## 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:

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

Where:

$D_m$ = $m$th differentiated payment;

$P$ = the loan principal;

$i$ = nominal interest rate. This is usually 1/12 of the annual interest rate and is 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.

In this stage, the user has to provide more inputs, so your program should be able to parse new command-line arguments. Let's add the `--type` one, so now the calculator should be run with 4 arguments:

```
python creditcalc.py --type=diff --principal=10000
```

The `--type` argument is indicating the type of payment: `"annuity"` or `"diff"` (differentiated). It must always be provided in any run.

Also, it is possible that the user will make a mistake in the provided input. The program should not fail, but inform the user, that he has provided `"Incorrect parameters"`.

- `--type` is specified neither as `"annuity"` nor as `"diff"` or not specified at all:

```
> python creditcalc.py --principal=1000000 --
Incorrect parameters
```

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

```
> python creditcalc.py --type=diff --principa
```

```
2 Incorrect parameters
```

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

```
1 > python creditcalc.py --type=annuity --princ
2 Incorrect parameters
```

- 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:

```
1 > python creditcalc.py --type=annuity --princ
2 Incorrect parameters
```

- Negative values should not be provided:

```
1 > python creditcalc.py --type=diff --principa
2 Incorrect parameters
```

## 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).
- Handling of invalid parameters. It's a good idea to show an error message if the user enters invalid parameters.
- The only thing left is to compute the overpayment: the amount of interest paid over the whole term of the loan.

## 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=100
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:

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

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

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

The second month ($m = 2$):

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

The third month ($m = 3$):

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

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=
Your annuity payment = 21248!
Overpayment = 274880
```

**Example 3:** *fewer than four arguments are given*

```
> python creditcalc.py --type=diff --principal=100
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=500
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
```

```
10
11  Overpayment = 14628
```

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

```
1  > python creditcalc.py --type=annuity --payment=87
2  Your loan principal = 800018!
3  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*

```
1  > python creditcalc.py --type=annuity --principal=
2  It will take 2 years to repay this loan!
3  Overpayment = 52000
```

See hint

**Write a program**

Report a typo

Code Editor IDE

```
1  import argparse
2  from math import ceil, log
3  import sys
4
5
6  def validate_argument(args):
7      is_invalid = args.type not in (
8      is_invalid = is_invalid or args.
9      is_invalid = is_invalid or args.
```

```
    is_invalid = is_invalid or (args
    if is_invalid:
        print('Incorrect parameters'
        sys.exit()


if __name__ == '__main__':
    parser = argparse.ArgumentParser
    parser.add_argument("--type", ty
    parser.add_argument("--payment",
    parser.add_argument("--principal
    parser.add_argument("--periods",
    parser.add_argument("--interest"
    args = parser.parse_args()

    validate_argument(args)

    i = args.interest / 1200
    periods = args.periods
    principal = args.principal
    payment = args.payment
    if args.type == 'diff':
        total = 0
        for m in range(periods):
            d = ceil(principal / per
            print(f'Month {m + 1}: p
            total += d
        print("")
        print(f'Overpayment = {total
    elif principal == 0:
        principal = round(payment /
        print(f'Your loan principal
        print(f'Overpayment = {payme
    elif periods != 0:
        payment = ceil(principal * i
        print(f'Your annuity payment
        print(f'Overpayment = {payme
    else:
        t  = ceil(log(payment / (pay
        y, m = divmod(t, 12)

        years = f"{y} year{'s' if y

```

```
53          months = f"{m} month{'s' if
54
55          time = years + ('and ' if y
56
57          print(f'It will take {time}
```

Continue Solve again

Solutions (259)

🎉 Correct

It was a tricky task, but you nailed it!

**407** learners liked this problem. **64** didn't like it. **What about you?**

😍 🙂 😐 ☹️ 😡

**Topics in stage**

- Declaring functions
- While loop
- Program execution
- Errors
- For loop
- Debugging Python applications in PyCharm

Comments (656) Hints (42) Useful links (11) 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

---

Resources
Blog
University

Subscription
For Business
Pricing

Hyperskill
About
Careers
For Content Creators

Support
Help Center
Terms

GET IT ON Google Play

Download on the App Store

😎 Become beta tester

Be the first to see what's new


Hyperskill