# Price Case Study

Vitali Mueller

June 16, 2018

## CONTENTS



## 1 INTRODUCTION - EXERCISE

Build a set of classes / functions which can price a basic consumer loan given the following inputs:

- Amount
- Term in Months
- APR
- Lending Date
- Repayment day of month

It should return:

- the annuity amount
- the future cashflow schedule (due date, amount due on date, capital component, interest component, and any other relevant info)

For Bonus points:

- write unit tests
- provide a brief wrote up of your approach

### 1.1 Consider two loans:

1. One which was taken out on the 1st of the month, but the regular repayment is on the 15th of each month
2. The other which was taken out on the 15th of the month, and the regular repayment is on the 15th of each month

If these loans are for the same amount (say 1000 pounds), term (12 months) and rate (10 percent, say) - what should the annuity be in each case?

## 2 Equations

To calculate the time value of money, we would need following arguments:

- Future Value of Money - (FV)
- Present Value of Money / Principal / Amount - (PV)
- Annual Percentage Rate - APR
- Number of Compounding Periods per Year - N
- Interest per Month - (i_m)
- Annuity - (a)
- Discount Factor - (DF)

First we calculate the interest per month:

$$i_m = \frac{APR}{12 * 100} \tag{2.1}$$

Calculate the Future Value:

$$FV = PV * (1 + i_m)^N \tag{2.2}$$

Calculate the Present Value:

$$PV = FV * \frac{1}{(}1 + i)^N \tag{2.3}$$

Calculate the Annuity Payment:

$$a = P * \frac{R(1+R)^N}{(1+R)^N - 1} \tag{2.4}$$

We will take the above equation and split it into 2 equation to simplify it for our Software.

$$a = \frac{PV}{DF} \tag{2.5}$$

Calculate the Discount Factor:

$$DF = \frac{(1+i)^N - 1}{i(1+i)^N} \tag{2.6}$$

Calculate first month interest:

$$i_m\_1 = \frac{(i_m}{31 * 100} * days\_left \tag{2.7}$$

Calculate Interest Component:

$$total\_interest = FV - PV \tag{2.8}$$

Calculate Total Credit:

$$total\_credit = Annuity * Period \tag{2.9}$$

# 3 Loan Payment Class - Python

The loan.py source code contains:

1. LoanCalculator class

2. LoanCalculatorTest class for Unit Test

All files can be retrieved from here: `https://github.com/vitaliAI/pricecasestudy`

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Created on Mon Jun 11 21:32:16 2018

@author: vmueller
"""

class LoanCalculator(object):
    """
    Loan Payment = Amount / Discount Factor
    P = A / D

    To calculate the discount factor,
    loan payment, total credit and total interest we need the following arguments

    N number of the periodic payments  = Payments per year times number of years
    Periodic Interest Rate (i) = APR divided by 12
    Discount Factor (D) = {[(1 + i) ^n] - 1} / [i(1 + i)^n]
    """

    def __init__(self, amount, period, apr, start=1, end=1):
        self.months = 12
        self.amount = amount
        self.period = period
        self.apr = apr
        self.i = self.apr / (self.months * 100)
        self.start = start
        self.end = end
        self.special_case = False
        if start != end:
            self.special_case = True

    def discount_factor(self):
        i = self.i
        discount_factor_v = (((1 + i) ** self.period) - 1) / (i * (1 + i) ** self.period)
        return discount_factor_v

    def loan_payment(self):
        if not self.special_case:
```

```
            return round(self.amount / self.discount_factor(), 2)
        else:
            fv = self.future_value(self.first_month_interest())
            i = self.apr / (100 * 12)
            return round(self.present_value(fv, i) / self.discount_factor(), 2)

    def total_credit(self):
        return round(self.loan_payment() * self.period, 2)

    def total_interest(self):
        return round(self.total_credit() - self.amount, 2)

    def first_months_days_to_pay(self):
        delta_days = 0
        if self.start < self.end:
            delta_days = self.end - self.start
        elif self.start > self.end:
            delta_days = 31 - self.start + self.end
        return delta_days

    def first_month_interest(self):
        i_daily = self.i / (31 * 100)
        i_first_month = i_daily *  self.first_months_days_to_pay()
        return i_first_month

    def future_value(self, i, n = 1):
        fv = self.amount * (1 + i)**n
        return fv

    def present_value(self, fv, i, n = 1):
        pv = fv * (1/ (1 + i)**n)
        return pv

    def display_all_information(self):
        if not self.special_case:
            print("\n")
            print("---- Normal Loan Case ----\n")
            print("Annuity: ", round(self.amount / self.discount_factor(), 2))
            print("Total Credit: ", self.total_credit())
            print("Total Interest: ", self.total_interest())
        else:
            fv = self.future_value(self.first_month_interest())
            print("\n")
            print("---- Special Loan Case ----\n")
```

```
            print("Monthly Interest Rate: ", self.i)
            print("Present Value: ", round(self.present_value(fv, self.i), 2))
            print("Annuity: ", round(self.present_value(fv, self.i) / self.discount_factor
            print("Total Credit: ", self.total_credit())
            print("Total Interest: ", self.total_interest())
            print("Days in the first month: ", self.first_months_days_to_pay())

from unittest import TestCase


class LoanCalculatorTest(TestCase):

    def setUp(self):
        # Normal Case
        self.loan_1 = LoanCalculator(3000, 12, 12, 11, 11)
        # Special case
        self.loan_2 = LoanCalculator(3000, 12, 12, 2, 17)


    def test_loan_payment(self):
        annuity = self.loan_1.loan_payment()
        # Verify the Amount
        self.assertEqual(266.55, annuity)

    def test_loan_payment_2(self):
        annuity = self.loan_2.loan_payment()
        # Verify the Amount
        self.assertEqual(263.92, annuity)

    def test_days_first_month(self):
        days = self.loan_2.first_months_days_to_pay()
        self.assertEqual(15, days)

    def run_all_tests(self):
        self.test_loan_payment()
        self.test_loan_payment_2()
        self.test_days_first_month()

if __name__ == "__main__":
    loan = LoanCalculator(3000, 12, 12, 11, 11)
    loan.display_all_information()
    loan = LoanCalculator(3000, 12, 12, 2, 17)
    loan.display_all_information()
```

```
t = LoanCalculatorTest()
t.setUp()
t.run_all_tests()
```

# 4 RESULTS:

**Normal Loan Case**

Annuity: 266.55
Total Credit: 3198.6
Total Interest: 198.6

**Special Loan Case**

Monthly Interest Rate: 0.01
Present Value: 2970.44
Annuity: 263.92
Total Credit: 3167.04
Total Interest: 167.04
Days in the first month: 15

# 5 PRICING CASE STUDY APP BUILD WITH DJANGO

You can test the Application by visiting the following link:

`https://pricingcasestudy.herokuapp.com/`