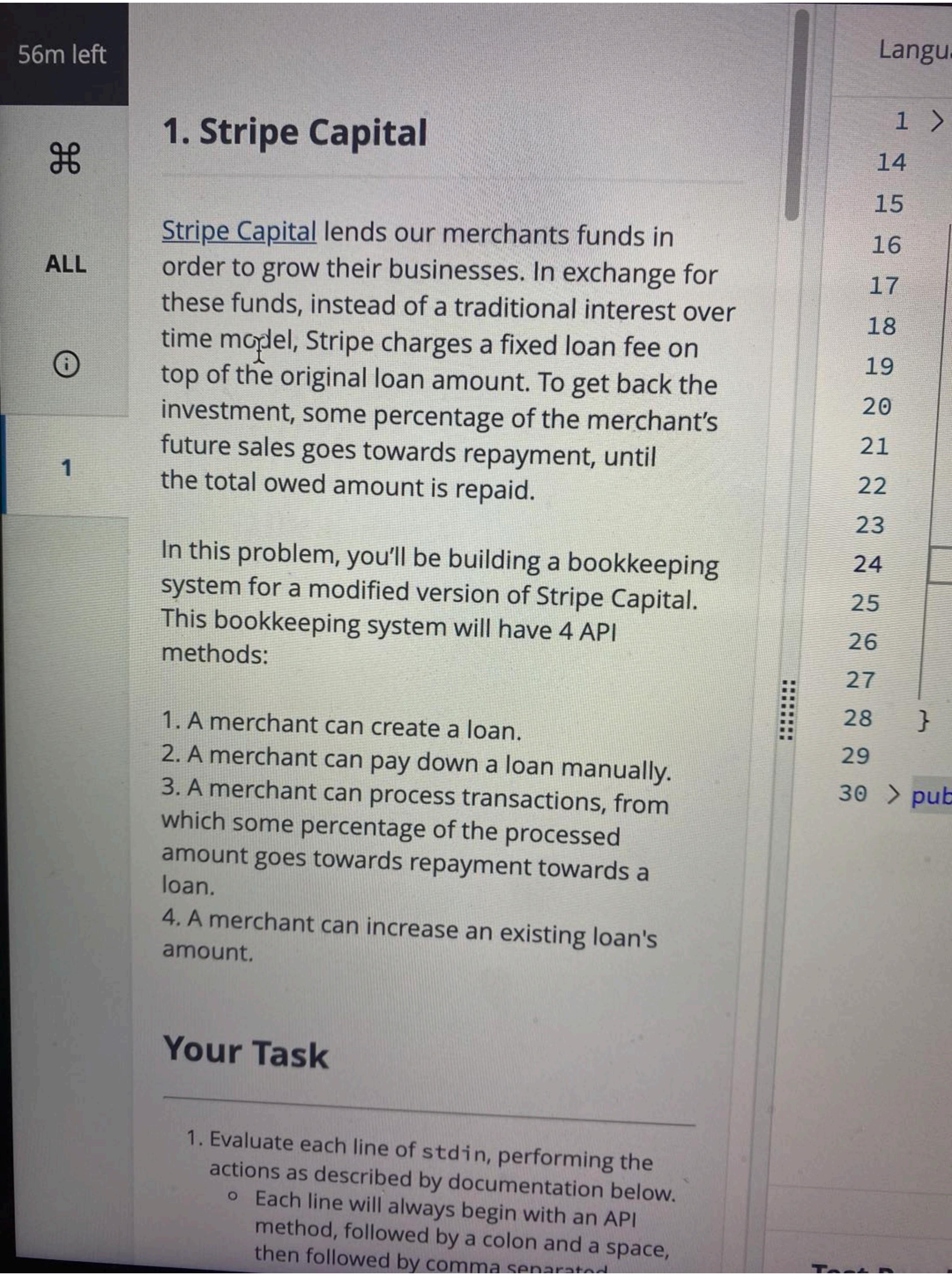


1. Stripe Capital

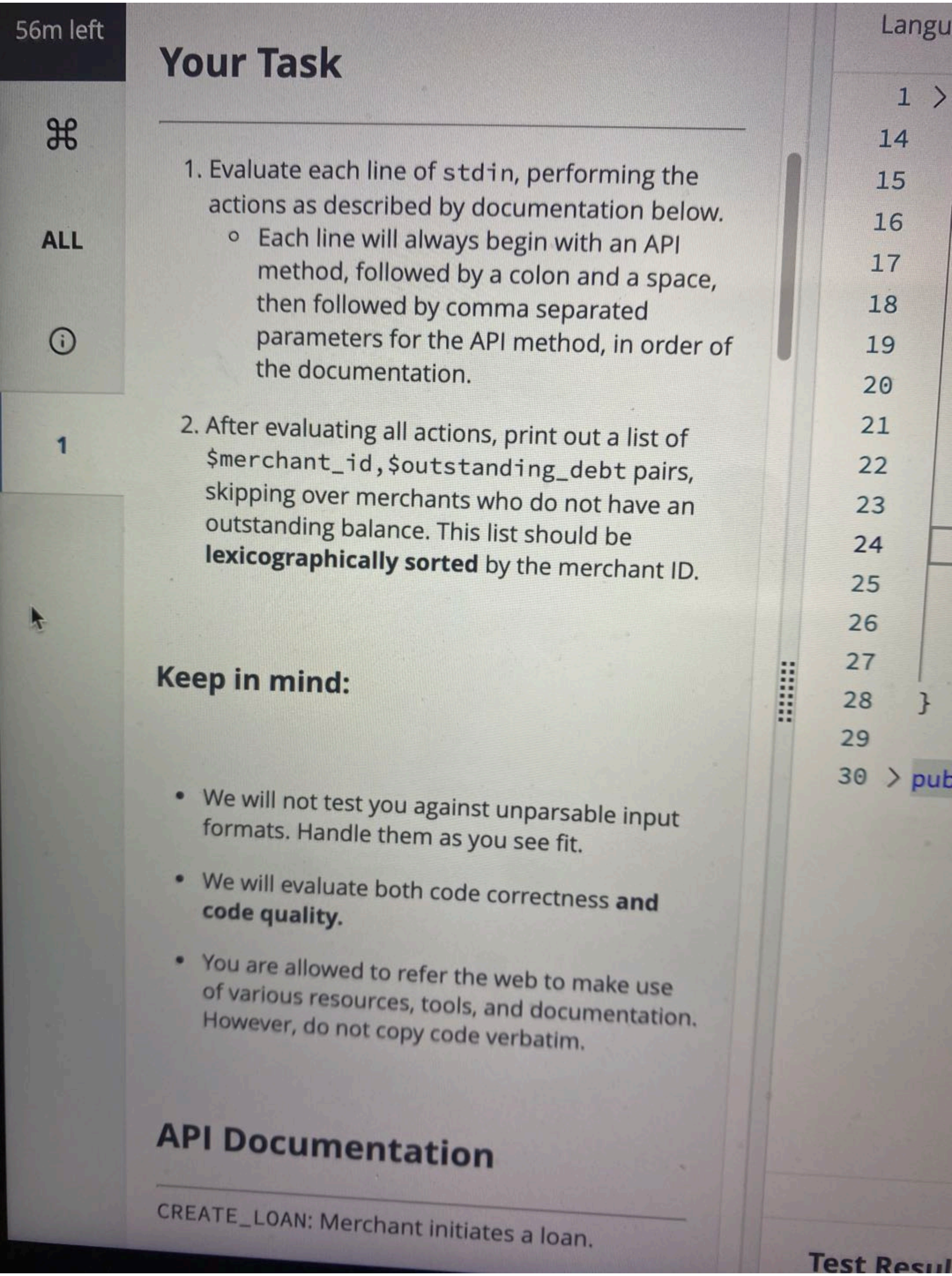
Stripe Capital lends our merchants funds in order to grow their businesses. In exchange for these funds, instead of a traditional interest over time model, Stripe charges a fixed loan fee on top of the original loan amount. To get back the investment, some percentage of the merchant's future sales goes towards repayment, until the total owed amount is repaid.

In this problem, you'll be building a bookkeeping system for a modified version of Stripe Capital. This bookkeeping system will have 4 API methods:

1. A merchant can create a loan.
2. A merchant can pay down a loan manually.
3. A merchant can process transactions, from which some percentage of the processed amount goes towards repayment towards a loan.
4. A merchant can increase an existing loan's amount.

Your Task

1. Evaluate each line of `stdin`, performing the actions as described by documentation below.
   - Each line will always begin with an API method, followed by a colon and a space, then followed by comma separated



Your Task

1. Evaluate each line of `stdin`, performing the actions as described by documentation below.
    - Each line will always begin with an API method, followed by a colon and a space, then followed by comma separated parameters for the API method, in order of the documentation.

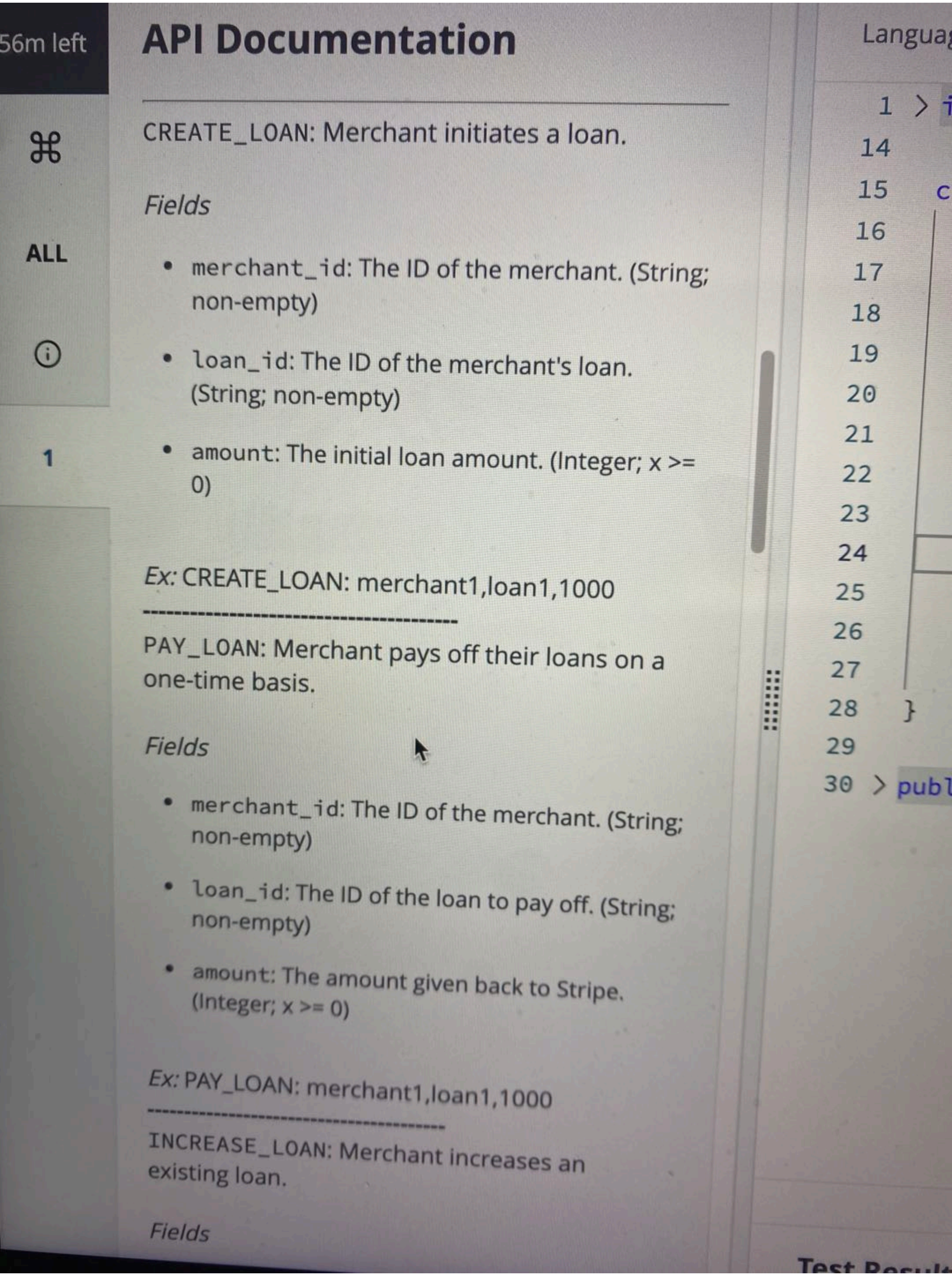
2. After evaluating all actions, print out a list of `$merchant_id,$outstanding_debt` pairs, skipping over merchants who do not have an outstanding balance. This list should be **lexicographically sorted** by the merchant ID.

Keep in mind:

- We will not test you against unparsable input formats. Handle them as you see fit.
- We will evaluate both code correctness **and code quality.**
- You are allowed to refer the web to make use of various resources, tools, and documentation. However, do not copy code verbatim.

API Documentation

CREATE_LOAN: Merchant initiates a loan.



API Documentation

CREATE_LOAN: Merchant initiates a loan.

Fields

- `merchant_id`: The ID of the merchant. (String; non-empty)
- `loan_id`: The ID of the merchant's loan. (String; non-empty)
- `amount`: The initial loan amount. (Integer; x >= 0)

Ex: CREATE_LOAN: merchant1,loan1,1000

PAY_LOAN: Merchant pays off their loans on a one-time basis.

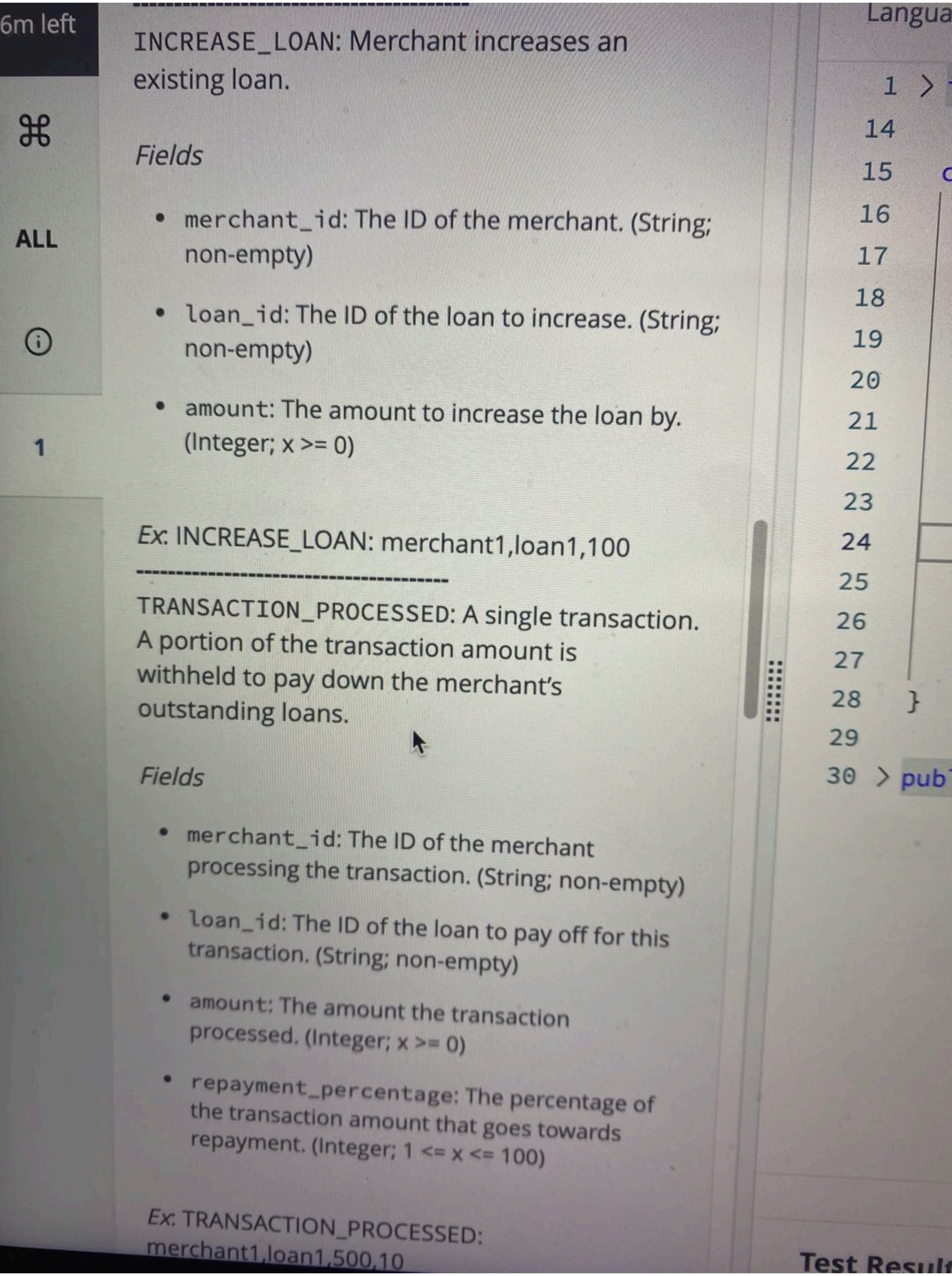
Fields

- `merchant_id`: The ID of the merchant. (String; non-empty)
- `loan_id`: The ID of the loan to pay off. (String; non-empty)
- `amount`: The amount given back to Stripe. (Integer; x >= 0)

Ex: PAY_LOAN: merchant1,loan1,1000

INCREASE_LOAN: Merchant increases an existing loan.

Fields



INCREASE_LOAN: Merchant increases an existing loan.

Fields

- `merchant_id`: The ID of the merchant. (String; non-empty)
- `loan_id`: The ID of the loan to increase. (String; non-empty)
- `amount`: The amount to increase the loan by. (Integer; x >= 0)

Ex: INCREASE_LOAN: merchant1,loan1,100

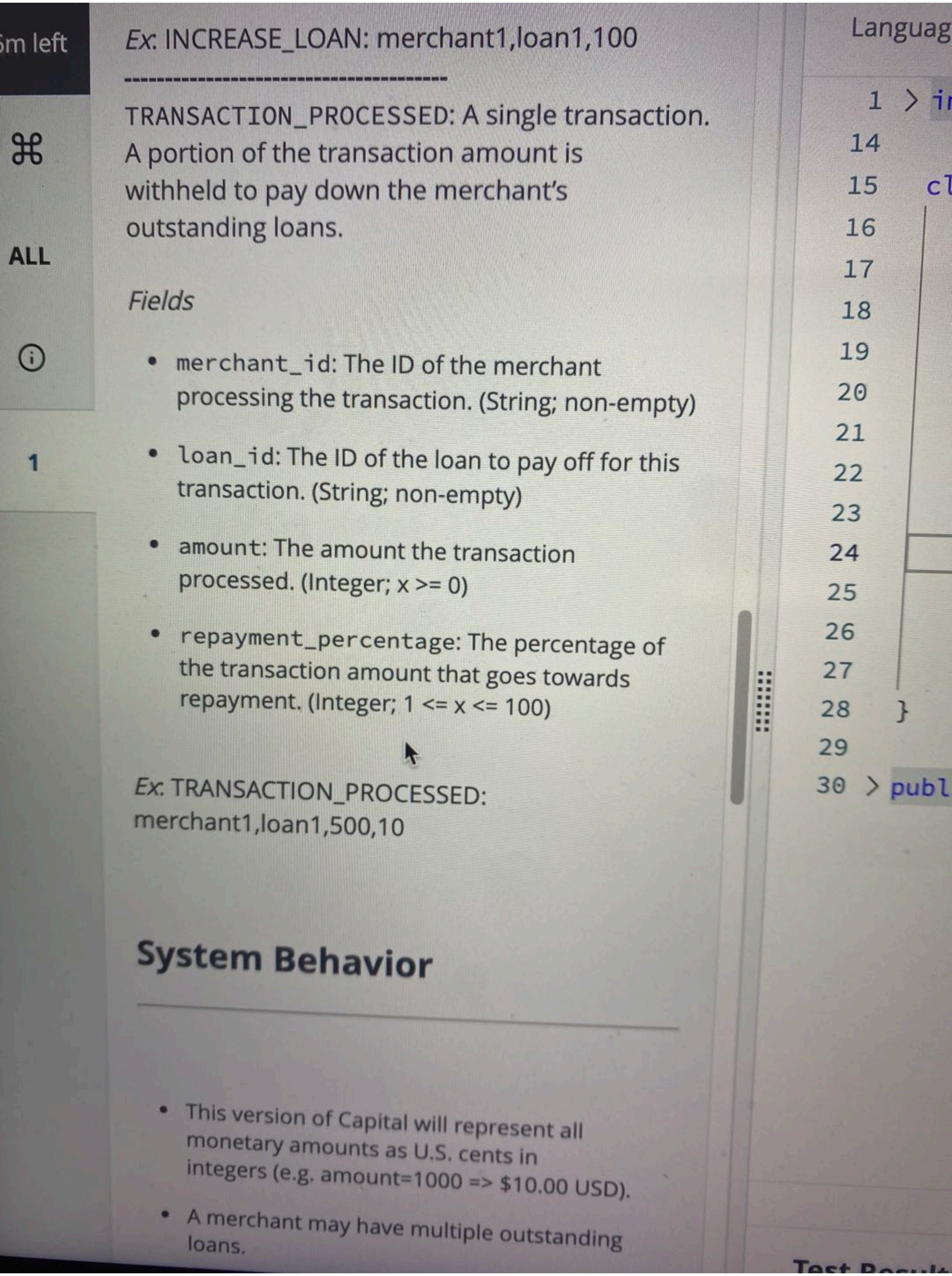
TRANSACTION_PROCESSED: A single transaction. A portion of the transaction amount is withheld to pay down the merchant's outstanding loans.

Fields

- `merchant_id`: The ID of the merchant processing the transaction. (String; non-empty)
- `loan_id`: The ID of the loan to pay off for this transaction. (String; non-empty)
- `amount`: The amount the transaction processed. (Integer; x >= 0)
- `repayment_percentage`: The percentage of the transaction amount that goes towards repayment. (Integer; 1 <= x <= 100)

Ex: TRANSACTION_PROCESSED: merchant1,loan1,500,10

```
}
pub
```



Ex: INCREASE_LOAN: merchant1,loan1,100

`TRANSACTION_PROCESSED`: A single transaction. A portion of the transaction amount is withheld to pay down the merchant's outstanding loans.

Fields

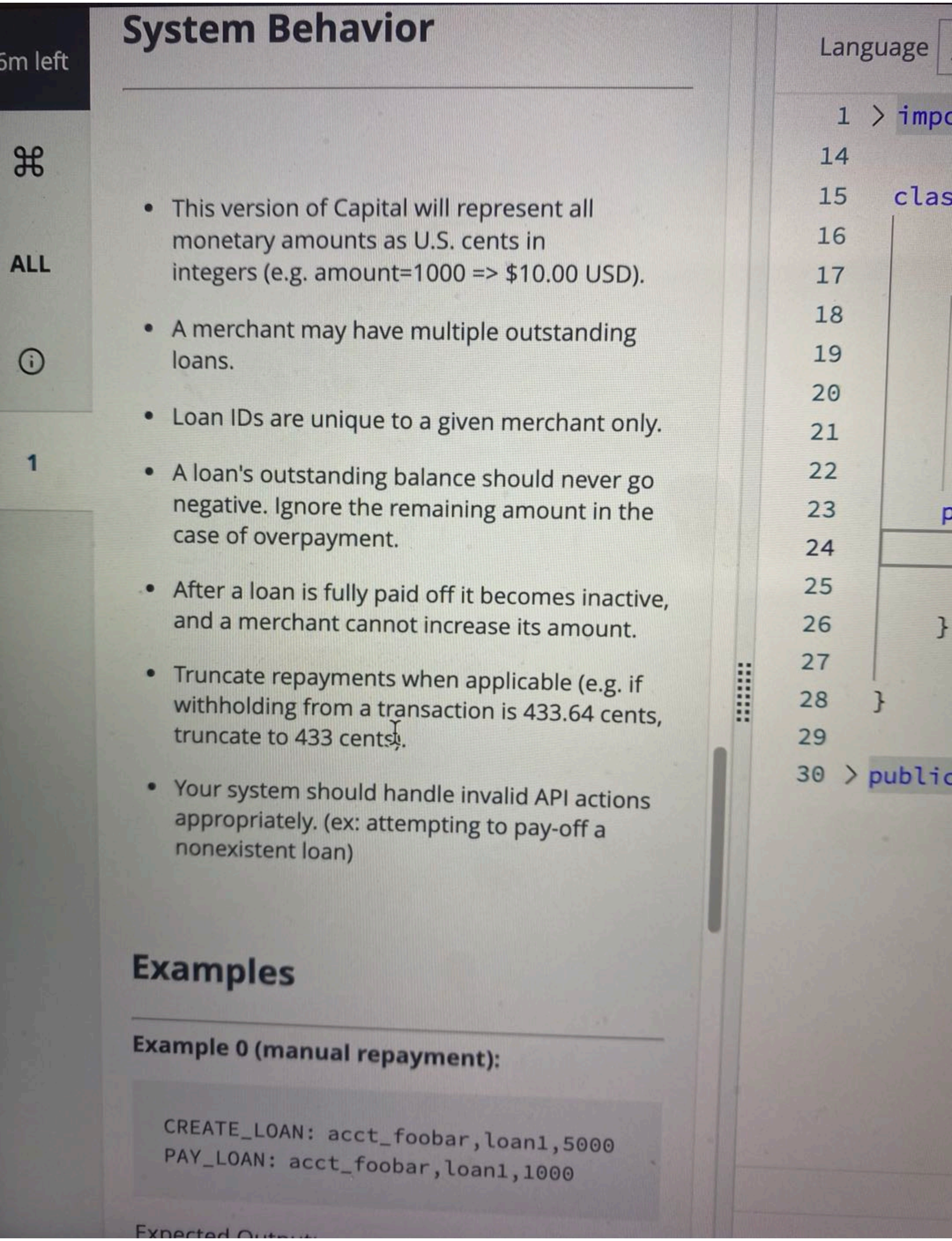
- `merchant_id`: The ID of the merchant processing the transaction. (String; non-empty)
- `loan_id`: The ID of the loan to pay off for this transaction. (String; non-empty)
- `amount`: The amount the transaction processed. (Integer; x >= 0)
- `repayment_percentage`: The percentage of the transaction amount that goes towards repayment. (Integer; 1 <= x <= 100)

Ex: TRANSACTION_PROCESSED: merchant1,loan1,500,10

System Behavior

- This version of Capital will represent all monetary amounts as U.S. cents in integers (e.g. amount=1000 => $10.00 USD).
- A merchant may have multiple outstanding loans.

```
> i
  cl
  }

> publ
```



```
> impo
  clas
    p
    }
  }
> public
```

System Behavior

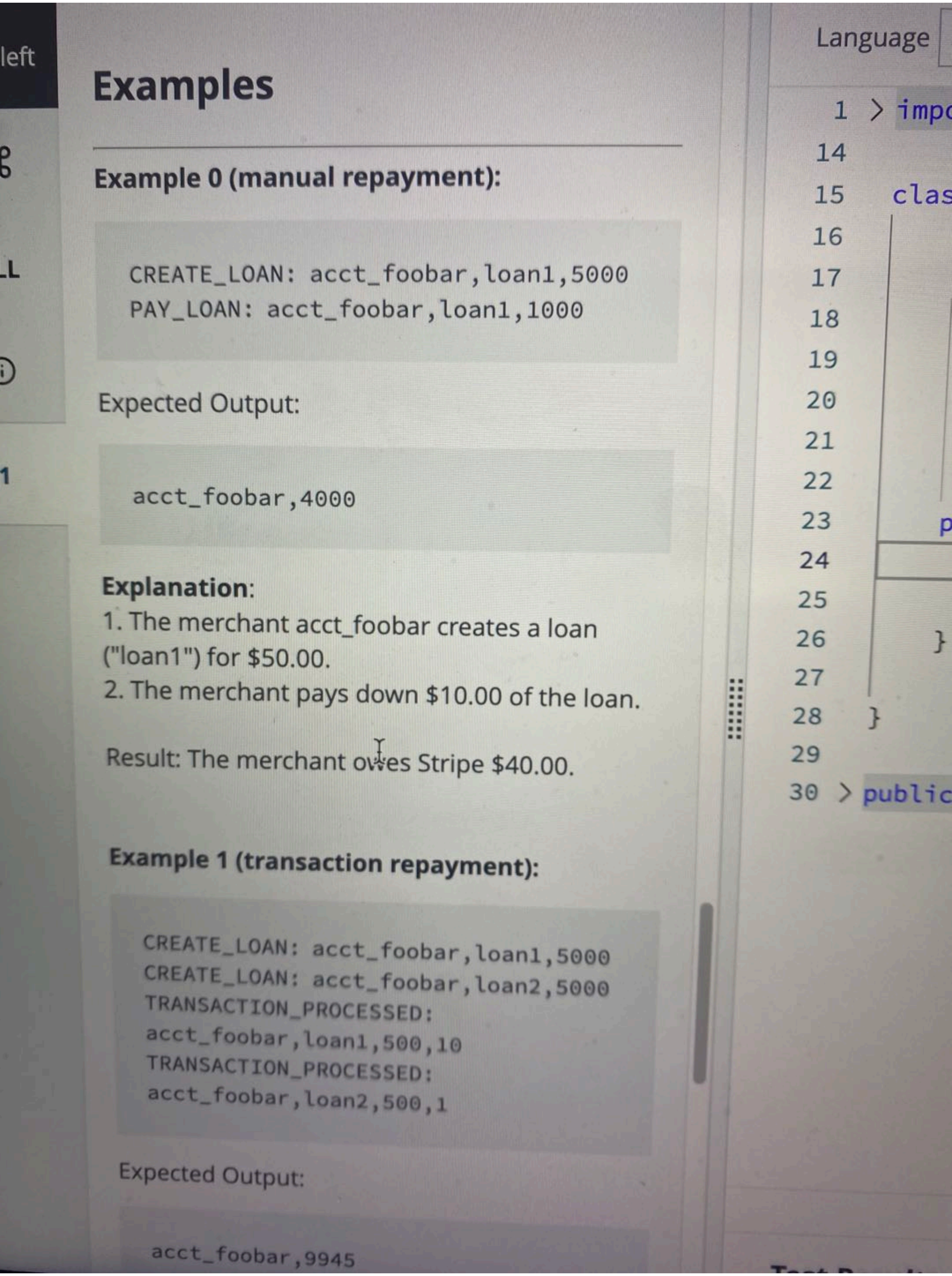
- This version of Capital will represent all monetary amounts as U.S. cents in integers (e.g. amount=1000 => $10.00 USD).
- A merchant may have multiple outstanding loans.
- Loan IDs are unique to a given merchant only.
- A loan's outstanding balance should never go negative. Ignore the remaining amount in the case of overpayment.
- After a loan is fully paid off it becomes inactive, and a merchant cannot increase its amount.
- Truncate repayments when applicable (e.g. if withholding from a transaction is 433.64 cents, truncate to 433 cents).
- Your system should handle invalid API actions appropriately. (ex: attempting to pay-off a nonexistent loan)

Examples

Example 0 (manual repayment):

```
CREATE_LOAN: acct_foobar,loan1,5000
PAY_LOAN: acct_foobar,loan1,1000
```

Expected Output



Examples

Example 0 (manual repayment):

```
CREATE_LOAN: acct_foobar,loan1,5000
PAY_LOAN: acct_foobar,loan1,1000
```

Expected Output:

```
acct_foobar,4000
```

Explanation:
1. The merchant acct_foobar creates a loan ("loan1") for $50.00.
2. The merchant pays down $10.00 of the loan.

Result: The merchant owes Stripe $40.00.

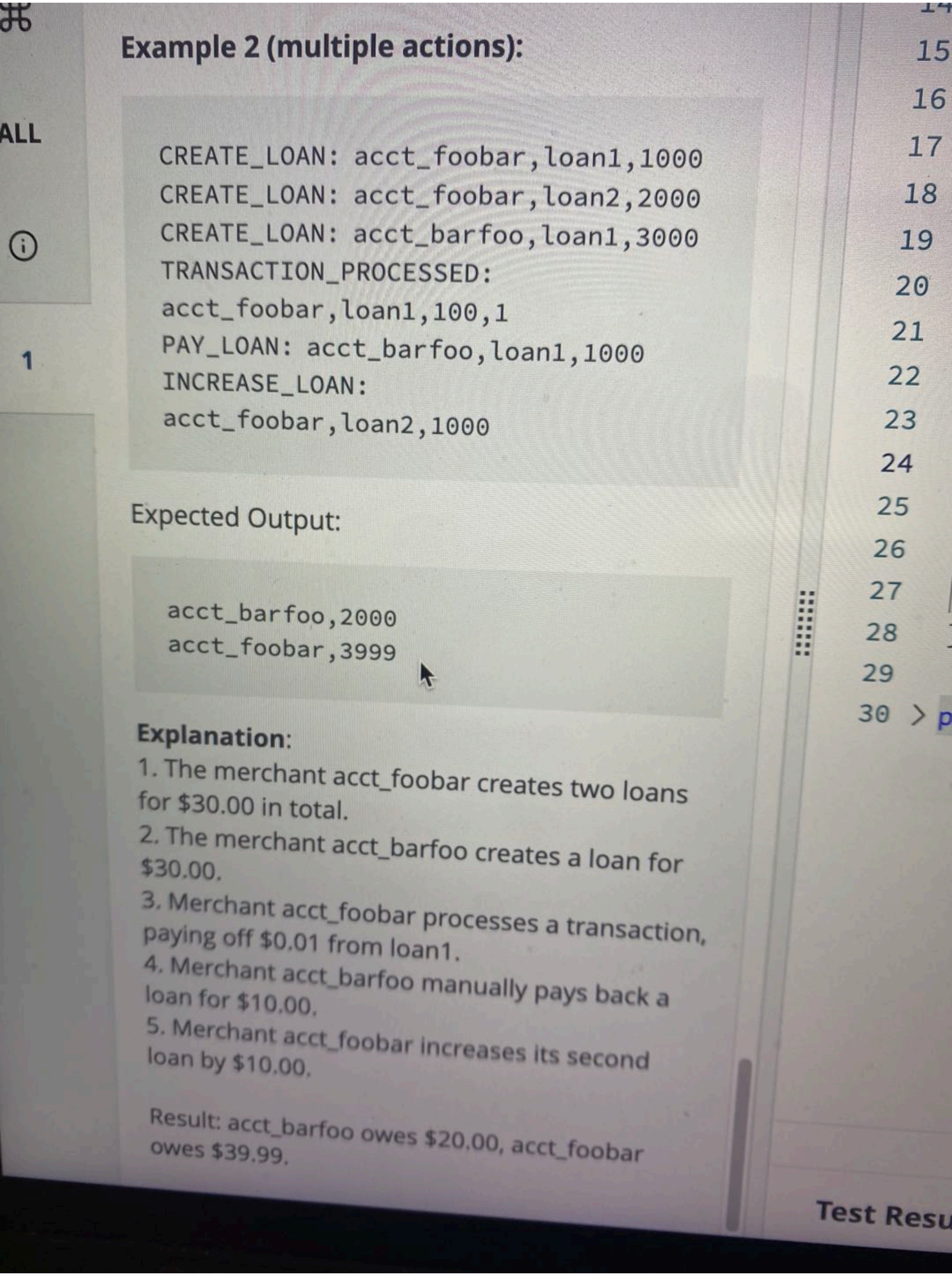
Example 1 (transaction repayment):

```
CREATE_LOAN: acct_foobar,loan1,5000
CREATE_LOAN: acct_foobar,loan2,5000
TRANSACTION_PROCESSED:
acct_foobar,loan1,500,10
TRANSACTION_PROCESSED:
acct_foobar,loan2,500,1
```

Expected Output:

```
acct_foobar,9945
```

```
> impo

  clas



              p

             }
  }

> public
```



Example 2 (multiple actions):

```
CREATE_LOAN: acct_foobar,loan1,1000
CREATE_LOAN: acct_foobar,loan2,2000
CREATE_LOAN: acct_barfoo,loan1,3000
TRANSACTION_PROCESSED:
acct_foobar,loan1,100,1
PAY_LOAN: acct_barfoo,loan1,1000
INCREASE_LOAN:
acct_foobar,loan2,1000
```

Expected Output:

```
acct_barfoo,2000
acct_foobar,3999
```

Explanation:

1. The merchant acct_foobar creates two loans for $30.00 in total.
2. The merchant acct_barfoo creates a loan for $30.00.
3. Merchant acct_foobar processes a transaction, paying off $0.01 from loan1.
4. Merchant acct_barfoo manually pays back a loan for $10.00.
5. Merchant acct_foobar increases its second loan by $10.00.

Result: acct_barfoo owes $20.00, acct_foobar owes $39.99.