```
"""Perform credit card calculations.
Adharsh Maheswaran
IS 326
Feb 15, 2023
"""
from argparse import ArgumentParser
import sys


# replace this comment with your implementation of get_min_payment(),
# interest_charged(), remaining_payments(), and main()
def parse_args(args_list):
    """Takes a list of strings from the command prompt and passes them through as
    arguments
    Args:
        args_list (list) : the list of strings from the command prompt
    Returns:
        args (ArgumentParser)
    """

    parser = ArgumentParser()
    parser.add_argument('balance_amount', type=float, help='The total amount of
balance left on the credit account')
    parser.add_argument('apr', type=int, help='The annual APR, should be an int
between 1 and 100')
    parser.add_argument('credit_line', type=int, help='The maximum amount of
balance allowed on the credit line.')
    parser.add_argument('--payment', type=int, default=None,
                        help='The amount the user wants to pay per payment, should
be a positive number')
    parser.add_argument('--fees', type=float, default=0, help='The fees that are
applied monthly.')

    # parse and validate arguments
    args = parser.parse_args(args_list)
    if args.balance_amount < 0:
        raise ValueError("balance amount must be positive")
    if not 0 <= args.apr <= 100:
        raise ValueError("APR must be between 0 and 100")
    if args.credit_line < 1:
        raise ValueError("credit line must be positive")
    if args.payment is not None and args.payment < 0:
        raise ValueError("number of payments per year must be positive")
    if args.fees < 0:
        raise ValueError("fees must be positive")
    return args


# Returns minimum payement
def get_min_payment(balance, fees):
    """
    Takes in balance and fees and returns the minimum monthly payment
    """
    b = balance
    f = fees
    m = 0.02
    minPayment = float(((b * m) + f))
    if minPayment < 25:
        minPayment = 25
```

```
    return minPayment


# Returns interest charged
def interest_charged(balance, apr):
    """
    Takes in balance and apr and returns the interest charged
    """
    a = float(apr / 100)
    b = balance
    y = 365
    d = 30
    i = float((a / y) * b * d)
    return i


# Return remaining payments
def remaining_payments(balance, apr, targetamount, credit_line, fees):

    # Counters
    counter = 0
    counter25 = 0
    counter50 = 0
    counter75 = 0

    # While balance is not zero and positive
    while balance > 0:
        if targetamount is None:
            payment = get_min_payment(balance, fees)
        else:
            payment = targetamount

        i = interest_charged(balance, apr)
        payment_to_balance = payment - i

        if payment_to_balance < 0:
            print("The balance cannot be paid off.")
            return [counter, counter25, counter50, counter75]
            break

        if payment_to_balance >= 0:
            balance -= payment_to_balance
            if balance > (0.25 * credit_line):
                counter25 += 1
            if balance > (0.5 * credit_line):
                counter50 += 1
            if balance > (0.75 * credit_line):
                counter75 += 1
            counter += 1

    return [counter, counter25, counter50, counter75]


# Main Method
def main(balance, apr, targetamount, credit_line, fees):
    """
    User inputs the information and the main function outputs the main
    """
    # Minimum Payments
```

```
    minPayment = get_min_payment(balance, fees)
    print("Your recommended minimum payment is $", minPayment)

    payMinimum = False

    # Checks if targetamount is not there
    if targetamount is None:
        payMinimum = True
    else:
        print("Your target payment is less than the minimum payment for this credit
card.")

    # Total Payments
    total_payments = remaining_payments(balance, apr, targetamount, credit_line,
fees)

    # If statement for if pay_minimum is true or false
    if payMinimum:
        print("If you pay the minimum payments each month, you will pay off the
balance in", total_payments[0],
              "payments.\n")
        print(
            "You will spend a total of", total_payments[1],
            "months over 25% of the credit line. \nYou will spend a total of",
total_payments[2],
            "months over 50% of the credit line \nYou will spend a total of",
total_payments[3],
            "months over 75% of the credit line \n")
    else:
        print("If you make payments of", targetamount, "you will pay off the
balance in", total_payments[0],
              "payments.\n")

        print("If you pay the minimum payments each month, you will pay off the
balance in", total_payments[0],
              "payments.\n")
        print(
            "You will spend a total of", total_payments[1],
            "months over 25% of the credit line. \nYou will spend a total of",
total_payments[2],
            "months over 50% of the credit line \nYou will spend a total of",
total_payments[3],
            "months over 75% of the credit line \n")


if __name__ == "__main__":
    try:
        arguments = parse_args(sys.argv[1:])
    except ValueError as e:
        sys.exit(str(e))
    print(main(arguments.balance_amount, arguments.apr, credit_line=arguments
               .credit_line, targetamount=arguments.payment, fees=arguments.fees))
```