第一次作业

买房问题

假设用户想购买一套总价100万元的房子，首付10万元，每月等额还款，年贷款利率5%，计划240个月还清，请计算用户每月需要偿还的银行贷款金额。

具体要求

1. 从键盘输入各项参数，以Enter键或Space键作为输入结束标志。最后一次输入完成后，程序自动显示计算结果。
2. 输入和输出要有适当的文字提示。
3. 从文件中读取多个用户条件并计算每个用户的每月付款，将结果保存到新文件中。
4. 最终的月还款额四舍五入到最接近的人民币金额。

代码

```
"""
* coding=utf-8
* python=3.11
* File: Experiment1.py
* Author: donghy23@mails.tsinghua.edu.cn
* Created: 2024-7-12
* Repo: https://github.com/FHYQ-Dong/Tsinghua-Program-Design-Assignment-
3/blob/main/Experiment1/Experiment1.py
"""


import argparse
import json

def calMonthPayment(totalPrice: float, downPayment: float, interestRate_perYear:
float, loanDuration: int) -> float:
    """计算每月还款额
    每年

    ### Args:
        totalPrice (float): 总价（单位：元）
        downPayment (float): 首付（单位：元）
        interestRate_perYear (float): 年利率（单位：%）
        loanDuration (int): 贷款期限（单位：月）

    ### Returns:
        float: 每月还款额（单位：元），保留两位小数（四舍五入）
    """

    loanAmount = totalPrice - downPayment
    interestRate_perMonth = interestRate_perYear / 12 / 100
    monthPayment = loanAmount * interestRate_perMonth * (1 + 1 / ((1 +
interestRate_perMonth)**loanDuration - 1))
    return "%.2f" % monthPayment
```

```
def main():
    """
    Calculate monthly payment based on the provided arguments or input file.

    ### Arguments:
    -t, --totalPrice: Total price of the loan (unit: RMB)
    -p, --downPayment: Down payment amount (unit: RMB)
    -i, --interestRate: Interest rate per year (unit: %)
    -d, --loanDuration: Loan duration in months
    -f, --file: Input file (JSON format)
    -o, --output: Output file (JSON format)
    -j, --showTemplateJsonFormat: Show template JSON format of input file

    ### Returns:
    - If -j or --showTemplateJsonFormat is provided, it prints the template JSON format of the input file.
    - If -f or --file is provided, it calculates the monthly payment for each item in the input file and either writes the output to the specified output file or prints it to the console.
    - If -t, -p, -i, and -d are provided, it calculates and prints the monthly payment based on the provided arguments.
    - If no arguments are provided, it prints the help message.
    """

    parser = argparse.ArgumentParser(description='Calculate monthly payment')
    parser.add_argument('-t', '--totalPrice', type=float, help='Total price (unit: RMB)')
    parser.add_argument('-p', '--downPayment', type=float, help='Down payment (unit: RMB)')
    parser.add_argument('-i', '--interestRate', type=float, help='Interest rate per year (unit: %%)')
    parser.add_argument('-d', '--loanDuration', type=int, help='Loan duration (unit: month)')
    parser.add_argument('-f', '--file', type=argparse.FileType('r'), help='Input file (JSON)')
    parser.add_argument('-o', '--output', type=argparse.FileType('w'), help='Output file (JSON)')
    parser.add_argument('-j', '--showTemplateJsonFormat', action='store_true', help='Show template JSON format of input file')

    try:
        args = parser.parse_args()
    except argparse.ArgumentError as e:
        print(e)
        parser.print_help()
        exit(1)

    if args.showTemplateJsonFormat:
        templateInput = [
            {
                "name": "Name1",
                "totalPrice": 1000000,
                "downPayment": 200000,
```

```
                "interestRate": 5,
                "loanDuration": 240
            },
            {
                "name": "Name2",
                "totalPrice": 2000000,
                "downPayment": 400000,
                "interestRate": 4,
                "loanDuration": 120
            }
        ]
        print(json.dumps(templateInput, indent=4))
        return

    if args.file:
        data = json.load(args.file)
        output = []
        for item in data:
            item['monthlyPayment'] = calMonthPayment(
                item['totalPrice'],
                item['downPayment'],
                item['interestRate'],
                item['loanDuration']
            )
            output.append(item)
        if args.output:
            args.output.write(json.dumps(output, indent=4))
        else:
            for i, item in enumerate(output):
                print(('\n' + '-'*48 + '\n') * bool(i))
                print(f'Name: {item["name"]}')
                print(f'Total price: {item["totalPrice"]}')
                print(f'Down payment: {item["downPayment"]}')
                print(f'Interest rate: {item["interestRate"]}')
                print(f'Loan duration: {item["loanDuration"]}')
                print(f'Monthly payment: {item["monthlyPayment"]}')
    else:
        if args.totalPrice and args.downPayment and args.interestRate and
args.loanDuration:
            monthPayment = calMonthPayment(args.totalPrice, args.downPayment,
args.interestRate, args.loanDuration)
            print(f'Monthly payment: {monthPayment}')
        else:
            parser.print_help()


if __name__ == '__main__':
    main()
```

结果

输入1：

```
python .\Experiment1.py -t 100000 -p 10000 -i 3 -d 240
```

输出1：

```
Monthly payment: 499.14
```

输入2：

```
python .\Experiment1.py -f .\input.json
```

输出2：

```
Name: AAA
Total price: 1000000
Down payment: 100000
Interest rate: 5
Loan duration: 240
Monthly payment: 5939.60

-------------------------------------------------

Name: BBB
Total price: 2000000
Down payment: 200000
Interest rate: 5
Loan duration: 240
Monthly payment: 11879.20

-------------------------------------------------

Name: CCC
Total price: 1000000
Down payment: 100000
Interest rate: 5
Loan duration: 120
Monthly payment: 9545.90

-------------------------------------------------

Name: DDD
Total price: 2000000
Down payment: 200000
Interest rate: 1
Loan duration: 12
Monthly payment: 150813.74

-------------------------------------------------

Name: EEE
```

```
Total price: 1000000
Down payment: 100000
Interest rate: 10
Loan duration: 120
Monthly payment: 11893.57
```