Полтавський університет економіки і торгівлі

Навчально-науковий інститут денної освіти

Форма навчання денна

Кафедра комп’ютерних наук та інформаційних технологій

**Кваліфікаційна робота**

**на тему:**

**РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ З ТЕМИ**

**«ОСНОВИ МОВИ ПРОГРАМУВАННЯ PYTHON»**

**Допускається до захисту**

«\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_ 2023 р.

Керівник\_\_\_\_\_\_\_\_\_ Чілікіна Т.В.

 (підпис)

зі спеціальності 122 «Комп’ютерні науки»

освітня програма «Комп’ютерні науки»

ступеня магістра

Виконавець роботи **Пипко Р.В.**

 **\_\_\_\_\_\_\_\_\_\_«\_\_»\_\_\_\_\_\_2023р.**

 **(підпис)**

Науковий керівник **к.ф.-м.н., доц. Чілікіна Т.В.**

 **\_\_\_\_\_\_\_\_\_\_«\_\_»\_\_\_\_\_\_2023р.**

 **(підпис)**

Рецензент

**Полтава – 2023**

**ЗАЯВА МАГІСТРА**

Завідувачу кафедри КНІТ

Олені ОЛЬХОВСЬКІЙ

 здобувача вищої освіти 2 курсу, групи КН м 11

 освітньо-професійної програми «Комп’ютерні науки»

 зі спеціальності 122 «Комп’ютерні науки»

 Навчально-наукового інституту денної освіти Пипко Руслана Вадимовича

 моб. тел. 099-028-66-80

**ЗАЯВА**

Прошу дозволити мені виконувати кваліфікаційну роботу на тему „Розробка елементів програмного забезпечення з теми «Основи мови програмування Python»”.

Здобувач вищої освіти \_\_\_\_\_\_\_\_\_\_\_\_\_\_Пипко Руслан Вадимович

Науковий керівник \_\_\_\_\_\_\_\_\_\_\_\_\_\_Чілікіна Тетяна Василівна

«\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_20\_\_ р.

 ПОЛТАВСЬКИЙ УНІВЕРСИТЕТ ЕКОНОМІКИ І ТОРГІВЛІ

**Затверджую**

Керівник \_\_\_\_\_\_\_\_\_ Чілікіна Т.В.

 *(підпис)*

 «\_\_\_\_» 2023 р.

**ЗАВДАННЯ ТА КАЛЕНДАРНИЙ ГРАФІК**

**ВИКОНАННЯ КВАЛІФІКАЦІЙНОЇ РОБОТИ**

на тему „Розробка елементів програмного забезпечення з теми «Основи мови програмування Python»”.

зі спеціальності 122 Комп’ютерні науки»

Освітня програма «Комп’ютерні науки»

Ступеня магістр

Прізвище, ім’я, по батькові Пипко Руслан Вадимович

Затверджена наказом ректора No \_\_\_\_-Н від « » \_\_\_\_\_\_ 202\_ р.

Термін подання студентом роботи «\_\_ » \_\_\_\_\_ 202\_ р.

Вихідні дані до кваліфікаційно роботи: публікації з теми, навчальні тренажери в дистанційних курсах з комп’ютерних наук.

Зміст пояснювальної записки (ВСТУП. РОЗДІЛ 1. ПОСТАНОВКА ЗАДАЧІ.РОЗДІЛ 2. ІНФОРМАЦІЙНИЙ ОГЛЯД. 2.1 Проблематика створення електронних засобів навчання. 2.2 Огляд програмних засобів для створення програмного забезпечення. 2.3 Необхідність та актуальність теми роботи. РОЗДІЛ 3.ТЕОРЕТИЧНА ЧАСТИНА. 3.1 Алгоритмізація процесу розробки програмного забезпечення. 3.2 Розробка блок-схеми з описом роботи тренажер. 3.3 Обґрунтування вибору програмного забезпечення. РОЗДІЛ 4. ПРАКТИЧНА ЧАСТИНА. 4.1 Опис програми. 4.2 Тестування розробленого програмного продукту. 4.3 Користувацька інструкція до програмного продукту. ВИСНОВКИ. СПИСОК ЛІТЕРАТУРИ. ДОДАТОК А. Код програми)

Перелік графічного матеріалу (*Блок-схеми, таблиці, графіки, рисунки.)*

Консультанти розділів кваліфікаційної роботи

|  |  |  |
| --- | --- | --- |
| Розділ | ПІП консультанта | Підпис, дата |
| Завдання видав | Завдання прийняв |
| Постанова задачі | Чілікіна Т.В. |  |  |
| Інформаційний огляд | Чілікіна Т.В. |  |  |
| Теоретична частина | Чілікіна Т.В. |  |  |
| Практична частина | Чілікіна Т.В. |  |  |

Календарний графік виконання кваліфікаційної роботи

|  |  |  |
| --- | --- | --- |
| Зміст роботи | Термін виконання | Фактичне виконання |
| 1.Вступ |  |  |
| 2.Вивчення методичних рекомендацій та стандартів та звіт керівнику  |  |  |
| 3.Постановка задачі |  |  |
| 4.Інформаційний огляд джерел бібліотек та інтернету |  |  |
| 5.Теоретична частина |  |  |
| 6.Практична частина  |  |  |
| 7.Закінчення оформлення |  |  |
| 8.Доповідь студента на кафедрі |  |  |
| 9.Доробка (за необхідністю), рецезування  |  |  |

Дата видачі завдання «\_\_\_\_»\_\_\_\_\_\_\_\_\_ 2023 р.

Здобувач вищої освіти \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

 *(підпис студента)*

Науковий керівник Чілікіна Т.В.

Кваліфікаційна робота оцінена на \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_
 (балів, оцінка за національною шкалою, оцінка за ECTS)

Протокол засідання ЕК No \_\_\_\_ від «\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_ 202\_ р.

Секретар ЕК\_\_\_\_\_\_\_\_\_\_\_\_\_\_ (підпис) \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_(ініціал та прізвище)

**Затверджую**  **Погоджено**

Зав. кафедрою\_\_\_\_\_\_\_\_\_\_\_ Науковий керівник\_\_\_\_\_\_\_\_\_\_\_

к.ф.-м.н. Олена Ольховська к.ф.-м.н. Чілікіна Т.В.

«\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2023 р. «\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2023 р.

**План**

кваліфікаційної роботи ступеня магістр

зі спеціальності 122 Комп’ютерні науки

освітня програма 122 Комп’ютерні науки

програмування Python” »

Пипко Руслан Вадимович

на тему «Розробка програмного забезпечення з теми “Основи мови програмування Python»

ВСТУП

РОЗДІЛ 1. ПОСТАНОВКА ЗАДАЧІ

РОЗДІЛ 2. ІНФОРМАЦІЙНИЙ ОГЛЯД

2.1 Проблематика створення електронних засобів навчання

2.2 Огляд програмних засобів для створення програмного забезпечення

2.3 Необхідність та актуальність теми роботи.

РОЗДІЛ 3. ТЕОРЕТИЧНА ЧАСТИНА

 3.1 Алгоритмізація процесу розробки програмного забезпечення

 3.2 Розробка блок-схеми з описом роботи тренажеру

 3.3 Обґрунтування вибору програмного забезпечення

РОЗДІЛ 4. ПРАКТИЧНА ЧАСТИНА

 4.1 Опис програми

 4.2 Тестування розробленого програмного продукту
 4.3 Користувацька інструкція до програмного продукту

ВИСНОВКИ

СПИСОК ЛІТЕРАТУРИ
ДОДАТОК А. Код програми

Здобувач вищої освіти \_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

 *(підпис)*

«\_\_\_\_» \_\_\_\_\_\_\_\_\_\_ 2023 р.

**АНОТАЦІЯ**

**Записка:** 55 ст, 4 розділи, додаток А.

*Ключові слова:* ТРЕНАЖЕР, МОВА ПРОГРАМУВАННЯ PYTHON, АВТОМАТИЧНА ГЕНЕРАЦІЯ ТЕСТІВ, МОДУЛЬ PYQ5, PYCHARM

***Об’єкт розробки*** – блок-схема, алгоритмізація та розробка елементів програми тренажера.

***Мета роботи*** – розробка програмного забезпечення з теми: «Розробка програмного забезпечення з теми “Основи мови програмування Python” »

***Методи дослідження*** – емпіричні, теоретичні, комплексні.

***Результати –*** створення тренажеру.

Розглянуто головні аспекти та загальні етапи створення програмного забезпечення для навчання мови програмування за темою «Основи мови програмування Python». Проведений огляд теоретичного матеріалу з теми: «Основи мови програмування Python», розкрита проблематика щодо створення електронного засобу навчання, виокремлено переваги та недоліки щодо мови програмування з теми, також розроблено алгоритм елементів та блок-схему програмного забезпечення навчального тренажера з теми: «Основи мови програмування Python». Програмно реалізовано елементи тренажеру.

**ЗМІСТ**

[ВСТУП 9](#_Toc153638150)

[РОЗДІЛ 1. ПОСТАНОВКА ЗАДАЧІ 11](#_Toc153638151)

[РОЗДІЛ 2. ІНФОРМАЦІЙНИЙ ОГЛЯД 12](#_Toc153638152)

[2.1 Проблематика створення електронних засобів навчання 12](#_Toc153638153)

[2.2 Огляд програмних засобів для створення програмного забезпечення 15](#_Toc153638154)

[2.3 Необхідність та актуальність теми роботи 20](#_Toc153638155)

[РОЗДІЛ 3. ТЕОРЕТИЧНА ЧАСТИНА 23](#_Toc153638156)

[3.1 Алгоритмізація процесу розробки програмного забезпечення 23](#_Toc153638157)

[3.2 Розробка блок-схеми з описом роботи тренажеру 25](#_Toc153638158)

[3.3 Обґрунтування вибору програмного забезпечення 27](#_Toc153638159)

[РОЗДІЛ 4. ПРАКТИЧНА ЧАСТИНА 30](#_Toc153638160)

[4.1 Опис процесу створення програмного продукту 30](#_Toc153638161)

[4.2 Тестування розробленого програмного продукту 38](#_Toc153638162)

[4.3 Користувацька інструкція до програмного продукту 46](#_Toc153638163)

[ВИСНОВКИ 51](#_Toc153638164)

[СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 52](#_Toc153638165)

[ДОДАТОК А. Код програми 53](#_Toc153638166)

|  |  |
| --- | --- |
|  | КН М-11 КП |
|  | П.І.Б | Підпис | Дата |
| Виконав | Пипко Р.В. |  |  | «Розробка програмного забезпечення з теми “Основи мови програмування Python” » | Стадія | Аркуш | Аркушів |
| Перевірив | Чілікіна Т.В. |  |  | КП |  | 2 |
| Н. контр. | Чілікіна Т.В. |  |  | ПУЕТКафедра КНІТ |
| Зав. каф. | Ольховська О.В. |  |  |

**ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ**

|  |  |
| --- | --- |
| **Умовні позначення, символи, скорочення, терміни** | **Пояснення умовних позначень, символів, скорочень** |
| Python | інтерпретована об'єктно-орієнтована мова програмування |
| PyCharm | інтегроване середовище розробки для мови програмування Python |
| ЕЗН | Електронний засіб навчання |
| ПЗ | Програмне забезпечення |
| ІDE | Інтегроване середовище розробки |
| Фреймворк  | Це програмне середовище, котре покращує та спрощує процес стврення ПЗ |
| Version Control Systems | Системи керування версіями |
| PEP 8 | Python Enhancement Proposal |

**ВСТУП**

***Актуальність*** даної роботи полягає у тому, що щорічно кількість студентів, котрі займаються вивченням комп’ютерних технологій, зокрема мов програмування зростає. Адже світ не стоїть на місці, він з кожним днем змінюється, з’являється щось нове, щось цікаве, незвичне, і цим обумовлена цікавість студентів до сфери інформаційних технологій, оскільки вона в першу чергу відображає ці зміни. Попри стрімкий розвиток технологій останніми роками суспільство стикається з рядом глобальних проблем, як от пандемія коронавірусу. Проте, і ця проблема в свою чергу спровокувала діджиталізацію багатьох процесів, таких як, дистанційна освіта. З початком пандемії всі студенти стали навчатись дистанційно, що суттєво ускладнило перевірку знань студентів. Оскільки, Python та різноманітні його компілятори є безкоштовними, студенти мають змогу вивчати його як самостійно, так в межах курсів в навчальних закладах. Відповідно, для самостійної перевірки засвоєного матеріалу студентам стане в нагоді тренажер з теорії основ цієї мови з автоматичною генерацією та перевіркою завдань. Окрім цього, впровадження такого тренажеру в навчальну систему студента дозволить підвищити якість та ефективність засвоєння матеріалу.

***Метою роботи*** є створення десктопної програми - тренажеру з автоматичною генерацією та перевіркою тестів з теми “Основи мови програмування Python” - мовою програмування Python.

Досягнення мети передбачає виконання наступних ***завдань***:

1. Аналіз існуючих тренажерів для перевірки засвоєного матеріалу.

2. Аналіз посібників та інших навчальних джерел для вивчення основ мови програмування.

3. Розробка концепції тренажеру.

4. Розробка десктопної програми-тренажеру з автоматичної генерації та перевірки тестових завдань.

Тренажер для вивчення основ мови програмування Python буде написаний також мовою Python з використанням модуля PYQ5. Даний модуль дає змогу розробити простий та зрозумілий для користувача інтерфейс. Таким чином, студенти зможуть перевірити отримані знання та рівень їх засвоєння.

***Завданням тренажеру*** є допомогти студентам визначити свій рівень знань та якість засвоєння матеріалу. Завдаки простому та зрозумілому інтерфесу, користувач зможе легко використовувати цей тренажер як помічник в вивченні мови програмування Pyhton.

***Методи дослідження:*** емпіричні, теоретичні, комплексні (узагальнення, порівняння, групування, систематизації).

***Об’єкт дослідження***: електронні засоби навчання.

***Предмет дослідження:*** програмне забезпечення з теми «Основи мови програмування Python».

***Інформаційна база:*** Дослідження становлять монографії, наукові статті, підручники, навчальні посібники, наукова література, аналітичні огляди, статистичні дані, інформація, яка отримана з мережі Internet з проблематики теми роботи.

**РОЗДІЛ 1. ПОСТАНОВКА ЗАДАЧІ**

1. **Опис алгоритму;**

На даному етапі розроблено детальний опис послідовності кроків, які потрібно виконати для того, щоб ми змогли створити працюючий тренажер в подальшому. Тобто, в описі алгоритму буде представлена покрокова інформація про те, як створити тренажер з теми «Основи мови програмування Python». Алгоритм буде містити всі необхідні дії та кроки, які потрібно виконати, починаючи з початкових даних і закінчуючи отриманням кінцевого результату. На цьому етапі, опис алгоритму буде представлений у вигляді блок-схеми у текстовій формі з використанням графіків та рисунків.

1. **Перенесення алгоритму у вигляд блок-схеми;**

Дане завдання полягає в перенесенні розробленого алгоритму у вигляд блок-схеми, що є одним з способів графічного представлення послідовності дій алгоритму за допомогою блоків, які відображають кроки алгоритму, рішення та зв'язки між ними. Блок-схема надає візуальне подання алгоритму, а це в свою чергу, полегшує розуміння його логіки та структури. У блок-схемі будуть використовуватись певні символи-блоки для представлення різних елементів алгоритму, такі як: процеси/дії, рішення/умови, цикли, та з’єднувачі.

1. **Розробка елементів тренажеру.**

Завдання полягає в створенні елементів навчального інструменту, тобто тренажера з автоматичною генерацією тестових завдань, котрий допоможе студентам самостійно перевіряти свої знання та вдосконалити навички з програмування на мові Python. Елементи тренажера: тести із теорії мови програмування Python та реакція на відповіді з поясненнями (тренажер може надавати пояснення у разі вибору неправильного варіанту для кожного завдання). Також буде розроблено калькулятор для обчислення результатів тестування.

**РОЗДІЛ 2. ІНФОРМАЦІЙНИЙ ОГЛЯД**

## 2.1 Проблематика створення електронних засобів навчання

Обговорення проблем створення навчальних засобів в електронному вигляді варто починати із визначення фундаментальних понять. До них відносяться такі, як: електронний засіб навчання, проблематика створення електронного засобу навчання.

**Електронний засіб навчання (далі ЕЗН)** - це комп'ютерна програма, веб-сайт або інше цифрове середовище, створене з метою навчання, освоєння конкретного предмету або розвитку навичок користувача. Він використовується для підтримки навчального процесу, забезпечуючи доступ до навчального матеріалу, інтерактивних завдань, вправ, тестів, інших навчальних засобів.

Електронні засоби навчання можуть бути використані як самостійні навчальні ресурси або в доповнення до традиційних методів навчання, таких як лекції та підручники. Вони можуть бути розроблені для типів і рівнів навчання - від шкільного освіти до професійного чи галузевого навчання.

Електронні засоби навчання можуть містити різноманітний навчальний контент, включаючи текстові матеріали, ілюстрації, анімацію, відео, аудіозаписи та інші мультимедійні елементи. Розроблюваний електронний засіб навчання буде містити тренажер з тестами. Вони часто мають інтерактивні функції, такі як вправи, тести, завдання для самоперевірки та можливості спілкування з викладачем або іншими студентами.

Електронні засоби навчання можуть бути доступні через веб-браузер на комп'ютері, планшеті або смартфоні, або ж вони можуть бути встановлені як окрема програма на комп'ютері або мобільному пристрої. Вони дозволяють студентам навчатися в зручний для них час та отримувати негайний фідбек та відстежувати свій прогрес.

**Проблематика створення ЕЗН** - це сукупність проблем чи складнощів, з якими можуть стикатися розробник при його створенні. Це можуть бути технічні, організаційні, концептуальні або інші труднощі, які необхідно вирішити або подолати для успішної реалізації.

Проблематика створення може включати наступні аспекти:

**1.** **Технічні проблеми;**

Це можуть бути складнощі, пов'язані з розробкою технологій, нестабільністю системи, несумісністю з іншими компонентами, проблемами безпеки та інші.

**2.** **Організаційні проблеми;**

Такі проблеми виникають у зв'язку з організацією робочих процесів, розподілом ресурсів.

**3.** **Концептуальні проблеми;**

Це проблеми, пов'язані з розробкою концепції, визначенням цілей та вимог, проектуванням архітектури, управлінням змінами та іншими аспектами, що визначають основи проекту.

**4.** **Людський фактор;**

Це можуть бути проблеми, пов'язані з недостатньою кваліфікацією розробника, відсутністю співпраці або комунікації, недостатнім управлінням ресурсами.

**Створення електронного засобу навчання** має кілька важливих переваг і може відповідати наступним цілям і потребам:

· **Доступність та гнучкість:**

Електронні засоби навчання дозволяють студентам отримувати доступ до навчального матеріалу у будь-який зручний для них час та місці. Це особливо важливо для дистанційного навчання, курсів самостійного навчання або навчання на відстані. Вони також надають можливість індивідуалізації навчання, дозволяючи студентам працювати у власному темпі та пристосовувати навчальний процес до своїх потреб.

· **Інтерактивність та зацікавленість;**

Електронні засоби навчання можуть бути розроблені з використанням інтерактивних елементів, які стимулюють зацікавленість та взаємодію студентів. Вони можуть включати відеоуроки, інтерактивні завдання, тести, та інші форми взаємодії, що допомагають залучити студентів та покращити їхню мотивацію.

· **Масштабованість;**

Електронні засоби навчання можуть бути ефективними для масштабування навчального процесу. Вони дозволяють досягти великої аудиторії студентів одночасно, без обмежень фізичного простору або кількості викладачів. Це особливо важливо для масових відкритих онлайн-курсів та інших форм дистанційного навчання.

· **Актуалізація та оновлення;**

Електронні засоби навчання дозволяють легко актуалізувати та оновлювати навчальний матеріал.

Отже, тепер можемо розкрити саме проблематику створення електронного засобу навчання. Вона виникає з ряду проблем, які можуть впливати на ефективність та якість навчального процесу. Основні проблеми, пов'язані з розробкою електронних засобів навчання, включають:

**Планування та структурування:** Однією з ключових проблем є розробка чіткого плану навчального процесу та структурування матеріалів. Електронний засіб навчання повинен бути організованим та послідовним, щоб студентам було легко орієнтуватися та отримувати інформацію в логічній послідовності.

**Відповідність потребам користувачів:** Розробники електронного засобу навчання повинні ретельно досліджувати та розуміти потреби та очікування своїх користувачів. Різні студенти можуть мати різні стилі навчання, рівні володіння та інтереси, тому електронний засіб навчання повинен бути гнучким та пристосовуватися до різних типів користувачів.

**Взаємодія та залучення:** Є важливим, щоб електронний засіб навчання сприяв взаємодії та активному залученню студентів. Це може бути досягнуто шляхом включення інтерактивних завдань, вправ, тестів та можливостей спілкування з викладачами або іншими студентами.

**Мотивація та заохочення:** Важливо, щоб електронний засіб навчання був мотивуючим для студентів. Це може включати використання системи винагород або можливість відстежування своїх досягнень.

Таким чином можна сказати, що електронні засоби навчання в сучасному світі мають багато переваг та користі для студентів, викладачів та навчальних закладів. Розроблюваний нами тренажер так само націлений на покращення якості навчання студентів за рахунок усіх вищезазначених переваг.

## 2.2 Огляд програмних засобів для створення програмного забезпечення

Після того, як була розкрита проблематика створення електронного засобу навчання та висвітлені ключові переваги IT-технологій над усталеними методами навчання, варто перейти до пояснення таких сутностей, як програмне забезпечення та програмні засоби. В цьому розділі досліджено такі питання, як: програмне забезпечення та програмний засіб, засоби та методи для створення програмних продуктів.

**Програмне забезпечення (далі - ПЗ)** - це сукупність програм, даних та інструкцій, що виконуються на пристроях, щоб забезпечити певну функціональність або виконати певні завдання. Програмне забезпечення може мати різноманітні застосування, від операційних систем, що керують роботою пристрою, до програмних додатків, які виконують специфічні завдання для користувачів. ПЗ може бути розроблене для різних платформ, включаючи комп'ютери, мобільні пристрої, сервери, та інші девайси.
**Програмний засіб** - це часткова підмножина програмного забезпечення, яка зазвичай використовується для реалізації конкретної задачі, функції або операції. Це може бути програма чи набір програм, спеціально розроблений для виконання певної задачі, діяльності чи процесу. Прикладами програмного засобу може бути редактор тексту, візуальна обробка, інструменти для розробки програм тощо.

Таким чином, різниця між програмним забезпеченням та програмним засобом полягає в тому, що програмне забезпечення охоплює всі типи програм, тоді як програмний засіб може вказувати на конкретні інструменти або програми, що використовуються для вирішення певних завдань чи проблем.

**Основні характеристики програмного забезпечення включають:** функціональність, надійність, продуктивність, безпеку та зручність використання.

В загальному розумінні, програмне забезпечення є ключовим компонентом сучасної технологічної інфраструктури, яке дозволяє виконувати широкий спектр завдань та забезпечує функціональність, яка допомагає нам у повсякденному житті.

**Створення програмного забезпечення** - це процес котрий включає кілька етапів, таких як: аналіз вимог, проектування, реалізацію, тестування, впровадження та підтримку програмного забезпечення.

Створення програмного забезпечення вимагає використання різноманітних інструментів та технологій, включаючи:

1. **Мови програмування;**

Мовою програмування є формальний набір правил і синтаксичних конструкцій, які використовуються для написання кодів програм для комп'ютерів або інших пристроїв. Мова програмування - це засіб комунікації між людьми та комп'ютерами, що дозволяє створювати програми, які виконують певні завдання.

Мовам програмування властива власна синтаксична структура, правила написання коду та набір інструкцій, які використовуються для створення ПЗ. Кожна мова має свої особливості, переваги та обмеження, що робить їх спеціалізованими для різних потреб та, відповідно,сценаріїв використання.

Мови програмування можуть бути використані для розробки різноманітних програм, включаючи веб-додатки, мобільні додатки, операційні системи, наукові обчислення, штучний інтелект, вбудоване програмування та багато інших завдань. Вони є важливим інструментом у всіх сферах ІТ та програмування.
Мови програмування діляться на різноманітні категорії: високі, низькі, скриптові, функціональні, мови обробки даних та статистики та ін. До найпоширеніших та найпопулярніших мов програмування належать: Python, Java, JavaScript, C++, C#, Ruby.

Для розробки тренажеру було обрано мову Python, що є простою та ефективною мовою програмування, яка підтримує об'єктно-орієнтований підхід. Вона широко використовується в наукових дослідженнях, веб-розробці, штучному інтелекті та інших областях.

Python має велику екосистему стандартних бібліотек і сторонніх модулів, які допомагають розробникам реалізувати різноманітні функціональні можливості.

Наприклад:

· Бібліотека NumPy забезпечує підтримку для наукових обчислень;

· Бібліотека Pandas - для обробки та аналізу даних;

· Бібліотека Django - для веб-розробки та інші.

Python також є інтерпретованою мовою програмування, що означає, що можливо виконувати код без необхідності компіляції. Це забезпечує швидку зміну та налагодження коду, що корисно під час розробки тренажера з тестування студентів як в нашому випадку;

Важливо також зазначити, що Python має велику та активну спільноту розробників, яка забезпечує підтримку, документацію, сторонні розширення та приклади коду. Це робить Python привабливим вибором для розробки тренажеру, оскільки можливо знайти багато ресурсів та підтримку в разі потреби.

Загалом, Python є популярною мовою програмування, яка добре підійде для створення тренажерів з тестування студентів завдяки своєму зручному синтаксису, гнучкістю, багатофункціональністю та підтримкою спільноти розробників.

1. **Інтегровані середовища розробки (IDEs);**

Інтегровані середовища розробки є основними інструментами для розробки програмного забезпечення. Вони надають розробникам інтерфейс для написання, тестування та налагодження коду. Деякі популярні IDE включають Visual Studio (для розробки NET-програм), PyCharm (для Python).
Для розробки тренажеру було обрано компілятор PyCharm, який є інтегрованим середовищем розробки (IDE), спеціально розробленим для роботи з Python. Він надає розширену підтримку мови, автодоповнення коду, налагоджувальний засіб та інші корисні функції для розробки на Python. PyCharm має багато корисних функцій, таких як система контролю версій, інструменти для управління пакетами та залежностями, підтримка віртуальних середовищ та інтеграція з іншими інструментами розробки. Також PyCharm доступний для різних операційних систем, що дозволяє розробляти на Python на будь-якій платформі.

1. **Текстові редактори;**

Текстові редактори - це програми для створення та редагування текстових файлів. Вони можуть мати різні функції, призначення та спеціалізації залежно від потреб користувача та типу роботи, яку вони виконують. Відповідно до цього, існують різні види текстових редакторів: від простих (Блокнот, Нотатки, Google docs) до більш складних, котрі містять інструменти для підсвічування синтаксису для різних мов програмування, плагіни для розширення функціональності, роботу з віддаленими репозиторіями (для роботи з Git)

1. **Системи керування версіями;**

Це інструменти, які дозволяють розробникам відстежувати зміни в коді, спільно працювати над проектами та керувати версіями програмного забезпечення. Існують локальні, централізовані системи керування версіями, а також розподілені системи. До найпопулярніших систем керування версіями відносять Git - розподілену систему керування версіями, яка зберігає повну копію коду на кожному комп'ютері розробника та дозволяє ефективно працювати офлайн із розподіленим репозиторієм.

1. **Бібліотеки та фреймворки;**

Це набори попередньо написаного коду та функцій, які допомагають розробникам виконувати певні завдання безпосередньо. Вони забезпечують готові рішення для розробки певних функціональностей, що полегшує розробку та прискорює процес створення програмного забезпечення. Розробники використовують ці засоби для створення, тестування та налагодження програмного забезпечення з метою забезпечення його якості та ефективності. Бібліотеки є колекціями функцій, методів та класів, які можна використовувати для конкретних завдань у розробці програм. Вони підвантажуються за потребою і можуть бути використані у проекті для конкретних функцій чи операцій. Приклади бібліотек: NumPy у Python для наукових обчислень, React у JavaScript для створення інтерфейсів, Pandas для обробки даних у Python.
Фреймворки, у свою чергу, виступають платформою для розробки програм, яка надає структуру та правила для побудови програмного продукту. Вони визначають архітектуру програми і надають розробникам інструменти для створення власної логіки на основі цієї структури. Фреймворки можуть включати в себе бібліотеки для певних завдань, але також надають правила та стандарти розробки. Прикладами фреймворків є Django для веб-розробки у Python, Angular у JavaScript для створення односторінкових додатків, .NET Framework для розробки програм на платформі Microsoft.

Створення програмного забезпечення також включає в себе процес збору вимог від користувачів, аналізу та проектування архітектури програми, розробки коду, тестування для перевірки правильності та надійності програми, а також впровадження та підтримку програмного забезпечення після його релізу.

Створення програмного забезпечення є складним процесом, який вимагає співпраці між розробниками, аналітиками, тестувальниками та іншими учасниками проекту. Використання певних програмних засобів та методологій розробки може сприяти покращенню якості та продуктивності розробки програмного забезпечення.

У загальному розумінні, програмний засіб для створення програмного забезпечення - це будь-який інструмент, який полегшує розробку, тестування та випуск програмного забезпечення. Вибір конкретного програмного засобу залежить від типу проекту, мови програмування та інших факторів.

Як було зазначено, для створення тренажеру з вивчення основ мови програмування Python, що передбачатиме користувацький інтерфейс з автоматичною генерацією тестів, використовуватиметься мова об’єктно-орієнтованого програмування Python та середовище розробки PyCharm. Процес розробки програмного засобу включатиме усі етапи розробки програмного забезпечення – від аналізу вимог до тестування, впровадження та підтримки готового продукту.

## 2.3 Необхідність та актуальність теми роботи

 Щоб підкреслити необхідність та актуальність теми роботи, слід зазначити, що в сучасному цифровому світі, де дистанційне навчання набуває все більшої популярності, інтерактивні та зручні інструменти для тестування важливі для створення ефективного навчального процесу. Такі тренажери сприяють покращенню якості навчання, дозволяючи студентам та викладачам миттєво оцінювати знання, отримувати зворотній зв'язок та адаптувати процес навчання до індивідуальних потреб. Також можна відзначити, що вони дозволяють зручно створювати та редагувати тести, використовувати різноманітні формати питань і допомагають у підготовці до оцінювань. Загалом, актуальність цієї теми полягає у підтримці сучасних методів навчання та оцінювання, що сприяють підвищенню ефективності освіти.

Отже, створення тренажеру для тестування студентів має декілька суттєвих переваг і відповідає сучасним потребам освітнього середовища:

**Гнучкість та доступність.** Тренажери для тестування можуть бути доступні онлайн, що дозволяє студентам зручно виконувати тести в будь-який зручний для них час та місце, забезпечуючи гнучкість у навчанні.

**Автоматизована перевірка.** Вбудована автоматична перевірка відповідей дозволяє отримувати миттєві результати тестування, що полегшує процес оцінювання та надає швидкий зворотний зв'язок студентам.

**Посилення навчального процесу.** Тестування може бути використане як інструмент для закріплення знань, перевірки розуміння матеріалу та підготовки до екзаменів. Тренажери дозволяють створювати різноманітні тести для різних цілей навчання.

**Аналіз результатів.** Системи тренажерів можуть надати детальну статистику та аналіз результатів тестування, що допомагає викладачам та студентам зрозуміти слабкі та сильні сторони, а також здійснювати індивідуальний підхід до навчання.

**Можливість редагування та оновлення матеріалу.** Зручний інтерфейс для створення та редагування тестів дозволяє легко вносити зміни та оновлювати матеріал відповідно до поточних потреб і вимог.

Тому, актуальність створення тренажерів для тестування студентів підкреслюється сучасними тенденціями в освіті, коли дистанційне навчання та онлайн-ресурси стають все більш популярними. Засоби цифрової освіти стають важливою складовою для підтримки ефективного навчання та оцінювання. Такі тренажери дозволяють створювати інтерактивне та зручне середовище для студентів, що сприяє покращенню їх освітнього процесу та результативності.

**РОЗДІЛ 3. ТЕОРЕТИЧНА ЧАСТИНА**

## 3.1 Алгоритмізація процесу розробки програмного забезпечення

В даному розділі варто розкрити, саме алгоритм роботи навчального тренажеру. В нашому випадку, алгоритм роботи навчального тренажеру буде розкрито з сторони користувача.

**Крок 1. Запуск програмного продукту.**

Першим етапом роботи є запуск електронного засобу навчання (ЕЗН). Відкривається десктопна сторінка, на якій міститься довідкова інформація про теоретичні основи мови програмування Python.

**Крок 2. Ознайомлення з навчальними матеріалами.**

Передбачається, що студент ознайомлюється з довідковою сторінкою та вивчає подані правила та визначення.

**Крок 3. Натискання на кнопку “Почати тест”.**

Користувач повинен натиснути на кнопку “Почати тест”, після чого відкривається друга сторінка з тестуванням.

На сторінці міститься елемент “надпис”, або “label”, в якому сформульовано питання. Питання формулюється автоматично, рандомним методом обравши інформацію із попередньо створеного словника.

**Крок 4. Вибір правильної відповіді.**

Окрім надпису на сторінці тестування розміщені також 4 радіокнопки, у яких автоматично рандомно підібрано 4 варіанти відповіді. Користувачу необхідно натиснути на варіант, котрий він вважає правильним.

**Крок 5. Натискання кнопки “відповісти”.**

Після того, як користувач обрав відповідь, він повинен натиснути на кнопку “відповісти”, що розміщено під радіокнопками посередині екрану. Після натискання користувачем на кнопку система починає автоматичну перевірку відповіді шляхом зіставлення відповіді та питання у попередньо створеному словнику.

**Крок 6. Отримання сповіщення про результат.**

Коли автоматичну перевірку завершено, користувач отримує сповіщення про результат у вигляді окремого вікна.

Якщо відповідь правильна, то у сповіщенні зазначено “Правильно!”, якщо неправильна - надається пояснення, яка відповідь була правильною.

**Крок 7. Натискання кнопки “ок”.**

Після ознайомлення зі сповіщенням, яке з’явилось в результаті автоматичної перевірки відповіді користувача, йому необхідно натиснути на кнопку “ок” у вікні даного сповіщення. При натисканні цієї кнопки запускається процес автоматичної генерації нового завдання. Система нанового рандомно підбирає з першої колонки словника - формулювання запитання, а з другої колонки - 4 рандомні варіанти відповіді. Система побудована таким чином, що 1 відповідь завжди обиратиметься з того ж поля, звідки було взято формулювання питання.

Таким чином, було представлено алгоритм роботи тренажеру з автоматичною генерацією тестових завдань за темою теоретичні основи мови програмування Python. Система має дуже простий та зручний інтерфейс, який дозволяє користувачу інтуїтивно переходити між сторінками, давати відповіді на запитання та поглиблювати свої теоретичні знання.

## 3.2 Розробка блок-схеми з описом роботи тренажеру



«Рисунок 3.1. «Блок схема з описом роботи тренажеру»

В цьому розділі представлена блок схема з описом роботи навчального тренажеру. Більш детальний опис представлений у пункті 3.1 розділу №3 даної кваліфікаційної роботи, тому повторюватись, не має сенсу.

Варто зазначити в чому ж полягає необхідність при створення блок схеми, адже створення блок-схеми має кілька переваг, особливо при розробці програмного забезпечення.

Основні плюси створення блок-схеми включають:

**Візуалізація логіки:** Блок-схема надає візуальне представлення логіки роботи програми. Вона дозволяє побачити послідовність дій, умови, цикли та розгалуження. Це допомагає зрозуміти загальну структуру програми та логіку її роботи.

**Виявлення помилок:** Блок-схема допомагає виявляти потенційні помилки або недоліки в логіці програми ще до фактичної реалізації. Вона може допомогти виявити неузгодженості, пропущені кроки або некоректні умови. Це зменшує кількість помилок та сприяє більш якісній розробці програмного забезпечення.

**Комунікація та співпраця:** Блок-схема є ефективним засобом комунікації між розробниками, проектними менеджерами та іншими зацікавленими сторонами. Вона допомагає уточнити вимоги, обговорити логіку та зрозуміти взаємозв'язки між частинами програми.

**Вдосконалення та оптимізація:** Завдяки блок-схемі можна легко аналізувати та вдосконалювати логіку програми. Вона дозволяє ідентифікувати зайві або неефективні кроки, шукати можливості оптимізації та зробити програму більш ефективною.

Усі ці переваги роблять створення блок-схеми важливим етапом у процесі розробки програмного забезпечення, що сприяє якіснішій роботі.

## 3.3 Обґрунтування вибору програмного забезпечення

Існує досить багато об'єктно-орієнтованих мов програмування, які дозволяють розробникам створювати програмне забезпечення, орієнтоване на об'єкти. Тому, щоб зробити вибір якою мовою програмування буде написаний тренажер, варто перелічити головні мови програмування, тому ось кілька найпопулярніших об'єктно-орієнтованих мов програмування:

**Java:** Java є однією з найпоширеніших мов програмування, яка використовує об'єктно-орієнтований підхід. Вона пропонує багатий набір функцій, велику екосистему бібліотек і фреймворків, і використовується в широкому спектрі додатків, від веб-розробки до мобільних додатків та багато іншого.

**C++:** C++ є потужною мовою програмування, яка поєднує можливості низькорівневого програмування з об'єктно-орієнтованими парадигмами. Вона широко використовується в системному програмуванні, графічних додатках, вбудованих системах та інших областях.

**C#:** C# є мовою програмування, розробленою Microsoft, яка є частиною платформи .NET. Вона має сильну підтримку об'єктно-орієнтованого програмування та використовується для розробки різноманітних додатків, включаючи веб-додатки, десктопні програми та мобільні додатки.

**Python:** Python є простою та ефективною мовою програмування, яка підтримує об'єктно-орієнтований підхід. Вона широко використовується в наукових дослідженнях, веб-розробці, штучному інтелекті та інших областях.

**Ruby:** Ruby є елегантною та експресивною мовою програмування, яка активно використовує об'єктно-орієнтований підхід. Вона відома своєю простотою та гнучкістю, і широко використовується в веб-розробці та створенні додатків.

Це лише кілька прикладів об'єктно-орієнтованих мов програмування, але насправді існує багато інших мов, які також підтримують цю парадигму. Вибір мови залежить від проєкту, потреб та особистих уподобань. Мій вибір мови програмування для створення тренажеру з тестування студентів, зупинився на використання мови програмування Python. Варто додати історію створення даної мови, щоб зрозуміти мій вибір.

Мова програмування Python була створена Гвідо ван Россумом і вперше випущена у 1991 році.

Ось деякі ключові етапи історії створення Python:

**1980-1990 роки:** Гвідо ван Россум працював над розробкою мови програмування, яку він назвав ABC. Ідея полягала в створенні простої і зрозумілої мови програмування для новачків.

**1989 рік:** Гвідо ван Россум розпочав роботу над новою мовою програмування, яку він назвав Python. Він хотів створити мову, яка поєднувала б простоту використання, зрозумілість та потужність.

**1991 рік:** Перша версія Python, відома як Python 0.9.0, була випущена. Це була прототипна реалізація мови, яка мала основні конструкції, такі як функції, змінні, цикли та умовні оператори.

**1994 рік:** Випуск Python 1.0, першої стабільної версії мови. Python 1.0 вже мав значну кількість функціональності, включаючи модулі, об'єктно-орієнтованість та обробку виключень.

**2000-ті роки:** Python набув популярності серед програмістів і став використовуватись в різних сферах, включаючи веб-розробку, наукові обчислення, штучний інтелект та багато іншого. Були випущені нові версії Python з поліпшеними можливостями та функціями.

**2008 рік:** Випуск Python 3.0, значної оновленої версії мови. Python 3.0 внесла кілька несумісних змін з попередніми версіями, зокрема щодо роботи з рядками символів та друку на екран.

**Після 2008 року** Python продовжував розвиватись і став однією з найпопулярніших мов програмування у світі. З'явилися нові версії Python з поліпшеннями та новими функціями, такими як асинхронне програмування, розширені можливості обробки даних та машинного навчання.

Сьогодні Python залишається однією з найпопулярніших мов програмування завдяки своїй простоті, ефективності, широкому спектру застосувань та активній спільноті розробників.

Оскільки я раніше цікавився програмуванням на цій мові та маю певні навички, тому це особисте уподобання даної мови. Оскільки, дана програма не є досить складним в написанні, а мова програмування Python є простою та легкою у вивченні, це особливо важливо, адже наша аудиторія складається зі студентів, які можуть мати різний рівень підготовки в галузі програмування, а чим простіша та легша мова в засвоєнні, тим студентам буде швидше та легше навчитися програмувати.

Загалом, можемо сказати, що Python є вдалим вибором для створення тренажеру з тестування студентів через свою простоту, гнучкість, багату екосистему та підтримку спільноти розробників.

**РОЗДІЛ 4. ПРАКТИЧНА ЧАСТИНА**

## 4.1 Опис процесу створення програмного продукту

1. Першим етапом розробки програмного продукту є імпорт необхідних бібліотек та модулів. Python має велику колекцію стандартних бібліотек, які включають різноманітні модулі для роботи з регулярними виразами, рядками, файлами, мережами та стандартними структурами даних Ці бібліотеки Python автоматично підвантажуються і стають доступними при створенні нового проекту. Але для розробки тренажеру, який передбачає створення віконного інтерфейсу, необхідно інсталювати додаткову бібліотеку - PyQt5, яка є фреймворком для розробки графічного інтерфейсу.

PyQt5 - це бібліотека, яка надає інтерфейс для багатьох функцій і можливостей інструментарію Qt (розробники - компанія Digia). Qt є фреймворком для створення графічних інтерфейсів користувача (GUI) на різних платформах, включаючи Windows, macOS, Linux.

PyQt5 дозволяє розробникам створювати такі елементи графічного інтерфейсу, як вікна, кнопки, меню, віджети та інші. Вона надає доступ до багатьох можливостей Qt, дозволяючи створювати стильні, функціональні та крос-платформенні програми з графічним інтерфейсом.

Бібліотека інсталюється через вікно налаштувань інтерпретатора PyСharm.

Програмний код нижче дозволяє імпортувати необхідні для роботи модулі, обʼєкти та класи.



***Рисунок 4.1 “Імпорт необхідних модулів та класів”***

Перший рядок імпортує модуль sys, який надає доступ до системних функцій та параметрів Python.

Наступний рядок імпортує різні класи та об'єкти з модуля QtWidgets, який входить до бібліотеки PyQt5. Ось що імпортується:

* + QApplication: Клас, який керує основними параметрами програми PyQt.
	+ QMainWindow: Клас головного вікна програми, яке може містити різні елементи.
	+ QWidget: Базовий клас для всіх інтерфейсних елементів.
	+ QVBoxLayout: Менеджер розташування, який організовує елементи у вертикальний стовпець.
	+ QPushButton: Клас кнопки, яку можна натискати.
	+ QLabel: Клас мітки, яка відображає текст або зображення.
	+ QRadioButton: Клас радіо-кнопки, яка дозволяє користувачеві обирати один з декількох варіантів.
	+ QMessageBox: Клас вікна повідомлення, яке може показати користувачеві певне повідомлення або попередження.

Ці імпорти дають можливість використовувати ці класи та об'єкти для створення вікон, кнопок, міток та інших елементів у графічному інтерфейсі програми на основі PyQt5.

2. Наступним етапом роботи є розробка списку словників questions, який міститиме словники, в яких відображатимуться запитання для тренажеру з можливими варіантами відповіді та правильною відповіддю.



***Рисунок 4.2. “Список словників”***

Кожен словник містить ключові пари “ключ: значення”:

* "question": Рядок, що містить питання.
* "options": Список рядків, які представляють варіанти відповідей на це питання.
* "answer": Рядок, який містить правильну відповідь на питання.

Розроблено 10 словників, відповідно попередньо сформовано 10 запитань з 4-ма варіантами відповіді до кожного. Таким чином, тест також міститиме 10 тестових завдань.

3.Далі викликається клас QMainWindow для створення головного вікна та відбувається його конструювання.


***Рисунок 4.3. “Створення та конструювання головного вікна”***

При конструюванні вікна встановлюються такі його параметри, як заголовок сторінки (self.setWindowTitle("Тест") - встановлює заголовок вікна на "Тест"), її розмір та положення (self.setGeometry(100, 100, 300, 200) - встановлює параметри, що відповідають координатам лівого верхнього кута вікна (100, 100) і його розміром у ширину 300 та висоту 200).
Далі ініціалізуються змінні self.currentQuestion = -1 та self.score = 0:, які необхідні для відстеження поточного номера питання та результату тестування.

Після цього створюється центральний віджет QWidget, на якому будуть розміщені всі наступні віджети. Він встановлюється як віджет головного вікна. В програмному коді також було реалізовано функцію вертикального розміщення для розміщення віджетів один під одним у центральному віджеті. Для цього використано обʼєкт QVBoxLayout(). Це робить вікно візуально читабельнішим та зручнішим для користування.

Після цього створюється обʼєкт кнопка з написом "Почати тестування" за допомогою self.startButton = QPushButton("Почати тестування", self. Усі віджети створюються з прив'язкою до головного віджету self.

Рядок self.startButton.clicked.connect(self.startTest) забезпечує привʼязку методу self.startTest, який ініціює початок тестування, до натискання кнопки.

І останній рядок цієї частини програмного коду забезпечує додавання кнопки в центр віджету.

4. Наступним етапом є розробка методу, що забезпечить власне початок тестування


***Рисунок 4.4. “Ініціалізація процесу тестування”***

Для цього використовується метод startTest(), який ініціалізує початкові значення для тесту (номер поточного питання і балів) і потім викликає метод displayQuestion(), що розпочинає процес тестування, виводячи перше питання. В методі startTest() вказується обʼєкт self, це означає, що метод працює в межах об'єкта даного класу.

Рядок self.currentQuestion = 0 використовується для відстеження номера питання в тесті. А self.score = 0 підраховує кількість балів користувача за правильні відповіді під час тестування.

Метод displayQuestion() відповідає за відображення питання та варіантів відповідей під час тестування.

5.Наступний фрагмент коду виконує очищення вмісту попереднього питання і його варіантів відповідей, які можуть бути вже відображені на екрані, перед тим, як відобразити нове питання в вікні.



***Рисунок 4.5. “Очищення попереднього вмісту вікна”***

Задано цикл for i in reversed(range(self.layout.count())):, який працює з усіма віджетами, які додані до макету. Функція (layout). self.layout.count() повертає кількість віджетів у макеті. Звернення до self.layout.itemAt(i)повертає віджет, який знаходиться на позиції i в макеті. Метод widget() повертає сам віджет, який міститься в даній позиції i. Після цього if widget is not None: перевіряє, чи не є віджет пустим. Якщо він не пустий - система видаляє його вміст.

6. Наступна частина коду відповідає за відображення нового питання або оцінки, у тому випадку, якщо всі тестові завдання було пройдено.



***Рисунок 4.6. “Умова для відображення питання або результату”***

Перший рядок перевіряє, чи поточний номер питання менший за загальну кількість питань у списку questions. Це потрібно для того, щоб розуміти, чи відображати нове питання, чи тест вже завершено.

Наступний рядок отримує питання зі списку questions за поточним індексом self.currentQuestion і за допомогою self.layout.addWidget(QLabel(question["question"])) питання додається у вікно у вигляді надпису як віджет.
Наступний метод забезпечує розміщення на вікні варіантів відповіді у вигляді радіокнопок, що дозволить користувачу вибрати правильну відповідь. Використовується self.radioButtons = [], який створює порожній список, куди будуть додаватися об'єкти радіокнопок. Цикл for option in question["options"]: перебирає всі варіанти відповідей, які відносяться до поточного питання. Обʼєкт самої радіокнопки для кожного варінату відповіді створює radioButton = QRadioButton(option). Створена радіокнопка додається до списку радіокнопок методом self.radioButtons.append(radioButton). Після чого радіокнопки розміщуються вертикально за допомогою self.layout. Кожна радіокнопка відображатиметься в інтерфейсі.

Додається кнопка з надписом "Submit", яка буде використовуватись для підтвердження відповіді користувачем. До кнопки прив'язується метод checkAnswer, а саме до події натискання на кнопку "Submit". Це дозволяє обробляти відповіді користувача після натискання на цю кнопку. Кнопка так само додається до вертикального розміщення.

Якщо умова if не виконується (тобто всі питання вже відповіли), програма переходить до else. За допомогою QMessageBox.information (self, "Оцінка", f"Твоя оцінка: {self.score}") виводиться інформація про оцінку користувача після завершення тестування. Показується кількість балів, яку користувач отримав. Функція self.close ( закриває вікно програми після завершення тестування.

7.Наведений нижче шматок коду відповідає за перевірку відповіді користувача та підрахунок балів.



***Рисунок 4.7. “Перевірка відповіді та підрахунок балів”***

Створюється змінна selectedOption, котра зберігатиме відповідь користувача. За допомогою self.currentQuestion з списка questions вибирається правильна відповідь correctAnswer. Цикл for radioButton in self.radioButtons: дозволяє перебрати усі 4 радіокнопки, що є у списку. Умова if radioButton.isChecked(): перевіряє, чи радіокнопка була обрана користувачем. Якщо кнопка була обрана, її текст зберігається у змінній selectedOption.
Далі відбувається порівння обраної відповіді з правильною за допомогою умови if selectedOption == correctAnswer:. Якщо відповіді збігаються, то лічильник self.score += 1 додає 1 бал.
Функції QMessageBox.information(self, "Результат", "Вірно!") та QMessageBox.information(self, "Результат", f"Не вірно, правильна відповідь: {correctAnswer}"): показують вікно з інформацією про правильність чи неправильність відповіді користувача.
Далі лічильник збільшує номер поточного питання, готуючи програму до відображення наступного. Викликається self.displayQuestion(): відповідає за відображення наступного питання або оцінку користувача після відповіді.

8. Остання частина коду відповідає за створення самого додатку: створює його головне вікно, відображає його і запускає головний цикл обробки подій, необхідний для функціонування інтерфейсу та взаємодії з користувачем.



***Рисунок 4.8. “Створення віконного інтерфейсу”***

У цій частині коду створюється обʼєкт додатку QApplication, він представляє головний контект програми PYQt. Після цього творюється екземпляр класу TestApp, який і є головним вікном програми. Викликається mainWin.show(), яка відображає вікно на екрані користувача.
Далі, за допомогою sys.exit(app.exec\_()):, запускається головний цикл програми PyQt. app.exec\_() розпочинає цикл обробки подій, що дозволяє програмі очікувати на події від користувача (натискання кнопок) і відреагувати на них. Він потрібен для забезпечення коректного виходу з програми після завершення головного циклу. Без цього може виникнути проблема з відповідним закриттям додатку.

## 4.2 Тестування розробленого програмного продукту

Для тестування програмного продукту ми запустимо програму і спробуємо пройти різноманітні сценарії подій як користувач.

При запуску програми у нас є лише один варіант вибору – натискання на кнопку «Почати тестування». Натискаємо на кнопку.



***Рисунок 4.8. «Знімок екрану з програми»***

Після натискання на екрані з’являється перше питання та 4 варіанти відповіді. Спробуємо обрати одну відповідь:



***Рисунок 4.9.*** «***«Знімок екрану з програми»***

Відповідь виділилась, але не відправилась без натискання на «Submit». Спробуємо обрати 2 відповіді одночасно:



***Рисунок 4.10.*** «***«Знімок екрану з програми»***

Обрати 2 відповіді одночасно не вдалося. При виборі варіанту «б», попередня відповідь затерлась. Відповідь знову не відправилась без натискання на «Submit».
Пробуємо почергово обрати відповіді «в» та «г» - попередня відповідь перезатирається, виділяється лише кнопка, на яку натиснули останньою.

Далі спробуємо обрати правильну та неправильну відповіді.
Спершу обираємо правильну відповідь та натискаємо «Submit».



***Рисунок 4.11.*** «***«Знімок екрану з програми»***

Отримуємо сповіщення та натискаємо «ОК».

 

***Рисунок 4.12.*** «***«Знімок екрану з програми»***

Тепер спробуємо обрати не правильну відповідь в цьому тесті:
Обираємо відповідь «а» та натискаємо «Submit»



***Рисунок 4.12.*** «***«Знімок екрану з програми»***

Отримуємо сповіщення про результат:



***Рисунок 4.13.*** «***«Знімок екрану з програми»***

Спробуємо також обрати інші варіанти:



***Рисунок 4.13.*** «***«Знімок екрану з програми»***



***Рисунок 4.14.*** «***«Знімок екрану з програми»***

При виборі обох відповідей отримуємо аналогічні сповіщення:



***Рисунок 4.15.*** «***«Знімок екрану з програми»***

Вищеописані кроки було пророблено для усіх десяти запитань. Багів та несправностей системи не було виявлено.

Проведемо також тестування калькулятору відповідей:
1) Дамо правильні відповіді на усі питання.
При відповіді на усі запитання система виводить наступе сповіщення:



 ***Рисунок 4.15.*** «***«Знімок екрану з програми»***

2) Дамо не правильні відповіді на усі питання.
При виборі неправильної відповіді на усі 10 запитань тесту система виводить наступне сповіщення:



***Рисунок 4.16.*** «***«Знімок екрану з програми»***

3) Дамо правильні відповіді на половину запитань.
При виборі правильної відповіді у половині тестів система вивела на екран наступний результат:



***Рисунок 4.16.*** «***«Знімок екрану з програми»***

Аналогічний тест було проведено і для іншої кількості правильних відповідей, система не показала жодних помилок.

Отже, було проведено тестування різноманітних сценаріїв поведінки користувача і не виявлено жодного багу чи несправності системи. Система проста, інтуїтивно зрозуміла користувачеві і дозволяє зручно пройти тестування з предметної галузі. Система не обмежує користувача в часі відповіді, що теж робить її більш універсальною для великої кількості користувачів.

## 4.3 Користувацька інструкція до програмного продукту

**Крок 1. Почніть тестування.**
Для того, щоб розпочати тестування, натисніть на кнопку «Почати тестування»



***Рисунок 4.17.*** «***«Знімок екрану з програми»***

**Крок 2. Прочитайте питання.**
Після того, як ви натиснули на кнопку «Почати тестування», на екрані з’являється питання та 4 варіанти відповіді. Уважно прочитайте питання.



***Рисунок 4.18.*** «***«Знімок екрану з програми»***

**Крок 3. Оберіть відповідь.**
На екрані ви бачите питання та 4 варіанти відповіді. До кожного питання є лише 1 правильна відповідь. Для вибору відповіді натисніть на кружечок поряд із відповіддю, яку вважаєте правильною. При натисканні на кнопку вона виділяється кольором. Для того, щоб відправити відповідь, натисніть «Submit»



***Рисунок 4.19.*** «***«Знімок екрану з програми»***

**Крок 4. Ознайомтесь з результатом.**
Після натискання на «Submit» на вашому екрані з’явиться сповіщення про результат відповіді.
а) Якщо відповідь правильна, ви отримаєте наступне сповіщення:



***Рисунок 4.20.*** «***«Знімок екрану з програми»***

б) Якщо відповідь не правильна, ви отримаєте наступне сповіщення з поясненням:



***Рисунок 4.21.*** «***«Знімок екрану з програми»***

**Крок 5. Перейдіть до наступного питання.**
Для того, щоб перейти до наступного питання, необхідно натиснути кнопку «ОК» у сповіщенні, яке транслює результат попереднього тесту:



***Рисунок 4.22.*** «***«Знімок екрану з програми»***

**Крок 6. Дайте відповіді на всі запитання.**
Тест містить 10 запитань. Проробіть описані вище кроки для усіх запитань. Після відповіді на останнє питання і натискання на «ОК» у сповіщенні, ви отримаєте сповіщення про результат:



***Рисунок 4.23.*** «***«Знімок екрану з програми»***

Результат відповідає кількості правильних відповідей. Кожне питання оцінюється в 1 бал. Максимальна оцінка – 10 балів.

**Крок 7. Закрийте тест.**
Для того, щоб закрити вікно програми, натисніть «ОК» на сповіщенні про результат.



***Рисунок 4.24.*** «***«Знімок екрану з програми»***

**ВИСНОВКИ**

У ході написання даної кваліфікаційної роботи та вивчення основ мови програмування Python і розробки елементів програмного забезпечення були розглянуті ключові аспекти цієї теми. Python, як мова програмування, має багато переваг, які роблять його привабливим вибором для розробки тренажеру з тестування студентів. Він простий у вивченні та читанні, має зрозумілу синтаксичну структуру та багатий екосистему бібліотек, що полегшують розробку програмного забезпечення.

PyCharm, у свою чергу, є потужним інтегрованим середовищем розробки (IDE) для мови програмування Python, яке надає широкий спектр інструментів для розробки, налагодження та тестування програмного забезпечення. Використання PyCharm сприяє покращенню продуктивності розробників та забезпечує зручне робоче середовище для створення тренажеру з тестування студентів.

У процесі розробки програмного забезпечення з використанням Python та PyCharm важливо мати глибокі знання основ мови програмування, включаючи роботу зі змінними, умовними операторами, циклами та функціями. Розуміння цих концепцій допомагає ефективно використовувати можливості Python та розробляти функціональний та ефективний тренажер для тестування студентів.

В цілому, розробка елементів програмного забезпечення з використанням мови програмування Python та інструментів, таких як PyCharm, виявляється перспективною і зручною для створення тренажерів з тестування студентів. Ці інструменти допомагають зробити процес розробки більш ефективним, забезпечуючи простоту, продуктивність та зручність для розробників. Знання основ мови програмування Python є необхідними для успішного створення програмного забезпечення та його подальшого розвитку.

**СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ**

1. <https://docs.python.org/3/>;
2. <https://uk.wikipedia.org/wiki/Python>;
3. Pythone для дітей. / Джейсон Бриггс ; пер. с англ. Станислава Ломакина ; [науч. ред. Д. Абрамова]. — М. : Манн, Иванов и Фербер, 2017. — 320 с.;
4. <https://github.com>;
5. <https://ru.stackoverflow.com>;
6. <https://dou.ua/forums/topic/40798/>;
7. «Python Crash Course» - автор Eric Matthes;
8. Automate the boring Stuff with Python – автор Al Sweigert;
9. <https://stackoverflow.com>;
10. <https://www.coursera.org>;
11. "Python Crash Course" Ерік Метеза;
12. Automate the Boring Stuff with Python" Аль Свейгарт;
13. "Fluent Python" Луціано Рамальйо;
14. "Learning Python" Марка Лутца.

**ДОДАТОК А. Код програми**

import sys

from PyQt5.QtWidgets import QApplication, QMainWindow, QWidget, QVBoxLayout, QPushButton, QLabel, QRadioButton, QMessageBox

# Тести (питання та відповіді )

questions = [

 {"question": "1. Який тип даних повертає функція input()?", "options": ["а) список", "б) рядок", "в) ціле число", "г) число з плаваючою точкою"], "answer": "б) рядок"},

 {"question": "2. Який із варіантів є рядком?", "options": ["а) (Line)", "б) 4563", "в) «This is a string»", "г) (1, 2, 3, «line»)"], "answer": "в) «This is a string»"},

 {"question": "3. Призначенням функції str() є:", "options": ["а) перетворення рядка в число", "б) перетворення числа в рядок", "в) запит даних в користувача", "г) усі вищеперелічені"], "answer": "б) перетворення числа в рядок"},

 {"question": "4. Тип даних НЕ визначає:", "options": ["а) діапазон можливих значень", "б) місце зберігання даних", "в) спосіб зберігання даних", "г) допустимі операції"], "answer": "а) діапазон можливих значень"},

 {"question": "5. Що НЕ є типом даних:", "options": ["а) словник", "б) рядок", "в) умова", "г) ціле число"], "answer": "в) умова"},

 {"question": "6. Який результат буде виведено у консоль при запуску програми:\n\t\tx = 23\n\t\t\t\tnum = 0 if x > 10 else 11\n\t\tprint(num)", "options": ["а) 10", "б) 0", "в) помилка", "г) 11"], "answer": "б) 0"},

 {"question": "7. Яка функція виводить результат у консоль?", "options": ["а) out()", "б) print()", "в) write()", "г) log()"], "answer": "б) print()"},

 {"question": "8. Скільки бібліотек можна імпортувати в один проект?", "options": ["а) 1", "б) 10", "в) 1000", "г) нескінченну кількість"], "answer": "г) нескінченну кількість"},

 {"question": "9. Який результат буде виведено у консоль при запуску програми:\n\t\tfor i in range(5):\n\t\t\tif i % 2 == 0:\n\t\t\t\tcontinue\n\t\t\tprint(i)", "options": ["а) помилка", "б) 0", "в) числа 1 і 3", "г) числа 2 і 4"], "answer": "в) числа 1 і 3"},

 {"question": "10. Оператор присвоювання позначається:", "options": ["а) =", "б) ==", "в) =:", "г) !="], "answer": "а) ="},

]

class TestApp(QMainWindow):

 def \_\_init\_\_(self):

 super().\_\_init\_\_()

 self.setWindowTitle("Тест")

 self.setGeometry(100, 100, 300, 200)

 self.currentQuestion = -1

 self.score = 0

 self.centralWidget = QWidget(self)

 self.setCentralWidget(self.centralWidget)

 self.layout = QVBoxLayout()

 self.centralWidget.setLayout(self.layout)

 self.startButton = QPushButton("Почати тестування", self)

 self.startButton.clicked.connect(self.startTest)

 self.layout.addWidget(self.startButton)

 def startTest(self):

 self.currentQuestion = 0

 self.score = 0

 self.displayQuestion()

 def displayQuestion(self):

 for i in reversed(range(self.layout.count())):

 widget = self.layout.itemAt(i).widget()

 if widget is not None:

 widget.deleteLater()

 if self.currentQuestion < len(questions):

 question = questions[self.currentQuestion]

 self.layout.addWidget(QLabel(question["question"]))

 self.radioButtons = []

 for option in question["options"]:

 radioButton = QRadioButton(option)

 self.radioButtons.append(radioButton)

 self.layout.addWidget(radioButton)

 submitButton = QPushButton("Submit", self)

 submitButton.clicked.connect(self.checkAnswer)

 self.layout.addWidget(submitButton)

 else:

 QMessageBox.information(self, "Оцінка", f"Твоя оцінка: {self.score}")

 self.close()

 def checkAnswer(self):

 selectedOption = None

 correctAnswer = questions[self.currentQuestion]["answer"]

 for radioButton in self.radioButtons:

 if radioButton.isChecked():

 selectedOption = radioButton.text()

 break

 if selectedOption == correctAnswer:

 self.score += 1

 QMessageBox.information(self, "Результат", "Вірно!")

 else:

 QMessageBox.information(self, "Результат", f"Не вірно, правильна відповідь: {correctAnswer}")

 self.currentQuestion += 1

 self.displayQuestion()

app = QApplication(sys.argv)

mainWin = TestApp()

mainWin.show()

sys.exit(app.exec\_())