Description

Let's compute all the parameters of the loan. There are at least two kinds of loan: those with annuity payment and those with differentiated payment. In this stage, you are going to calculate only the former. **Annuity** type of payment consists in paying a fixed sum of money at specified intervals, for example, each month or each year. The **annuity payment** amount is precisely this fixed sum of money that you need to pay at regular intervals.

Let's assume that annuity payments are made monthly. Thus, we can use the following formula to calculate the **monthly payment**:

$$A_{\text{ordinary annuity}} = P \cdot \frac{i \cdot (1+i)^n}{(1+i)^n - 1}$$

Where:

A = annuity payment;

P = loan principal;

i = nominal (monthly) interest rate. Usually, it is 1/12 of the annual interest rate and is a floating value, not

a percentage. For example, if your annual interest rate = 12%, then i = 0.01;

n = number of payments. This is usually the number of months in which repayments will be made.

So far, in this stage you are interested in four values: the number of monthly payments required to repay the loan, the monthly payment amount, the loan principal, and the loan interest. Each of these values can be calculated if others are known:

Loan principal:

$$P = \frac{A}{\left(\frac{i \cdot (1+i)^n}{(1+i)^n - 1}\right)}$$

The number of payments:

$$n = \log_{1+i}\left(\frac{A}{A - i \cdot P}\right)$$

As the number of input parameters increases from this stage onwards, it might be convenient to use command-line arguments to input all known values. The `argparse` module can help you parse them. You can run your loan calculator via command line using arguments like this:

```
python creditcalc.py --principal=1000000 --periods
```

The missing parameter is the one that needs to be calculated. As you can see above, only the values for principal, periods and interest are given. This means that the program should calculate the monthly

payment amount, because it was not specified in arguments.

Check the description and examples below for more details.

Objectives

In this stage, you should change the behavior of the calculator:

1. First, you should parse the provided parameters to define, which ones are known and which one is missing.
2. Compute the missing value using the formulas mentioned above. The calculator should be able to calculate the number of monthly payments, the monthly payment amount, and the loan principal.
3. Finally, output the value users wanted to compute.

The final version of your program is supposed to work from the command line and parse the following arguments:

- `--payment` is the payment amount. It can be calculated using the provided principal, interest, and number of months
- `--principal` You can get its value if you know the interest, annuity payment, and number of months.
- `--periods` denotes the number of months needed to repay the loan. It's calculated based on the interest, annuity payment, and principal.
- `--interest` is specified without a percent sign. Note that it can accept a floating-point value.

> Our loan calculator can't calculate the interest, so it must always be provided.

Please be careful converting "**X months**" to "**Y years and Z months**". Avoid writing "0 years and 11 months" (output "11 months" instead) and "1 years and 0 months" (output "1 year" instead).

Examples

The greater-than symbol followed by a space (`> `) represents the user input. Note that this is not part of the input.

Example 1: *calculating the number of monthly payments*

```
> python creditcalc.py --principal=1000000 --payme
It will take 8 years and 2 months to repay this lo
```

Let’s take a closer look at **Example 1**.

You know the loan principal, the monthly payment (annuity payment), and the loan interest. You didn't provide the `--periods` argument, so you want to calculate the number of monthly payments. What do you do?

1) You need to know the nominal interest rate. It is calculated like this:

$$i = \frac{10\%}{12 \cdot 100\%} = 0.008333...$$

2) Now you can calculate the number of months:

$$n = \log_{1+0.008333...}\left(\frac{15000}{15000 - 0.008333... \cdot 1000000}\right)$$

$$97.71...$$

3) You need an integer number, so let's round it up. Notice that the user will pay the same amount every month for 97 months, and in the 98th month the user will pay *0.71...* of the monthly payment. So, there are 98 months to pay.

$$n = 98$$

4) Finally, you need to convert "98 months" to "8 years and 2 months" so that it is more readable and understandable for the user.

Example 2: *calculating the monthly payment (the annuity payment)*

```
> python creditcalc.py --principal=1000000 --perio
Your monthly payment = 21248!
```

Example 3: *calculating the loan principal*

```
 python creditcalc.py --payment=8721.8 --periods=1
Your loan principal = 800000!
```

See hint

Write a program

Report a typo

Code Editor IDE

IDE integration ✕ IDE not installed or not running

Unlock your coding potential with IDE

Experience the power of professional development. IDEs offer smart code assistance, faster coding, and a smoother workflow. Please open a JetBrains IDE or or install and configure it from this page

Solve in IDE | Learn more about IDE

Continue

Solutions (69)

🎉 Correct

Seems like you've got all the theory down, way to go!

513 learners liked this problem. 76 didn't like it. What about you?

😍 🙂 😐 🙁 😡

Topics in stage

✔ List | ✔ String formatting | ✔ Indexes

- PyCharm basics
- Type casting
- Command line arguments
- Argparse module
- Running Python applications in PyCharm

This content was created 1 year ago and updated 7 days ago.
Share your feedback below in comments to help us improve it!

Comments (317) Hints (35) Useful links (3) Solu

Help us develop a better product for you and other learners!

I agree to receive invites to research activities (interviews, surveys)

All courses
Top courses
Beginner-friendly
Career paths
Python
Java
JavaScript
Kotlin

Full catalog

Go
Android
C++
Generative AI
Math
Frontend
SQL and Databases
Data Science
Bioinformatics
Backend

DevOps
Data Analysis
Machine Learning
Drafts