What you'll do in this stage 4/4: Differentiate payment

Project: Loan Calculator

3835 users solved this stage. Latest completion was **8 minutes ago**.

Stages completed: **4 / 4**

Topics you need to learn in order to complete the current stage:

Declaring a function 2 | List 4 | For loop 1 | While loop 4 | Introduction to operating systems 1 | Command line overview 1 | Parameters and options 1 | Indexes 4 | Type casting 1 | Program execution 1 | Errors 1 | Command line arguments 1 | Argparse module 1

Use the material from these topics and the skills you've learned to successfully complete this stage of the project.

You will be working on the following in this project stage

Make the loan calculator able to work with different types of payment and accept command-line arguments. Hide details ↑

Description

Finally, let's add to our calculator the capacity to compute differentiated payments. We'll do this for the type of repayment where the loan principal is reduced by a constant amount each month. The rest of the monthly payment goes toward interest repayment and it is gradually reduced over the term of the loan. This means that the payment is different each month. Let's look at the formula:

$$D_m = \frac{P}{n} + i * \left(P - \frac{P * (m-1)}{n}\right)$$

Where:

D_m = mth differentiated payment;

P = the loan principal;

i = nominal interest rate. This is usually 1/12 of the annual interest rate, and it's usually a float value, not a percentage. For example, if our 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.

m = current repayment month.

The user has to input a lot of parameters, so it might be convenient to use command-line arguments.

You can run your loan calculator via command line like this:

```
python creditcalc.py
```

Using command-line arguments you can run your program this way:

```
python creditcalc.py --type=diff --principal=1000000 --periods=10 --interest=10
```

This way, your program can get different values without prompting the user to input them. It can be useful when you are developing your program and trying to find a mistake, and you want to run the program with the same parameters again and again. It's also convenient if you made a mistake in a single parameter: you don't have to input all the other values again.

Objectives

In this stage, you are going to implement the following features:

- Calculation of differentiated payments. To do this, the user can run the program specifying interest, number of monthly payments, and loan principal.

- Ability to calculate the same values as in the previous stage for annuity payment (principal, number of monthly payments, and monthly payment amount). The user specifies all the known parameters with command-line arguments, and one parameter will be unknown. This is the value the user wants to calculate.
- Handling of invalid parameters. It's a good idea to show an error message if the user enters invalid parameters (they are discussed in detail below).

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

- `--type` indicates the type of payment: `"annuity"` or `"diff"` (differentiated). If `--type` is specified neither as `"annuity"` nor as `"diff"` or not specified at all, show the error message.

```
> python creditcalc.py --principal=1000000 --periods=60 --interest=10
Incorrect parameters
```

- `--payment` is the monthly payment amount. For `--type=diff`, the payment is different each month, so we can't calculate months or principal, therefore a combination with `--payment` is invalid, too:

```
> python creditcalc.py --type=diff --principal=1000000 --interest=10 --payment=100000
Incorrect parameters
```

- `--principal` is used for calculations of both types of payment. 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. These parameters are incorrect because `--interest` is missing:

```
> python creditcalc.py --type=annuity --principal=100000 --payment=10400 --periods=8
Incorrect parameters
```

You may have noticed that for differentiated payments you will need 4 out of 5 parameters (excluding payment), and the same is true for annuity payments (the user will be calculating the number of payments, the payment amount, or the loan principal). Thus, you should also display an error message when fewer than four parameters are provided:

```
> python creditcalc.py --type=annuity --principal=1000000 --payment=104000
Incorrect parameters
```

You should also display an error message when negative values are entered:

```
> python creditcalc.py --type=diff --principal=30000 --periods=-14 --interest=10
Incorrect parameters
```

The only thing left is to compute the overpayment: the amount of interest paid over the whole term of the loan. Voila: you have a real loan calculator!

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 differentiated payments*

```
> python creditcalc.py --type=diff --principal=1000000 --periods=10 --interest=10
Month 1: payment is 108334
Month 2: payment is 107500
Month 3: payment is 106667
Month 4: payment is 105834
Month 5: payment is 105000
Month 6: payment is 104167
Month 7: payment is 103334
Month 8: payment is 102500
Month 9: payment is 101667
Month 10: payment is 100834

Overpayment = 45837
```

In this example, the user wants to take a loan with differentiated payments. You know the principal, the count of periods, and interest, which are 1,000,000, 10 months, and 10%, respectively.

The calculator should calculate payments for all 10 months. Let's look at the formula above. In this case:

$P = 1000000$
$n = 10$
$i = \frac{interest}{12 * 100\%} = \frac{10\%}{12 * 100\%} = 0.008333...$

Now let's calculate the payment for the first month:

$$D_1 = \frac{P}{n} + i * \left(P - \frac{P * (m-1)}{n}\right) = \frac{1000000}{10} + 0.008333... * \left(1000000 - \frac{1000000 * (1-1)}{10}\right) = 108333.333...$$

The second month ($m = 2$):

$$D_2 = \frac{P}{n} + i * \left(P - \frac{P * (m-1)}{n}\right) = \frac{1000000}{10} + 0.008333... * \left(1000000 - \frac{1000000 * (2-1)}{10}\right) = 107500$$

The third month ($m = 3$):

$$D_3 = \frac{P}{n} + i * \left(P - \frac{P * (m-1)}{n}\right) = \frac{1000000}{10} + 0.008333... * \left(1000000 - \frac{1000000 * (3-1)}{10}\right) = 106666.666...$$

And so on. You can see other monthly payments above.

Your loan calculator should output monthly payments for every month as in the first stage. Also, round up all floating-point values.

Finally, your loan calculator should add up all the payments and subtract the loan principal so that you get the overpayment.

Example 2: *calculate the annuity payment for a 60-month (5-year) loan with a principal amount of 1,000,000 at 10% interest*

```
> python creditcalc.py --type=annuity --principal=1000000 --periods=60 --interest=10
Your annuity payment = 21248!
Overpayment = 274880
```

Example 3: *fewer than four arguments are given*

```
> python creditcalc.py --type=diff --principal=1000000 --payment=104000
Incorrect parameters.
```

Example 4: *calculate differentiated payments given a principal of 500,000 over 8 months at an interest rate of 7.8%*

```
> python creditcalc.py --type=diff --principal=500000 --periods=8 --interest=7.8
Month 1: payment is 65750
Month 2: payment is 65344
Month 3: payment is 64938
Month 4: payment is 64532
Month 5: payment is 64125
Month 6: payment is 63719
Month 7: payment is 63313
Month 8: payment is 62907

Overpayment = 14628
```

Example 5: *calculate the principal for a user paying 8,722 per month for 120 months (10 years) at 5.6% interest*

```
> python creditcalc.py --type=annuity --payment=8722 --periods=120 --interest=5.6
Your loan principal = 800018!
Overpayment = 246622
```

Example 6: *calculate how long it will take to repay a loan with 500,000 principal, monthly payment of 23,000, and 7.8% interest*

```
> python creditcalc.py --type=annuity --principal=500000 --payment=23000 --interest=7.8
It will take 2 years to repay this loan!
Overpayment = 52000
```

Report a typo

Go to study plan and keep learning