ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ

НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

Кафедра інженерії програмного забезпечення

Пояснювальна записка

до бакалаврської роботи
на ступінь вищої освіти бакалавр

на тему: «**РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ ОБЛІКУ КОМУТАЦІЙНОГО ОБЛАДНАННЯ З ВИКОРИСТАННЯМ МОВИ ПРОГРАМУВАННЯ PYTHON»**

Виконав: студент 4 курсу, групи ПД-42
спеціальності 121 Інженерія програмного забезпечення
освітня програма «Інженерія програмного забезпечення»
(шифр і назва спеціальності)
Францев А.В.
(прізвище та ініціали)
Керівник Гаманюк І.М.
(прізвище та ініціали)
Рецензент_____ _____
(прізвище та ініціали)
Нормоконтроль _____
(прізвище та ініціали)

Київ – 2022

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ

Навчально-науковий інститут інформаційних технологій

Кафедра Інженерії програмного забезпечення

Ступінь вищої освіти - «Бакалавр»

Спеціальність - 121 Інженерія програмного забезпечення

ЗАТВЕРДЖУЮ

Завідувач кафедри

Інженерії програмного забезпечення

_____І.М. Гаманюк

«____»_____2022 року

ЗАВДАННЯ
НА БАКАЛАВРСЬКУ РОБОТУ СТУДЕНТУ

Францев Антон Володимирович
(прізвище, ім'я, по батькові)

1. Тема роботи: «Розробка програмного забезпечення для обліку комутаційного обладнання мовою з використанням мови програмування Python»

Керівник роботи Гаманюк І.М.
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом вищого навчального закладу від "16" лютого 2022 року №22.

2. Строк подання студентом роботи 06.06.2022

3. Вихідні дані до роботи:

3.1. Робота із протоколом SNMP.
3.2. Pycharm.
3.3. Існуючі інструменти для обліку комутаційного обладнання
3.4. Наукова-технічна література

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити):

4.1. Аналіз та огляд існуючих додатків та інструментів для реалізації системи.
4.2. Розробка структури додатку для керування обліку комутаційного обладнання.
4.3. Програмна реалізація додатку.
4.4. Висновки.

5. Перелік графічного матеріалу

5.1. Титульний слайд
5.2. Аналоги
5.3. Мета, об'єкт, предмет та наукова новизна дослідження
5.4. Технічні завдання
5.5. Програмні засоби та інструменти використані для реалізації
5.6. Діаграма прецендентів системи
5.7. Блок схема роботи програми

5.8. Екранні форми (Зчитування данних із комутатора)
5.9. Екранні форми (Інформація про розробника програми)
5.10. Екранні форми (Виведення помилки про неправильно введені данні)
5.11. Моделювання процесу сканування мережі на наявність SNMP клієнтів
5.12. Діаграма пакетів
5.13. Діаграма класів
5.14. Відпрацювання коду
5.15. Апробація результатів дослідження
5.16 Висновки

6. Дата видачі завдання 11.03.2022

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів бакалаврської роботи	Строк виконання етапів роботи	Примітка
1	Вибір теми бакалаврської роботи	11.03.2022	Виконано
2	Підбір науково-технічної літератури	12.03.2022-20.03.2022	Виконано
3	Дослідження аналогів та актуальності додатку	20.03.2022-04.04.2022	Виконано
4	Аналіз та вибір інструментів для розробки додатку	04.04.2022-25.04.2022	Виконано
5	Проектування та реалізація	25.04.2022-06.05.2022	Виконано
6	Висновки, оформлення роботи	06.05.2022-13.05.2022	Виконано
7	Передзахист	16.05.2022-01.06.2022	
8	Захист роботи		

Студент _____ А.В. Францев
(підпис) (прізвище та ініціали)

Керівник роботи_____ І. М. Гаманюк

РЕФЕРАТ

Текстова частина роботи 72 с., 29 рис., 28 джерел.

КОМУТАЦІЙНЕ ОБЛАДНАННЯ, PYHTON, ОБЛІК КОМУТАЦІЙНОГО ОБЛАДНАННЯ

Об'єкт дослідження – облік комутаційного обладнання.

Предмет дослідження – розробка програмного забезпечення для обліку комутаційного обладнання.

Мета роботи – автоматизація обліку комутаційного обладнання.

Методи дослідження – методи розпізнавання та групування інформації, емпіричні методи.

У роботі проведено аналіз існуючих версій мови Python, а також середовищ розробки, таких як PyCharm, Qt Creator, а також реалізацію обліку комутаційного обладнання за допомогою Python.

Загальною проблемою цих комутаційного обладнання наразі є безпека. За допомогою шифрування можна замаскувати дані про облік комутаційного обладнання.

Було описано необхідні дії для запуску системи, проблеми які можуть виникнути при першій установці, та наведено інтерфейс користувача. Роз'яснено як користуватися даною системою, за рахунок чого вона працює і як може бути збільшена її продуктивність і швидкодія. Як приклад зроблено запуск програми, варіанти введення даних. Показано приклади помилок, що можуть виникнути під час роботи.

У якості вихідних даних є зображення з текстовими даними.

Даний додаток може бути використано у всіх сферах, які потребують облік комутаційного обладнання.

***Галузь використання* – облік комутаційного обладнання.**

ЗМІСТ

ВСТУП

Актуальність теми. Організація взаємодії між пристроями у мережі є складним завданням. Для вирішення складних завдань використовується універсальний прийом - декомпозиція, або розбиття однієї складної задачі на кілька більш простих задач-модулів.

Процедура декомпозиції включає визначення функцій кожного модуля, що вирішує окреме завдання, і інтерфейсів між ними. В результаті досягається логічне спрощення завдання, і з'являється можливість модифікації окремих модулів без зміни решти системи.

При декомпозиції часто використовують багаторівневий підхід. Усі безліч модулів розбивають на рівні. Рівні утворюють ієрархію, тобто є вищі і нижчі рівні. Безліч модулів, що становлять кожен рівень, сформовано таким чином, що для виконання своїх завдань вони звертаються із запитами тільки до модулів, що безпосередньо примикає до нижчележачого рівня.

Результати роботи всіх модулів, що належать деякому рівню, можуть бути передані лише модулям сусіднього рівня. Ієрархічна декомпозиція задачі передбачає визначення функції кожного рівня та інтерфейсів між рівнями. Інтерфейс визначає набір функцій, які нижчий рівень надає вищого рівня. В результаті ієрархічної декомпозиції досягається відносна незалежність рівнів та можливість їх заміни.

Модулі, що реалізують протоколи сусідніх рівнів та знаходяться в одному вузлі, також взаємодіють один з одним відповідно до певних правил та за допомогою стандартизованих форматів повідомлень. Ці правила називають інтерфейсом. Інтерфейс визначає набір служб, що надається цим рівнем сусіднього рівня. Протокол та інтерфейс висловлюють одне й те саме поняття, але традиційно в мережах за ними закріпили різні області дії: протоколи визначають правила взаємодії модулів одного рівня у різних вузлах, а інтерфейси визначають правила взаємодії модулів сусідніх рівнів в одному вузлі. Протоколи реалізуються комп'ютерами та іншими мережними пристроями — концентраторами, мостами,

комутаторами, маршрутизаторами тощо. буд. Залежно від типу пристрою, реалізують певний набір протоколів.

Для того, щоб їх розробити слід програмувати, а оскільки мова Python є найбільш оптимальною для цього, варто розглянути детальніше розробку програмного забезпечення для обліку комутаційного обладнання мовами Python.

Мета дослідження – автоматизація обліку комутаційного обладнання.

Об'єкт дослідження – облік комутаційного обладнання.

Предмет дослідження - розробка програмного забезпечення для обліку комутаційного обладнання.

Завдання дослідження:

- Розглянути особливості комутаційного обладнання в сучасному світі;
- Дослідити особливості мови Python;
- Зробити аналіз засобів розробки програмного забезпечення для обліку комутаційного обладнання мовами Python;
- Описати реалізацію програмного забезпечення для обліку комутаційного обладнання мовами Python.

Структура роботи. Загальний обсяг роботи – 70 сторінок. Робота складається зі вступу, чотирьох розділів, висновків, списку використаних джерел та додатків.

РОЗДІЛ 1. АНАЛІЗ ТА ОГЛЯД ІСНУЮЧИХ ДОДАТКІВ ТА ІНСТРУМЕНТІВ ДЛЯ РЕАЛІЗАЦІЇ СИСТЕМИ

1.1 Комутаційне обладнання та протокол SNMP

В даний час більшість IT компаній не стоїть на місці, та постійно розширює свій штат співробітників та інфраструктуру. Однак для підключення більшої кількості клієнтів і розширення пропускної спроможності мережі закупається і встановлюється велика кількість додаткового комутаційного обладнання яке потребує обліку.

На сьогодні існує дуже багато протоколів для взаємодії із комутаційним обладнанням. Найбільш популярний із яких SNMP (Simple Network Management Protocol).

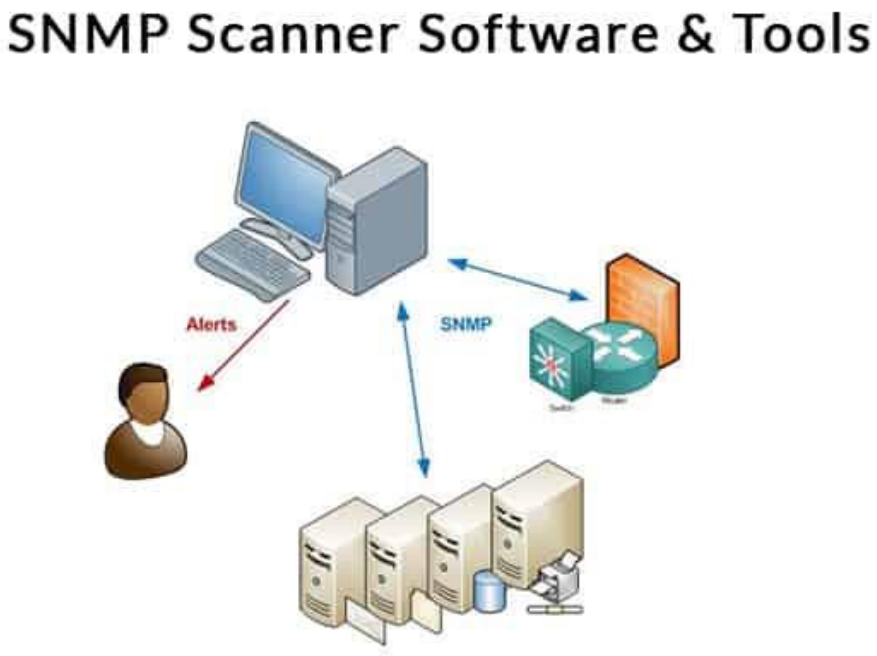


Рисунок 1.1 – Традиційний інтерфейс взаємодії із протоколом SNMP.

На початку 1988 року стало ясно, що існують загальні набори засобів управління мережними і застосовуються різними виробниками, які досі створюють власні продукти спостереження та конфігурування для своїх пристроїв пристроїв, що підтримують прояви відчуттів з ними.

Після багатьох засідань IAB (Рада з архітектури Інтернету) опублікувала в минулому 1988 році епохальний документ RFC 1052: Рекомендації IAB з розробки стандартів управління мережею Інтернету, які вводяться до якнайшвидшого використання елементів простого мережевого управління (Simple Network Management).

Керування мережею за допомогою SNMP базується на стандартній моделі клієнт/сервер, яка широко використовується в мережевих програмах, що використовують TCP/IP. На кожному хості, на який поширюється керування, виконується процес-агент. Агент – це обслуговуючий процес, який працює з Базою інформації управління (MIB) хоста. На хостах, застосовуваних управління мережею, може виконуватися процес-диспетчер.

Спочатку протокол повинен був надати системним адміністраторам інструмент управління інтернетом. Однак, гнучка архітектура SNMP дозволила проводити моніторинг всіх мережевих пристроїв та керувати ними з однієї консолі. Це стало причиною поширення SNMP.

Безпека протоколу SNMP - це розділ специфікації протоколу, що змінюється, з часу його створення. З кожною версією SNMP, намагалися посилити безпеку. Перша версія протоколу SNMPv1 була найпростішою та небезпечною. Повідомлення протоколу могли бути схильні до можливості модифікації, підміни або прослуховування. Безпека протоколу базувалася на моделі безпеки на основі спільнот (т.зв. Community-based Security Model), що передбачало аутентифікацію на основі єдиного текстового рядка - своєрідного пароля (т.зв. community-sting), яка передавалася в тілі повідомлення у відкритому вигляді . Хоча, ця версія протоколу найбільш незахищена, вона часто застосовується у сучасних мережах, т.к. є найпростішим.

1.2. Шифрування інформації та захист обладнання

В даний час інформаційних технологій, що постійно розвиваються, кожне підприємство, що має розгорнуту мережеву інфраструктуру, прагне забезпечення високого рівня безпеки своєї інфраструктури. Цей етап життєвого циклу підприємства став одним з найважливіших у зв'язку з тим, що інфраструктура, що використовується, застосовується для доступу до мережевих ресурсів, в які став входити великий потік конфіденційної інформації, за рахунок переходу від паперового до повного електронного документообігу. Одночасно з таким потоком зростають кількості атак різних категорій на мережну інфраструктуру. За статистикою за 2019 рік, найпоширенішим видом атак є web - атаки, про що свідчить зібрана статистика компанією "Symantec Corporation". З початку 2019 по червень 2019 кількість web - атак постійно зростає і за 6 місяців становив майже 1000 атак.

Отже, найбільш актуальною проблемою сучасного світу є забезпечення максимально високого рівня безпеки мережної інфраструктури від несанкціонованого доступу або різноманітних атак. Для вирішення описаної проблеми необхідно розглянути можливі методи її вирішення

Одним з найбільш примітивних рішень є використання антивірусних програм, однак інструмент дозволяє позбутися лише деяких загроз, так як принцип роботи антивірусних рішень полягає в перевірці файлів, що знаходяться на жорсткому диску на предмет знаходження шкідливих ділянок коду з сигнатурною базою, яка повинна постійно оновлюватися.

Наступним найпоширенішим інструментом є система виявлення вторгнень, наприклад Suricata або Snort. Такі системи, як правило, використовують два режими роботи, а саме сигнатурний і поведінковий [2]. Сигнатурний режим роботи полягає в тому, що циркулюючий мережевий трафік аналізується встановленою системою виявлення вторгнень і порівнюється з наявною базою даних сигнатур.

Такий підхід запозичений у роботі будь-якого антивірусу, тільки як сигнатура представляється не шкідливий код в заголовку мережного кадру. Поведінковий режим полягає в тому, що встановлена система виявлення вторгнень проводить постійне дослідження автентифікованого користувача, наприклад, в доменній мережі, а також його мережну активність та взаємодію додатків, що працюють під цим обліковим записом, після чого відбувається порівняння із заздалегідь побудованою моделлю на предмет наявності аномалій або некоректної поведінки користувача [12].

Після аналізу запропонованих рішень можна зробити висновок, що такі методи витрачають багато часу на виявлення, тому що їм доводиться пропускати заражений мережевий потік і проводити аналіз зліпків мережевого трафіку, виявленого аналізатором мережевого потоку, а також відсутня можливість визначення атаки нульового дня за рахунок прихильності до оновлюваної основі сигнатур. Для зменшення часу виявлення вторгнення в даний час використовую методи машинного навчання, які дозволяють сигналізувати про наявність атаки на ранніх етапах і з певною ймовірністю вказати який вид атаки буде використовуватися, що дозволяє співробітнику, який відповідає за інформаційну безпеку своєї організації з високою точністю визначити набір інструментів захисту від атаки певного виду.

При використанні машинного навчання вибирають зазвичай два основних алгоритми, а саме логістична регресія та дерево рішень. Логістична регресія застосовується для прогнозування ймовірності виникнення деякої події за значенням множини ознак. Для цього вводиться так звана залежна змінна { y} y, що приймає лише одне з двох значень - як правило, це числа 0 (подія не відбулася) і 1 (подія відбулася), і безліч незалежних змінних (також званих ознаками, предикторами або регресорами) - речових {x_{1},x_{2},…,x_{n}}, на основі значень яких потрібно обчислити ймовірність прийняття того чи іншого значення залежної змінної [3]. DecisionTreeClassifier — це метод, який зазвичай використовується в інтелектуальному аналізі даних. Ціль полягає в тому, щоб

створити модель, яка прогнозує значення цільової змінної на основі декількох вхідних змінних [4].

Розглянутий алгоритм має вхідні параметри, які дозволяють досягти максимальної продуктивності методу дерево рішень, одним із таких параметрів є рівень занурення у дерево рішень. Тим не менш, існують також шифри, які можна застосувати при програмуванні на Python.

Історія шифрів заміни налічує близько 4 тисяч років. До нашого часу шифри заміни мали роль виключно забезпечення конфіденційності повідомлень (тобто шифруванням) — перетворенням повідомлень із зрозумілої форми в незрозумілу і зворотне відновлення на стороні одержувача, роблячи його неможливим для прочитання для того, хто перехопив або підслухав без секретного знання (а саме ключа, необхідного для дешифрування повідомлення). В останні десятиліття 21 сторіччя сфера застосування шифрів заміни розширилася і включає не лише таємну передачу повідомлень, але і методи перевірки цілісності повідомлень, ідентифікація відправника/одержувача (аутентифікація), цифрові підписи, інтерактивні підтвердження та технології безпечного спілкування тощо [2].

В найдавніші часи в Стародавньому Римі існував Атбаш - простий шифр підстановки для івриту. Даним алгоритмом зашифровано частину біблійних текстів. Правило шифрування полягає у заміні і-тої літери абетки літерою зномером n − i + 1, де n — кількість літер в алфавіті. Таким чином, перша буква алфавіту замінюється останньою, друга — передостанньою і так далі.

Приклад для латинського алфавіту виглядає так:

Вхідний текст: abcdefghijklmnopqrstuvwxyz

Зашифрований текст: ZYXWVUTSRQPONMLKJIHGFEDCBA

Застосування алгоритму до українського алфавіту:

Вхідний текст: а б в г ґ д е є ж з и і ї й к л м н о п р с т у ф х ц ч ш щ ь юя

Зашифрований текст: я ю ь щ ш ч ц х ф у т с р п о н м л к й ї і и з ж є е дґ г в б а

Походження слова «Атбаш» пояснюється принципом заміни літер. Слово אתבש складено з літер «леф», «Тав», «Бет» та «Шин», тобто першої таостанньої, другої та передостанньої літер давньосемітського алфавіту.

Шифр тбаш був, швидше за все, винайдений ессеями, іудейською сектою повстанців. Вони розробили безліч різних кодів і шифрів, які використовувалися для приховування важливих імен і назв, щоб потім уникнути переслідування. Знання цих кодів і шифрів були потім передані гностикам, які, в свою чергу, передали їх Катарам. Пізніше Орден Тамплієрів завербував Катарський дворян і перейняв знання шифрів. Таким чином, шифр використовувався протягом багатьох років, від близько 500 до н. е. до 1300 р. н.е. - моменту, коли Орден Тамплієрів був розпущений.

У криптографії замінний шифр - це метод шифрування, при якому одиниці відкритого тексту замінюються на зашифрований текст, визначеним чином, за допомогою ключа; "одиницями" можуть бути одиничні літери (найпоширеніші), пари букв, триплети літер, суміші вищезазначених тощо. Приймач розшифровує текст, виконуючи зворотний процес заміщення для вилучення вихідного повідомлення.

Замінні шифри можна порівняти з шифрами транспозиції. У транспортному шифрі одиниці відкритого тексту переставляються в інший і, як правило, досить складний порядок, але самі одиниці залишаються незмінними. Навпаки, у замінному шифрі одиниці відкритого тексту зберігаються в тій же послідовності в зашифрованому тексті, але самі одиниці змінюються.

Існує ряд різних типів замінних шифрів. Якщо шифр оперує окремими літерами, його називають простим замінним шифром; шифр, який оперує більшими групами літер, називається поліграфічним. Моноалфавітний шифр використовує фіксовану заміну по всьому повідомленню, тоді як поліалфавітний шифр використовує ряд заміщень у різних положеннях повідомлення, де одиниця з відкритого тексту відображається в одну з декількох можливостей в шифротексті і навпаки.

Замінні шифри можуть замінювати лише літери стандартного алфавіту зашифрованим текстом або застосовувати заміни також до пробілів та розділових знаків.

Прості замінні шифри (або моноальфавітні замінні шифри)

Прості або моноальфавітні замінні шифри покладаються на відображення окремих букв алфавіту із відкритим текстом до певної літери алфавіту шифротексту [3].

Для простої заміни кожна буква стандартного алфавіту замінюється однаковою буквою або символом зашифрованого тексту згідно з фіксованим правилом. Так, наприклад, якщо в кодованому повідомленні букву "a" замінити символом "#", така сама заміна відбуватиметься у кожному повідомленні, кодованому відповідно до цього конкретного правила заміни.

Оскільки кожна буква стандартного алфавіту теоретично може бути замінена будь-якою іншою буквою чи символом, існує нескінченна кількість потенційних моноальфавітних шифрів заміщення. В операціях ручного кодування найпростіше спочатку створити алфавіт зашифрованого тексту, а потім виконувати шифрування, порівнюючи це з відкритим текстом.

Генератори ключових слів

Популярний метод, який використовується для створення алфавітів зашифрованого тексту для простої заміни, починається з ключового слова - бажано слова, в якому кожна буква стандартного алфавіту зустрічається лише один раз (наприклад, "запас" або "кабінка"). Це слово розміщується на початку таблиці, а решта букв алфавіту позначаються кінцем у звичайному порядку (тобто пропускаючи ті літери, що трапляються в ключовому слові).

Шифр Atbash

Один із найраніших зафіксованих шифрів заміщення, шифр Атбаша наклав моноальфавітні заміни на єврейський алфавіт. Це була проста система, в якій кожен зашифрований фрагмент відкритого тексту використовував один і той же алфавіт зашифрованого тексту.

Шифр Atbash створив свій алфавіт зашифрованого тексту, просто змінивши алфавіт із відкритим текстом, відобразивши першу літеру стандартного алфавіту до останньої, другу до другої останньої тощо. Система отримала свою назву фонетично, замінивши іврит «алеф» його шифрованою формою «тав» та «бет» на «гомілка».

Природно, що метод може застосовуватися до будь-якого мовного алфавіту і є швидким і простим інструментом для маскування повідомлень некритичного значення з очей випадкових спостерігачів. Елемент додаткової безпеки може бути застосований до техніки, позначаючи рядок цифр, символів або розділових знаків на кінці алфавіту зашифрованого тексту перед виконанням заміни [5].

Шифр Цезаря

Трохи безпечнішим за Атбаш був Шифр, який Юлій Цезар використовував для надсилання зашифрованих повідомлень своїм арміям на місцях. Shift або Цезарський шифр працює, зміщуючи алфавіт на певну кількість ходів і замінюючи кожну букву відкритого тексту на її зміщений еквівалент зашифрованого тексту. У Стародавньому Римі Цезар замінив букву "a" на букву "d" - "зміщення на 3", що, по суті, створило ключ шифрування для мови.

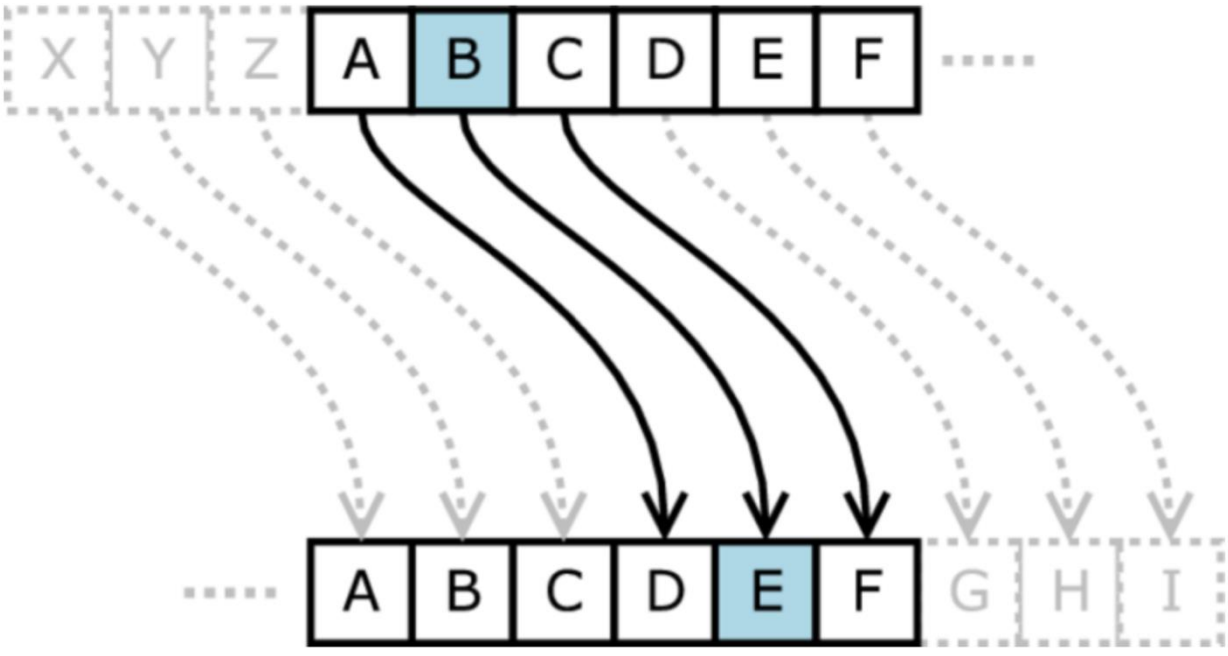


Рисунок 1.1. – Приклад шифра Цезаря

Отже, прийнявши рішення про номер зміни, створення алфавіту зашифрованого тексту є відносно простим питанням зміщення букв стандартного алфавіту на узгоджену кількість місць ліворуч. Це дає "d" як першу літеру алфавіту Цезарського шифру, за якою слідує "efgh" тощо, а літери "abc" позначаються після "z". Шифрування вихідного повідомлення відбувається шляхом простої заміни літери відповідним зашифрованим текстом [7].

Дешифрування зашифрованого тексту може бути здійснено за допомогою посилання на таблицю з зміщеним алфавітом, створену для шифрування повідомлення, або шляхом створення нової таблиці, яка зміщує букви стандартного алфавіту на однакову кількість позицій у зворотному напрямку.

Шифр Пігпена (Шифр масонів)

Цей замінний шифр замінює кожну літеру відкритого тексту алфавіту відповідним символом. Незважаючи на те, що його історичне походження незрозуміле, шифр Пігпен протягом багатьох років використовувався кількома групами, зокрема, серед яких були солдати Союзу, ув'язнені Конфедерацією під час громадянської війни в Америці, та напівсекретним товариством масонів з 18 століття.

Масони використовували шифр Пігпен у багатьох аспектах свого повсякденного життя і навіть у смерті: криптографічні повідомлення були виявлені на надгробних написах передбачуваних членів суспільства.

Сітка символів налаштована для створення «алфавіту» зашифрованого тексту, і кожна буква кодованого повідомлення просто узгоджується і замінюється відповідним символом.

Шифри підстановки диграфів

На відміну від моноальфавітних замінних шифрів, заміна підписних шифрів замінює пари букв із стандартного алфавіту парою букв зашифрованого тексту [8].

Відкритий текст повідомлення спочатку розділити на пари літер або диграфів. Пари заміщення можна визначити, використовуючи комбінацію

моноальфабітних методів, таких як дві зміни Цезаря. Перша зміна буде застосована до першої літери кожної пари, а друга зміна буде диктувати другу літеру.

Порушення кодексу

Незважаючи на їх успіх в історичному застосуванні, прості, моноальфавітні та диграфні замінні шифри досить легко зламати - за умови часу та належних методів.

Грубої сили (спроб і помилок) може бути достатньо, щоб зламати шифр Цезарського зрушення, оскільки існує лише 26 можливих алфавітів шифротексту, які використовують усі стандартні літери.

В інших випадках застосування гучного процесу, який називається Частотний аналіз, може зламати код. Цей відносно простий прийом передбачає розгляд зашифрованого тексту із заміною літер з точки зору відомих правил, що регулюють використання стандартної мови: Найчастіше зустрічаються букви ("e" англійською мовою), загальновживані слова ("the", "та") та подібні умови можуть дати справедливий початковий показник змісту повідомлення. Деякі здогадки можуть бути використані для заповнення решти прогалин.

Замінні поліальфабітні шифри

Через ці властиві слабкі сторони були зроблені спроби розробити сильніші коди заміщення. Приблизно в 1467 р. Леон Баттіста Альберті створив перший відомий поліальфабетний шифр заміщення. Шифр Альберті використовував змішаний алфавіт для шифрування, який перемикався на інший алфавіт зашифрованого тексту у випадкових точках тексту. Ці пункти були позначені в коді великою літерою. Ключі цієї системи, закодовані Альберті, на наборах дисків із шифром.

```
Keyword:    M E C M E C M E C M E C M E C M E C M E C M
Plaintext:  w e n e e d m o r e s u p p l i e s f a s t
Ciphertext: I I P Q I F Y S T Q W W B T N U I U R E U F
```

Рисунок 1.2. – Приклад шифру Альберті

Спираючись на роботу Альберті, група, що включала німецького ченця Йоганнеса Тритемія, італійського вченого Джованні Порту, і французького дипломата XVI століття Блеза де Віженера, перегнали поліальфабітну заміну в шифр Відженера, який повністю використовував до 26 різних алфавітів шифротексту і залишався до 1854 року як "Le Chiffre Undechiffrable" або "Unbreakable Cipher" [5].

Більшість з цих шифрів можна використовувати в Python, що робить його ефективним для реалізації комутаційного обладнання.

ВИСНОВКИ ДО РОЗДІЛУ 1

Отже, можна прийти до висновку що комутаційне обладнання наразі є важливою частиною життя суспільства та застосовується в багатьох технологіях. Варто розуміти, що шифрування в сучасному світі – важливий елемент безпечного існування світу.

РОЗДІЛ 2. ОБЛІК КОМУТАЦІЙНОГО ОБЛАДНАННЯ НА ОСНОВІ PYTHON

2.1 Мова Python

В даний час з розвитком мережевих технологій, глобальна мережа Internet стала використовуватися в багатьох областях нашого життя. Все більше людей користуються інтернетом, і у зв'язку з цим з'явилося безліч технологій, які надають можливість створення та розробки web-додатків та інших високотехнологічних новаторств.

Однією з найпопулярніших є мова програмування Python на середовищі PyCharm. PyCharm – це найінтелектуальніша Python IDE (Integrated Development Environment – Інтегроване середовище розробки) з повним набором засобів для ефективної розробки мовою Python.

Випускається у двох варіантах – безкоштовна версія PyCharm Community Edition та платна, що підтримує більший набір можливостей PyCharm Professional Edition. PyCharm виконує інспекцію коду «на льоту», автодоповнення, у тому числі ґрунтуючись на інформації, отриманій під час виконання коду, навігації за кодом, забезпечує безліч рефакторингів.

Наведемо порівняльний аналіз можливостей даних середовищ [6]:

Community Edition:

1. Спрощена IDE для розробки тільки на Python.

2. Безкоштовна, з відкритим кодом, під ліцензією Apache 2.

3. Розумний контекст редактор, відладчик, рефакторинги, інспекції, інтеграція з VCS.

4. Навігація по проекту, підтримка тестування, UI, що настроюється, гарячі клавіші Vim.

Professional Edition:

1. Повнофункціональна IDE для розробки на Python, зокрема багатомовних веб-додатків з фреймворками.

2. Підтримка фреймворків Django, Flask, Google App Engine, Pyramid, web2py.

3. Підтримка мов JavaScript, CoffeeScript, TypeScript, CSS, Cython та ін.

4. Віддалена розробка, підтримка роботи з БД та мови SQL.

5. Виявлення коду, що дублюється.

6. Діаграми UML і SQLAlchemy.

7. Python Profiler [1].

Дана мова програмування є порівняно немолодою мовою для web-розробки, задуманою 1980-го, а реалізована ближче до дев'яностих. Його автор, Гвідо ван Россум, хотів удосконалити мову «ABC» (ABC – імперативну, процедурну, структурну високорівневу мову програмування загального призначення та IDE), яка використовувалася для навчання, але мала ряд недоліків. У результаті, після тривалої роботи Россума вийшов високорівневий, скриптовий PL (PL – Procedural Language).

Python підтримує кілька стилів програмування. Він змушує розробника дотримуватися певної парадигми. Python підтримує об'єктно-орієнтоване та процедурне програмування. Існує обмежена підтримка функціонального програмування. Мова має чіткий і послідовний синтаксис, продуману модульність і масштабованість, завдяки чому вихідний код написаних на Python програм легко читаємо.

Творці наголосили не на потужності самого коду, а на продуктивності розробників, які з ним працюють. На «препроцесорі» найкраще виходить код, який може прочитати лише автор. Для роботи у команді такий підхід не підійде. «Python» використовують багато програмістів, оскільки він ефективний і має великий перелік переваг, які характеризують роботу з цією мовою програмування:

– простота мови. Його можна використовувати не тільки як у web-розробці, а й у будь-якій іншій галузі, де фахівці не мають якихось глибоких знань у програмуванні. Сам синтаксис даної мови програмування схожий із

звичайними математичними операціями, які не несуть у собі жодних особливих складнощів;

– різноманітність реалізацій. Найвідоміша і канонічніша – це «CPython», реалізація на «C». Це означає, що код, написаний на ньому, повністю взаємодіє з «C», і бібліотеки цієї мови також можна використовувати для реалізації. Те саме стосується мови «Java», існує і реалізація на ньому – «Jython». Таких прикладів маса, аж до взаємодії Пітона з "Android" та "iOS";

– широке розповсюдження. "Python" використовують навіть Disney. Наслідком цього факту є те, що багато про нього вже відомо. Як тільки ви стикаєтеся з проблемою при програмуванні на Python, відразу можете звернутися за допомогою інтернету: швидше за все, вашу проблему вже хтось вирішував. До того ж для реалізації практично будь-якого проекту вже існують заготівлі, які можна застосувати для себе;

– не потребує компіляції. «Python» – мова, що інтерпретується, а значить, запустити програму можна відразу після внесення змін до її файлу.

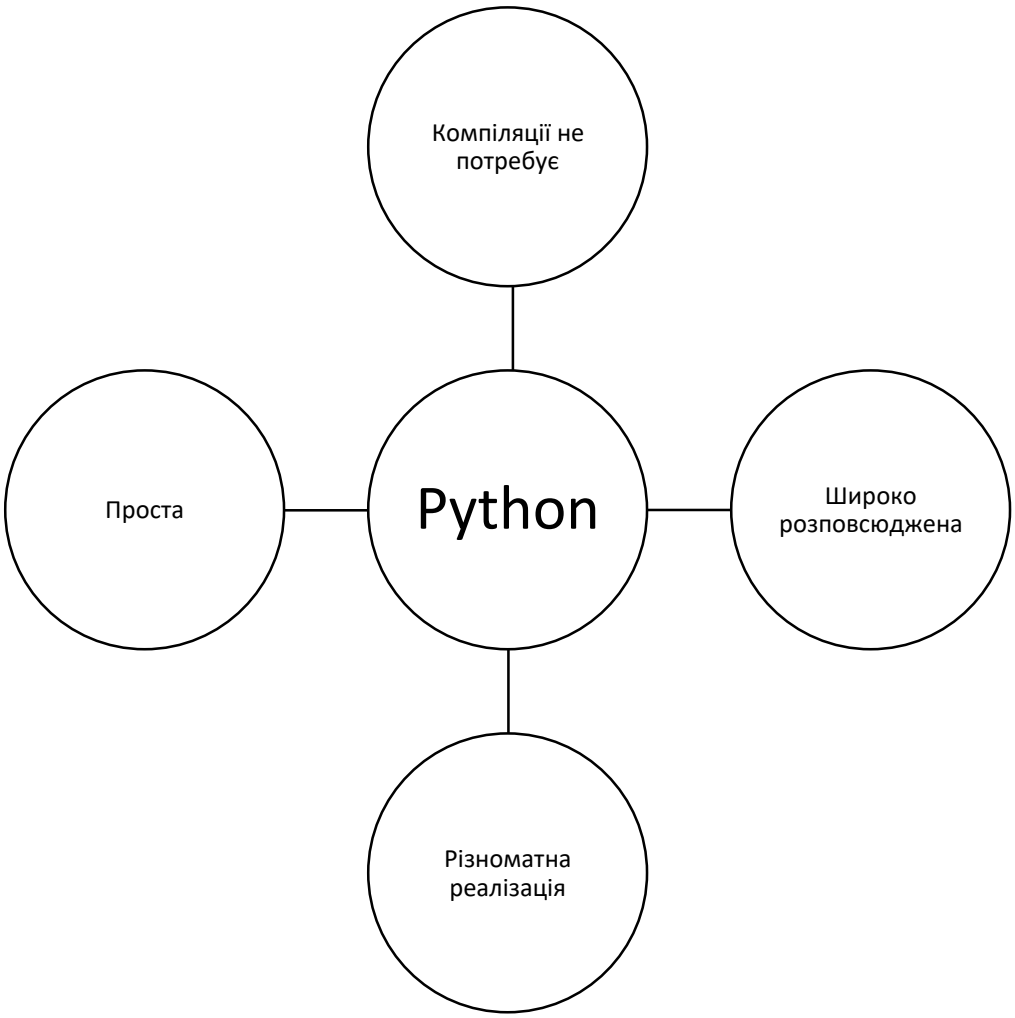


Рисунок 2.1. – Особливості мови Python.

Це призводить до того, що доопрацювання, переробка та налагодження програм відбувається набагато швидше, ніж у багатьох інших мовах. Розглянемо приклад у реалізації коду на Python «Текстовий редактор» (рис. 2.2, рис. 2.3), створений у графічній бібліотеці Tkinter (Tkinter – це графічна бібліотека, що дозволяє створювати програми з віконним інтерфейсом). Ця бібліотека є інтерфейсом до популярної мови програмування та інструменту створення графічних програм tcl/tk. Tkinter, як і tcl/tk, є кроссплатформенной бібліотекою і можна використовувати у більшості поширених операційних систем (Windows, Linux, Mac OS X та інших.).

```
import tkinter
from tkinter.filedialog import asksaveasfile, askopenfile
from tkinter.messagebox import showerror

FILE_NAME = tkinter.NONE


def new_file():
    global FILE_NAME
    FILE_NAME = "Untitled"
    text.delete('1.0', tkinter.END)


def save_file():
    data = text.get('1.0', tkinter.END)
    out = open(FILE_NAME, 'w')
    out.write(data)
    out.close()


def save_as():
    out = asksaveasfile(mode='w', defaultextension='.txt')
    data = text.get('1.0', tkinter.END)
    try:
        out.write(data.rstrip())
    except Exception:
        showerror(message="Ошибка при сохранении файла")

def open_file():
    global FILE_NAME
    inp = askopenfile(mode="r")
    if inp is None:
        return
    FILE_NAME = inp.name

    data = inp.read()
    text.delete('1.0', tkinter.END)
    text.insert('1.0', data)
```

Рисунок 2.2 – Приклад реалізації коду

```
def about_the_program():
    root = tkinter.Tk()
    root.title("О программе")
    lab = tkinter.Label(root,
                        text="Txt Editor \nВерсия 0.1 "
                        "\n05.01.2018 \nCopyright©2018 "
                        "\nВсе права защищены "
                        "\n \nСведения об участнике проекта: "
                        "\nБухаров Тимур - Разработчик "
                        "\nСтерлитамак СФ БашГУ",font="Arial 12",
                        justify=tkinter.LEFT)
    lab.pack()

root = tkinter.Tk()
root.title("Txt Editor v0.1")
root.minsize(width=400, height=400)
root.maxsize(width=400, height=400)

text = tkinter.Text(root, width=400, height=400)
text.pack()

menuBar = tkinter.Menu(root)
fileMenu = tkinter.Menu(menuBar)
fileMenu1 = tkinter.Menu(menuBar)
fileMenu1.add_command(label="О программе", command=about_the_program)
fileMenu.add_command(label="Новый файл", command=new_file)
fileMenu.add_command(label="Открыть", command=open_file)
fileMenu.add_command(label="Сохранить", command=save_file)
fileMenu.add_command(label="Сохранить как", command=save_as)
fileMenu.add_separator()
fileMenu.add_command(label="Выход", command=root.quit)
menuBar.add_cascade(label="Файл", menu=fileMenu)
menuBar.add_cascade(label="Справка",menu=fileMenu1)



root.config(menu=menuBar)
root.mainloop()
```

Рисунок 2.3. – Приклад реалізації коду.

Оскільки Tkinter є досить прозорим інтерфейсом до tcl/tk, то основним джерелом для неї є man-сторінки tcl/tk. Ці сторінки є у будь-якій Unix-системі (у розділі n або 3tk). Насамперед під час роботи з Tkinter необхідно створити

головне (кореневе) вікно (рис. 1.2), у якому розміщуються інші графічні елементи – віджети. Існує великий набір віджетів на всі випадки життя: для введення тексту, виведення тексту, меню, що випадають тощо. Серед віджетів є кнопка при натисканні на яку відбувається задана подія. Деякі віджети (фрейми) використовуються для групування інших віджетів у собі. Також у Tkinter існує три стандартні GM (Geometry Manager – менеджер геометрії), які керують трьома розміщеннями: «Grid», «Pack» та «Place». "Grid" ділить весь простір віджету на осередки, кількість яких визначається дочірніми віджетами.

Сам осередок ідентифікується за номерами рядків і стовпців, а також вони об'єднуються як по горизонталі, так і по вертикалі. «Pack» має віджети один за одним по тій чи іншій стороні. Воно підійде для більш простого схемного розташування.

«Place» вказує віджету на його місце за допомогою координат як в абсолютних, так і відносних значеннях загального вікна. Відомо, що кожному ідеальний той керівник розміщеннями, користуватися яким можеш досконало. Однак якщо метою ставити створення найбільш адекватного коду, необхідно вміти користуватися трьома методами і грамотно їх комбінувати; але потрібно знати одну річ, що не можна використовувати два менеджери всередині одного вікна.

Установка Python є досить простою. Його можна встановити в будь-яку операційну систему, як Windows, Mac OS, Linux. Установка Python у Windows.

1. Слід перейти до https://www.python.org/downloads
2. Натиснути "Завантажити Python", версії щороку можна оновити.
3. Як тільки файл python.exe закінчить завантаження, після ви можете запустити виконуваний файл для встановлення Python.

Установка включає IDLE, pip і документацію.

IDLE – це інтегроване середовище розробки (IDE) для Python, яке постачається в комплекті з реалізацією мови за умовчанням. IDLE - це графічний інтерфейс (GUI), який має ряд функцій для розробки програм. Так само Python може бути встановлений в Linux/Unix, Mac OS X.

Також можна встановити Pycharm, IDE для Python, розроблену JetBrains. Компанія стверджує, що працює краще, ніж будь-яка інша IDE для Python. Pycharm допомагає розробникам писати акуратний та підтримуваний код, а також надає всі інструменти, необхідні для продуктивної розробки на Python.

Тепер, коли є необхідне налаштування IDE, можна розпочати написання першої програми. Якщо людина використовує PyCharm, слід виконати такі дії:

1. Натиснути “Створити новий проект” на екрані вітання PyCharm.
2. Вказати дійсну назву проекту
3. Створити новий файл python, клацнувши правою кнопкою миші на ім'я папки та обрати Створити -> Файл Python
4. Написати код: Print ('Hello World!')
5. Зберегти файл як HelloWorld
6. Запустити файл HelloWorld

Висновок буде видно на екрані як - Hello World! Перша програма на Python готова.

2.2. Особливості використання мови Python

Отже, чи вартує вивчення Python витраченого часу та зусиль? Насправді, він існує з 1991 року (довше, ніж Java), і постійно входить до 10 найпопулярніших комп’ютерних мов. Людям платять за написання програм на Python — серйозних речей, якими кожен користуєтеся щодня, як-от Google, YouTube, Dropbox, Netflix і Hulu.

Python має репутацію продуктивної мови, яка подобається організаціям, що швидко розвиваються.

Python можна знайти у багатьох обчислювальних середовищах, включаючи такі:

- командний рядок на моніторі або вікні терміналу;
- графічні інтерфейси користувача, включаючи Інтернет;
- Інтернет, на стороні клієнта та сервера;

- сервери, що підтримують великі популярні сайти;
- хмари (сервери, якими керують треті сторони);
- мобільні пристрої;
- вбудовані пристрої.

Програми Python варіюються від одноразових сценаріїв до систем з мільйоном рядків.

Python — це хороша мова високого рівня загального призначення. Її дизайн робить її дуже читабельною, що важливо. Кожна комп'ютерна програма написана лише один раз, але читається й переглядається багато разів, часто багатьма людьми. Читабельність також полегшує вивчення та запам'ятовування.

Порівняно з іншими популярними мовами, Python має м'який процес навчання, який робить людину продуктивною швидше, але він має глибини, які слід досліджувати, набираючись досвіду. Відносна лаконічність Python дозволяє написати програму, яка набагато менша, ніж її еквівалент на статичній мові. Дослідження показали, що програмісти, як правило, виробляють приблизно однакову кількість рядків коду на день — незалежно від мови — тому написання половини рядків коду подвоює продуктивність.

Python — найпопулярніша мова для вступних курсів інформатики в провідних американських коледжах. Це також найпопулярніша мова для оцінки навичок програмування понад двома тисячами роботодавців. Пишіть все, що забажаєте, за допомогою Python і використовуйте це будь-де, вільно. Python працює майже скрізь і має «батареї в комплекті» — метричну кількість корисного програмного забезпечення в його стандартній бібліотеці. Але, можливо, найкраща причина використовувати Python – несподівана: людям це зазвичай подобається. Їм насправді подобається програмувати з ним, а не розглядати його як ще один інструмент для виконання завдань. Часто вони кажуть, що упускають якусь функцію Python, коли їм потрібно працювати іншою мовою. І це те, що відрізняє Python від більшості його аналогів.

Python — не найкраща мова для кожної ситуації. За замовчуванням вона встановлюється не скрізь. Вона достатньо швидка для більшості програм, але

може бути недостатньо швидкою для деяких більш вимогливих випадків. Якщо програма витрачає більшу частину свого часу на обчислення (технічний термін пов'язаний з процесором), програма, написана на C, C++ або Java, як правило, працюватиме швидше, ніж її еквівалент на Python.

– Іноді кращий алгоритм (поетапне рішення) в Python перевершує неефективний в C. Більша швидкість розробки в Python дає більше часу для експериментів з альтернативами.

– У багатьох програмах довгий час очікування, очікуючи відповіді від якогось сервера в мережі. ЦП (центральний процесор, чіп комп'ютера, який виконує всі обчислення) практично не задіяний; отже, кінцевий час між статичними та динамічними програмами буде близьким.

– Стандартний інтерпретатор Python написаний на C і може бути розширений кодом C.

– Інтерпретатори Python стають швидшими. Java була страшенно повільною в зародковому стані, і багато досліджень і грошей пішло на її прискорення. Python не належить корпорації, тому його вдосконалення відбувалися поступово.

– Можливо, є надзвичайно вимоглива програма, і що б ви не робили, Python не відповідає потребам. Звичайними альтернативами є C, C++ і Java, але новіша мова під назвою Go (яка схожа на Python, але працює як C) може бути відповіддю.

Найбільша проблема, з якою зараз зіткається людина, полягає в тому, що існують дві версії Python. Python 2 існує завжди і попередньо встановлено на комп'ютерах Linux та Apple. Це була чудова мова, але немає нічого ідеального. У комп'ютерних мовах, як і в багатьох інших областях, деякі помилки є косметичними і їх легко виправити, тоді як інші важко виправити. Жорсткі виправлення несумісні: нові програми, написані з ними, не працюватимуть у старій системі Python, а старі програми, написані до виправлення, не працюватимуть у новій системі. Творець Python (Гвідо ван Россум) та інші вирішили об'єднати жорсткі виправлення разом і назвати його Python 3. Python 2

– це минуле, а Python 3 – майбутнє. Остання версія Python 2 — 2.7, і вона буде підтримуватися протягом тривалого часу, але не буде Python 2.8. Нова розробка буде в Python 3.

ВИСНОВКИ ДО РОЗДІЛУ 2

Відповідно, мова Python залишається актуально, та використовується в багатьох різних випадках. Таким чином, варто зауважити, що мова Python – це правильний вибір для реалізації досліджуваного проекту.

РОЗДІЛ 3. ЗАСОБИ РОЗРОБКИ

3.1 Вступ

Основним середовищем розробки було вибрано середовище PyCharm, яке надає інструменти аналізу коду, відладки та рефакторингу. PyCharm розроблене компанією «JetBrains». Також використовується Qt Creator - інтегроване середовище розробки, призначене для створення крос-платформових додатків за допомогою бібліотеки Qt. Бібліотека реалізована в Python-модулях, та охоплює близько 1000 класів. PyQt розробляється англійською компанією «Riverbank Computing». Підтримуються операційні системи Microsoft Windows, Linux, OS X, iOS та Android.

Для розробки застосунка використовувалась мова програмування Python версії 3.9.7. Python - це мова програмування високого рівня загального призначення, яка використовується в тому числі і для розробки веб-додатків.

3.2 Мова програмування Python

Python – це мова комп’ютерного програмування, яка часто використовується для створення веб-сайтів і програмного забезпечення, автоматизації завдань і аналізу даних. Python є мовою загального призначення, що означає, що його можна використовувати для створення різноманітних програм. Ця універсальність разом зі зручністю для початківців робить її однією з найбільш широко використовуваних мов програмування сьогодні.

Мова зосереджена на підвищенні продуктивності розробників і здатності читати код. Python вміщає в собі декілька парадигм програмування: структуроване, об’єктно-орієнтоване, функціональне, імперативне і аспектно-орієнтоване. Ця мова динамічно типізована, автоматичне керування пам'яттю, повний самоаналіз, механізм обробки винятків, підтримка багато поточних обчислень і зручні високо рівневі структури даних. Код на Python організований у

вигляді функцій та класів, які можна об'єднати в модулі, які, у свою чергу, можна об'єднати в пакети.

Python популярний з кількох причин. Ось більш глибокий погляд на те, що робить його настільки універсальним і простим у використанні для програмістів:

– Він має простий синтаксис, який імітує природну мову, тому його легше читати та розуміти. Це дозволяє швидше створювати проекти та швидше їх покращувати.

– Він універсальний. Python можна використовувати для багатьох різних завдань, від веб-розробки до машинного навчання.

– Він зручний для початківців, що робить його популярним серед програмістів початкового рівня.

– Це відкритий вихідний код, що означає, що його можна безкоштовно використовувати та розповсюджувати навіть у комерційних цілях.

– Архів модулів і бібліотек Python — пакети коду, які сторонні користувачі створили для розширення можливостей Python — величезний і зростає.

– Python маFє велику та активну спільноту, яка вносить свій внесок у архів модулів і бібліотек Python, а також виступає як корисний ресурс для інших програмістів. Велика спільнота підтримки означає, що якщо програміст зможе знайти відповідь на своє питання відносно легко; хтось обов'язково стикався з такою ж проблемою раніше.

Python неважкий і через те його просто вивчити, через те що він вимагає унікального синтаксису, зосередженого на читанні. Розробники можуть читати та розбиратися в коді Python легше, ніж в решті мов. Це, у свою чергу, зменшує витрати на підтримку та розробку програм, тому що дозволяє командам програмувати спільно без значних бар'єрів у мові та досвіді. З огляду на те що Python підтримує використання модулів і пакетів, що означає, що програми дозволено розробляти модульно, а код можна ще раз застосовувати в різних проектах. Після того, як ви розробили необхідні модулі або пакети, ви можете розширити їх для застосування в інших проектах і просто імпортувати чи то

експортувати ці модулі. Однією з найбільш багатообіцяючих переваг Python є те, що і стандартна бібліотека, і інтерпретатор є вільно доступними, як у двійковому, так і у формі вихідного коду. Ексклюзивності у свою чергу немає, через те що Python і всі необхідні інструменти доступні на всіх основних платформах. Так що це привабливий варіант для компаній, які не хочуть піклуватися про оплату високих витрат на розробку.

3.3 Середовище розробки

3.3.1 Середовище розробки PyCharm

PyCharm - одна з найпопулярніших IDE Python. Є багато причин для цього, включаючи той факт, що він розроблений JetBrains, розробником популярної IDE IntelliJ IDEA, яка входить до трійки кращих Java IDE і є" найрозумнішою IDE JavaScript " WebStorm. Доступність використання підтримки веб-розробки з Django є ще однією вагомою причиною.

Доступний у вигляді кросплатформенного додатка, PyCharm сумісний з платформами Linux, macOS і Windows. Витончено вписуючись в число кращих IDE Python, PyCharm забезпечує підтримку як версій Python 2 (2.7), так і Python 3 (3.5 і вище).

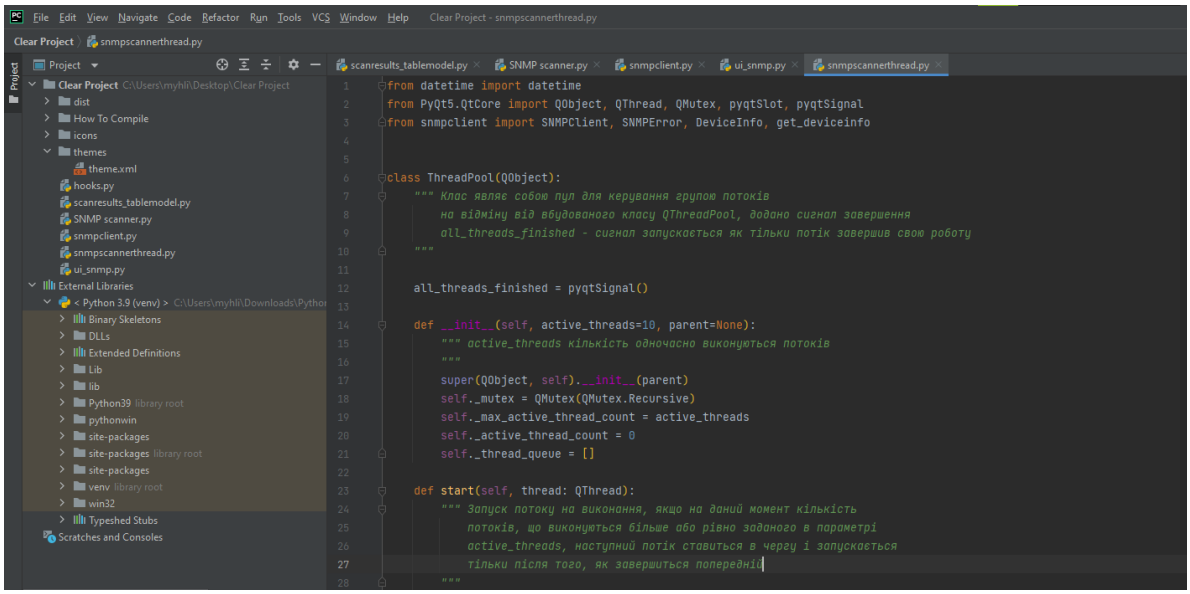


Рисунок 3.1 – Графічний інтерфейс PyCharm

PyCharm поставляється з безліччю модулів, пакетів та інструментів, що прискорюють розробку Python і одночасно скорочують зусилля, необхідні для виконання того ж самого в значній мірі. Крім того, PyCharm може бути налаштований відповідно до вимог розробки та особистих уподобань. Вперше він був представлений широкій публіці ще в лютому 2010 року. На додаток до аналізу коду, функції PyCharm:

- Графічний відладчик
- Вбудований модульний тестер
- Підтримка інтеграції для систем контролю версій (VCSs)
- Підтримка науки про дані за допомогою Anaconda

3.3.2 Середовище розробки Qt Creator

Qt Creator - це повністю інтегроване середовище розробки (IDE), яке надає вам інструменти проектування і розробки складних додатків для безлічі настільних і мобільних платформ. Ви можете або створити проект з нуля, або імпортувати існуючий проект. Qt Creator генерує всі необхідні файли в залежності від типу створюваного проекту. Наприклад, якщо ви оберете створення Програми з графічним інтерфейсом користувача (GUI), Qt Creator створить порожній .ui файл, який ви можете змінити в інтегрованому Qt Designer.

Qt Creator інтегрований з кроссплатформенними системами автоматизації збірки: qmake і CMake. Також ви можете імпортувати існуючі проекти, які не використовують qmake або CMake, і вказати Qt Creator просто проігнорувати вашу систему збірки.

Ви можете використовувати Qt Designer щоб розташовувати і налаштовувати ваші віджети або діалоги і тестувати їх використовуючи різні стилі і розміру екрану. Створені за допомогою Qt Designer віджети і форми легко інтегруються в програмний код з використанням механізму сигналів і слотів Qt, які дозволять вам легко визначити поведінку графічних елементів. Всі властивості, встановлені в Qt Designer, можуть бути динамічно змінені в коді.

Більш того, такі особливості як просування віджетів і власні модулі дозволять вам використовувати власні віджети з Qt Designer.

3.4 Технології, використані при розробці програми

Для розробки програми було використано декілька бібліотек Python, а саме sys, os, shutil, typing, datetime, PyQt5, PySide2, qt_material, pysnmp, ipaddress, netifaces, pandas, sqlite3.

Модуль sys надає програмісту набір функцій, які дають інформацію про те, як інтерпретатор Python взаємодіє з операційною системою.

Модуль «sys» дає наступну інформацію:

– Яка версія пітона запущена.

– Шлях до інтерпретатора Python, що виконує поточний скрипт.

– Параметри командного рядка, використовувані при запуску на виконання скрипта.

– Прапори, встановлені інтерпретатором.

– Представлення значень з плаваючою точкою.

– Багато іншого.

Модуль sys часто використовують з модулем os. За допомогою sys отримують потрібну інформацію про операційну систему, щоб уникнути непередбачених помилок, а за допомогою os взаємодіють з нею (робота з файлами, запуск програм на виконання, обробка шляхів і так далі).

```
import os
os.remove("D:\\test.txt")
```

Рисунок 3.2 – Код для видалення файлу

Всі бібліотеки (модулі) в Python 3 підключаються за допомогою команди «import ім'я бібліотеки». Існують різні варіанти підключення, які використовуються при певних обставинах:

– import sys - згаданий спосіб використовується, коли у вашому проекті можуть застосовуватися такі ж імена змінних або функцій, як і в бібліотеці. При такому підключенні виклик функцій або змінних з модуля виконується так « " sys.імя_функції"»

– from sys import * - спосіб використовується тоді, коли програміст впевнений, що перетинів Імен немає. При виклику методів або функцій не потрібно ставити префікс «sys.", що зменшує кількість коду. Зірочку можна замінити на імена конкретних функцій, перераховані через кому. Це дозволяє імпортувати не весь модуль, а лише деякі його частини.

– import sys as s - при такому записі назва "sys «замінюється на»s". Спосіб використовують, щоб замінити довгу назву модуля на більш короткий і зручний. Це дозволяє і зменшити кількість коду, і уникнути перетину імен.

```
import sys
print(sys.exec_prefix)
>> C:\Users\myhli\AppData\Local\Programs\Python\Python39
```

Рисунок 3.3 – Код який показує, в який каталог встановлено Python

Бібліотека під назвою shutil включає в себе кілька корисних функцій для створення копій об'єктів на жорсткому диску. Щоб швидко скопіювати файл у вихідний каталог, варто скористатися методом copyfile, попередньо підключивши модуль shutil.

У ролі першого аргументу тут виступає оригінальний документ, в той час як другим параметром потрібно поставити передбачуваний новий файл. Варто враховувати, що копіюється тільки вміст, але не метадані. У наступному прикладі

відбувається копіювання даних з файлу test.txt в test2.txt на диску D. Функція copyfile також повертає адресу створеного документа.

```
import shutil

shutil.copyfile("D:\\test.txt", "D:\\test2.txt")
```

Рисунок 3.4 – Код для копіювання вмісту файлу

Вбудований метод copy з модуля shutil дозволяє в Python копіювати файл в зазначену папку, зберігаючи при цьому його початкове ім'я.

Модуль «typing» в Python, представлений в Python 3.5, намагається надати спосіб підказки типів, щоб допомогти засобам перевірки статичних типів і лінтерам точно передбачати помилки. Через те, що Python повинен визначати тип об'єктів під час виконання, розробникам іноді дуже складно дізнатися, що саме відбувається в коді.

Навіть зовнішні засоби перевірки типів, такі як PyCharm IDE, не дають кращих результатів. В середньому тільки правильно прогнозує помилки приблизно в 50% випадків, згідно з цією відповіддю на StackOverflow. Python намагається пом'якшити цю проблему, вводячи так зване вказівку типу (Анотація типу), щоб допомогти зовнішнім засобам перевірки типів ідентифікувати будь-які помилки. Це хороший спосіб для програміста натякнути на тип використовуваного об'єкта (ів) під час самої компіляції і переконатися, що засоби перевірки типів працюють правильно. Це також робить код Python більш читабельним і надійним для інших читачів!

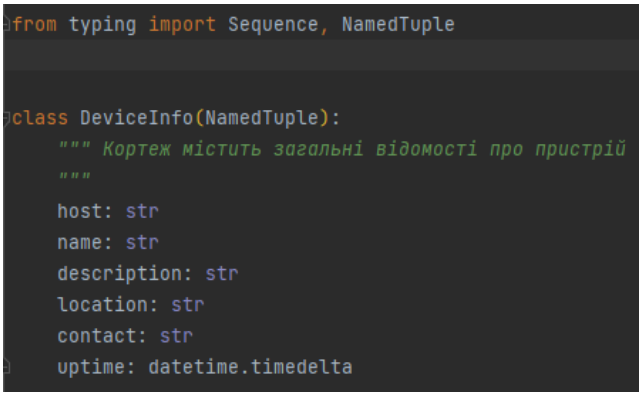


Рисунок 3.5 – Код для створення іменованого кортежу

Бібліотека «datetime» зі стандартної бібліотеки мови програмування Python являє собою збірник з самих різних класів для комфортної роботи з часом і датами. За рахунок безлічі вбудованих методів, призначених для зручного відображення, а також маніпуляції над часом і датами, підвищується функціональність деяких програм.

Бібліотека datetime використовується для роботи в Python з часом і датами, дозволяючи представляти дану інформацію в найбільш зручній формі.

Вона складається з декількох класів. Завдяки їх наявності, програміст отримує доступ до багатьох корисних методів:

– Отримання поточних системних дати і часу;
– Обчислення різниці між датами та інші арифметичні операції;
– Операцій, які дозволяють порівнювати час;
– Форматований висновок інформації про дату і час.

У модулі використовуються константи MINYEAR і MAXYEAR, які дорівнюють 1 і 9999 відповідно. Це мінімальне і максимально можливе значення року, використовувані в бібліотеці.

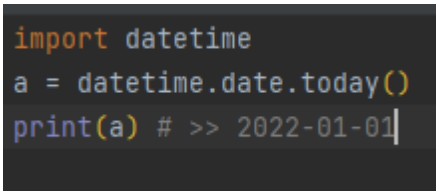


Рисунок 3.6 – Код для отримання сьогоднішньої дати

PyQt5, PySide2-це набір інструментів для графічного інтерфейсу. Це інтерфейс Python для Qt, однієї з найпотужніших і популярних кроссплатформенних бібліотек графічного інтерфейсу. Основна відмінність PyQt5 і PySide2 - це Ліцензії під якими поширюються ці дві обгортки над Qt. PyQt5 поширюється під GPL і комерційною ліцензією. PySide2 поширюється як Qt під GPL, LGPL і комерційною ліцензією.PyQt був розроблений компанією Riverbank Computing Ltd.

PyQt API-це набір модулів, що містять велику кількість класів і функцій. У той час як модуль QtCore містить функції не-GUI для роботи з файлами, каталогами і т.д., Модуль QtGui містить всі графічні елементи управління. Крім того , існують модулі для роботи з XML (Getxml), SVG (Svg), SQL (Tsql) і т. д.

Створення простого додатка з графічним інтерфейсом з використанням PyQt включає в себе наступні кроки:

– Імпортувати модуль QtGui;

– Створіть об'єкт програми;

– Об'єкт QWidget створює вікно верхнього рівня. Додайте в нього об'єкт Qlabel;

– Встановіть заголовок ярлика ”Привіт, світ”;

– Визначте розмір і положення вікна методом setGeometry ();

– Введіть основний цикл програми методом app.exec_ ()

```
import sys
from PyQt5.QtWidgets import QApplication, QWidget, QLabel
def window():
   app = QApplication(sys.argv)
   widget = QWidget()
   textLabel = QLabel(widget)
   textLabel.setText("Hello World!")
   textLabel.move(110,85)
   widget.setGeometry(50,50,320,200)
   widget.setWindowTitle("PyQt5 Example")
   widget.show()
   sys.exit(app.exec_())
if __name__ == '__main__':
   window()
```

Рисунок 3.7 – Код програми Hello World

Скомпілюємо даний код та отримаємо результат:

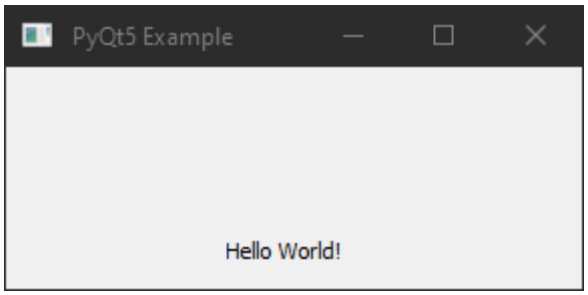


Рисунок 3.8 – Результат програми Hello World

Для PyQt5 існують бібліотеки для розширення її функціонування одною з таких є «qt_material». «qt_material» – таблиця XML стилів для PySide2, PyQt5.

```
import sys
from PyQt5 import QtWidgets
from qt_material import apply_stylesheet
app = QtWidgets.QApplication(sys.argv)
window = QtWidgets.QMainWindow()
apply_stylesheet(app, theme='dark_teal.xml')
window.show()
app.exec_()
```

Рисунок 3.9 – Код програми із XML стилями

При запуску коду, отримаємо вікно до якого була використана таблиця XML стилів.

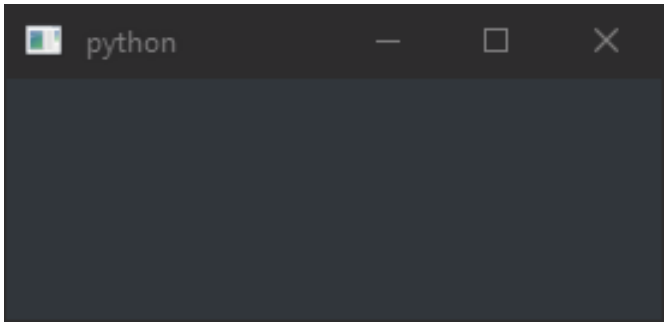


Рисунок 3.10 – Результат програми

Модуль «Pysnmp» - це чисто пітоновська реалізація, з відкритим вихідним кодом і безкоштовною реалізацією SNMP-движка v1 / v2c / v3, поширювана за ліцензією BSD з 2 пунктами.

Функції які реалізовані:

– Повна підтримка SNMPv1 / v2c і SNMPv3;

– Платформа SMI для підтримки інформації MIB і реалізації об'єктів, керованих SMI;

– Повна реалізація сутності SNMP;

– Підтримка розширених параметрів безпеки USM (3DES, 192/256-бітове шифрування AES);

– Розширювана платформа мережевих протоколів (UDP / IPv4, UDP / IPv6);

– Підтримка API вводу-виводу на основі асинхронних сокетів;

– Інтеграція «Twisted», «Asyncio» і «Trollius»;

– Інтеграція «Pesmi» для динамічної компіляції MIB;

– Вбудовані прилади, що розкривають операції механізму протоколів;

– 100% Python, працює з Python 2.6, через версію 3.7.

Модуль «ipaddress» - надає можливості для створення, управління і роботи з адресами і мережами IPv4 і IPv6.

Функції та класи в цьому модулі спрощують виконання різних задач, пов'язаних з IP-адресами, включаючи перевірку того, чи знаходяться два хоста в одній підмережі, перебір всіх хостів в певній підмережі, перевірку того, чи представляє рядок допустимий IP-адреса або визначення мережі, і так далі.

Модуль «netifaces» - портативна стороння бібліотека на Python для перерахування мережевих інтерфейсів на локальній машині.

Історично склалося так, що було важко безпосередньо отримати мережеву адресу (адреси) комп'ютера, на якому виконуються ваші скрипти Python, без шкоди для переносимості вашого скрипта. «netifaces» піклується про перерахування інтерфейсів, мережевих адрес, а також зберігає переносимість.

pandas - це високорівнева пітон бібліотека для аналізу даних. Чому її називають високорівневою, тому що побудована вона поверх більш низькорівневої бібліотеки NumPy (написана на C), що є великим плюсом в продуктивності. екосистемі Python pandas є найбільш продвинутою та швидкорозвиваючою бібліотекою для обробки та аналізу даних.

Модуль «sqlite3» -це автономна файлова база даних SQL. SQLite поставляється в комплекті з Python і може використовуватися в будь-якому з ваших додатків на Python без необхідності установки будь-якого додаткового програмного забезпечення.

SQLite для Python пропонує менше типів даних, ніж є в інших реалізаціях SQL. З одного боку, це накладає обмеження, але, з іншого боку, в SQLite багато чого зроблено простіше. Ось основні типи:

– NULL-значення NULL;

– INTEGER-ціле число;

– REAL-число з плаваючою точкою;

– TEXT-текст;

– BLOB-бінарне представлення великих об'єктів, що зберігається в точності з тим, як його ввели.

ВИСНОВКИ ДО РОЗДІЛУ 3

У даному розділі було наведено перелік інструментів, з допомогою яких створювався SNMP сканер. Були використані переваги мови програмування Python. Також були використані пакети PyQt5, PySide2, за допомогою якого було створено GUI. Для побудови файлу бази даних було використано пакет Pandas та його методи перетворення даних з PyQt5 віджетів в excel підтримуваний тип файлу.

РОЗДІЛ 4. ОПИС ПРОГРАМНОЇ РЕАЛІЗАЦІЇ

4.1 Вхідна і вихідна інформація

Розроблена система повинна мати такі функції програмних модулів:

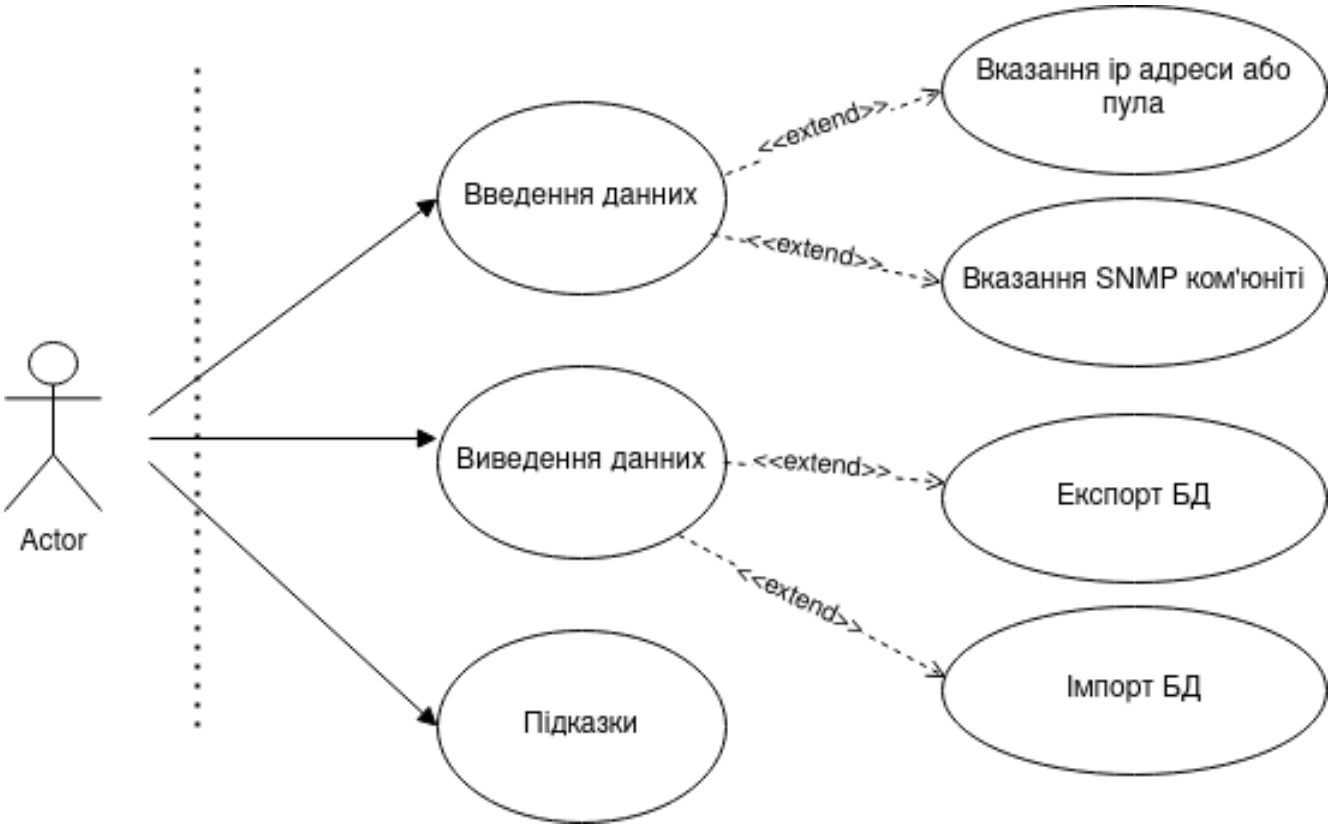


Рисунок 4.1 – Функції програмних модулів

При розробці системи були підключені бібліотеки, описані в попередньому розділі. За допомогою імпорту код Python в одному модулі може одержувати доступ до коду в іншому модулі. Інакше кажучи тоді як модуль імпортується, Python виконує цілий код у файлі модуля. Оголошення імпорту поєднує дві операції; воно шукає іменований модуль, а потім прив’язує результати цього пошуку до імен у локальній області.

Процедура пошуку імпорту оператора визначається як виклик функції __import__() з відповідними аргументами. Повернене значення __import__() використовується для виконання операції зв’язування імені оператора імпорту. Прямий виклик __import__() тільки виконує пошук модуля, а якщо знайдено,

виконує операцію створення модуля. Хоча можуть з'являтися деякі побічні ефекти, скажімо імпорт батьківських пакетів та оновлення різних кешів (включаючи sys. modules), тільки оператор імпорту виконує операції зв’язування імен.

Коли виконується оператор імпорту, викликається стандартна вбудована функція __import__(). Інші механізми виклику імпорту (такі як importlib. import_module()) можуть обійти __import__() і застосовувати власні рішення для реалізації семантики імпорту. Тоді як ви імпортуєте модуль вперше, Python шукає модуль, і якщо він його знаходить, він створює модульний об’єкт та ініціалізує його. Якщо названий модуль не знайдено, виникає помилка ModuleNotFoundError.

Під час виклику import Python реалізує різні стратегії пошуку для іменованих модулів. Ці стратегії дозволено модифікувати та розширювати кількома способами. Конструкція "import . as . " час від часу використовуються для полегшення використання бібліотек. Інакше кажучи при його використанні код запише розробнику не повну назву, а назву бібліотеки, яку вказав сам розробник. У моєму випадку модуль pandas. викликається як pd (рисунок 4.2).

```
from ipaddress import IPv4Network
import netifaces
import pandas as pd
import sqlite3
```

Рисунок 4.2 – Використання import … as …

4.2 Вимоги до апаратного і програмного забезпечення

Висуваються такі вимоги до апаратного забезпечення користувача:

– процесор Intel Pentium 4 або будь-який інший сумісний з підтримкою набору інструкцій SSE2;

– RAM 1 ГБ і більше;

– HDD 20 Гб і більше.

– СКБД SQLite 3, або будь-яка інша сумісна.

Вимоги до програмного забезпечення користувача: – операційна система Windows 7 або Windows 8, або Windows 8.1, або Windows 10

4.3 Запуск системи

Для запуску розробленої програми у форматі «.py» необхідно:

1. Скачати і встановити PyCharm з офіційного сайту.

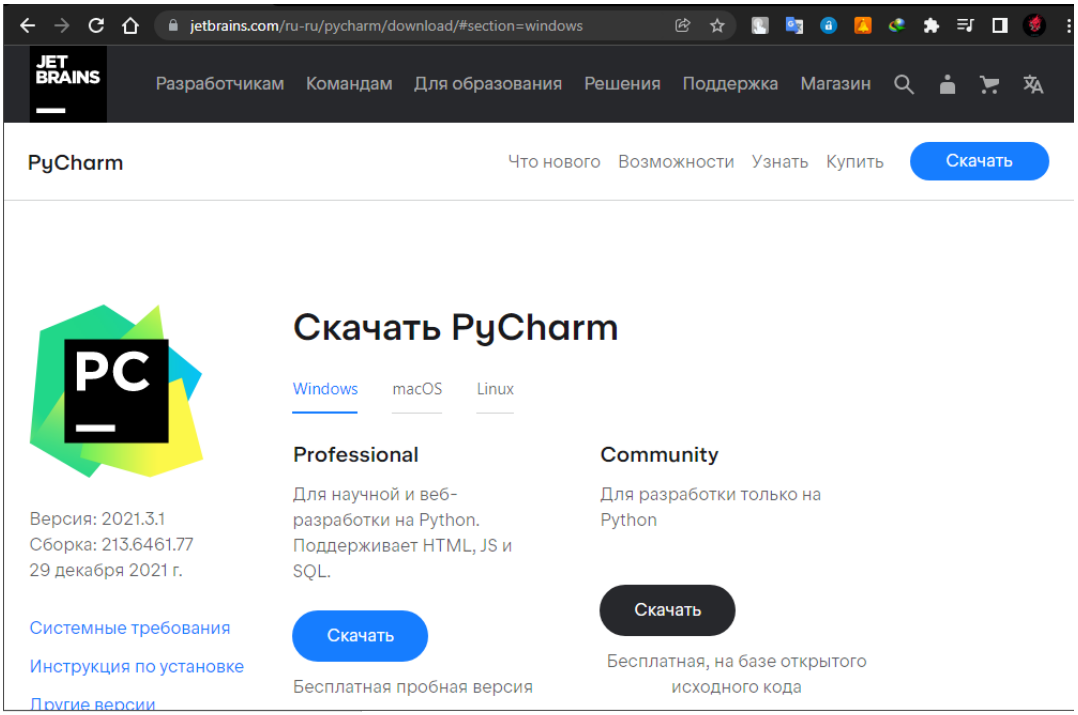


Рисунок 4.3 – Сторінка загрузки IDE PyCharm

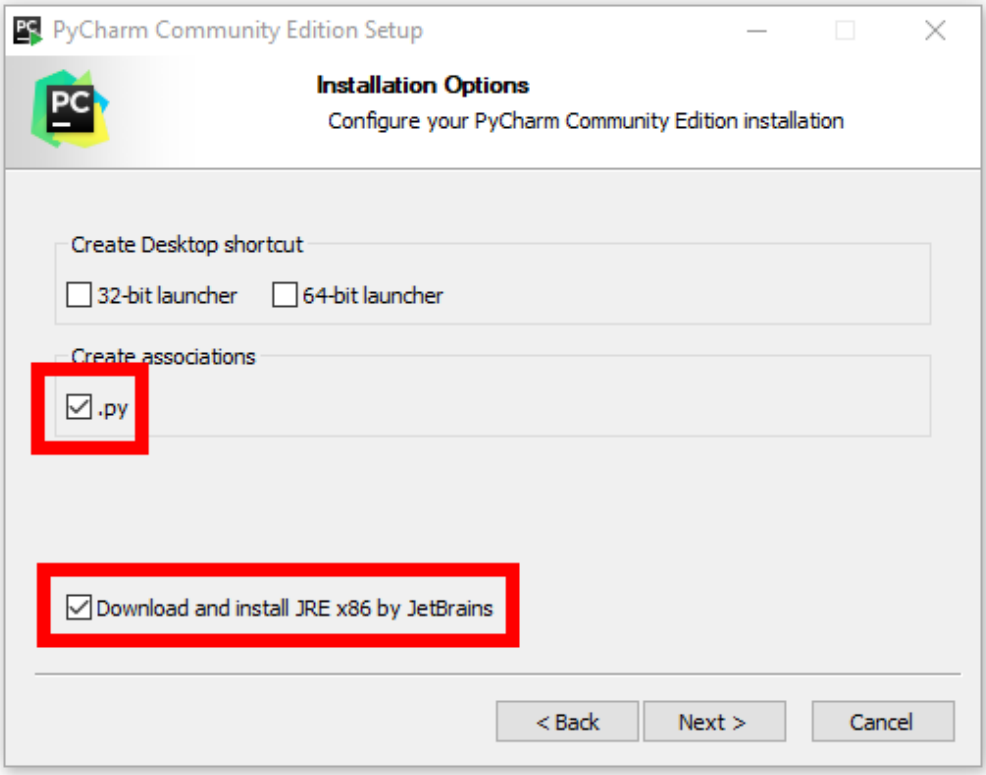


Рисунок 4.4 – Процес установки IDE PyCharm

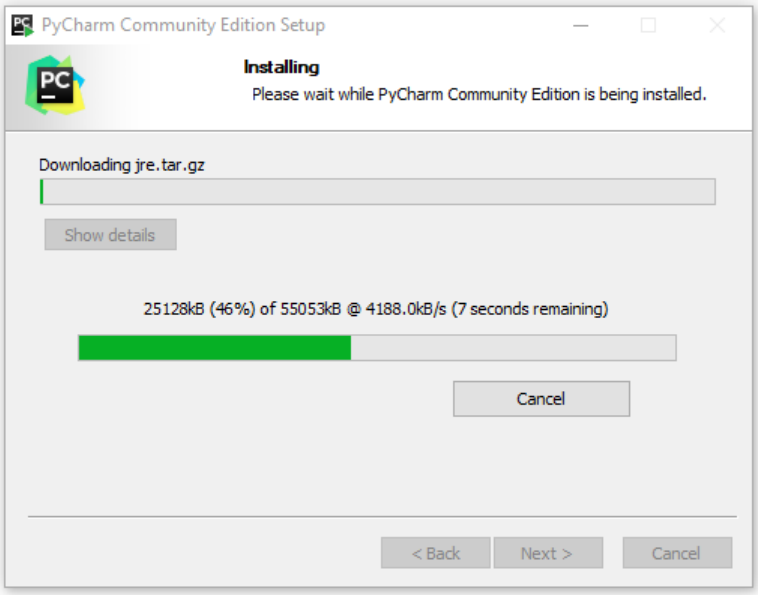


Рисунок 4.5 – Закінчення установки IDE PyCharm

При загрузці PyCharm, якщо у вас не буде установлено python в системі, IDE сама запропонує встановити інтерпретатор мови програмування.

Після чого потрібно викликати вбудовану консоль і виконайтие наступні команди:

1) cd '.\How To Compile\'
2) python .\get-pip.py

Першою командою ми переходимо в папку з нашими додатками для запуску основгого коду, там знаходиться pip - система керування пакунками, яка використовується для встановлення та управління програмними пакетами, які написані на Python. Наступною командою ми встановлюємо pip в нашу систему, для подальшої роботи з ним.

Не виходячи з консолі виконаємо команди, які встановлять команди для повного функціонування, назви пакетів знаходяться в файлі під назвою «requirements.txt» - файл залежностей програми, для автоматичного встановлення пакетів python за допомогою утиліти pip

Даний формат є звичайним текстовим файлом, де вказано назву пакета python. В консолі нам портібно буде написати і виконати:

1) pip install -r requirements.txt
2) pip install .\netifaces-0.11.0-cp39-cp39-win_amd64.whl

2. Запустити програму, двічі натиснувши на «SNMP scanner.py».

3.

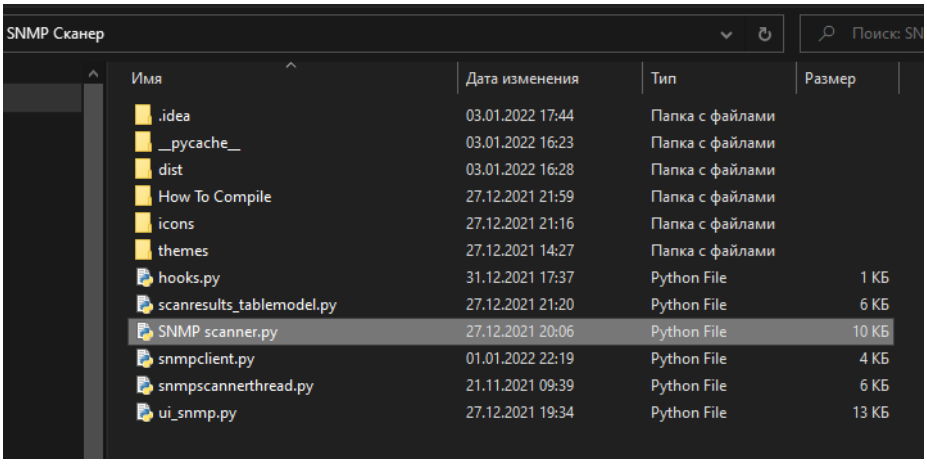


Рисунок 4.6 – Запуск програми через «.py» файл

Або через командний рядок, вказавши назву файлу.

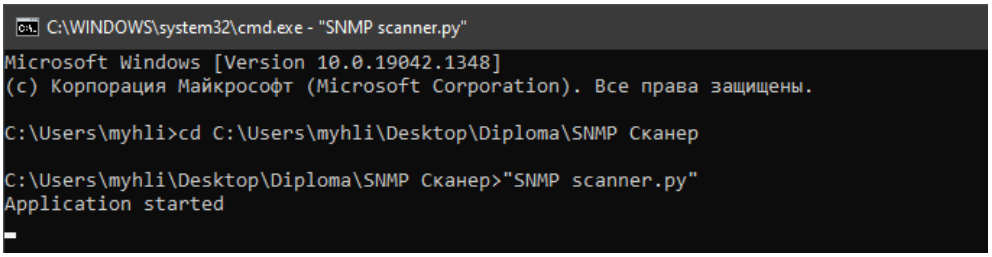


Рисунок 4.7 – Запуск програми через консоль

4. Для запуску програми у форматі .exe потрібно всього лише два рази клацнути на неї.

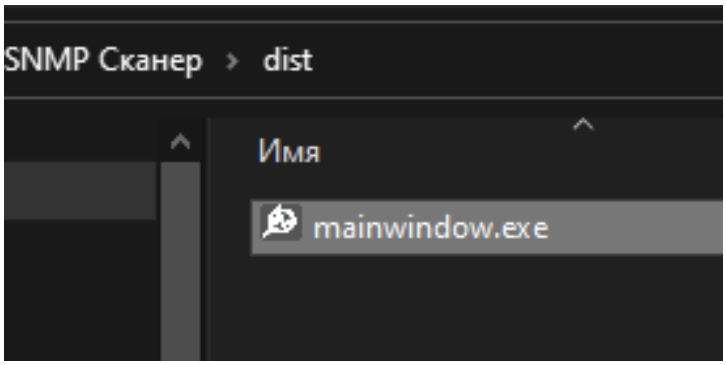


Рисунок 4.8– Запуск програми через виконуючий файл

Після запуску виконуючий файл в папці запуску створить 2 додаткові папки:

1. icons;
2. themes.

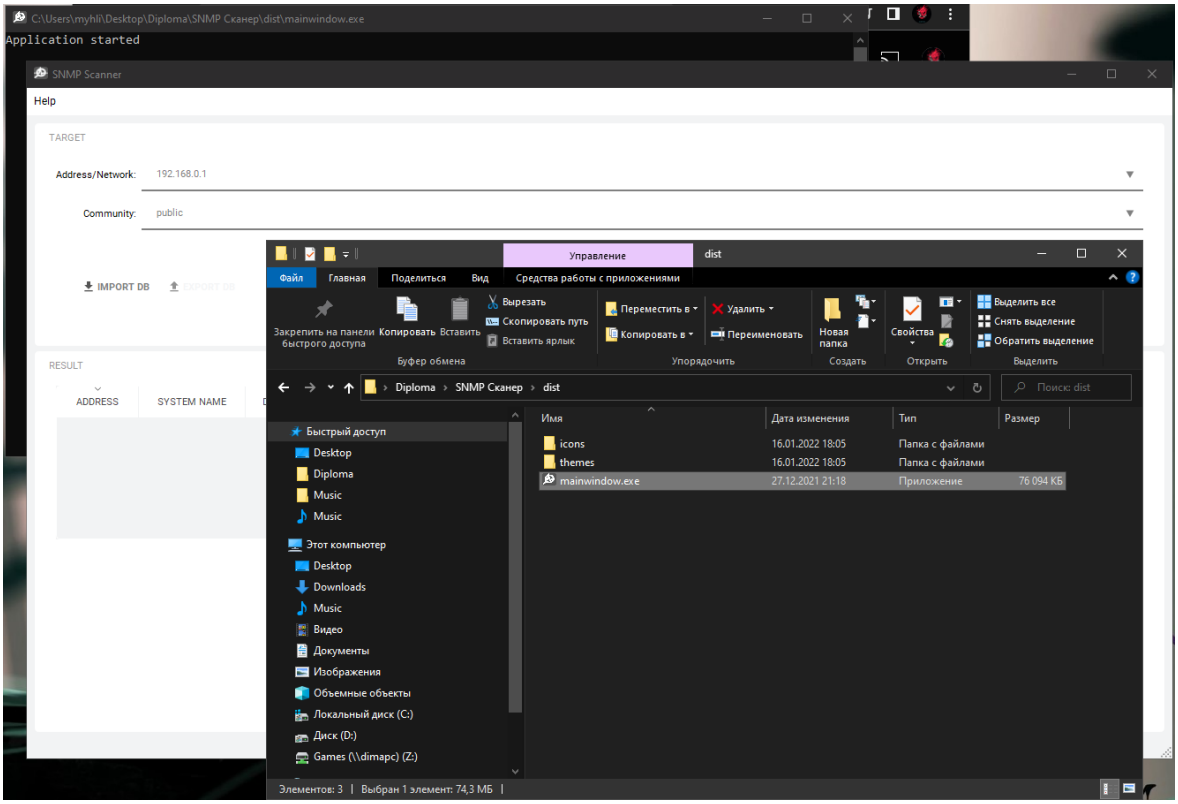


Рисунок 4.9– Результат запуску програми через виконуючий файл

4.3.1 Інтерфейс користувача

Інтерфейс користувача для надання вказання даних наведено на рисунку 4.10

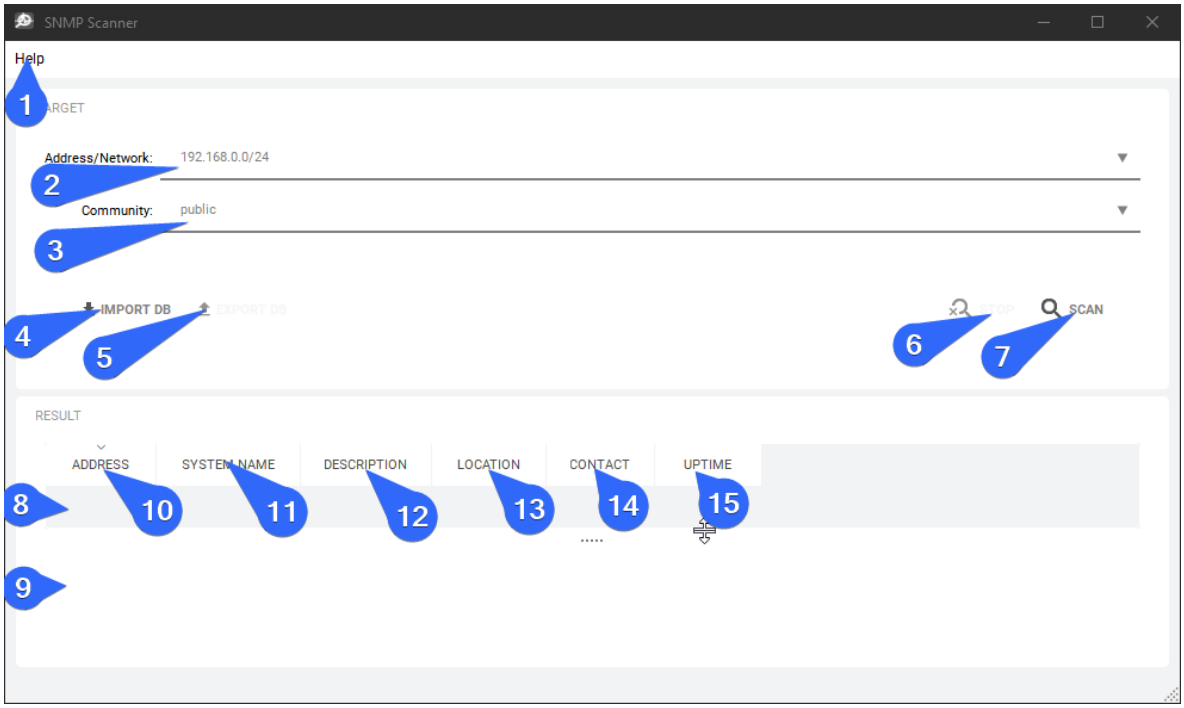


Рисунок 4.10 – Графічний інтерфейс екрану реєстрації нового користувача

Цифрами позначені:

1) Кнопка інформації про програму.

2) Поле вводу IP адреси. Поле має валідацію вводу.

3) Поле для вводу SNMP спільноти. Поле має валідацію вводу.

4) Кнопка імпортування існуючої БД в програму

5) Кнопка експортування таблиці SNMP в зовнішню БД.

6) Кнопка закінчення сканування мережі на SNMP хости.

7) Кнопка початку сканування мережі на SNMP хости.

8) Зона відображення даних знайдених SNMP хостів.

9) Зона відображення статусу програми.

10) Колонка адрес SNMP хостів.

11) Колонка імені систем SNMP хостів.

12) Колонка опису SNMP хостів.

13) Колонка місця знаходження SNMP хостів.

14) Колонка контактної інформації SNMP хостів.

15 Колонка загального часу роботи SNMP хостів.

Після того як користувач натисне кнопку «Start» під формами вводу, у разі успішного вводу даних, почнеться сканування мережі на наявність SNMP хостів.(рисунок 4.4).

Рисунок 4.11 – Процес сканування програми

У перше поле потрібно ввести IP адресу хоста або пул адрес з маскою, у разі не коректного вводу поля «Address/Network» - адреси і підмережі в програмі спрацює виключення і з’явиться помилка(рисунок 4.5)

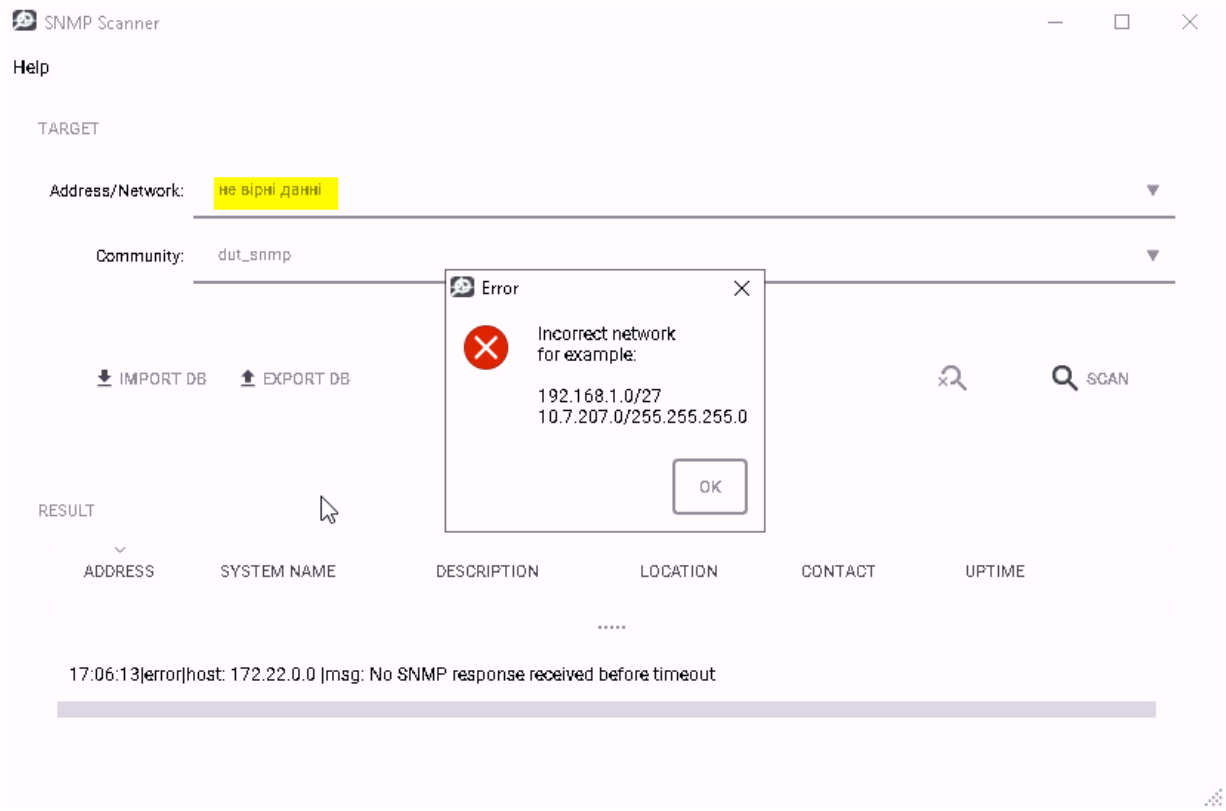


Рисунок 4.12 – Процес сканування програми

В помилці користувачу повідомляється «Некоректна мережа» і дається приклад як дані слід ввести для коректної ініціалізації програми.

В залежності від того яку підмережу IP адрес введе користувач, залежить швидкість сканування і того на скільки швидко програма закінчить свою роботу. Сама програма може використовувати багатопотоковість процесора, що значно збільшує її продуктивність і швидкодію, як результат – чим більше ядер і потоків в процесорі тим швидше відбуватиметься сканування. Результат сканування можливо споглядати в ході роботи програми в реальному часі в зоні відображення статусу програми, який ми розглянули вище.

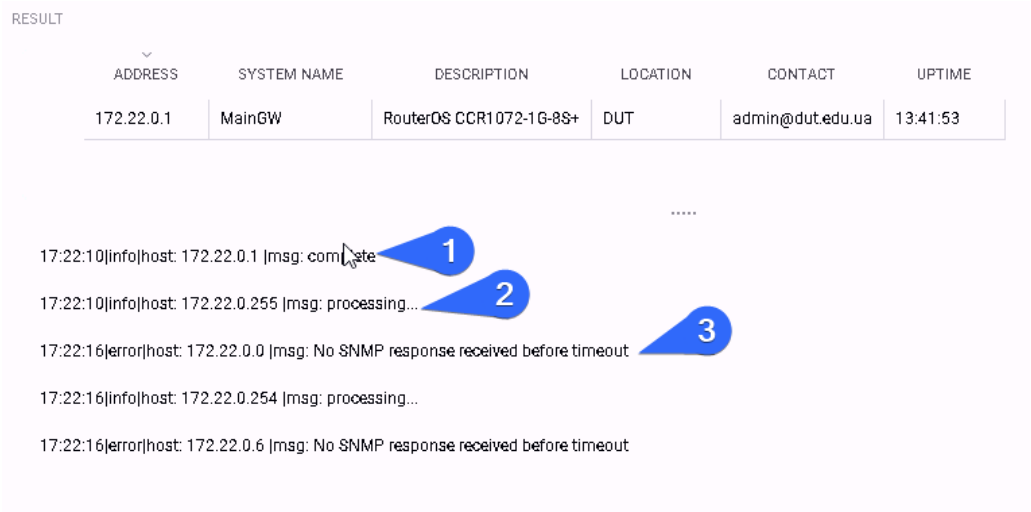


Рисунок 4.13 – Процес сканування програми в зоні статусу

Цифрами позначені:

1) Рядок статусу з даними про сканування IP адреси «172.22.0.1» завершено успішно.

2) Рядок статусу з даними про сканування IP адреси «172.22.0.255» розпочалось.

3) Рядок статусу з даними про сканування IP адреси «172.22.0.0» без відповіді від SNMP хоста.

ВИСНОВКИ ДО РОЗДІЛУ 4

У даному розділі було описано необхідні дії для запуску системи, проблеми які можуть виникнути при першій установці, та наведено інтерфейс користувача. Роз’яснено як користуватися даною системою, за рахунок чого вона працює і як може бути збільшена її продуктивність і швидкодія. Як приклад зроблено запуск програми, варіанти введення даних. Показано приклади помилок, що можуть виникнути під час роботи.

ВИСНОВКИ

Python — добре розроблена мова, яку можна використовувати для програмування в реальному світі. Python — це дуже високорівнева, динамічна, об'єктно-орієнтована мова програмування загального призначення, яка використовує інтерпретатор і може використовуватися у величезній області програм.

Python був розроблений так, щоб його було легко зрозуміти та використовувати. Останнім часом Python називають дуже зручною для початківців мовою. Python завоював популярність як мова, зручна для початківців, і він замінив Java як найпопулярнішу мову для ознайомлення. Як динамічно типізована мова, Python дійсно гнучкий.

Крім того, Python також більш прощає помилки, тому ви все одно зможете компілювати та запускати свою програму, поки не потрапите в проблемну частину. Python — це гнучка, проста мова програмування для кодування. Ця мова може підтримувати різні стилі програмування, включаючи структурні та об’єктно-орієнтовані. Можна використовувати й інші стилі. Python дуже гнучкий через його здатність використовувати модульні компоненти, розроблені іншими мовами програмування. Наприклад можна написати програму на C++ та імпортувати її в Python як модуль. Потім додати до нього щось інше (наприклад, створити для нього графічний інтерфейс).

Саме тому було обрано Python для розробки програмного забезпечення для обліку комутаційного обладнання мовами Python, яке було представлене в даній роботі.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Базурін В. М. Особливості python як першої мови програмування. 2021. 125 с.
2. Базурін В. М., Омелечко Є. А., Ковтун А. В. Порівняльний аналіз середовищ програмування мовою Python, 2018. 292 с.
3. Васильєв О. М. Програмування мовою Python. 2022.
4. Ковтанюк М. С., Криворучко І. І. Вивчення мови програмування python за допомогою вебресурсів. 2020.
5. Копей В. Б., Семанишин Л. М. Застосування мови програмування Python для побудови баз знань та експертних систем. 2018. 67 с.
6. Кухар М. А. Аналіз можливостей мови програмування Python для роботи з просторовими даними. 2019. 46 с.
7. Пасішніченко І. С. Дослідження технічних характеристик і використання вакуумного комутаційного обладнання. 2018.
8. Ascher D. et al. Numerical python. 2019.
9. Beazley D. M. Python essential reference. 2019.
10. Benmouyal G. et al. A unified approach to controlled switching of power equipment. 2018.
11. Davis J. T. Automation of a Production Switching System. 2015. 727 с.
12. De Smedt T., Daelemans W. Pattern for python 2018. 260 с.
13. Downey A. Think python. 2019
14. Dubois P. F., Hinsen K., Hugunin J. Numerical python. 1996. 267 с.
15. Fisher E. B., O'Neill R. P., Ferris M. C. Optimal transmission switching. 2018. 420 с.
16. Holmgren W. F. et al. PVLIB python 2015. 2015. 25 с.
17. Ito H. (ed.). Switching Equipment, 2019.
18. Kuhlman D. A python book: Beginning python, advanced python, and python exercises. 2019. 227 с.
19. Lutz M. Programming python. 2015.

20. Niayesh K., Runde M. Power switching components. NY: Springer International Publishing, 2017.

21. O'Boyle N. M., Morley C., Hutchison G. R. Pybel: Python 2018. 17 c.

22. Oliphant T. E. Python for scientific computing. 2017. 20 c.

23. Raschka S. Python machine learning. NY: Packt publishing ltd, 2015.

24. Shein E. Python for beginners. 2016. 21 c.

25. Vallat R. Pingouin: statistics in Python. 2018. 500 c.

26. Van Rossum G. An introduction to Python. Bristol : Network Theory Ltd., 2013. 115 c.

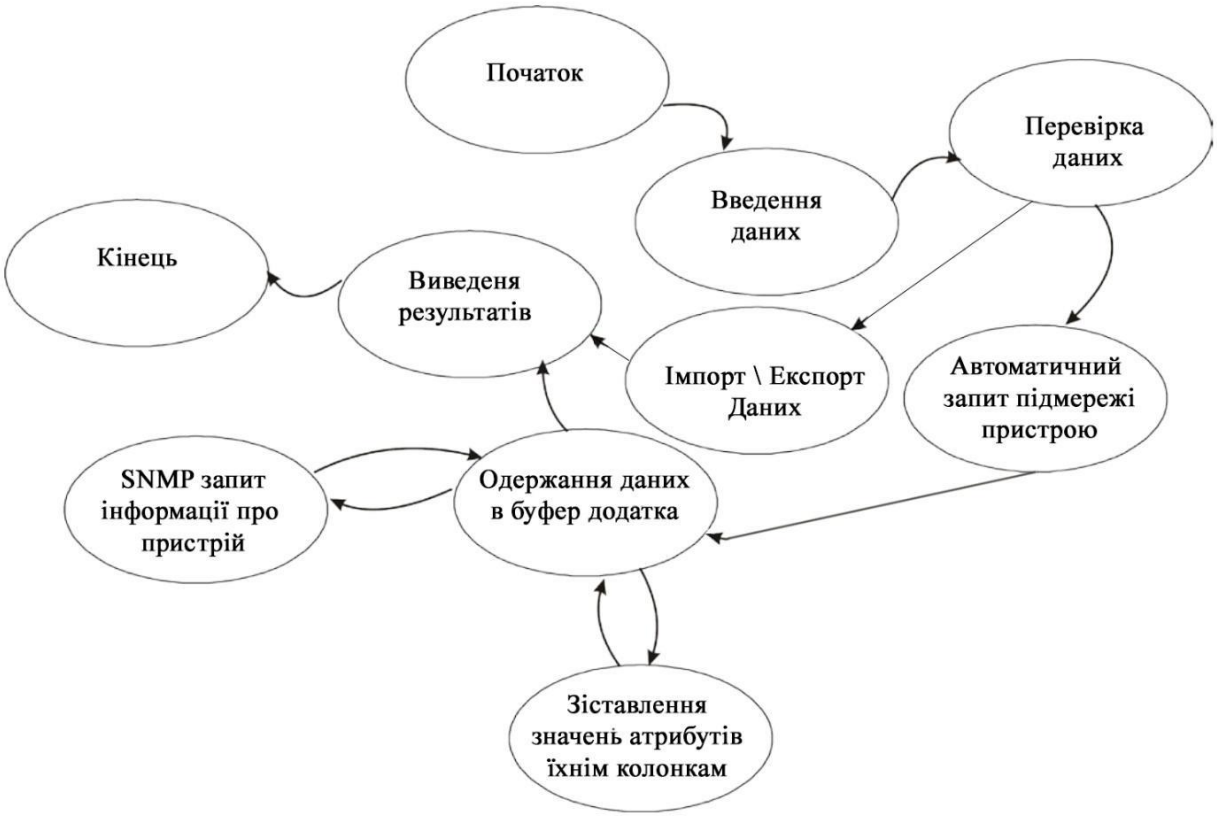
27. Van Rossum G. et al. Python Programming. 2017. 36 c.

28. Van Rossum G., Drake Jr F. L. Python tutorial. – Amsterdam, The Netherlands: Centrum voor Wiskunde en Informatica, 2018. 620 c.

ДОДАТКИ

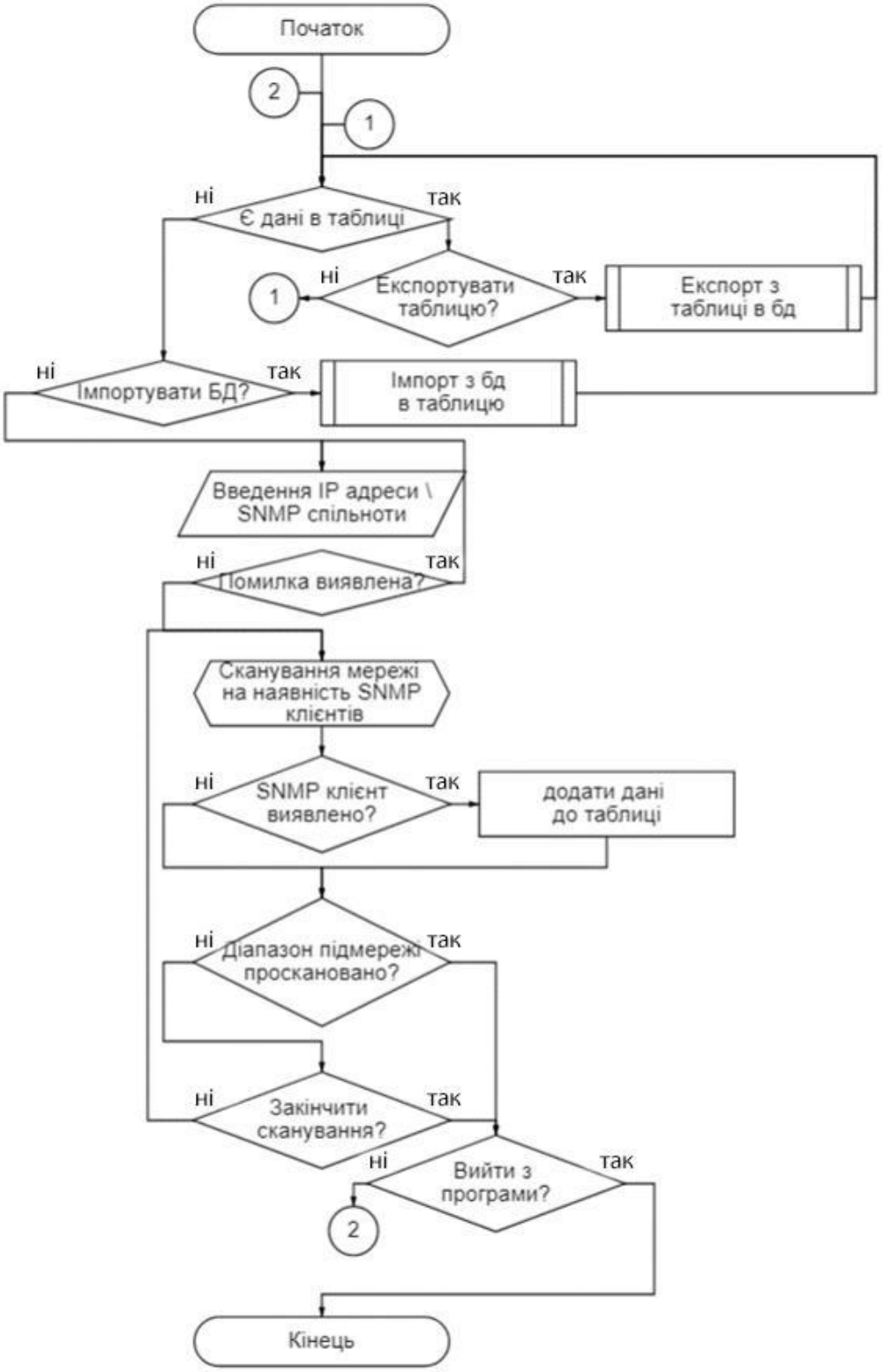
ДОДАТОК А

Функціональна діаграма основних компонентів системи



ДОДАТОК Б

Блок-схема роботи програми



ДОДАТОК В

Код файлу «compile exe.bat» для упакування додатку у виконуючий (.exe) файл

```
@ECHO OFF
echo !!!!!!!!
echo !!!!!!!!
echo «.bat» files work only in OS Windows
echo Put «compile exe.bat» in folder with your main script file to compile .exe file
echo !!!!!!!!
echo !!!!!!!! press ENTER to continue
pause
setlocal enableextensions
pyinstaller -F --onefile --noconsole --clean --icon="icons\appicon.ico" --add-data=themes;themes --add-data=icons;icons
--runtime-hook=hooks.py "SNMP scanner.py"
pause
```

ДОДАТОК Г

Код файлу «ui_compile.bat» для перетворення «.ui» файлу в pyhton (.py) файл

```
@ECHO OFF
echo !!!!!!!!
echo !!!!!!!!
echo «.bat» files work only in OS Windows
echo Put «ui_compile.bat» in folder with pyQt «.ui» file to convert it to python code,
echo then replace created python file to folder with main code
echo !!!!!!!!
echo !!!!!!!! press ENTER to continue
pause
setlocal enableextensions
pyuic5 -x snmp.ui -o ui_snmp.py
echo .
echo .
echo .
echo .
echo .
echo !DONE!
pause
```

ДОДАТОК Д

Код файлу «hooks.py» для кешування елементів графічного інтерфейсу при упакуванні коду у виконуваний файл

```
import sys
import os
import shutil

path = getattr(sys, '_MEIPASS', os.getcwd())
# Вказуємо шлях для збереження іконок інтерфейсу при упакуванні коду у виконуваний файл «.exe»
full_path = path+"\\icons"
try:
    shutil.move(full_path, ".\\icons")
except:
    print("Cannot create 'images' folder. Already exists.")

# Вказуємо шлях для збереження тем при упакуванні коду у виконуваний файл «.exe»
full_path = path+"\\themes"
try:
    shutil.move(full_path, ".\\themes")
except:
    print("Cannot create 'images' folder. Already exists.")
```

ДОДАТОК Е

Код файлу «scanresults_tablemodel.py» для перетворення отриманих SNMP даних у вигляд таблиці

```
from ipaddress import IPv4Address, get_mixed_type_key
from typing import List
from PyQt5.QtCore import Qt, QAbstractTableModel, QModelIndex, QVariant, QCoreApplication
from snmpclient import DeviceInfo

class ScanResultsTableModel(QAbstractTableModel):
    """ Модель реалізує обгортку, для відображення контейнера, що містить
        дані класу DeviceInfo у поданні таблиці
    """
    HEADERS = [QCoreApplication.translate('ScanResultsTableModel', 'Address'),
               QCoreApplication.translate('ScanResultsTableModel', 'System Name'),
               QCoreApplication.translate('ScanResultsTableModel', 'Description'),
               QCoreApplication.translate('ScanResultsTableModel', 'Location'),
               QCoreApplication.translate('ScanResultsTableModel', 'Contact'),
               QCoreApplication.translate('ScanResultsTableModel', 'UpTime')]

    COLUMN_COUNT = len(HEADERS)

    def __init__(self, parent=None, *args):
        """ Конструктор класа
        """
        QAbstractTableModel.__init__(self, parent, *args)
        self._data: List[DeviceInfo] = []
        self._to_insert = []

    def rowCount(self, parent) -> int:
        """ Повертає кількість рядків у таблиці
            перевизначений метод батьківського класу
        """
        return len(self._data)

    def columnCount(self, parent: QModelIndex) -> int:
        """ Повертає кількість колонок у таблиці
            перевизначений метод батьківського класу
        """
        return self.COLUMN_COUNT

    def data(self, index: QModelIndex, role: int = Qt.DisplayRole) -> QVariant:
        """ Повертає дані, що знаходяться за індексом у таблиці
            перевизначений метод батьківського класу
        """
        if (not index.isValid()) or role != Qt.DisplayRole:
            return None

        row = index.row()
        column = index.column()

        devinfo = self._data[row]

        if (row > self.rowCount(QModelIndex()) or row < 0):
            return None

        devinfo = self._data[row]

        # Порядок членів класу DeviceInfo відповідає порядку
        # заголовків у списку, тобто. якщо заголовок Description
```

```
        # у змінній HEADERS має індекс 2, то для devinfo[2]
        # буде повернено значення змінної devinfo.description
        return str(devinfo[column])

    def headerData(self, section: int, orientation: Qt.Orientation,
                   role: int = Qt.DisplayRole) -> QVariant:
        """ Отримання даних заголовка
            перевизначений метод батьківського класу
        """
        result = None
        if role != Qt.DisplayRole:
            return result

        if orientation == Qt.Horizontal:
            result = self.HEADERS[section]

        return result

    def insertRows(self, position: int, rows: int, parent: QModelIndex) -> bool:
        """ Вставка рядків
            перевизначений метод батьківського класу
        """
        if len(self._to_insert) == 0:
            return False

        self.beginInsertRows(QModelIndex(), position, position + rows - 1)
        for row in self._to_insert:
            self._data.append(row)

        self._to_insert.clear()
        self.endInsertRows()
        self.dataChanged.emit(parent, parent)

        return True

    def removeRows(self, row: int, count: int, parent: QModelIndex) -> bool:
        """ Видалення кількості рядків дорівнює count,  починаючи з row
            перевизначений метод батьківського класу
        """
        rowcount = self.rowCount(QModelIndex)
        if ((rowcount < row) or (rowcount > (row + count)) or (rowcount == 0)):
            return False

        self.beginRemoveRows(parent, row, row + count - 1)
        for index in range(row + count - 1, row, -1):
            del self._data[index]

        self.endRemoveRows()
        return True

    def sort(self, column: int, order: int):
        """ Сортування таблиці по колонці
            перевизначений метод батьківського класу
        """
        self.layoutAboutToBeChanged.emit()

        sortkey = lambda x: x[column]
        if column == 0:
            sortkey = lambda x: get_mixed_type_key(IPv4Address(x.host))

        self._data.sort(key=sortkey, reverse=True if order == Qt.DescendingOrder else False)
        self.layoutChanged.emit()
```

```
def add_row(self, devinfo: DeviceInfo):
    """ Додавання рядка, що є даними об'єкта DeviceInfo
    """
    self._to_insert.append(devinfo)
    self.insertRows(self.rowCount(QModelIndex()), 1, QModelIndex())

def remove_all_rows(self):
    """ Очищення всіх рядків у таблиці
        однак колонки не будуть видалені
    """
    self.beginRemoveRows(QModelIndex(), 0, self.rowCount(QModelIndex()) - 1)
    self._data.clear()
    self.endRemoveRows()
```

ДОДАТОК Є

Код файлу «snmpscannerthread.py» для розподілення потоків сканування на всі ядра центрального процесора системи

```
from datetime import datetime
from PyQt5.QtCore import QObject, QThread, QMutex, pyqtSlot, pyqtSignal
from snmpclient import SNMPClient, SNMPError, DeviceInfo, get_deviceinfo

class ThreadPool(QObject):
    """ Клас являє собою пул для керування групою потоків
        на відміну від вбудованого класу QThreadPool, додано сигнал завершення
        all_threads_finished - сигнал запускається як тільки потік завершив свою роботу
    """

    all_threads_finished = pyqtSignal()

    def __init__(self, active_threads=10, parent=None):
        """ active_threads кількість одночасно виконуються потоків
        """
        super(QObject, self).__init__(parent)
        self._mutex = QMutex(QMutex.Recursive)
        self._max_active_thread_count = active_threads
        self._active_thread_count = 0
        self._thread_queue = []

    def start(self, thread: QThread):
        """ Запуск потоку на виконання, якщо на даний момент кількість
            потоків, що виконуються більше або рівно заданого в параметрі
            active_threads, наступний потік ставиться в чергу і запускається
            тільки після того, як завершиться попередній
        """
        self._mutex.lock()
        try:
            if self._active_thread_count < self._max_active_thread_count:
                self._active_thread_count += 1
                thread.finished.connect(self._thread_finished)
                thread.start()
            else:
                self._thread_queue.append(thread)
        finally:
            self._mutex.unlock()

    def stop(self):
        """ Зупинка запуску потоків, що стоять у черзі
            вже запущені потоки продовжать працювати
        """
        self._mutex.lock()
        try:
            self._thread_queue = []
        finally:
            self._mutex.unlock()

    @pyqtSlot()
    def _thread_finished(self):
        """ Слот, що обробляє подію завершення виконання потоку в пулі
        """
        self._mutex.lock()
        try:
            self._active_thread_count -= 1
            if len(self._thread_queue) > 0:
```

```
            self.start(self._thread_queue.pop())

        elif self._active_thread_count == 0:
            self.all_threads_finished.emit()
    finally:
        self._mutex.unlock()

@property
def active_threads(self) -> int:
    """ Кількість потоків, що виконуються в даний момент
    """
    return self._active_thread_count

@property
def max_active_threads_count(self) -> int:
    """ Функція поверне максимальну кількість
        одночасно виконуються потоків
    """
    return self._max_active_thread_count

class SNMPScannerThread(QThread):
    """ Потік, що збирає інформацію про хост за протоколом SNMP
        за нормального завершення запускає сигнал recived_message з інформацією
        зібраної про хост у параметрі (об'єкт DeviceInfo)
        також для передачі текстових повідомлень про хід виконання
        використовується сигнал recived_message
    """

    work_done = pyqtSignal(DeviceInfo)
    recived_message = pyqtSignal(str)

    def __init__(self, host: str, community: str, parent=None):
        """ host - ім'я хоста наприклад 192.168.1.1
            community - SNMP ком'юніті з доступом на читання
        """
        super(QThread, self).__init__(parent)

        self._host = host
        self._community = community

    def run(self):
        """ Метод, що збирає інформацію про хост, за результатами виконання
            викликає два сигнали SNMPScannerThread.work_done(DeviceInfo)
            та SNMPScannerThread.recived_message(str)
        """
        devinfo = None

        msg = self.tr('{}|info|host: {} |msg: {}').format(datetime.now().strftime('%H:%M:%S'),
                                                          self._host, self.tr('processing...'))

        self.recived_message.emit(msg)
        try:
            client = SNMPClient(self._host, community=self._community)
            devinfo = get_deviceinfo(client)

        except SNMPError as err:
            msg = self.tr('{}|error|host: {} |msg: {}').format(datetime.now().strftime('%H:%M:%S'), self._host, err)
            self.recived_message.emit(msg)
            return

        if devinfo:
            msg = self.tr('{}|info|host: {} |msg: {}').format(datetime.now().strftime('%H:%M:%S'),
```

```
                self._host, self.tr('complete'))

        self.recived_message.emit(msg)
        self.work_done.emit(devinfo)
```

ДОДАТОК Ж

Код файлу «snmpclient.py» для взаємодії з SNMP драйвером через код на вибраній мові програмування

```
import datetime
from typing import Sequence, NamedTuple
from pysnmp.entity.rfc3413.oneliner import cmdgen

class DeviceInfo(NamedTuple):
    """ Кортеж містить загальні відомості про пристрій
    """
    host: str
    name: str
    description: str
    location: str
    contact: str
    uptime: datetime.timedelta

class SNMPError(Exception):
    """ Помилка під час роботи з протоколом SNMP
        вхідні параметри це значення таких функцій, що повертаються
        як getCmd з бібліотеки pysnmp
    """

    def __init__(self, error_indication, error_status, error_index, var_binds):
        self.error_indication = error_indication
        self.error_status = error_status
        self.error_index = error_index
        self.message = ''

        if error_indication:
            self.message = error_indication
        elif error_status:
            self.message = '{} at {}'.format(error_status.prettyPrint(),
                                    error_index and var_binds[int(error_index) - 1][0] or '?')

        super().__init__(self.message)

class SNMPClient:
    """ SNMP клієнт, що працює з PySNMP """

    def __init__(self, host="localhost", port=161,
                 community="public", version="SNMPv2-MIB"):

        self.cmd_gen = cmdgen.CommandGenerator()
        self.host = host
        self.port = port
        self.community = community
        self.version = version

    def getbulk(self, non_repeaters: int, max_repetitions: int, *oid) -> Sequence:
        """
        SNMP getbulk запит
        non_repeaters: Це визначає кількість наданих змінних, які не слід повторювати.
        max_repetitions: вказує максимальну кількість ітерацій для повторюваних змінних.
        oid: список oid
        """
        error_indication, error_status, error_index, var_binds = self.cmd_gen.bulkCmd(
            cmdgen.CommunityData(self.community),
            cmdgen.UdpTransportTarget((self.host, self.port)),
```

```
            non_repeaters,
            max_repetitions,
            *oid
        )
        result = []
        if error_indication or error_status:
            raise SNMPError(error_indication, error_status, error_index, var_binds)

        for row in var_binds:
            item = {}
            for name, val in row:
                if str(val) == '':
                    item[name.prettyPrint()] = ''
                else:
                    item[name.prettyPrint()] = val.prettyPrint()
            result.append(item)
        return result

def get_deviceinfo(client: SNMPClient) -> DeviceInfo:
    """ Отримання основної інформації від пристрою, взятої з
    iso.org.dod.internet/mgmt/mib2/system
    OID_SYSTEM - '1.3.6.1.2.1.1'
    """
    data = client.getbulk(0, 10, '1.3.6.1.2.1.1')
    info_table = {}

    for elem in data:
        for key, item in elem.items():
            info_table[key] = item

    ticks = int(info_table['SNMPv2-MIB::sysUpTime.0'])
    result = DeviceInfo(
        client.host,
        info_table['SNMPv2-MIB::sysName.0'],
        info_table['SNMPv2-MIB::sysDescr.0'],
        info_table['SNMPv2-MIB::sysLocation.0'],
        info_table['SNMPv2-MIB::sysContact.0'],
        datetime.timedelta(seconds=ticks / 100))
    return result
```

ДОДАТОК Л

Код файлу «SNMP scanner.py» в якому пов’язуються і взаємодіють всі зовнішні модулі програми в один повнофункціональний додаток

```
import os
import sys
from ipaddress import IPv4Network
import netifaces
import pandas as pd
import sqlite3
from PySide2 import QtCore
from PyQt5 import QtGui
from PyQt5.QtCore import QCoreApplication, pyqtSlot, QModelIndex
from PyQt5.QtWidgets import (QMessageBox, QMainWindow, QApplication,
                             QComboBox, QFileDialog)
from qt_material import apply_stylesheet
from scanresults_tablemodel import ScanResultsTableModel
from snmpclient import DeviceInfo
from snmpscannerthread import ThreadPool, SNMPScannerThread
from ui_snmp import Ui_SNMPScannerWindow

GATEWAY = netifaces.gateways()['default'][netifaces.AF_INET][0]
ICONS_PATH = 'icons'
ABOUT_MESSAGE = QCoreApplication.translate('About', """
Автор: студент Державного Університету Телекомунікацій Францев Антон Владимирович
Версія 0.0.1
Рік 2022
Іконки взяті з відкритого ресурсу Google Fonts
""")

class SNMPScannerWindow(QMainWindow):
    """ Клас головного вікна програми
    """

    def __init__(self):
        """ Конструктор класу
        """
        super(QMainWindow, self).__init__()

        self._thread_pool = ThreadPool()
        self.ui = Ui_SNMPScannerWindow()
        self.ui.setupUi(self)
        self.init_ui()
        self._init_signals_and_slots()

    def init_ui(self):
        """ ініціалізація інтерфейсу
        """
        self.setWindowIcon(QtGui.QIcon(os.path.join(ICONS_PATH, 'appicon.png')))
        self.ui.scanButton.setIcon(QtGui.QIcon(os.path.join(ICONS_PATH, 'start.png')))
        self.ui.stopButton.setIcon(QtGui.QIcon(os.path.join(ICONS_PATH, 'stop.png')))
        self.ui.import_dbButton.setIcon(QtGui.QIcon(os.path.join(ICONS_PATH, 'import.png')))
        self.ui.export_dbButton.setIcon(QtGui.QIcon(os.path.join(ICONS_PATH, 'export.png')))
        self.ui.actionAbout.setIcon(QtGui.QIcon(os.path.join(ICONS_PATH, 'about.png')))
        self.ui.scanResultsTableView.setModel(ScanResultsTableModel())
        self.ui.scanResultsTableView.resizeColumnsToContents()
        self.ui.hostsCBox.setCurrentText('{}'.format(GATEWAY))
        self.ui.communityCBox.setCurrentText('public')

    def _init_signals_and_slots(self):
```

```
        """ початкове підключення сигналів
            до слотів
        """
        self.ui.scanButton.clicked.connect(self.start_scan)
        self.ui.stopButton.clicked.connect(self.stop_scan)
        self.ui.stopButton.setEnabled(False)
        self.ui.import_dbButton.clicked.connect(self.import_db)
        self.ui.export_dbButton.clicked.connect(self.export_db)
        self.ui.export_dbButton.setEnabled(False)
        self.ui.actionAbout.triggered.connect(self.show_about)
        self._thread_pool.all_threads_finished.connect(self._scan_completeted)

    def scan_parameters_is_correct(self):
        """ Перевірка коректності даних користувача
            введених ним у вікні програми, у разі неправильних даних
            буде виведено повідомлення
        """
        network = None
        try:
            network = IPv4Network(self.ui.hostsCBox.currentText())
        except ValueError:
            QMessageBox.critical(self, self.tr('Error'),
                                 self.tr('Incorrect network\nfor example:\n\n192.168.1.0/27\n10.7.207.0/255.255.255.0'))
            return False

        if not self.ui.communityCBox.currentText():
            QMessageBox.critical(self, self.tr('Error'), self.tr('Community is empty'))
            return False
        return True

    def append_current_params_to_completions(self):
        """ Додавання поточних введених користувачем параметрів
            у списки автозавершення
        """

        def append_to_combobox(cbox: QComboBox):
            current = cbox.currentText()
            completions = set(cbox.itemText(i) for i in range(cbox.count()))
            completions.add(current)
            cbox.clear()
            cbox.addItems(completions)
            cbox.setCurrentIndex(cbox.findText(current))

        append_to_combobox(self.ui.hostsCBox)
        append_to_combobox(self.ui.communityCBox)

    @pyqtSlot(DeviceInfo)
    def add_record_to_results_table(self, devinfo: DeviceInfo):
        """ Додавання інформації про пристрій (перший параметр)
            у загальну таблицю
        """
        if not devinfo:
            return

        self.ui.scanResultsTableView.model().add_row(devinfo)
        self.ui.scanResultsTableView.resizeColumnsToContents()

    @pyqtSlot(str)
    def add_record_to_message_list(self, message: str):
        """ Додавання повідомлення про аномальне завершення до списку
        """
        if not message:
```

```
            return

        self.ui.messageListWidget.addItem(message)
        self.ui.messageListWidget.scrollToBottom()

    @pyqtSlot()
    def start_scan(self):
        """ Слот реагує на натискання кнопки початку сканування
            перевіряє початкові параметри та запускає процес дослідження
            мережі
        """
        if not self.scan_parameters_is_correct():
            return
        self.append_current_params_to_completions()

        network = IPv4Network(self.ui.hostsCBox.currentText())
        community = self.ui.communityCBox.currentText()
        # очистка даних
        self.ui.scanResultsTableView.model().remove_all_rows()
        self.ui.messageListWidget.clear()

        self.add_record_to_message_list(
            self.tr('{}::{} scanning...'.format(network, community)))

        self.ui.scanButton.setEnabled(False)
        self.ui.stopButton.setEnabled(True)

        for host in network:
            scanner = SNMPScannerThread(host.exploded, community, self)
            scanner.recived_message.connect(self.add_record_to_message_list)
            scanner.work_done.connect(self.add_record_to_results_table)
            self._thread_pool.start(scanner)

    @pyqtSlot()
    def stop_scan(self):
        """ Слот реагує на кнопку зупинки сканирования
        """
        self.ui.stopButton.setEnabled(False)
        self._thread_pool.stop()
        self.add_record_to_message_list(self.tr(
            'scanning finished, wait, please  ...'))

    @pyqtSlot()
    def import_db(self):
        """ Слот реагує на кнопку імпортування бази данних
        """
        self._thread_pool.stop()
        self.ui.scanResultsTableView.model().remove_all_rows()
        self.add_record_to_message_list(self.tr('importing...'))
        filename, _ = QFileDialog.getOpenFileName(caption='Select file')
        db_connect = sqlite3.connect(filename)
        cur = db_connect.cursor()
        cur.execute("SELECT * FROM pysnmp_table")
        content = cur.fetchall()
        db_connect.commit()
        cur.close()
        db_connect.close()

        for i in content:
            self.ui.scanResultsTableView.model().add_row(DeviceInfo(i[0], i[1], i[2], i[3], i[4], i[5]))
            self.ui.scanResultsTableView.resizeColumnsToContents()
```

```
        self.add_record_to_message_list(self.tr('done!'))

    @pyqtSlot()
    def export_db(self):
        """ Слот реагує на кнопку експортування бази данних
        """
        self._thread_pool.stop()
        self.add_record_to_message_list(self.tr('exporting...'))

        filename, _ = QFileDialog.getSaveFileName(self,
                                        self.tr('Save results'), 'snmp.db')

        if filename:
            model = self.ui.scanResultsTableView.model()
            table_column_count = int(model.columnCount(self.ui.scanResultsTableView))
            table_row_count = int(model.rowCount(self.ui.scanResultsTableView))

            d = {}
            for i in range(table_column_count):
                l = []
                for j in range(table_row_count):
                    it = self.ui.scanResultsTableView.model().index(j, i)
                    l.append(str(model.data(it)) if str(model.data(it)) is not None else "")

                h_item = model.headerData(i, orientation=QtCore.Qt.Horizontal)
                n_column = str(i) if h_item is None else h_item
                d[n_column] = l

            df = pd.DataFrame(data=d)
            engine = sqlite3.connect(filename)
            df.to_sql('pysnmp_table', con=engine, index=False, if_exists='replace')

        self.add_record_to_message_list(self.tr('done!'))
        self.ui.export_dbButton.setEnabled(False)

    @pyqtSlot()
    def _scan_completeted(self):
        """ Слот який викликається при закінченні всіх робочих потоків
        """
        self._thread_pool = ThreadPool()
        self._thread_pool.all_threads_finished.connect(self._scan_completeted)

        self.add_record_to_message_list(self.tr('scanning completed'))
        self.ui.scanButton.setEnabled(True)
        if self.ui.scanResultsTableView.model().rowCount(QModelIndex()) != 0:
            self.ui.export_dbButton.setEnabled(True)
        self.ui.stopButton.setEnabled(False)

    @pyqtSlot()
    def show_about(self):
        """ Відображенняя вікна із відомостями про програму
        """
        QMessageBox.about(self, 'SNMP Scanner', ABOUT_MESSAGE)


def main():
    try:
        print('Application started')
        app = QApplication(sys.argv)
        apply_stylesheet(app, theme=r'themes\theme.xml')
        main_window = SNMPScannerWindow()
        main_window.show()
```

```
        retcode = app.exec_()

    except Exception as error:
        print('catched excetion: {}'.format(str(type(error))))
        raise

    sys.exit(retcode)

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

ДОДАТОК М

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ

НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ ОБЛІКУ КОМУТАЦІЙНОГО ОБЛАДНАННЯ З ВИКОРИСТАННЯМ МОВИ ПРОГРАМУВАННЯ PYTHON

Виконав студент 4 курсу
групи ПД-42
Францев Антон Володимирович
Керівник роботи
Старший викладач Гаманюк Ігор Михайлович

Київ – 2022

АНАЛОГИ

Назва продукту	Переваги	Недоліки
Solarwinds SNMP Scanner	1. Автоматично вимальовує топологію мережі 2. Приведе вас до першопричини події за допомогою аналізу даних	1. Висока ціна на корпоративне рішення унеможливлює використання програми малими компаніями 2. Застарілий, складний інтерфейс, потребує багато часу на навчання.
Adrem Software	1. Підтримує всі версії SNMP 2. Сканує як всю підмережу, так і лише машину 3. Повністю безкоштовна.	1. На роботу потребує багато часу. 2. Не зручний інтерфейс, складно парсима строка виводу інформації, та кореляції подій.
SoftinventiveLab	1. 1) Створює детальний підсумок вашої мережі в автоматизованому режимі. 2. 2) Створює вичерпні звіти з детальною інформацією про структуру мережі та її компоненти 3. 3) Використання конкурентних ліцензій що дозволяє масштабувати програму під кожну компанію окремо	1. Незважаючи на масштабоване ліцензування ціна на додаток зависока. 2. Відсутність портів на різні системи, можливість запуска тільки ОС сім'ї Windows.

2

МЕТА, ОБ’ЄКТ ТА ПРЕДМЕТ ДОСЛІДЖЕННЯ

Мета роботи – спрощення обліку та моніторингу комутаційного обладнання.

Об’єкт дослідження – процес обліку та моніторингу комутаційного обладнання.

Предмет дослідження – програмне забезпечення обліку та моніторингу комутаційного обладнання.

3

ТЕХНІЧНІ ЗАВДАННЯ

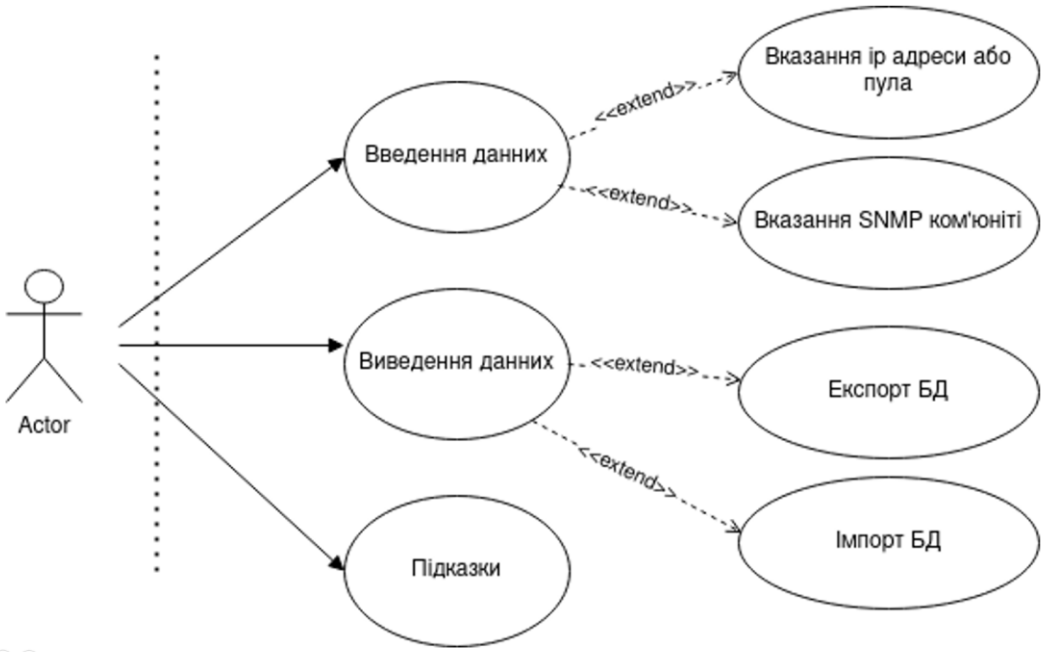
1. *Автоматичний пошук пристроїв у підмережі;*
2. *Можливість роботи із різними SNMP ком'юніті;*
3. *Легке масштабування програми;*
4. *Сумісність із базою даних для обліку і зберігання даних;*
5. *Отримання таких даних як Uptime та локацію пристрою;*
6. *Вміння сортувати та візуалізувати отримані дані;*

4

ПРОГРАМНІ ЗАСОБИ ТА ІНСТРУМЕНТИ ВИКОРИСТАНІ ДЛЯ РЕАЛІЗАЦІЇ

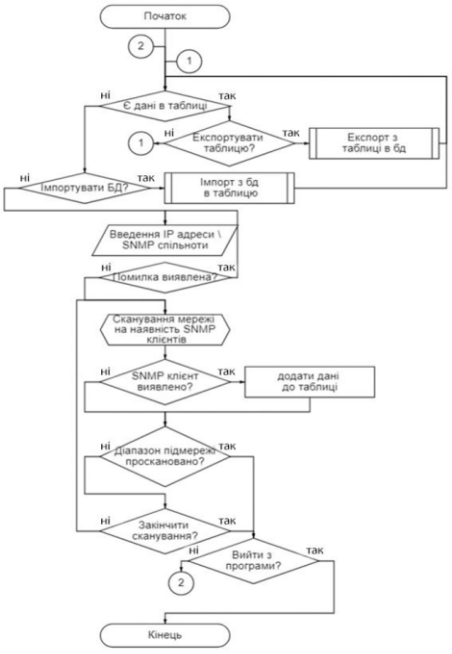
5

Діаграма прецедентів



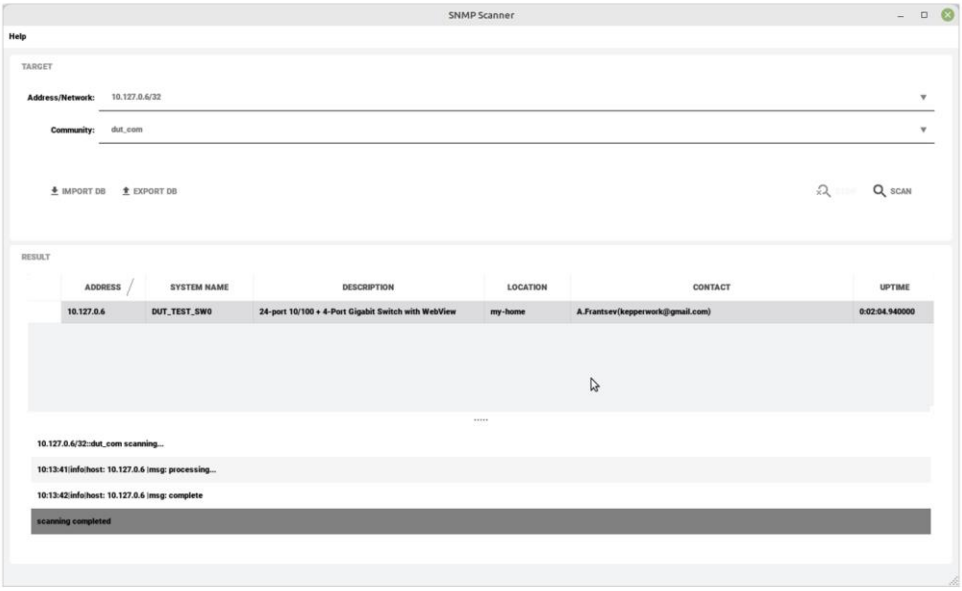
6

Блок схема роботи програми



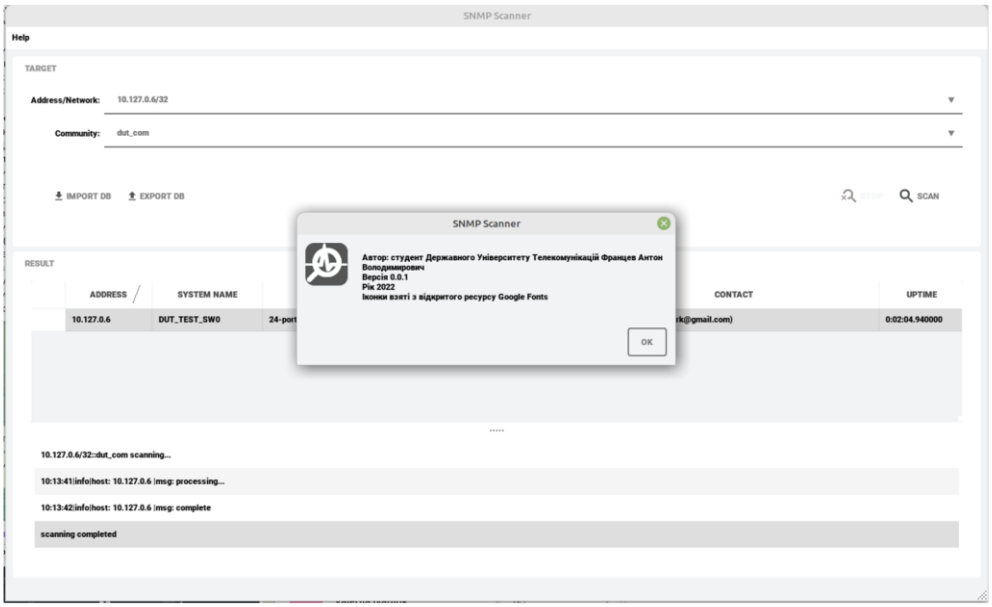
7

Екранні форми(*Зчитування даних із комутатора*)



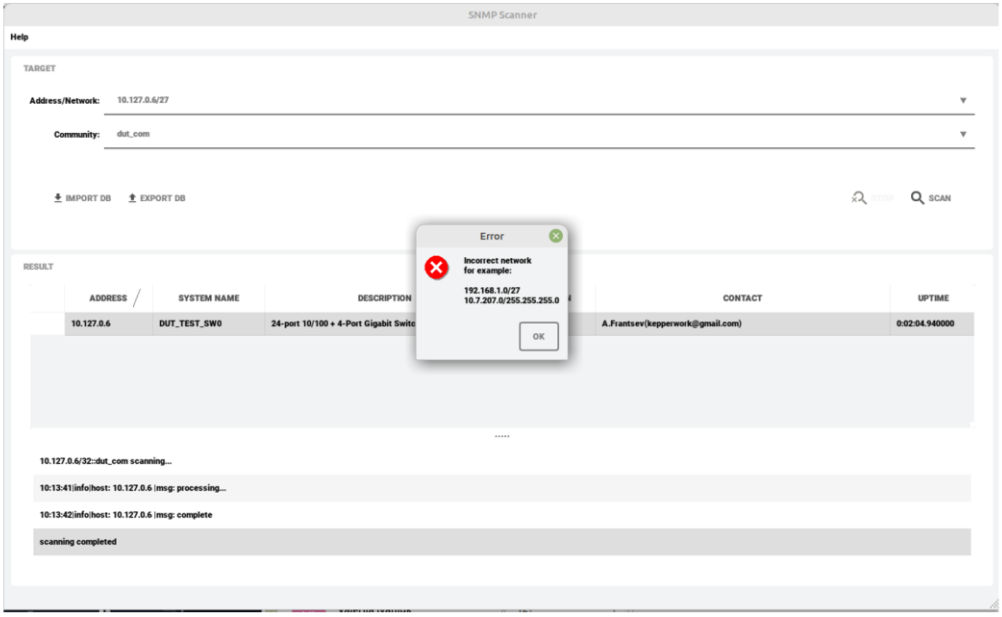
8

Екранні форми(*Інформація про розробника програми*)



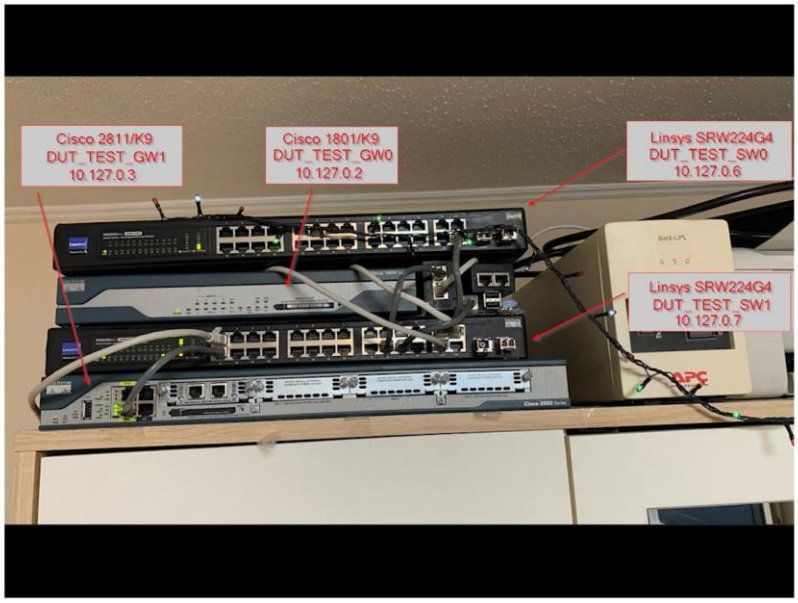
9

Екранні форми(*Виведення помилки про неправильно введені дані*)



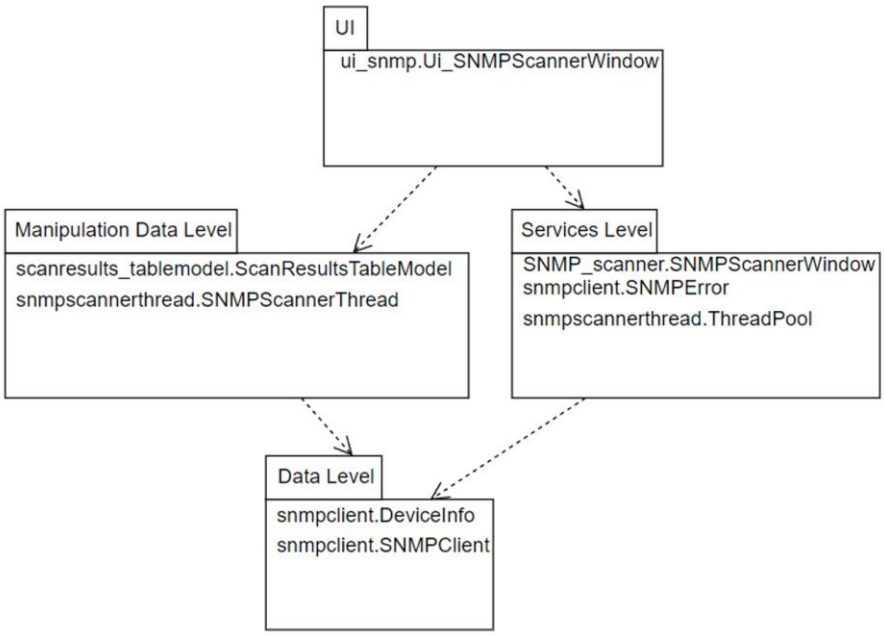
10

Моделювання процесу сканування мережі на наявність SNMP клієнтів



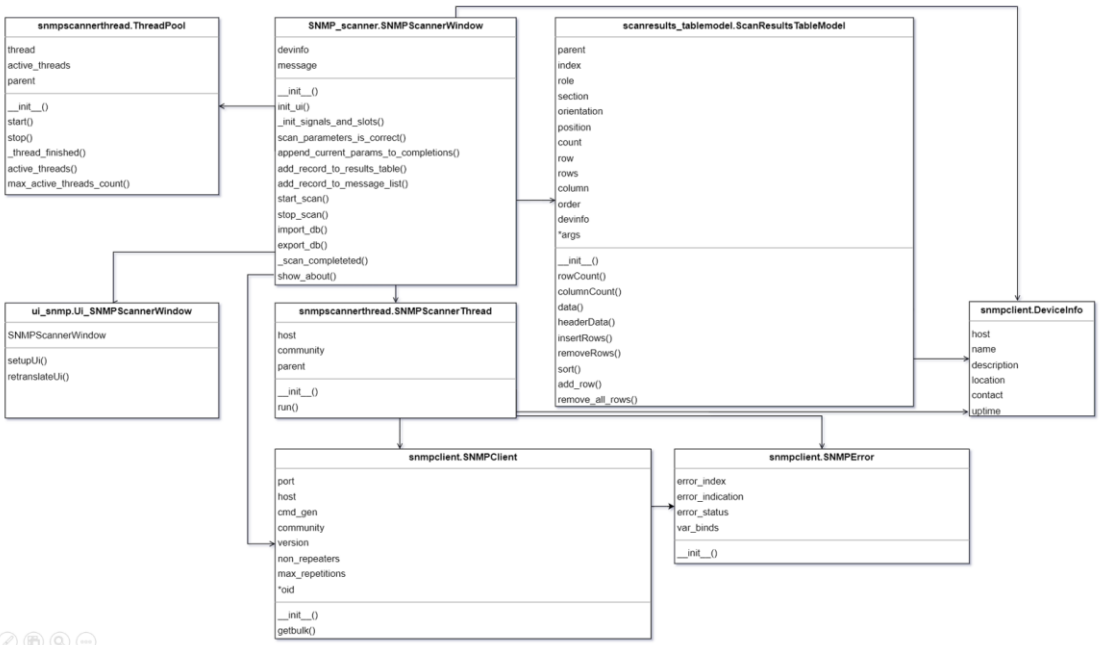
11

Діаграма пакетів



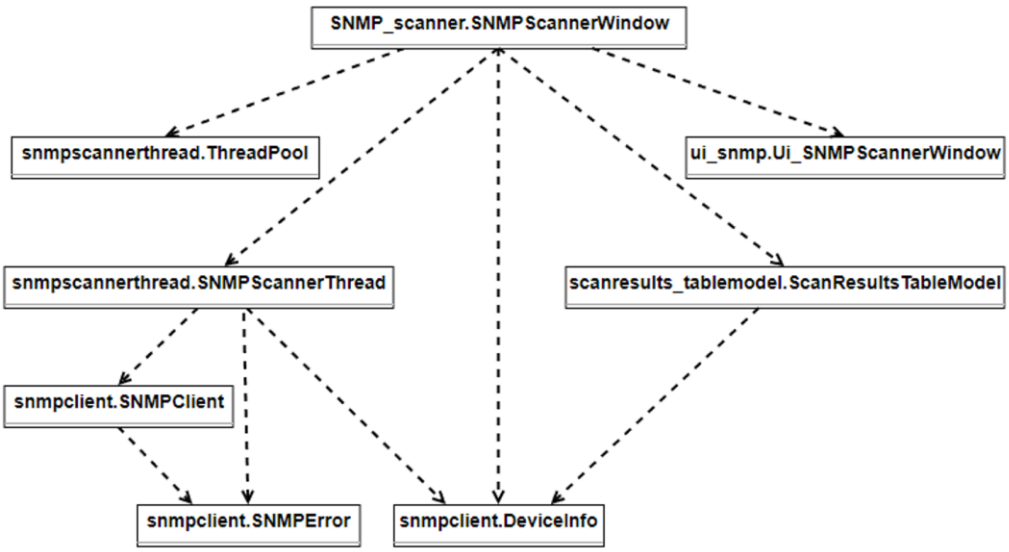
12

Діаграма класів.



13

Відпрацювання коду



14

АПРОБАЦІЯ РЕЗУЛЬТАТІВ ДОСЛІДЖЕННЯ

1. Францев А.В. Засоби розробки програмного забезпечення на мові програмування Python : Матеріали всеукраїнської науково-технічної конференції «Застосування програмного забезпечення в ІКТ». Збірник тез.\-20.04.2022, ДУТ, м.Київ – К.: ДУТ, 2022. – С. 28 – 29.
2. Францев А.В. Геймифікація з використанням мови програмування Python : Матеріали всеукраїнської науково-технічної конференції «Застосування програмного забезпечення в ІКТ». Збірник тез.\-20.04.2022, ДУТ, м.Київ – К.: ДУТ, 2022. – С. 98 – 99.

15

ВИСНОВКИ

1. Проведено аналіз існуючих програм, утиліт, та повних систем для контролю та обліку мережі. Було вирішено розробити програму для моніторингу та обліку мережевих пристроїв виходячи із недоліків існуючих систем.
2. Проведено аналіз інструментів, програмних засобів, і існуючих протоколів взаємодії із комутаційним обладнанням, таких як “Solarwinds SNMP Scanner”, “Adrem Software”, “SoftinventiveLab”, протоколи взаємодії “CDP”, “LLDP”, “SNMP”.
3. Описано програмні засоби та інструменти, котрі було застосовано для розробки програмного забезпечення: IDE PyCharm, DB SQLite, SNMP.
5. Змодельовано процес сканування мережі на наявність SNMP клієнтів.
6. Зроблена система для моніторингу, автоматичного збору інформації та обліку комутаційного обладнання що підійде для автоматизації бізнес процесів, та пришвидшення вирішення проблемних ситуацій у компанії пов'язаних із недоступністю обладнання.

16

ДЯКУЮ ЗА УВАГУ!