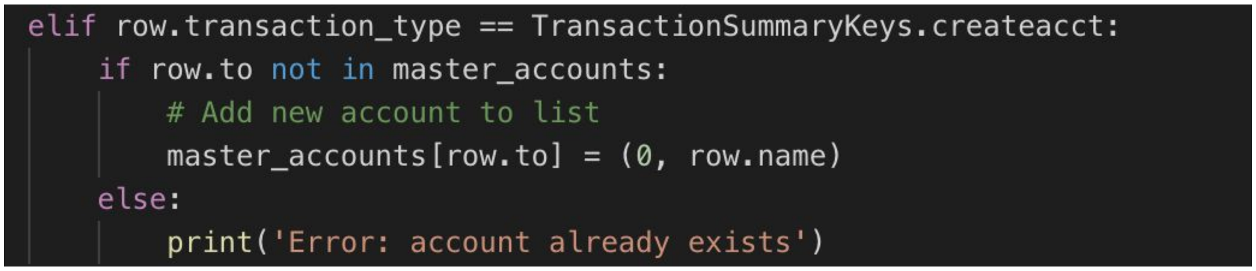
Combined test set report

Account Creation Tests:

For the account creation section, we decided to use statement coverage testing (i.e. each statement in the block of code covering account creation must be executed at least once for testing). We created a table of input values required to reach the lines pertaining to account creation (assuming that there is a "create account" transaction in the merged transaction summary file). Based on the table of simplest inputs to reach all parts of code we came up with two test cases. Creating an account when the inputted account number is not in the master accounts and creating an account when the inputted account number is already in the master accounts list.

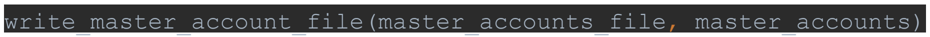


```
elif row.transaction_type == TransactionSummaryKeys.createacct:
    if row.to not in master_accounts:
        # Add new account to list
        master_accounts[row.to] = (0, row.name)
    else:
        print('Error: account already exists')
```

For account creation we decided to use statement coverage testing. P1 tests lines 83-85 in the account creation block of code. P2 tests lines 86-87.

Statement	To account	Master Accounts file	Merged transaction summary file	Test	Description
83	1234567	*Empty*	NEW 1234567 000 0000000 nam EOS 0000000 000 0000000 ***	create_p 1	Create account with new account number
84	1234567	*Empty*	NEW 1234567 000 0000000 nam EOS 0000000 000 0000000 ***		
85	1234567	*Empty*	NEW 1234567 000 0000000 nam EOS 0000000 000 0000000 ***		
86	1234567	1234567 0 abc	NEW 1234567 000 0000000 nam EOS 0000000 000 0000000 ***	create_p 2	Create account with existing account number
87	1234567	1234567 0 abc	NEW 1234567 000 0000000 nam EOS 0000000 000 0000000 ***		

After line 85/87 is executed, the following code is ran on lines 97 and 98 respectively:



```
write_master_account_file(master_accounts_file, master_accounts)
```

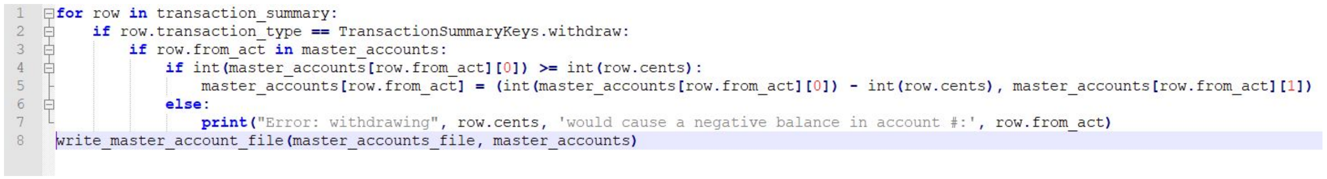
```
write_new_valid_accounts_file(master_accounts)
```

The execution of these lines is used in the test cases to verify the output.

Test	**Description**	**Input Master Accounts file**	**Input Merged transaction summary file**	**Output master accounts file**	**Output master valid accounts file**	**Expected terminal output**
create_p1	Create account with new account number	*Empty*	NEW 1234567 000 0000000 nam EOS 0000000 000 0000000 ***	1234567 0 nam	1234567 0000000	None
create_p2	Create account with existing account number	1234567 0 abc	NEW 1234567 000 0000000 nam EOS 0000000 000 0000000 ***	1234567 0 abc	1234567 0000000	Error: account #: 1234567 already exists

Withdraw Tests

For the withdraw section, we decided to use decision coverage testing. The code shown below is reduced to just the sections that affect withdraw.



```
1 for row in transaction_summary:
2     if row.transaction_type == TransactionSummaryKeys.withdraw:
3         if row.from_act in master_accounts:
4             if int(master_accounts[row.from_act][0]) >= int(row.cents):
5                 master_accounts[row.from_act] = (int(master_accounts[row.from_act][0]) - int(row.cents), master_accounts[row.from_act][1])
6             else:
7                 print("Error: withdrawing", row.cents, 'would cause a negative balance in account #:', row.from_act)
8 write_master_account_file(master_accounts_file, master_accounts)
```

It can be observed that there are three lines at which a decision is made; lines 1, 2, and 3. From this a flow graph can be constructed to assist with the creation of the necessary test cases.

From the flow graph it can be observed that there are four independent paths:

I. 1->8
II. 1->2->1->8
III. 1->2->3->4->1->8
IV. 1->2->3->6->1->8

For withdrawals we decided to use decision coverage testing, creating a test case to cover each of these paths. The paths and descriptions, along with the test, are detailed below:

Path	**Description**	**List**	**master_accounts_file**	**merged_transaction_summary_file**	**output master accounts file**	**Expected terminal output**
P1	Empty merged_transaction_summary_file	1->8	*Empty*	EOS 0000000 000 0000000 ***	*Empty*	*Empty*
P2	From_Account in the transaction is not present in the master_accounts_file	1->2->1->8	1234567 1000 nam	WDR 7654321 1000 0000000 nam EOS 0000000 000 0000000 ***	1234567 1000 nam	*Empty*
P3	Successful transaction	1->2->3->4->1->8	1234567 1000 nam	WDR 1234567 1000 0000000 nam EOS 0000000 000 0000000 ***	1234567 0 nam	*Empty*
P4	Attempted withdrawal	1->2->3->6->	1234567 100 nam	WDR 1234567 1000 0000000 nam	1234567 100 nam	Error: withdrawing

	would take more money than is present in the account	1->8		EOS 0000000 000 0000000 ***		1000 would cause a negative balance in account #: 1234567

Failures Uncovered by Tests:

Test	Test Description	Nature of Failure	Fix
test_withdraw_P3	Test a successful withdraw transaction	The account's balance does not properly decrease after the withdrawal is made.	Change the comparison from > to >=
test_withdraw_P4	Test a withdrawal of a greater amount than is in the user's account	The expected error message is not properly printed out	Test input was the wrong prompt. Change input to work properly

A5 work:

Group Member	**Hours Spent on Assignment**	**Aspects involved in**
Bruce Nishimura	5 hrs	Coded test suite helper and test_create_p1 and test_create_p2. Account creation tests and test cases. Creating/editing a5 report.
Meara Donovan	4 hrs	Backend design doc, commenting backend code, fixes for a5 report.
Sam McPhail	4 hours	Test creation for withdrawal section. Uncovered failures test_withdraw_P3 and test_withdraw_P4, and implemented fixes. Creation and formatting of final report.