Міністерство освіти і науки України  
Національний технічний університет України

«Київський політехнічний інститут  ім. І. Сікорського»

Кафедра інженерії програмного забезпечення в енергетиці

Практична робота № 2

з курсу: «Проектування кібер-фізичних систем»

**Виконав:**студент 2-го курсу,  
групи ТВ-32  
Щур Роман Ігорович

 Посилання на GitHub репозиторій:

https://github.com/RomanSchur/DCPS\_PR-1

Київ 2025

**Завдання**

Написати программу на довільній мові програмування (Java, Python, JS, C, C#, С++...), яка б на вхід отримувала дані двофазного лічильника "день-ніч" та за різним тарифом, рахувала різницю між попередніми показниками та поточними, виставляючи рахунок.

У випадку отримання даних від нового лічильника, вони мають дописатись у масив з усіма даними.

Після опрацювання нових даних, вони мають записатись замість старих як наступні початкові.

У випадку надходження менших даних, ніж були попередні, накрутити задане число квт (нехай, 100 день та 80 ніч) та видати рахунок.

До програми мають бути написані тести:

\* оновлення показників вже існуючого лічильника

\* отримання показників від нового лічильника

\* отримання показників з заниженими нічними показниками

\* отримання показників з заниженими денними показниками

\* отримання показників з заниженими нічними та денними показниками

Вхідні дані, які мають легко змінюватись в одному місці:  
\* тарифи на день та ніч (наприклад, у константах програми, у файлі, у БД)  
\* кількість накручуваних квт на день та ніч (зручно там же, де і тарифи)  
\* початковий масив даних з номерами лічильників, датами та попередніми тарифами (в залежності від обраної стратегії програми, аналогічно, у списку-мапі в коді, у файлі, у БД)  
\* массив з минулими обрахунками вартості електроенергії (так само)  
  
Вихідні дані:   
обновлений масив з вартістю електроенергії длі всіх лічильників.   
  
Додатки до програми:  
а) Зберігати історію всіх показників лічильників. \* (3 бали)  
б) Зберігати всі вхідні та вихідні дані в БД (в SQL - MySQL, Postgres чи ін - 5 балів), але за бажанням можна в NoSQL (Mongo чи ін - 5 + 5 балів) \*   
в) Поставляти нові дані в програму через чергу зі вхідними даними (Kafka, RabbitMQ, ін) Нові дані треба геренурвати. На додаткові бали можна зробити мікросервісну архітектуру. (5 балів)  
г) UI - якщо реалізується консольний чи справжній інтерфейс, там варто передбачити User Friendly поведінку для попередження користувача про накрутку показників та дати йому можливість її оминути вводячи коректні дані.  
  
\* Реалізуючи збереження, не забудьте реалізувати зчитування історії. (Кому ж потрібна валіза без ручки?) Проконтролюйте, що зберігаєте не прислані показники, а коректні показники, на основі яких рахуються квитанції.

**Програмна реалізація**

Програма написана мовою програмування Python. Вона представлена у вигляді однієї функції, яка приймає список з інформацією про певний лічильник. За допомогою циклів, умовних операторів і додаткових змінних виконується обчислення спожитої електроенергії як вдень, так і вночі. Потім, використовуючи тарифи, вираховується ціна за один день для вхідного лічильника. Після цього інформація про ціну та споживання кВт виводиться на екран у текстовому вигляді, а список початкових даних перезаписується, замінюючи старі дані про вхідний лічильник на нові. Для зручності та спрощення, тарифи на день і ніч, кількість кВт, що накручуються вдень і вночі, а також початковий масив даних з номерами лічильників, датами й попередніми тарифами, збережені в константних змінних. Також програма здійсню запис показників лічильників у файл.

import copy  
standard\_tariff\_day = 26.4  
standard\_tariff\_night = 21.6  
nakrytka\_day = 100  
nakrytka\_night = 80  
  
Information\_storage = [  
 {"ID": 1, "Дата": "01.03.2025", "Денне\_споживання(кВт)": 40.5, "Нічне\_споживання(кВт)": 10.5,"Загальна сума оплати на теперішню дату":1296},  
 {"ID": 2, "Дата": "01.03.2025", "Денне\_споживання(кВт)": 50.1, "Нічне\_споживання(кВт)": 20,"Загальна сума оплати на теперішню дату":2000},  
 {"ID": 3, "Дата": "01.03.2025", "Денне\_споживання(кВт)": 40, "Нічне\_споживання(кВт)": 20.5,"Загальна сума оплати на теперішню дату":3000},  
]  
  
def Calculate\_price(new\_data):  
 old\_data = next((i for i in Information\_storage if i["ID"] == new\_data["ID"]), None)  
 old\_index = next((index for index, i in enumerate(Information\_storage) if i["ID"] == new\_data["ID"]), None)  
 new\_data\_copy = copy.deepcopy(new\_data)  
  
 if old\_data is None:  
 res\_price=new\_data\_copy['Денне\_споживання(кВт)']\*standard\_tariff\_day+new\_data\_copy['Нічне\_споживання(кВт)']\*standard\_tariff\_night  
 print(f"-------------------------------- Результат обрахунку для лічильника № {new\_data\_copy['ID']} --------------------------------")  
 print(f"Станом на {new\_data\_copy['Дата']} лічильник №{new\_data\_copy['ID']} нарахував: {new\_data\_copy['Денне\_споживання(кВт)']} кВт вночі.")  
 print(f"Станом на {new\_data\_copy['Дата']} лічильник №{new\_data\_copy['ID']} нарахував: {new\_data\_copy['Нічне\_споживання(кВт)']} кВт вночі.")  
 print(f"--------------------------------------------------------------------------------------------------------------------")  
 print(f"Плата за {new\_data\_copy['Дата']} для лічильника №{new\_data\_copy['ID']} становить: {res\_price:.2f} грн")  
 new\_data\_copy['Загальна сума оплати на теперішню дату'] = res\_price  
 Information\_storage.append(new\_data\_copy)  
 return

Табл. №1 – Програмний код завдання.

if new\_data\_copy["Денне\_споживання(кВт)"] < old\_data["Денне\_споживання(кВт)"]:  
 new\_data\_copy["Денне\_споживання(кВт)"] += nakrytka\_day  
 differens\_day = new\_data\_copy["Денне\_споживання(кВт)"] - old\_data["Денне\_споживання(кВт)"]  
 price\_day = differens\_day \* standard\_tariff\_day  
 else:  
 price\_day = (new\_data\_copy["Денне\_споживання(кВт)"] - old\_data["Денне\_споживання(кВт)"]) \* standard\_tariff\_day  
  
 if new\_data\_copy["Нічне\_споживання(кВт)"] < old\_data["Нічне\_споживання(кВт)"]:  
 new\_data\_copy["Нічне\_споживання(кВт)"] += nakrytka\_night  
 differens\_night=new\_data\_copy["Нічне\_споживання(кВт)"]-old\_data["Нічне\_споживання(кВт)"]  
 price\_night = differens\_night \* standard\_tariff\_night  
 else:  
 price\_night = (new\_data\_copy["Нічне\_споживання(кВт)"] - old\_data["Нічне\_споживання(кВт)"]) \* standard\_tariff\_night  
  
 res\_day\_price = price\_night + price\_day  
 res\_price = old\_data["Загальна сума оплати на теперішню дату"]+res\_day\_price  
  
 new\_data\_copy['Загальна сума оплати на теперішню дату']=res\_price  
  
 print(f"-------------------------------- Результат обрахунку для лічильника № {new\_data\_copy['ID']} --------------------------------")  
 print(f"Станом на {old\_data['Дата']} лічильник №{old\_data['ID']} нарахував: {old\_data['Денне\_споживання(кВт)']} кВт вдень.")  
 print(f"Станом на {old\_data['Дата']} лічильник №{old\_data['ID']} нарахував: {old\_data['Нічне\_споживання(кВт)']} кВт вночі.")  
 print(f"--------------------------------------------------------------------------------------------------------------------")  
 print(f"Станом на {new\_data\_copy['Дата']} лічильник №{new\_data\_copy['ID']} нарахував: {new\_data\_copy['Денне\_споживання(кВт)']} кВт вдень.")  
 print(f"Станом на {new\_data\_copy['Дата']} лічильник №{new\_data\_copy['ID']} нарахував: {new\_data\_copy['Нічне\_споживання(кВт)']} кВт вночі.")  
 print(f"--------------------------------------------------------------------------------------------------------------------")  
 print(f"Плата за {new\_data\_copy['Дата']} для лічильника №{new\_data\_copy['ID']} становить: {res\_day\_price:.2f} грн")  
  
 Information\_storage[old\_index] = copy.deepcopy(new\_data\_copy)  
test\_data = [{"ID": 1, "Дата": "02.03.2025", "Денне\_споживання(кВт)": 30.5, "Нічне\_споживання(кВт)": 8.5 ,"Загальна сума оплати на теперішню дату":0},  
 {"ID": 2, "Дата": "02.03.2025", "Денне\_споживання(кВт)": 43.1, "Нічне\_споживання(кВт)": 9.5 ,"Загальна сума оплати на теперішню дату":0},  
 {"ID": 3, "Дата": "02.03.2025", "Денне\_споживання(кВт)": 55.5, "Нічне\_споживання(кВт)": 25.5, "Загальна сума оплати на теперішню дату":0},  
 {"ID": 4, "Дата": "02.03.2025", "Денне\_споживання(кВт)": 66, "Нічне\_споживання(кВт)": 45.5, "Загальна сума оплати на теперішню дату":0},]  
  
for i in test\_data:  
 Calculate\_price(i)  
print("\n")  
for Element in Information\_storage:  
 print(f"Лічильник №{Element['ID']}, Дата: {Element['Дата']}, "  
 f"Денне споживання: {Element['Денне\_споживання(кВт)']} кВт, "  
 f"Нічне споживання: {Element['Нічне\_споживання(кВт)']} кВт "  
 f"Загальна сума оплати на теперішню дату: {Element['Загальна сума оплати на теперішню дату']} грн ")  
  
with open('hist.txt', 'a') as file:  
 for Element in Information\_storage:  
 file.write(f"Лічильник №{Element['ID']}, Дата: {Element['Дата']}, "  
 f"Денне споживання: {Element['Денне\_споживання(кВт)']} кВт, "  
 f"Нічне споживання: {Element['Нічне\_споживання(кВт)']} кВт "  
 f"Загальна сума оплати на теперішню дату: {Element['Загальна сума оплати на теперішню дату']} грн\n")  
 file.write("\n")

Зображення, що містить текст, знімок екрана, Шрифт, меню

Вміст, створений ШІ, може бути неправильним.

Рис №1 – Результат виконання програми

Зображення, що містить текст, знімок екрана, Шрифт, число

Вміст, створений ШІ, може бути неправильним.

Рис №2 – Результат запису історії

Додатково для перевірки коректності роботи програми були створені unit-тести для кожної з п’яти умов,а саме:

\* оновлення показників вже існуючого лічильника  
\* отримання показників від нового лічильника  
\* отримання показників з заниженими нічними показниками  
\* отримання показників з заниженими денними показниками  
\* отримання показників з заниженими нічними та денними показниками

#\* тест оновлення показників вже існуючого лічильника  
import unittest  
import copy  
  
class test1\_pr2(unittest.TestCase):  
 def test(self):  
 standard\_tariff\_day = 26.4  
 standard\_tariff\_night = 21.6  
 nakrytka\_day = 100  
 nakrytka\_night = 80  
  
 Information\_storage = [{"ID": 1, "Дата": "01.03.2025", "Денне\_споживання(кВт)": 40.5, "Нічне\_споживання(кВт)": 10.5, "Загальна сума оплати на теперішню дату": 1296},]  
 def Calculate\_price(new\_data):  
 old\_data = next((i for i in Information\_storage if i["ID"] == new\_data["ID"]), None)  
 old\_index = next((index for index, i in enumerate(Information\_storage) if i["ID"] == new\_data["ID"]), None)  
 new\_data\_copy = copy.deepcopy(new\_data)  
  
 if old\_data is None:  
 res\_price = new\_data\_copy['Денне\_споживання(кВт)'] \* standard\_tariff\_day + new\_data\_copy['Нічне\_споживання(кВт)'] \* standard\_tariff\_night  
 new\_data\_copy['Загальна сума оплати на теперішню дату'] = res\_price  
 Information\_storage.append(new\_data\_copy)  
 return

if new\_data\_copy["Денне\_споживання(кВт)"] < old\_data["Денне\_споживання(кВт)"]:  
 new\_data\_copy["Денне\_споживання(кВт)"] += nakrytka\_day  
 differens\_day = new\_data\_copy["Денне\_споживання(кВт)"] - old\_data["Денне\_споживання(кВт)"]  
 price\_day = differens\_day \* standard\_tariff\_day  
 else:  
 price\_day = (new\_data\_copy["Денне\_споживання(кВт)"] - old\_data["Денне\_споживання(кВт)"]) \* standard\_tariff\_day  
  
 if new\_data\_copy["Нічне\_споживання(кВт)"] < old\_data["Нічне\_споживання(кВт)"]:  
 new\_data\_copy["Нічне\_споживання(кВт)"] += nakrytka\_night  
 differens\_night = new\_data\_copy["Нічне\_споживання(кВт)"] - old\_data["Нічне\_споживання(кВт)"]  
 price\_night = differens\_night \* standard\_tariff\_night  
 else:  
 price\_night = (new\_data\_copy["Нічне\_споживання(кВт)"] - old\_data["Нічне\_споживання(кВт)"]) \* standard\_tariff\_night  
  
 res\_day\_price = price\_night + price\_day  
 res\_price = old\_data["Загальна сума оплати на теперішню дату"] + res\_day\_price  
  
 new\_data\_copy['Загальна сума оплати на теперішню дату'] = res\_price  
  
 Information\_storage[old\_index] = copy.deepcopy(new\_data\_copy)  
  
new\_data = {"ID": 1, "Дата": "02.03.2025", "Денне\_споживання(кВт)": 45.0, "Нічне\_споживання(кВт)": 15.0, "Загальна сума оплати на теперішню дату": 0}  
Calculate\_price(new\_data)  
updated\_data = next(i for i in Information\_storage if i["ID"] == 1)  
self.assertEqual(updated\_data["Денне\_споживання(кВт)"], 45.0)  
self.assertEqual(updated\_data["Нічне\_споживання(кВт)"], 15.0)  
self.assertEqual(updated\_data["Загальна сума оплати на теперішню дату"], 1512)

Табл. №2 – Приклад програмного коду unit-теста.

new\_data\_copy['Загальна сума оплати на теперішню дату'] = res\_price  
  
 Information\_storage[old\_index] = copy.deepcopy(new\_data\_copy)  
  
new\_data = {"ID": 1, "Дата": "02.03.2025", "Денне\_споживання(кВт)": 45.0, "Нічне\_споживання(кВт)": 15.0, "Загальна сума оплати на теперішню дату": 0}  
Calculate\_price(new\_data)  
updated\_data = next(i for i in Information\_storage if i["ID"] == 1)  
self.assertEqual(updated\_data["Денне\_споживання(кВт)"], 45.0)  
self.assertEqual(updated\_data["Нічне\_споживання(кВт)"], 15.0)  
self.assertEqual(updated\_data["Загальна сума оплати на теперішню дату"], 1512)

Зображення, що містить текст, Шрифт, програмне забезпечення, знімок екрана

Вміст, створений ШІ, може бути неправильним.

Рис.№3 – Результат виконання.