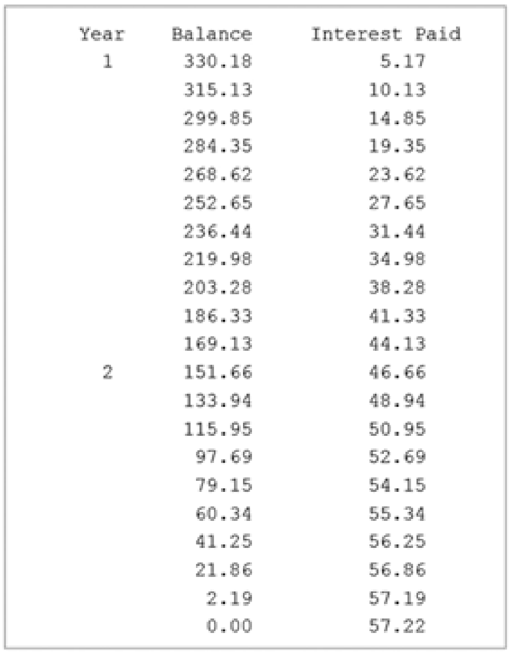
# Credit Card Calculation  Program

For this UPA, you will design, implement, and test a program that will allow you to determine the length of time needed to pay off a credit card balance, as well as the total interest paid.

## The Problem

The problem is to generate a table showing the decreasing balance and accumulating interest paid on a credit card account for a given credit card balance, interest rate, and monthly payment, as shown below.



```
Year    Balance      Interest Paid
  1      330.18            5.17
         315.13           10.13
         299.85           14.85
         284.35           19.35
         268.62           23.62
         252.65           27.65
         236.44           31.44
         219.98           34.98
         203.28           38.28
         186.33           41.33
         169.13           44.13
  2      151.66           46.66
         133.94           48.94
         115.95           50.95
          97.69           52.69
          79.15           54.15
          60.34           55.34
          41.25           56.25
          21.86           56.86
           2.19           57.19
           0.00           57.22
```

### Problem Analysis

The factors that determine how quickly a loan is paid off are the amount of the loan, the interest rate charged, and the monthly payments made. For a fixed-rate home mortgage, the monthly payments are predetermined so that the loan is paid off within a specific number of years. Therefore, the total interest that will be paid on the loan is made evident at the time the loan is signed.

For a credit card, there is only a minimum payment required each month. It is not always explicitly stated by the credit card company, however, how long it would take to pay off the card by making only the minimum payment. The minimum payment for a credit card is dependent on the particular credit card company. However, it is usually around 2–3% of the outstanding loan amount each month, and no less than twenty dollars. Thus, calculating this allows us to project the amount of time that it would take before the account balance becomes zero, as well as the total interest paid.

## Program Design

### Meeting the Program Requirements

No particular format is specified for how the output is to be displayed. All that is required is that the user be able to enter the relevant information and that the length of time to pay off the loan and the total interest paid is displayed. The user will also be given the choice of assuming the monthly payment to be the required minimum payment, or a larger specified amount.

### Data Description

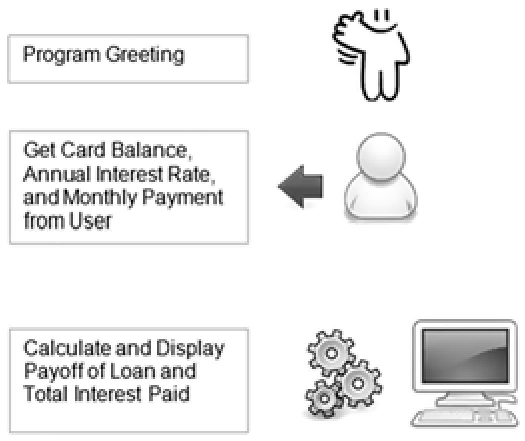
All that needs to be represented in this program are numerical values for the loan amount, the interest rate, and the monthly payment made. There is no need to create a data structure as the table of payments can be generated as it is displayed.

### Algorithmic Approach

The only algorithm needed for this problem is the calculation of the required minimum payment. The minimum payment is usually calculated at 2% or 3% of the outstanding balance, with a lower limit of around $20. Therefore, we will assume a worst case scenario of a minimum payment calculated at 2%, with a minimum payment of $20.

## Overall Program Steps

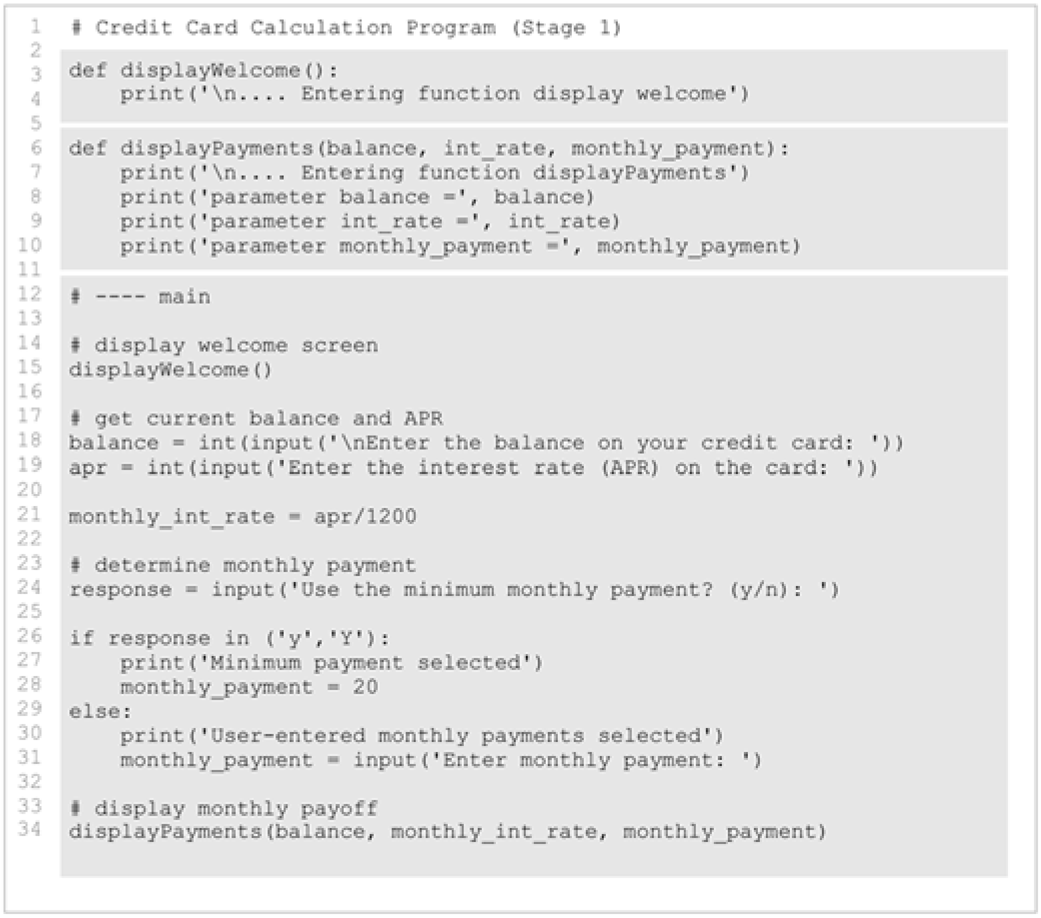
The overall steps in this program design are illustrated below:


Program Greeting
Get Card Balance, Annual Interest Rate, and Monthly Payment from User
Calculate and Display Payoff of Loan and Total Interest Paid

## Program Implementation and  Testing

### Stage 1 - Developing the Overall Program Structure

Enter the Stage 1 Code from the sample on the next page.



```
1  # Credit Card Calculation Program (Stage 1)
2
3  def displayWelcome():
4      print('\n.... Entering function display welcome')
5
6  def displayPayments(balance, int_rate, monthly_payment):
7      print('\n.... Entering function displayPayments')
8      print('parameter balance =', balance)
9      print('parameter int_rate =', int_rate)
10     print('parameter monthly_payment =', monthly_payment)
11
12 # ---- main
13
14 # display welcome screen
15 displayWelcome()
16
17 # get current balance and APR
18 balance = int(input('\nEnter the balance on your credit card: '))
19 apr = int(input('Enter the interest rate (APR) on the card: '))
20
21 monthly_int_rate = apr/1200
22
23 # determine monthly payment
24 response = input('Use the minimum monthly payment? (y/n): ')
25
26 if response in ('y','Y'):
27     print('Minimum payment selected')
28     monthly_payment = 20
29 else:
30     print('User-entered monthly payments selected')
31     monthly_payment = input('Enter monthly payment: ')
32
33 # display monthly payoff
34 displayPayments(balance, monthly_int_rate, monthly_payment)
```

**Notes:**

The program begins on line 15 with a call to function `displayWelcome()`. Next, the current credit card balance and annual interest rate (APR) are input from the user (lines 18–19), each read as an integer value. Since the monthly interest rate is what will be used in the calculations, the value in `apr` is divided by 1200 (on line 21). This converts the value to a monthly interest rate, as well as converting it to decimal form (for example, 18% as 0.18).

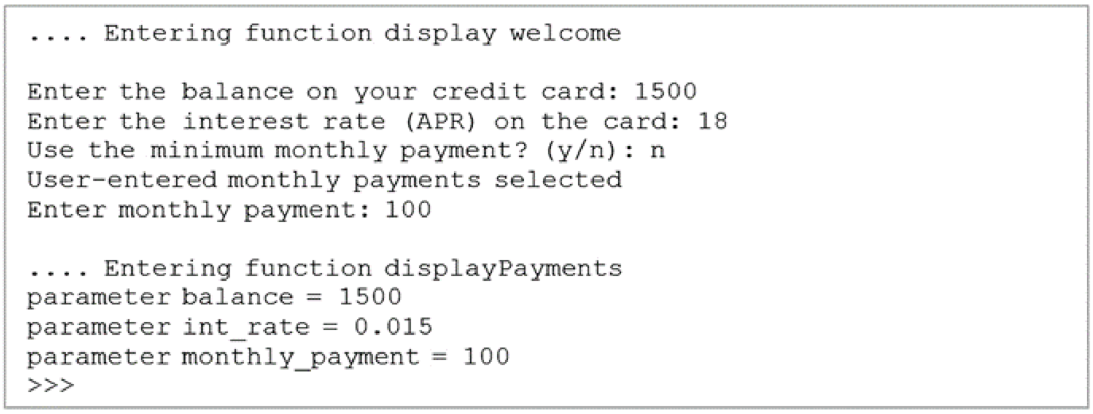
The final value input from the user is the monthly payments that they wish to have the payoff calculated with. They have a choice of either going with the minimum required monthly payment, assumed to be $20 for testing purposes (line 28), or a specified monthly payment (line

31). The credit card balance, annual percentage rate, and the assumed monthly payments are passed to func- tion `displayPayments` (on line 34) to calculate and display the pay down of the balance as well as the interest paid over each month of the payoff period.

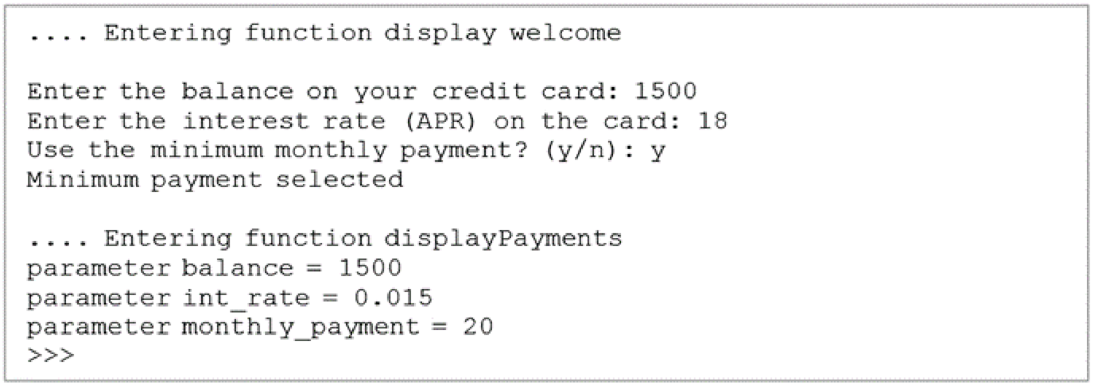
Functions `displayWelcome` (line 3) and `displayPayments` (line 6) consist only of trace statements. A trace statement prints, for testing purposes, a message indicating that a certain point in the program has been reached. Trace statements are also used to display the value of certain variables. Once this part of the program is working, we can focus on implementing the functions and further developing the main program section.

## Stage 1 - Testing

Test the Stage 1 code before continuing. A sample test run of this stage of the program is shown here:



```
.... Entering function display welcome

Enter the balance on your credit card: 1500
Enter the interest rate (APR) on the card: 18
Use the minimum monthly payment? (y/n): n
User-entered monthly payments selected
Enter monthly payment: 100

.... Entering function displayPayments
parameter balance = 1500
parameter int_rate = 0.015
parameter monthly_payment = 100
>>>
```

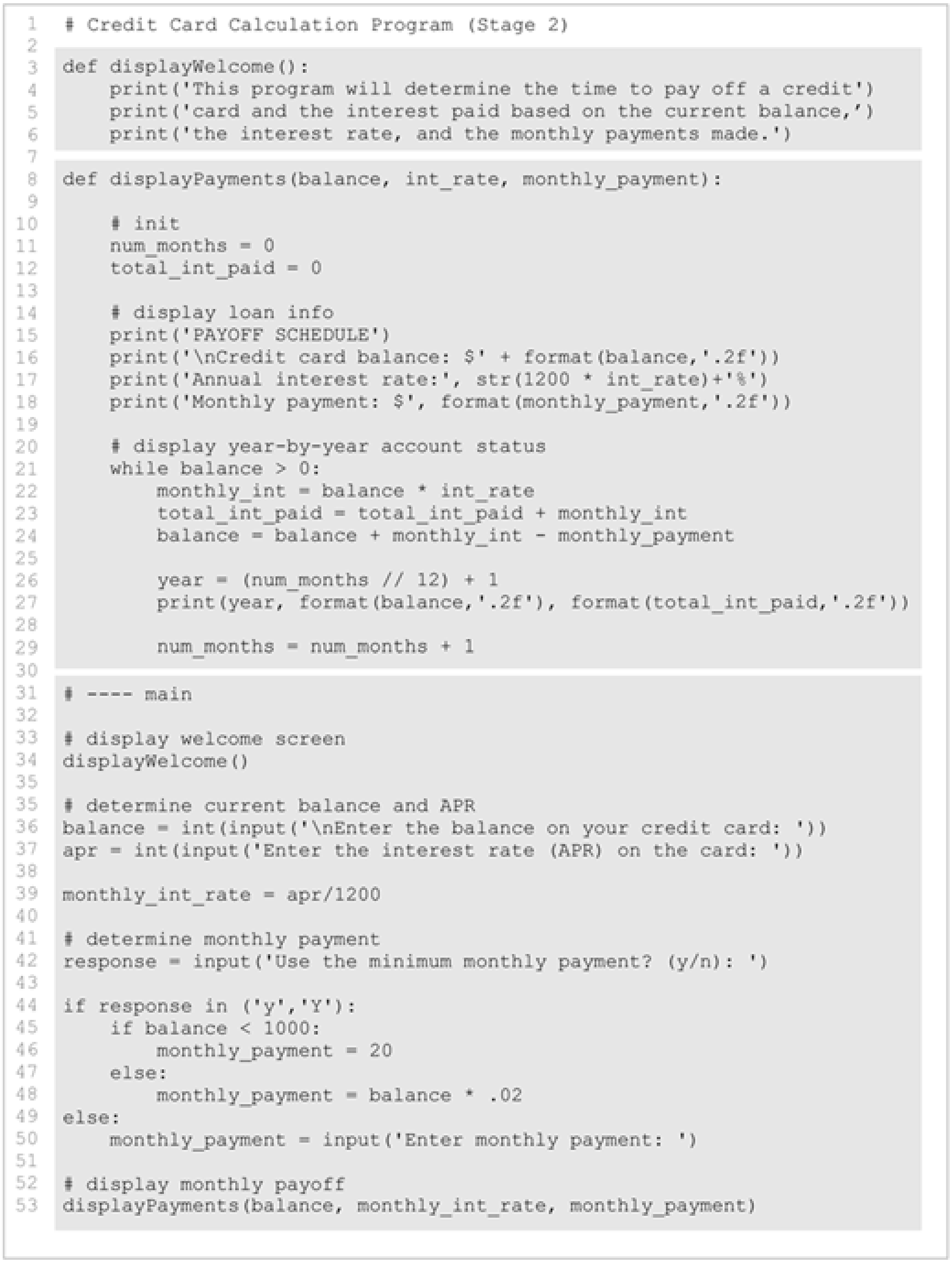


```
.... Entering function display welcome

Enter the balance on your credit card: 1500
Enter the interest rate (APR) on the card: 18
Use the minimum monthly payment? (y/n): y
Minimum payment selected

.... Entering function displayPayments
parameter balance = 1500
parameter int_rate = 0.015
parameter monthly_payment = 20
>>>
```

From the test results, we see that the appropriate values are being input and passed to function `displayPayments`. So it looks like the overall structure of this stage of the program is working correctly.

## Stage 2—Generating an Unformatted Display of Payments

Next, you will implement function `displayWelcome`, and develop an initial implementation of function `displayPayments,as seen below.` Refer to the Notes, for details.



```python
# Credit Card Calculation Program (Stage 2)

def displayWelcome():
    print('This program will determine the time to pay off a credit')
    print('card and the interest paid based on the current balance,’)
    print('the interest rate, and the monthly payments made.')

def displayPayments(balance, int_rate, monthly_payment):

    # init
    num_months = 0
    total_int_paid = 0

    # display loan info
    print('PAYOFF SCHEDULE')
    print('\nCredit card balance: $' + format(balance,'.2f'))
    print('Annual interest rate:', str(1200 * int_rate)+'%')
    print('Monthly payment: $', format(monthly_payment,'.2f'))

    # display year-by-year account status
    while balance > 0:
        monthly_int = balance * int_rate
        total_int_paid = total_int_paid + monthly_int
        balance = balance + monthly_int - monthly_payment

        year = (num_months // 12) + 1
        print(year, format(balance,'.2f'), format(total_int_paid,'.2f'))

        num_months = num_months + 1

# ---- main

# display welcome screen
displayWelcome()

# determine current balance and APR
balance = int(input('\nEnter the balance on your credit card: '))
apr = int(input('Enter the interest rate (APR) on the card: '))

monthly_int_rate = apr/1200

# determine monthly payment
response = input('Use the minimum monthly payment? (y/n): ')

if response in ('y','Y'):
    if balance < 1000:
        monthly_payment = 20
    else:
        monthly_payment = balance * .02
else:
    monthly_payment = input('Enter monthly payment: ')

# display monthly payoff
displayPayments(balance, monthly_int_rate, monthly_payment)
```

**Notes**:

Remove the two print instructions that were included only for test purposes in stage 1 of the program (previously on lines 27 and 30). Also, the minimum required monthly payment is computed (lines 45–48) rather than being set to 20.

Function `displayPayments` is where most of the work is done in the program. Therefore, we shall develop this function in stages as well. At this point, we develop the function to display, for each month during the loan payoff, the year, the current balance, and the total interest paid to date. We delay issues of screen formatting for the alignment of numbers, and only include formatting for rounding numeric values to two decimal places.

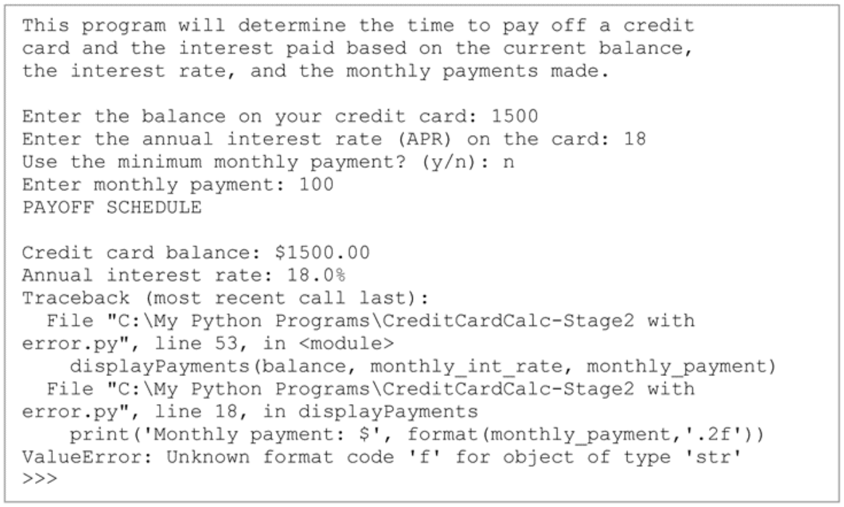
The while loop on line 21 iterates while `balance`, passed as an argument to the function, is greater than zero. The function will keep count of the number of months (lines) displayed, as well as the total interest paid. Variables `num_months` and `total_int_paid` are used for this purpose, and are therefore initialized before the loop to 0 (lines 11–12). On lines 15–18 the initial information for the calculation is displayed. Within the while loop, on line 22, the monthly interest paid (`monthly_int`) is computed as the current balance of that month during the payoff period (`balance`), times the monthly interest rate (`int_rate`). The total interest paid is then updated on line 23. On line 24, the new balance is computed as the current balance, plus the interest for the month, minus the monthly payment.

The next step is to display these computed values. Since time is kept track of in terms of months, the current year to be displayed is computed using integer division (line 26), adding one so that the first year is displayed as 1, and not 0. Then on line 27, the line representing the payment for the current month is displayed. Formatting is used so that all numerical values are displayed with two decimal places. Finally, variable `num_months` is incremented by one for the next iteration of the loop.

## Stage 2 Testing

Test this program once for a specified monthly payment amount, and once for the option of minimum monthly payments. The results are shown below:

Test 1:

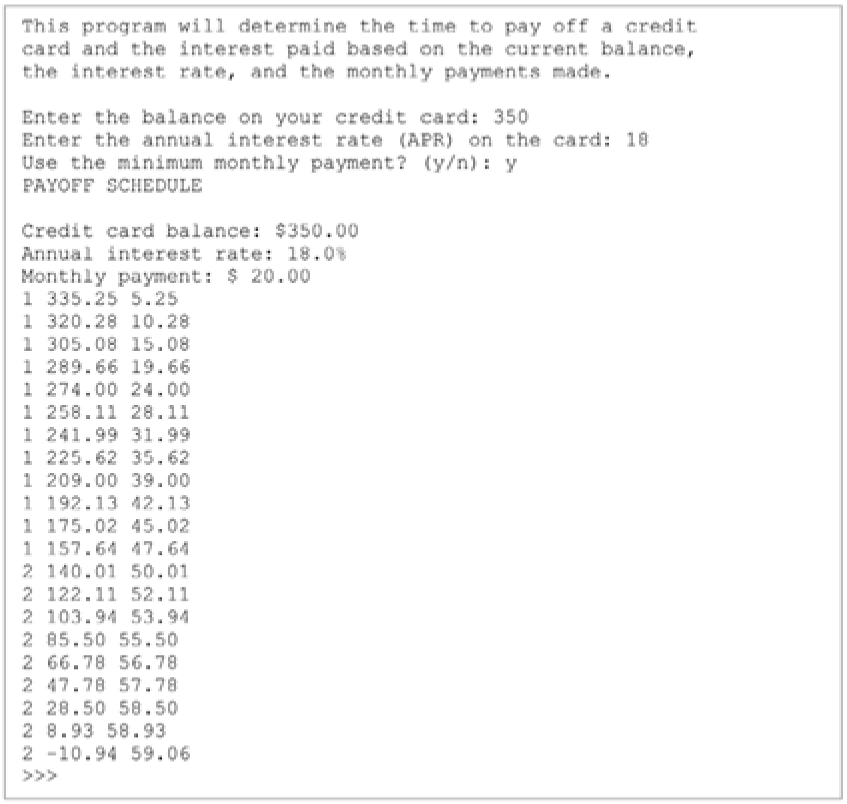


```
This program will determine the time to pay off a credit
card and the interest paid based on the current balance,
the interest rate, and the monthly payments made.

Enter the balance on your credit card: 1500
Enter the annual interest rate (APR) on the card: 18
Use the minimum monthly payment? (y/n): n
Enter monthly payment: 100
PAYOFF SCHEDULE

Credit card balance: $1500.00
Annual interest rate: 18.0%
Traceback (most recent call last):
  File "C:\My Python Programs\CreditCardCalc-Stage2 with
error.py", line 53, in <module>
    displayPayments(balance, monthly_int_rate, monthly_payment)
  File "C:\My Python Programs\CreditCardCalc-Stage2 with
error.py", line 18, in displayPayments
    print('Monthly payment: $', format(monthly_payment,'.2f'))
ValueError: Unknown format code 'f' for object of type 'str'
>>>
```

Clearly, there is something wrong with this version of the program. The `ValueError` generated on the previous page indicates that the format specifier `.2f` is an unknown format code for a string type value, referring to line 18. Thus, this must be referring to variable `monthly_payment`. But that should be a numeric value, and not a string value! How could it have become a string type?

Check if the problem also occurs when selecting the minimum payment option (your Test 2 should look like the sample below).

Test 2:



```
This program will determine the time to pay off a credit
card and the interest paid based on the current balance,
the interest rate, and the monthly payments made.

Enter the balance on your credit card: 350
Enter the annual interest rate (APR) on the card: 18
Use the minimum monthly payment? (y/n): y
PAYOFF SCHEDULE

Credit card balance: $350.00
Annual interest rate: 18.0%
Monthly payment: $ 20.00
1 335.25 5.25
1 320.28 10.28
1 305.08 15.08
1 289.66 19.66
1 274.00 24.00
1 258.11 28.11
1 241.99 31.99
1 225.62 35.62
1 209.00 39.00
1 192.13 42.13
1 175.02 45.02
1 157.64 47.64
2 140.01 50.01
2 122.11 52.11
2 103.94 53.94
2 85.50 55.50
2 66.78 56.78
2 47.78 57.78
2 28.50 58.50
2 8.93 58.93
2 -10.94 59.06
>>>
```

In this case the program works. Since the problem only occurred when the user entered the monthly payment (as opposed to the minimum payment option), we next try to determine what differences there are in the program related to the assignment of variable `monthly_payment`.

```
# determine monthly payment
response = input('Use the minimum monthly payment? (y/n): ')
if response in ('y', 'Y'):
    if balance < 1000:
        monthly_payment = 20
    else:
        monthly_payment = balance * .02
else:
    monthly_payment = input('Enter monthly payment: ')
```

When the user selects the minimum monthly payment option, variable `monthly_payment` is set to integer value `20` (or 2% of the current balance if balance is greater than 1000). Otherwise, its value is input from the user. This variable is not redefined anywhere else in the program. Since the variable `monthly_payment` is not a local variable, we can display its value directly from the Python shell,

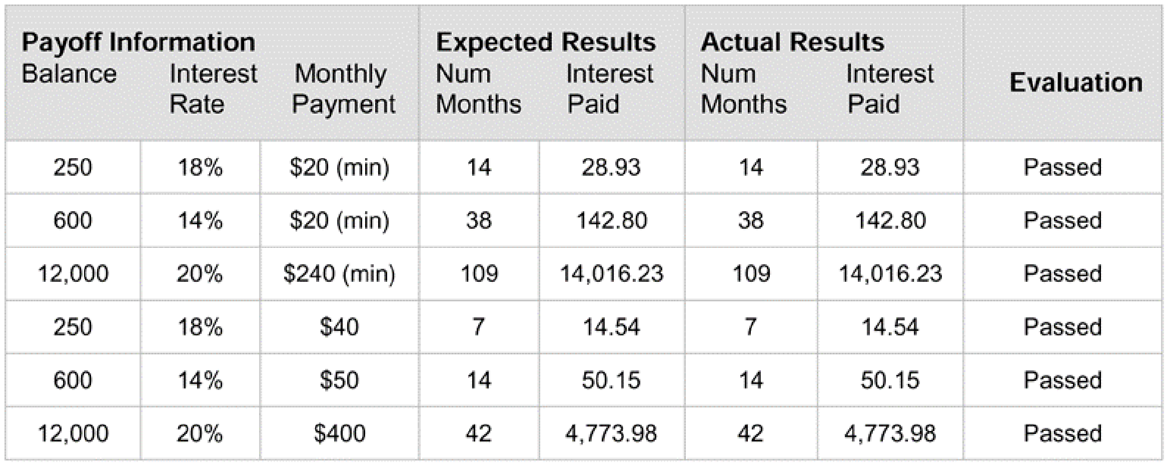
```
>>> monthly_payment
'140'
```

It *is* a string value. We immediately realize that the input value for variable `monthly_payment` was not converted to an integer type, and was thus left as a string type! We fix this problem by replacing the line with the following,

```
monthly_payment = int(input('Enter monthly payment: '))
```

This explains why the problem did not appear in the testing of stage 1 of the program. In that version, variable `monthly_payment` was never formatted as a numeric value, and also never used in a numerical calculation (both of which would have generated an error).

At this point, execute a number of test cases for various initial balances, interest rates, and monthly payments. You may use the values from the following table which were checked against online loan payoff calculator tools.



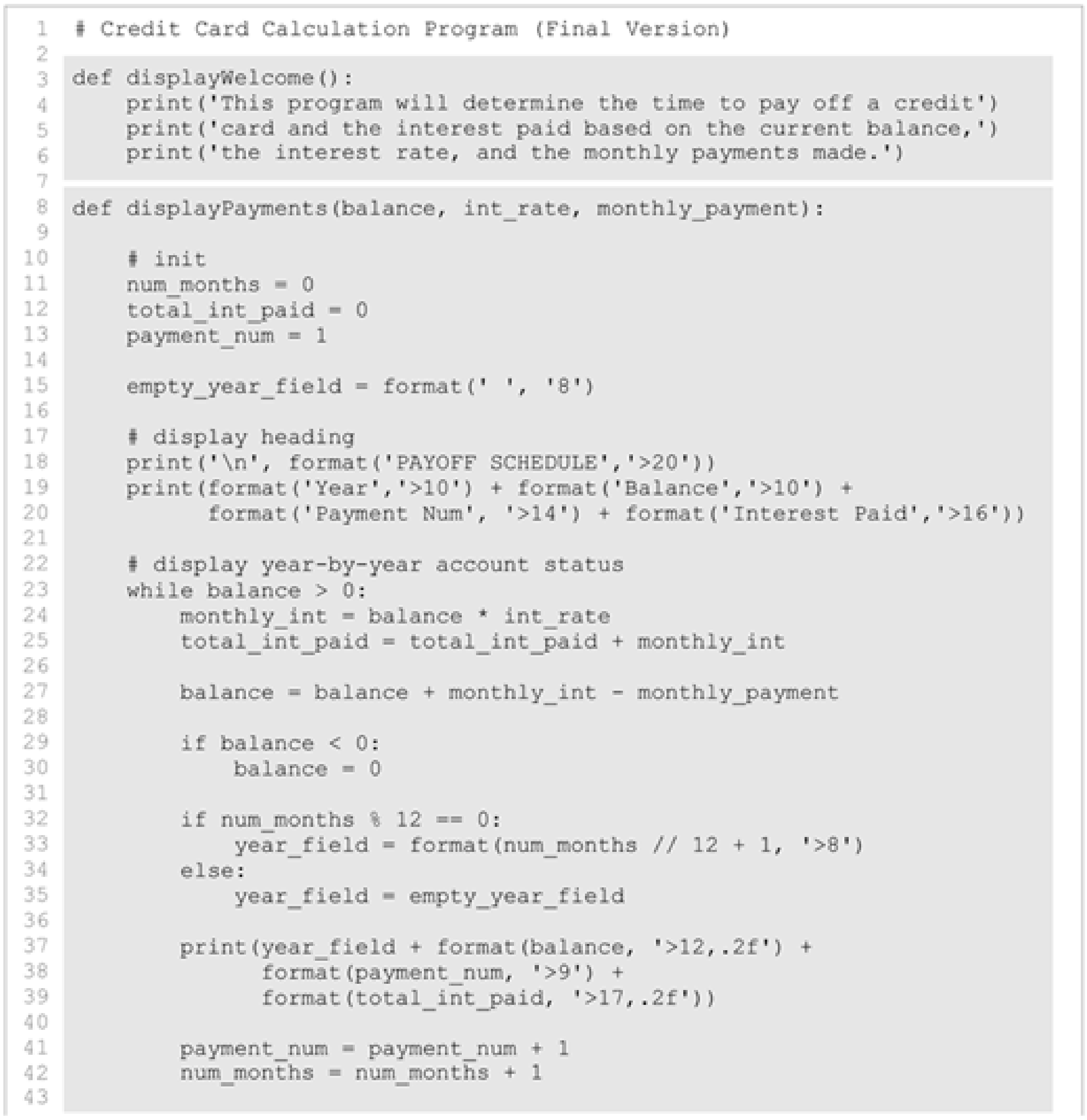
| **Payoff Information** | | | **Expected Results** | | **Actual Results** | | |
|---|---|---|---|---|---|---|---|
| Balance | Interest Rate | Monthly Payment | Num Months | Interest Paid | Num Months | Interest Paid | **Evaluation** |
| 250 | 18% | $20 (min) | 14 | 28.93 | 14 | 28.93 | Passed |
| 600 | 14% | $20 (min) | 38 | 142.80 | 38 | 142.80 | Passed |
| 12,000 | 20% | $240 (min) | 109 | 14,016.23 | 109 | 14,016.23 | Passed |
| 250 | 18% | $40 | 7 | 14.54 | 7 | 14.54 | Passed |
| 600 | 14% | $50 | 14 | 50.15 | 14 | 50.15 | Passed |
| 12,000 | 20% | $400 | 42 | 4,773.98 | 42 | 4,773.98 | Passed |

If your Stage 2 Test cases pass, you may move on to the final stage of program development.

## Stage 3—Formatting the Displayed Output

In the final stage of the program, input error checking is added. The program is also modified to allow the user to continue to enter various monthly payments for recalculating a given balance pay- off. Output formatting is added to make the displayed information more readable. Finally, you will correct the display of a negative balance at the end of the payoff schedule.

The final version of the program is shown on the next three pages. Please refer to the Notes that follow.



```
# Credit Card Calculation Program (Final Version)

def displayWelcome():
    print('This program will determine the time to pay off a credit')
    print('card and the interest paid based on the current balance,')
    print('the interest rate, and the monthly payments made.')

def displayPayments(balance, int_rate, monthly_payment):

    # init
    num_months = 0
    total_int_paid = 0
    payment_num = 1

    empty_year_field = format(' ', '8')

    # display heading
    print('\n', format('PAYOFF SCHEDULE','>20'))
    print(format('Year','>10') + format('Balance','>10') +
          format('Payment Num', '>14') + format('Interest Paid','>16'))

    # display year-by-year account status
    while balance > 0:
        monthly_int = balance * int_rate
        total_int_paid = total_int_paid + monthly_int

        balance = balance + monthly_int - monthly_payment

        if balance < 0:
            balance = 0

        if num_months % 12 == 0:
            year_field = format(num_months // 12 + 1, '>8')
        else:
            year_field = empty_year_field

        print(year_field + format(balance, '>12,.2f') +
              format(payment_num, '>9') +
              format(total_int_paid, '>17,.2f'))

        payment_num = payment_num + 1
        num_months = num_months + 1

```

```
# ---- main

# display welcome screen
displayWelcome()

# get current balance and APR
balance = int(input('\nEnter the balance on your credit card: '))
apr = int(input('Enter the interest rate (APR) on the card: '))

monthly_int_rate = apr/1200

yes_response = ('y','Y')
no_response = ('n','N')

calc = True
while calc:

    # calc minimum monthly payment
    if balance < 1000:
        min_monthly_payment = 20
    else:
        min_monthly_payment = balance * .02

    # get monthly payment
    print('\nAssuming a minimum payment of 2% of the balance ($20 min)')
    print('Your minimum payment would be',
          format(min_monthly_payment, '.2f'),'\n')

    response = input('Use the minimum monthly payment? (y/n): ')
    while response not in yes_response + no_response:
        response = input('Use the minimum monthly payment? (y/n): ')

    if response in yes_response:
        monthly_payment = min_monthly_payment
    else:
        acceptable_payment = False

        while not acceptable_payment:
            monthly_payment = int(input('\nEnter monthly payment: '))

            if monthly_payment < balance * .02:
                print('Minimum payment of 2% of balance required ($' +
                      str(balance * .02) + ')')

            elif monthly_payment < 20:
                print('Minimum payment of $20 required')
            else:
                acceptable_payment = True

```

```
 93     # check if single payment pays off balance
 94     if monthly_payment >= balance:
 95         print('* This payment amount would pay off your balance *')
 96     else:
 97         # display month-by-month balance payoff
 98         displayPayments(balance, monthly_int_rate, monthly_payment)
 99
100         # calculate again with another monthly payment?
101         again = input('\nRecalculate with another payment? (y/n): ')
102         while again not in yes_response + no_response:
103             again = input('Recalculate with another payment? (y/n): ')
104
105         if again in yes_response:
106             calc = True     # continue program
107             print('\n\nFor your current balance of $' + str(balance))
108         else:
109             calc = False    # terminate program
```

**Notes:**

The first set of changes in the program provides some input error checking. (We will address means of more complete error checking in Chapter 7.) In lines 55–56, tuples `yes_response` and `no_ response` are defined. These are used to check if input from the user is an appropriate yes/no response. For example, the while statement on line 73 checks that the input from line 72 is either `'y'`, `'Y'` ,`'n'`, or `'N'`,

```
while response not in yes_response + no_response:
```

by checking if `response` is in the concatenation of tuples `yes_response` and `no_response`. For determining the specific response, the tuples can be used individually (line 76),

```
if response in yes_response:
```

Similar input error checking is done on line 101.

The next set of changes allows a number of payoff schedules for an entered balance to be calculated. A while statement is added at line 59, with its condition based on the value of Boolean variable `calc` (initialized to `True` on line 58). To accommodate the recalculation of payoff schedules, variables `num_months`, `total_int_paid` and `payment_num` are each reset to 0 in function `displayPayments` (lines 11–13).

Output formatting is added in function `displayPayments`. On line 18, '`PAYOFF SCHEDULE`' is displayed right-justified within a field of twenty. On lines 19–20, the column headings are displayed with appropriate field widths. Lines 37–39 display the balance, payment number and in- terest of each month, aligned under the column headings. Lines 32–35 ensure that each year is displayed only once. Finally, in lines 29–30 variable `balance` is set to zero if it becomes negative so that negative balances are not displayed.

## Final Testing

Complete the testing by executing the program on a set of test cases.

| Payoff Info | | | Expected Results | | Actual Results | | Evaluation |
|---|---|---|---|---|---|---|---|
| Balance | Interest Rate | Monthly Payment | # of Months | Interest Paid | # of Months | Interest Paid | |
| 250 | 18% | $20 (min) | 14 | 28.93 | | | |
| 600 | 14% | $20 (min) | 38 | 142.80 | | | |
| 12,000 | 20% | $240 (min) | 109 | 14,016.23 | | | |
| 250 | 18% | $40 | 7 | 14.54 | | | |
| 600 | 14% | $50 | 14 | 50.15 | | | |
| 12,000 | 20% | $400 | 42 | 4,773.98 | | | |