]) # appending the value of the index inside the answer list
                                                                                      # adding 1 to it since our index starts with 0 and our id starts with 1


def print_status(self): # function so that we can display the balance of the members of the group
    print("- - - - - - - - - - - - - - - ")
    for i in range(len(self.transaction_bal_list)):
        print("Person", self.transaction_bal_list[i][0][0], "needs to give", "Person", self.transaction_bal_list[i][0][1], self.transaction_bal_list[i][1], "PHP")
```

The for loop simulates the transactions between accounts, it iterates each of the balances and once it finds a positive balance it will get distributed to the other accounts until the end of the list is reached. The record will get appended at the transac_bal_list.