Київський національний торговельно-економічний університет

Кафедра комп'ютерних наук та інформаційних систем

ВИПУСКНА КВАЛІФІКАЦІЙНА РОБОТА

на тему:

«Розробка інтерактивного Web - додатку тестування знань студентів»

Студента 4 курсу, 10 групи,
спеціальності
122 «Комп'ютерні науки» _____ *підпис студента* Дудченко Денис Юрієвич

Науковий керівник
Кандидат фізико-математичних наук _____ *підпис керівника* Філімонова Тетяна Олегівна

Гарант освітньої програми
кандидат технічних наук, доцент _____ *підпис керівника* Демідов Павло Георгійович

Київ 2021

Факультет інформаційних технологій
Кафедра комп'ютерних наук та інформаційних систем
Спеціальність 122 «Комп'ютерні науки»

Затверджую
Зав. кафедри _____Пурський О.І.
«18» грудня 2020р.

Завдання
на випускну кваліфікаційну роботу (проект) студенту

Дудченко Денис Юрієвич
(прізвище, ім'я, по батькові)

1. Тема випускної кваліфікаційної роботи (проекту)

«Розробка інтерактивного Web - додатку тестування знань студентів»
Затверджена наказом ректора від *«15» грудня 2020 р. № 3780*
2. Строк здачі студентом закінченої роботи *31 травня2021 року*
3. Цільова установка та вихідні дані до роботи
Мета роботи: розробка інтерактивного Web - додатку тестування знань студентів
Об'єкт дослідження: процеси тестування знань студентів
Предмет дослідження: інформаційні системи і технології в системі тестування знань студентів
4. Перелік графічного матеріалу: рисунки, таблиці, додатки, лістинг, презентація

5. Консультанти по роботі із зазначенням розділів, за якими здійснюється консультування:

Розділ	Консультант (прізвище, ініціали)	Підпис, дата	
		Завдання видав	Завдання прийняв
1	Філімонова Т.О.	22.12.2020 р.	22.12.2020 р.
2	Філімонова Т.О.	22.12.2020 р.	22.12.2020 р.
3	Філімонова Т.О.	22.12.2020 р.	22.12.2020 р.

6. Зміст випускної кваліфікаційної роботи (проекту) (перелік питань за кожним розділом)

ВСТУП
РОЗДІЛ 1. ТЕОРЕТИЧНІ АСПЕКТИ РОЗРОБКИ ІНТЕРАКТИВНОГО ДОДАТКУ ТЕСТУВАННЯ ЗНАНЬ
1.1 Поняття та зміст тестування знань
1.1.1 Типи завдань
1.1.2 Основні правила побудови тестування
1.2 Методи і особливості проведення тестування знань
РОЗДІЛ 2. МОДЕЛЬ ІНТЕРАКТИВНОГО ДОДАТКУ ДЛЯ ПРОВЕДЕННЯ ТЕСТУВАННЯ ЗНАНЬ
2.1 Аналіз системи показників для проведення тестування знань
2.2 Формування моделі проведення тестування знань
2.2.1 Відповіді
2.2.2 Якість тестування
РОЗДІЛ 3. РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ ПРОВЕДЕННЯ ОНЛАЙН АНКЕТУВАННЯ
3.1 Інформаційно-логічна модель проведення онлайнового анкетування
3.1.1 Діаграма варіантів використання
3.1.2 Діаграма класів
3.2 Програмна реалізація проведення онлайнового анкетування
3.2.1 Мова програмування Python
3.2.2 Фреймворк Django
3.2.3 Розробка графічного інтерфейсу користувача
ВИСНОВКИ
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ
ДОДАТКИ

. Календарний план виконання роботи

№ пор.	Назва етапів випускного кваліфікаційного проекту	Строк виконання етапів роботи	
		За планом	фактично
1	2	3	4
1	*Вибір теми випускного кваліфікаційного проекту*	*05.10.2020*	*05.10.2020*
2	*Розробка та затвердження завдання на випускний кваліфікаційний проект*	*18.12.2020*	*18.12.2020*
3	*Вступ*	*03.02.2021*	*03.02.2021*
4	*Розділ 1. Теоретичні аспекти розробки інтерактивного додатку тестування знань*	*26.02.2021*	*26.02.2021*
5	*Розділ 2. Модель інтерактивного додатку для проведення тестування знань*	*06.04.2021*	*06.04.2021*
6	*Розділ 3. Розробка програмного забезпечення для проведення тестування онлайн*	*12.05.2021*	*12.05.2021*
7	*Висновки*	*14.05.2021*	*14.05.2021*
8	*Здача випускного кваліфікаційного проекту на кафедру науковому керівнику*	*20.05.2021*	*20.05.2021*
9	*Попередній захист випускного кваліфікаційного проекту*	*26.05.2021*	*26.05.2021*
10	*Виправлення зауважень, зовнішнє рецензування випускного кваліфікаційного проекту*	*27.05.2021*	
12	*Представлення готового зшитого випускного кваліфікаційного проекту на кафедру*	*31.05.2021*	
13	*Публічний захист випускного кваліфікаційного проекту*	*За розкладом роботи ЕК*	

8. Дата видачі завдання «22» грудня 2020 р.

9. Керівник випускної кваліфікаційної роботи (проекту)
Філімонова Т.О.
(прізвище, ініціали, підпис)

10. Гарант освітньої програми Демідов П.Г.
(прізвище, ініціали, підпис)

11. Завдання прийняв до виконання студент-дипломник

Дудченко Д.Ю

(прізвище, ініціали, підпис)

12. Відгук керівника випускної кваліфікаційної роботи (проекту)

Керівник випускної кваліфікаційної роботи (проекту)

_____ 31.05.2021 р.

(підпис, дата)

13. Висновок про випускну кваліфікаційну роботу (проект)

Випускна кваліфікаційна робота (проект) Дудченко Дениса Юрійовича

(прізвище, ініціали)

може бути допущена до захисту в екзаменаційній комісії.

Гарант освітньої програми_____Демідов П.Г.

(підпис, прізвище, ініціали)

Завідувач кафедри _____Пурський О.І.

(підпис, прізвище, ініціали)

«_____»_____2021 р.

АНОТАЦІЯ

Тема дипломної роботи: «Інформаційна система інтерактивного оцінювання знань»

Виконав:Дудченко Денис Юрієвич

У випускній кваліфікаційній роботі розглядається питання створення системи інтерактивного оцінювання знань. Дипломна робота складається з восьми частин, а саме зі: вступу; першого розділу, в якому розглядається і описуються теоретико-методологічні основи дослідження, поняття та зміст тестування знань, методи і особливості проведення тестування знань; другого розділу, у якому проведено побудову моделі інтерактивного додатку тестування знань, проведено аналіз системи показників для проведення тестування знань, сформовано моделі проведення тестування; третього розділу, в якому відбувається вибір засобів програмної реалізації системи, її проектування за допомогою діаграм та, власне, розробка інформаційної системи; висновків, списку використаних джерел та додатків.

Випускна кваліфікаційна робота є науково-практичним дослідженням з питання «Розробка інтерактивного додатку тестування знань». Метою кваліфікаційної випускної роботи виступає розроблення інтерактивного додатку тестування знань засобами мови програмування Python і фреймворку Django.

Досліджено сучасні методи і засоби створення тестувань знань, виявлено відомі типи тестування та їх особливості, виявлено методи і особливості проведення тестування знань.

Завдяки обґрунтуванню теоретичних і методологічних положень спроектовано функціональні вимоги до інформаційної системи, проведено

аналіз системи показників для проведення тестування знань, сформовано моделі проведення тестування знань.

На основі результатів, отриманих в ході моделювання інтерактивного додатку тестування знань, було спроектовано і розроблено інформаційну систему тестування знань, яка в повній мірі виконує закладений в неї функціонал та готова до використання в умовах реальних тестувань знань в будь-якій предметній області. .

В ході проведеного дослідження було виявлено низку інструментів та ключових етапів впровадження інформаційних систем і комп'ютерних технологій в тематику тестування знань.

Апробація отриманих результатів допоможе вітчизняним спеціалістам сформувати єдиний алгоритм щодо впровадження і формування конкурентної стратегії, а також у функціонування технологій тестування знань на вітчизняному та закордонному ринках.

До ключових слів належать наступні: веб-сайт, тестування знань, оцінювання, HTML, CSS, Python, Django.

SUMMARY

Thesis topic: "Information system of interactive assessment of knowledge"

Completed:Dudchenko Denis Yurievich

This thesis deals with the creation of a system of interactive assessment of knowledge. Thesis consists of eight parts, namely: introduction; the first section, which considers and describes the theoretical and methodological foundations of the study, the concept and content of knowledge testing, methods and features of knowledge testing; the second section, which builds a model of an interactive application for testing knowledge, analyzes the system of indicators for testing knowledge, formed models for testing; the third section, which is the choice of means of software implementation of the system, its design using diagrams and, in fact, the development of information systems; conclusions, list of sources and appendices used.

The final qualifying work is a scientific and practical study on "Development of an interactive application for knowledge testing." The purpose of the qualifying thesis is to develop an interactive application for testing knowledge using Python programming language and Django framework.

Modern methods and means of creating knowledge testing are studied, known types of testing and their features are revealed, methods and features of knowledge testing are revealed.

Due to the substantiation of theoretical and methodological provisions, the functional requirements to the information system are designed, the analysis of the system of indicators for knowledge testing is carried out, the models of knowledge testing are formed.

Based on the results obtained during the modeling of an interactive application of knowledge testing, an information system for testing knowledge was designed and

developed, which fully performs its functionality and is ready for use in real knowledge testing in any subject area. .

The study identified a number of tools and key steps in implementing information systems and computer technology in the field of knowledge testing.

Approbation of the obtained results will help domestic specialists to form a unified algorithm for the implementation and formation of a competitive strategy, as well as in the functioning of knowledge testing technologies in domestic and foreign markets.

Keywords include website, knowledge testing, assessment, HTML, CSS, Python, Django.

ЗМІСТ

ВСТУП

В сучасному світі тенденція створення програмного забезпечення для полегшення роботи і життя стає все більш явною.

Спрощення різноманітних процесів проникає в усі сфери людського життя, починаючи роботою і побутом, і закінчуючи освітою.

Сферу освіти подібні тенденції змінюють в першу чергу, адже відхід від стандартних методик, що використовуються, призводить до спрощення навчального процесу як для студентів, так і для викладачів.

Саме з цією тематикою і пов’язана тема даної роботи.

Об’єкт дослідження представлено інтерактивними системами тестування знань.

Предмет дослідження — методи і засоби створення інтерактивних додатків тестування знань.

Мета роботи: розробка інтерактивного додатку тестування знань засобами мови програмування Python.

Для досягнення обраної мети слід виконати такі завдання:

- Проаналізувати предметну область
- Обрати засоби реалізації
- Створити діаграму варіантів використання
- Створити діаграму класів
- Спроектувати графічний інтерфейс користувача

Методи дослідження — аналіз літературних та інших джерел, комп’ютерне моделювання та проектування, розроблення програмного забезпечення.

Випускна кваліфікаційна робота є науково-практичним дослідженням з питання «Розробка інтерактивного додатку тестування знань». Метою

кваліфікаційної випускної роботи виступає розроблення інтерактивного додатку тестування знань засобами мови програмування Python і фреймворку Django.

Досліджено сучасні методи і засоби створення тестувань знань, виявлено відомі типи тестування та їх особливості, виявлено методи і особливості проведення тестування знань.

Завдяки обґрунтуванню теоретичних і методологічних положень спроектовано функціональні вимоги до інформаційної системи, проведено аналіз системи показників для проведення тестування знань, сформовано моделі проведення тестування знань.

На основі результатів, отриманих в ході моделювання інтерактивного додатку тестування знань, було спроектовано і розроблено інформаційну систему тестування знань, яка в повній мірі виконує закладений в неї функціонал та готова до використання в умовах реальних тестувань знань в будь-якій предметній області. .

В ході проведеного дослідження було виявлено низку інструментів та ключових етапів впровадження інформаційних систем і комп’ютерних технологій в тематику тестування знань.

Апробація отриманих результатів допоможе вітчизняним спеціалістам сформувати єдиний алгоритм щодо впровадження і формування конкурентної стратегії, а також у функціонування технологій тестування знань на вітчизняному та закордонному ринках.

РОЗДІЛ 1. ТЕОРЕТИЧНІ АСПЕКТИ РОЗРОБКИ ІНТЕРАКТИВНОГО ДОДАТКУ ТЕСТУВАННЯ ЗНАНЬ

1.1 Поняття та зміст тестування знань

Педагогічне тестування - це форма вимірювання знань учнів, заснована на застосуванні педагогічних тестів. Включає в себе підготовку якісних тестів, власне проведення тестування і подальшу обробку результатів, яка дає оцінку навченості тестованих [1].

Педагогічний тест - це інструмент оцінювання навченості учнів, що складається з системи тестових завдань, стандартизованої процедури проведення, обробки та аналізу результатів.

Тести можна класифікувати за різними ознаками [1]:

- за програмними цілями - інформаційні, діагностичні, навчальні, мотиваційні, атестаційні;
- за процедурою створення - стандартизовані, що не стандартизовані;
- за способом формування завдань - детерміновані, стохастичні, динамічні;
- за технологією проведення - паперові, в тому числі паперові з використанням оптичного розпізнавання, натурні, з використанням спеціальної апаратури, комп'ютерні;
- за формою завдань - закритого типу, відкритого типу, встановлення відповідності, упорядкування послідовності [1];
- по наявності зворотного зв'язку - традиційні і адаптивні

Традиційний тест містить список питань і різні варіанти відповідей. Кожне питання оцінюється в певну кількість балів. Результат традиційного тесту залежить від кількості питань, на які було дано правильну відповідь. На думку Аванесова В. С. традиційний тест - система завдань, що пред'являється в

порядку збільшення складності в один і той же час, з однаковою системою оцінювання для всіх тестованих

Адаптивний тест — Особливий вид тесту, в якому кожне наступне завдання вибирається залежно від відповідей на попередні завдання. Послідовність завдань і їх кількість в такому вигляді тесту визначається динамічно. Самими значущими перевагами комп'ютерного адаптивного тестування перед традиційним є:

- можливість адаптації під рівень знань тестованого (не доведеться відповідати на занадто складні або занадто прості питання);
- економія часу і сил за рахунок скорочення кількості завдань (довжина тесту може бути зменшена до 60%) без втрати рівня достовірності.

Тестове завдання - складова частина педагогічного тесту, що відповідає вимогам технологічності, форми, змісту і, крім того, зі статистичними вимогам:

- відомої складності;
- достатньою варіації тестових балів;
- позитивної кореляцією балів завдання з балами по всьому тесту.

1.1.1 Типи завдань

закриті:

- завдання альтернативних відповідей;
- завдання множинного вибору;
- завдання на відновлення відповідності;
- завдання на встановлення правильної послідовності.

відкриті:

- завдання вільного викладу;
- завдання-доповнення.

- Завдання з вибором однієї правильної відповіді:

При наборі тексту слова відокремлюються один від одного ...

а) двокрапкою;

б) коми;

в) прогалиною;

г) точкою.

- Завдання з вибором однієї неправильної відповіді:

Операція не має ознаки, за яким підібрані інші операції, представлені в списку ...

а) збереження тексту;

б) форматування тексту;

в) видалення фрагмента тексту;

г) переміщення фрагмента тесту;

д) копіювання фрагмента тексту.

- Завдання на установлення відповідності:

Установіть відповідність між командами і натисніть сполучення клавіш.

команда поєднання клавіш

1. Вирізати фрагмент тексту; а) CTRL + X;

2. Копіювати фрагмент тексту; б) CTRL + C;

3. Вставити фрагмент тексту. в) CTRL + V.

- Завдання з вибором декількох правильних відповідей:

Використання сліпого десятипальцевого методу веде до ...

а) зниження напруги на пальці;

б) зменшення швидкості друку;

в) зменшення кількості помилок і помилок;

г) швидкої стомлюваності пальців.

- Сортування послідовності:

Розмістіть в хронологічному порядку

iii. Бородинська битва

i. Льодове побоїще

ii. Куликовська битва

- Завдання з відкритою відповіддю:

Існує два способи освоєння клавіатури при друкуванні сліпим десятипальцевим методом:

1. соло на клавіатурі ergosolo.ru _____
2. stamina_____

1.1.2 Основні правила побудови тестування

- Використовуйте твердження, які тлумачаться однаково представниками різних субпопуляцій зацікавленої сукупності.
- Використовуйте твердження, коли особи, які мають різні думки чи риси, даватимуть різні відповіді.
- Подумайте про наявність категорії "відкритих" відповідей після списку можливих відповідей.
- Використовуйте лише один аспект конструкції, яка вас цікавить, для кожного елемента.
- Використовуйте позитивні твердження та уникайте негативів чи подвійних негативів.
- Не робіть припущень щодо респондента.
- Використовуйте чіткі та зрозумілі формулювання, легко зрозумілі для всіх рівнів освіти
- Використовуйте правильну орфографію, граматику та розділові знаки.
- Уникайте предметів, які містять більше одного запитання на предмет (наприклад, чи любите ви полуницю та картоплю?).

- Питання не повинно бути упередженим або навіть вести учасника до відповіді.

Для грамотної побудови структури тестування знань необхідно слідувати описаним вище правилам завжди, адже в противному випадку тестування може бути незрозумілим, мати більше однієї правильної відповіді, тощо, що в свою чергу призведе то отримання помилкових результатів оцінювання знань.

1.2 Методи і особливості проведення тестування знань

Тестування в педагогіці виконує три основні взаємопов'язані функції: діагностичну, навчальну і виховну:

- Діагностична функція полягає у виявленні рівня знань, умінь, навичок учня. Це основна і найочевидніша функція тестування. За об'єктивності, широті і швидкості діагностування, тестування перевершує всі інші форми педагогічного контролю.
- Навчальна функція тестування полягає в мотивуванні учня до активізації роботи по засвоєнню навчального матеріалу. Для посилення навчальної функції тестування можуть бути використані додаткові заходи стимулювання студентів, такі як: роздача викладачем примірного переліку питань для самостійної підготовки, наявність в самому тесті навідних запитань і підказок, спільний розбір результатів тесту.
- Виховна функція проявляється в періодичності та неминучості тестового контролю. Це дисциплінує, організовує і направляє діяльність учнів, допомагає виявити і усунути прогалини в знаннях, формує прагнення розвинути свої здібності.

Переваги:

- Тестування є більш якісним і об'єктивним способом оцінювання, його об'єктивність досягається шляхом стандартизації процедури проведення, перевірки показників якості завдань і тестів цілком.
- Тестування - більш справедливий метод, воно ставить всіх учнів в рівні умови, як в процесі контролю, так і в процесі оцінки, практично виключаючи суб'єктивізм викладача. За даними Британської асоціації NEAB, що займається підсумковою атестацією учнів Великобританії, тестування дозволяє знизити кількість апеляцій більш ніж в три рази, зробити процедуру оцінювання однаковою для всіх учнів незалежно від місця проживання, типу і виду освітньої установи, в якому займаються учні.
- Тести це більш об'ємний інструмент, оскільки тестування може включати в себе завдання з усіх тем курсу, в той час як на усний іспит зазвичай виноситься 2-4 теми, а на письмовий - 3-5. Це дозволяє виявити знання учня по всьому курсу, виключивши елемент випадковості при витягуванні квитка. За допомогою тестування можна встановити рівень знань учня по предмету в цілому і по окремих його розділів.
- Тест це більш точний інструмент, так, наприклад, шкала оцінювання тесту з 20 питань, складається з 20 ділень, в той час, як звичайна шкала оцінки знань - тільки з чотирьох.
- Тестування більш ефективно з економічної точки зору. Основні витрати при тестуванні припадають на розробку якісного інструментарію, тобто мають разовий характер. Витрати ж на проведення тесту значно нижче, ніж при письмовому або усному контролі. Проведення тестування і контроль результатів в групі з 30 чоловік займає півтори-дві години, усний або письмовий іспит - не менше чотирьох годин.

- Тестування - це більш м'який інструмент, вони ставлять всіх учнів в рівні умови, використовуючи єдину процедуру і єдині критерії оцінки, що призводить до зниження передекзаменаційні нервових напружень.

Недоліки:

- Розробка якісного тестового інструментарію - тривалий, трудомісткий і дорогий процес. Стандартні набори тестів для більшості дисциплін ще не розроблені, а розроблені зазвичай мають дуже низьку якість.
- Дані, одержувані викладачем в результаті тестування, хоча і включають в себе інформацію про прогалини в знаннях по конкретних розділах, але не дозволяють судити про причини цих прогалин.
- Тест не дозволяє перевіряти і оцінювати високі, продуктивні рівні знань, пов'язані з творчістю, тобто імовірнісні, абстрактні і методологічні знання.
- Широта охоплення тем в тестуванні має і зворотну сторону. Учень при тестуванні, на відміну від усного або письмового іспиту, не має достатньо часу для скільки-небудь глибокого аналізу теми.
- Забезпечення об'єктивності і справедливості тесту вимагає прийняття спеціальних заходів щодо забезпечення конфіденційності тестових завдань. При повторному застосуванні тесту бажано внесення в завдання змін.
- У тестуванні присутній елемент випадковості. Наприклад, учень, що не відповів на просте питання, може дати правильну відповідь на більш складний. Причиною цього може бути, як випадкова помилка в першому питанні, так і вгадування відповіді у другому. Це спотворює результати тесту і призводить до необхідності врахування ймовірнісної складової при їх аналізі.

Висновки

Після написання даного розділу, в якому в свою чергу здійснено аналіз за літературними джерелами стосовно порушеного питання. Власне, уся зібрана інформація, а саме створення теоретичної бази стосовно технологій інтерактивного оцінювання знань, дала змогу оцінити ситуацію та сформувати базову картину та план, щодо створення інформаційної системи.

Таким чином, перший розділ бакалаврської роботи дав змогу зрозуміти та ознайомитись з наявною теоретичною інформацією стосовно функціонування інтерактивних оцінювань знань.

РОЗДІЛ 2. МОДЕЛЬ ІНТЕРАКТИВНОГО ДОДАТКУ ДЛЯ ПРОВЕДЕННЯ ТЕСТУВАННЯ ЗНАНЬ

2.1 Аналіз системи показників для проведення тестування знань

Розробленню будь-якого тестування знань слід приділити достатню кількість часу і ретельно обдумати його зміст. Існує декілька сучасних парадигм для побудови тестувань:

Аналіз та вибір тематики тестування

- Мету тестування слід переглянути декілька разів, щоб обрати інформацію, яка буде подаватися студентові.
- Подання завдань має бути чітким, без можливості дати декілька відповідей (якщо це не задумано форматом завдання).

Формат анкети в Інтернеті

- Запитання для тестування слід створювати у найбільш підходящому форматі, що полегшує розуміння.

Довжина анкети

- Як правило, тестування не повинно перевищувати 45 хвилин по довжині.
    - Як правило, 1 запитання з декількома варіантами займають 3 хвилини.
    - Одне питання з короткою відповіддю еквівалентно 3 питанням із множинним вибором.

Прототипування

- Перед початком використання завдань для тестування викладачеві слід пройти їх самостійно, або залучитися допомогою колег, аби зрозуміти, що на кожне питання можна дати однозначну правильну відповідь.
    - Важливо розрізнити, чи виникали у учасників труднощі з будь-яким із запитань.
    - Відгуки суб’єктів слід використовувати для внесення будь-яких необхідних змін до тесту.

2.2 Формування моделі проведення тестування знань

2.2.1 Відповіді

Зачасту, є велика кількість студентів, які пропускають деякі запитання або взагалі не беруть участі в тестуванні.

Спеціалісти стверджують, що наступні три фактори визначають успішність тестування та ймовірність досягнення гідних рівнів відповіді.

- Зрозумілість і однозначність відповідей
- Рівень підготовленості студентів
- Складність завдання

Існує принаймні сім способів реагування на тестування знань.

1. Повні відповіді - це ті студенти, які переглядають усі питання та відповідають на всі питання.
2. Невідповідачі підрозділів - це ті особи, які не беруть участі в опитуванні. Існує дві можливі варіації одиниці, яка не відповідає. Такій особі може бути технічно перешкоджено брати участь, або вона може цілеспрямовано

відмовитись після відображення екрана привітання, але до перегляду будь-яких питань.

3. Випадаючі, що кидають відповіді, складаються з осіб, які дають відповіді на ті питання, що відображаються, але звільняються перед тим, як заповнити опитування.

4. Луркери переглядають усі питання опитування, але не відповідають на жодне з них.

5. Залишаються ті, хто кидає навчання, являє собою комбінацію 3 і 4. Такий учасник переглядає деякі питання, не даючи відповіді, але також кидає опитування до того, як дійде до кінця.

6. Невідповідачі, які не відповідають на запитання, переглядають всю анкету, але лише відповідають на деякі запитання.

7. Відсутні елементи, які не відповідають, представляють собою суміш 3 і 6. Особи, що демонструють цю поведінку відповідей, переглядають деякі питання, відповідають на деякі, але не на всі переглянуті питання, а також кидають до закінчення опитування.

2.2.2 Якість тестування

Якість тестування можна виміряти за отриманими оцінками та відгуками студентів. Щоб підтримувати високу якість тестування, слід враховувати стислість та послідовність запитань. По-перше, довжина тестування повинна бути лише така, яка необхідна для однозначної оцінки знань. Лаконічності можна досягти шляхом видалення зайвих і нерелевантних питань, що може додати студенту розчарування, але не мати значення для перевірки. Нарешті,

розміщення запитань у логічній послідовності також дає учасникам кращу ментальну карту під час проходження тестування.

2.3 Інформаційно-логічна модель проведення онлайнового анкетування

2.3.1 Діаграма варіантів використання

Діаграма використання найпростіша - це представлення взаємодії користувача із системою, яка показує взаємозв'язок між користувачем та різними випадками використання, в яких користувач бере участь. Діаграма випадків використання може ідентифікувати різні типи користувачів системи та різні випадки використання, і часто вона супроводжується також іншими типами діаграм. Варіанти використання представлені кругами або еліпсами.

Незважаючи на те, що сам випадок використання може детально вивчити кожну можливість, діаграма прикладів використання може допомогти забезпечити огляд системи на більш високому рівні. Раніше вже було сказано, що "схеми використання - це принципи вашої системи".

Через їх спрощений характер, схеми використання можуть бути хорошим інструментом комунікації для зацікавлених сторін. Креслення намагаються імітувати реальний світ і дають зацікавленій стороні уявлення про те, як буде розроблена система. Сіау та Лі провели дослідження, щоб визначити, чи взагалі існувала дійсна ситуація для схем використання або вони були непотрібними. Було виявлено, що діаграми випадків використання передають намір системи більш спрощеним чином зацікавленим сторонам і що вони "інтерпретуються більш повно, ніж діаграми класів".

Метою діаграми використання є відображення динамічного аспекту системи. Додаткові схеми та документація можуть бути використані для забезпечення повного функціонального та технічного уявлення про систему.

Вони забезпечують спрощене та графічне представлення того, що система насправді повинна робити.

Елементи:

- рамки системи (англ. system border) - прямоугольник із назвою у верхніх частинах та еліпсами (прецедентами) всередині. Часто може бути опущено без корисної інформації про полезну інформацію,
- актор (англ. актор) - стилізований людський персонаж, обозначаючий набір ролей користувача (розуміється в широкому змісті: людина, внешняя сутність, клас, інша система), взаємодіючого з деякою сутністю (системною, підсистемою, класом). Актори не можуть бути пов'язані між собою з іншим (за вимкнення відносин щодо обробки / дослідження),
- прецедент - еліпс із надписом, що обозначає виконувану систематичну дію (може включати можливі варіанти), що призводить до спостерігаємих акторами результатів. Надпис може бути ім'ям або описом (з точки зору актора) того, "що" робить система (а не "як"). Ім прецедента зв’язано з неперервним (атомарним) сценарієм - конкретною послідовністю дій, ілюстріруючою повіддю. Під час сценарію актори обмінюються із систематичними повідомленнями. Сценарій може бути приведений на діаграмі прецедентів у відео UML-коментарі. З одним прецедентом може бути пов'язано кілька різних сценаріїв

Діаграму варіантів використання, яка описує можливі дії користувача в системі можна знайти в додатку Б.

2.3.2 Діаграма класів

У програмній інженерії діаграма класів в Уніфікованій мові моделювання (UML) - це тип статичної структурної діаграми, що описує структуру системи, показуючи класи системи, їх атрибути, операції (або методи) та взаємозв'язки між об'єктами.

Діаграма класів є основним будівельним елементом об’єктно-орієнтованого моделювання. Він використовується для загального концептуального моделювання структури програми та для детального моделювання переведення моделей у програмовий код. Діаграми класів також можуть бути використані для моделювання даних. Класи на діаграмі класів представляють як основні елементи, взаємодії в програмі, так і класи, що програмуються.

На схемі класи представлені вікнами, які містять три відділення:

- У верхньому відділенні міститься назва класу. Надруковано жирним шрифтом і відцентровано, а перша літера написана великими літерами.
- Середній відсік містить атрибути класу. Вони вирівняні за лівим краєм, а перша буква мала.
- У нижньому відділенні містяться операції, які може виконувати клас. Вони також вирівняні за лівим краєм, а перша буква - мала.

При проектуванні системи ряд класів ідентифікується та згруповується у схему класів, яка допомагає визначити статичні відносини між ними. При детальному моделюванні класи концептуального проекту часто поділяються на ряд підкласів.

Залежність - це семантичний зв’язок між залежними та незалежними елементами моделі. Він існує між двома елементами, якщо зміни у визначенні одного елемента (сервера або цілі) можуть спричинити зміни для іншого (клієнта або джерела). Ця асоціація є односпрямованою. Залежність

відображається у вигляді штрихової лінії з відкритою стрілкою, яка вказує від клієнта до постачальника.

Для подальшого опису поведінки систем ці діаграми класів можуть бути доповнені діаграмою стану або машиною стану UML.

Асоціація представляє родину посилань. Двійкова асоціація (з двома кінцями) зазвичай представляється у вигляді рядка. Асоціація може пов’язувати будь-яку кількість класів. Асоціація з трьома ланками називається потрійною асоціацією. Асоціацію можна назвати, а кінці асоціації можна прикрасити іменами ролей, показниками власності, кратністю, видимістю та іншими властивостями.

Існує чотири різні типи асоціацій: двонаправлена, односпрямована, агрегаційна (включає агрегацію композиції) та рефлексивна. Двонаправлені та односпрямовані асоціації є найбільш поширеними.

Наприклад, клас польоту асоціюється з класом літака двонаправлено. Асоціація представляє статичне відношення, яке ділиться між об'єктами двох класів.

Агрегація є варіантом взаємозв'язку "має"; агрегація є більш конкретною, ніж асоціація. Це асоціація, яка представляє частково цілі або часткові стосунки. Як показано на зображенні, професор "має" клас для викладання. Як тип асоціації, агрегація може бути названа та мати ті самі прикраси, що і асоціація. Однак агрегація не може включати більше двох класів; це має бути бінарна асоціація. Крім того, навряд чи існує різниця між агрегаціями та асоціаціями під час реалізації, і діаграма може взагалі пропустити відносини агрегування. [7]

Агрегація може відбуватися, коли клас є колекцією або контейнером інших класів, але вміщені класи не мають сильної залежності життєвого циклу від контейнера. Вміст контейнера все ще існує, коли контейнер знищений.

В UML він графічно представлений у вигляді порожнистої форми ромба на вміщуючому класі одним рядком, що зв’язує його із вміщеним класом. Сукупність - це семантично розширений об’єкт, який у багатьох операціях трактується як одиниця, хоча фізично він складається з декількох менших об’єктів.

Приклад: Бібліотека та студенти. Тут студент може існувати без бібліотеки, зв'язок між студентом і бібліотекою є агрегацією.

Це вказує на те, що один із двох пов’язаних класів (підклас) вважається спеціалізованою формою іншого (супер тип), а суперклас - узагальненням підкласу. На практиці це означає, що будь-який екземпляр підтипу є також екземпляром суперкласу. Зразкове дерево узагальнень цієї форми зустрічається в біологічній класифікації: людина - це підклас маймуни, який є підкласом ссавців тощо. Зв’язок найлегше зрозуміти за допомогою фрази „A - це B” (людина - це ссавець, ссавець - тварина).

Графічне представлення UML узагальнення - це форма порожнистого трикутника на кінці суперкласу рядка (або дерева рядків), що зв’язує його з одним або кількома підтипами.

Відносини узагальнення також відомі як спадщина або відносини "є".

Суперклас (базовий клас) у відносинах узагальнення також відомий як "батьківський", суперклас, базовий клас або базовий тип.

Підтип у відносинах спеціалізації також відомий як "дочірній", підклас, похідний клас, похідний тип, клас успадкування або тип успадкування.

Зверніть увагу, що ці стосунки нічим не схожі на біологічні стосунки батьків та дітей: використання цих термінів надзвичайно поширене, але може ввести в оману.

A - це тип B

Наприклад, "дуб - це тип дерева", "автомобіль - це тип транспортного засобу"

Узагальнення може бути показано лише на діаграмах класів та на діаграмах використання.

При моделюванні UML взаємозв'язок реалізації - це взаємозв'язок між двома елементами моделі, в яких один елемент моделі (клієнт) реалізує (реалізує або виконує) поведінку, яку вказує інший елемент моделі (постачальник).

Графічне представлення UML реалізації - це порожниста форма трикутника на кінці інтерфейсу штрихової лінії (або дерева рядків), яка з'єднує її з одним або кількома реалізаторами. Проста головка стрілки використовується на кінці інтерфейсу штрихової лінії, що з'єднує її з користувачами. У діаграмах компонентів використовується графічна умова «м'яч і сокет» (реалізатори виставляють кульку або льодяник, тоді як користувачі показують сокет). Реалізації можна показати лише на діаграмах класів або компонентів. Реалізація - це взаємозв'язок між класами, інтерфейсами, компонентами та пакетами, що з'єднує елемент клієнта з елементом постачальника. Зв'язок реалізації між класами / компонентами та інтерфейсами показує, що клас / компонент реалізує операції, пропоновані інтерфейсом.

Залежність - це слабша форма зв'язку, яка вказує на те, що один клас залежить від іншого, оскільки він використовує його в певний момент часу. Один клас залежить від іншого, якщо незалежний клас є змінною параметра або локальною змінною методу залежного класу. Це відрізняється від асоціації, де атрибут залежного класу є екземпляром незалежного класу. Іноді відносини між двома класами дуже слабкі. Вони взагалі не реалізовані зі змінними-членами. Швидше вони можуть бути реалізовані як аргументи функції-члена.

інший. Ці відносини зазвичай описуються як "А має В" (у матері-кота є кошенята, у кошенят - мати-кішка).

Представлення UML асоціації - це лінія, що з'єднує два пов'язані класи. На кожному кінці рядка є додаткові позначення. Наприклад, ми можемо вказати,

використовуючи наконечник стрілки, що загострений кінець видно з хвоста стрілки. Ми можемо вказати власність шляхом розміщення кульки, ролі, яку відіграють елементи цього кінця, вказавши ім'я ролі та множинність екземплярів цієї сутності (діапазон кількості об'єктів, які беруть участь в асоціації з точки зору іншого кінця).

Класи сутності моделюють довгоживучу інформацію, якою обробляє система, а іноді і поведінку, пов'язану з цією інформацією. Їх не слід ідентифікувати як таблиці баз даних чи інших сховищ даних.

Вони намальовані як кола з короткою лінією, прикріпленою до нижньої частини кола. Як варіант, їх можна намалювати як звичайні класи із позначенням стереотипу «сутність» над назвою класу.

Діаграму класів програмного продукту, яка описує його внутрішню будову, можна знайти в додатку В.

Висновки

Мета даного розділу полягала в розробці моделі інтерактивного оцінювання знань, формуванні системи показників і виявленні основних структурних систем інтерактивних оцінювань знань, їх складових частин та особливостей.

Модель і система показників, розроблені в даному розділі, в подальшому зіграють ключову роль в створенні власної інформаційної системи, адже стануть базовим фундаментом побудови системи і чітким вказанням щодо її структури.

Таким чином, другий розділ бакалаврської роботи дав змогу виробити підхід до розробки майбутньої системи і зрозуміти її загальну структуру.

РОЗДІЛ 3. РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ ПРОВЕДЕННЯ ОНЛАЙН АНКЕТУВАННЯ

3.1. Програмна реалізація проведення онлайнового анкетування

3.1.1. Мова програмування Python

Python (в російській мові зустрічаються назви пітон [16] або пайтон [17]) - високорівнева мова програмування загального призначення з динамічної строгою типізацією і автоматичним управлінням пам'яттю [18] [19], орієнтований на підвищення продуктивності розробника, читання коду і його якості, а також на забезпечення переносимості написаних на ньому програм [20]. Мова є повністю об'єктно-орієнтованим - все є об'єктами [18]. Незвичайною особливістю мови є виділення блоків коду пробільними відступами [21]. Синтаксис ядра мови мінімалістичний, за рахунок чого на практиці рідко виникає необхідність звертатися до документації [20]. Сам же мова відома як інтерпретується і використовується в тому числі для написання скриптів [18]. Недоліками мови є часто більш низька швидкість роботи і більш високе споживання пам'яті написаних на ньому програм в порівнянні з аналогічним кодом, написаним на компільованих мовах, таких як Сі або C ++ [20] [18].

Python є мультипарадигмальності мовою програмування, що підтримує імперативне, процедурне, структурний, об'єктно-орієнтоване програмування [18], метапрограмування [22] і функціональне програмування [18]. Завдання узагальненого програмування вирішуються за рахунок динамічної типізації [23] [24]. Аспектно-орієнтоване програмування частково підтримується через декоратори [25], більш повноцінна підтримка забезпечується додатковими фреймворками [26]. Такі методики як контрактне і логічне програмування

можна реалізувати за допомогою бібліотек або розширень [27]. Основні архітектурні риси - динамічна типізація, автоматичне керування пам'яттю [18], повна інтроспекція, механізм обробки виключень, підтримка багатопоточних обчислень з глобальної блокуванням інтерпретатора (GIL) [28], високорівневі структури даних. Підтримується розбиття програм на модулі, які, в свою чергу, можуть об'єднуватися в пакети [29].

Еталонної реалізацією Python є інтерпретатор CPython, що підтримує більшість активно використовуваних платформ [30] і є стандартом де-факто мови [31]. Він поширюється під вільною ліцензією Python Software Foundation License, що дозволяє використовувати його без обмежень в будь-яких додатках, включаючи пропрієтарні [32]. CPython компілює вихідні тексти в високорівнева байт-код, який виконується в стековой віртуальній машині [33]. До іншим трьом основним реалізацій мови відносяться Jython (для JVM), IronPython (для CLR / .NET) і PyPy [18] [34]. PyPy написаний на підмножині мови Python (RPython) і розроблявся як альтернатива CPython з метою підвищення швидкості виконання програм, в тому числі за рахунок використання JIT-компіляції [34]. Підтримка версії Python 2 закінчилася в 2020 році [35]. На поточний момент активно розвивається версія мови Python 3 [36]. Розробка мови ведеться через пропозиції щодо розширення мови PEP (англ. Python Enhancement Proposal), в яких описуються нововведення, робляться коригування відповідно до зворотного зв'язку від спільноти і документуються підсумкові рішення [37].

Стандартна бібліотека включає великий набір корисних переносяться функцій, починаючи від функціоналу для роботи з текстом і закінчуючи засобами для написання мережевих додатків. Додаткові можливості, такі як математичне моделювання, робота з обладнанням, написання веб-додатків або розробка ігор, можуть реалізовуватися за допомогою великого кількості сторонніх бібліотек, а також інтеграцією бібліотек, написаних на Сі або C ++, при цьому і сам інтерпретатор Python може інтегруватися в проекти, написані

на цих мовах [18]. Існує і спеціалізований репозиторій програмного забезпечення, написаного на Python, - PyPI [38]. Даний репозиторій надає кошти для простої установки пакетів в операційну систему і став стандартом де-факто для Python [39]. Станом на 2019 рік у ньому містилося понад 175 тисяч пакетів [38].

Python став одним з найпопулярніших мов, він використовується в аналізі даних, машинному навчанні, DevOps і веб-розробки, а також в інших сферах, включаючи розробку ігор. За рахунок читабельності, простого синтаксису і відсутності необхідності в компіляції мову добре підходить для навчання програмуванню, дозволяючи концентруватися на вивченні алгоритмів, концептів і парадигм. Налагодження ж і експериментування в значній мірі полегшуються тим фактом, що мова є інтерпретується [40] [18]. Застосовується мову багатьма великими компаніями, такими як Google або Facebook [18]. Станом на квітень 2021 року Python займає третє місце в рейтингу TIOBE популярності мов програмування з показником 11,03% [41]. «Мовою року» за версією TIOBE Python оголошувався в 2007, 2010, 2018 і 2020 году [42].

Задумка по реалізації мови з'явилася в кінці 1980-х років, а розробка його реалізації почалася в 1989 році співробітником голландського інституту CWI Гвідо ван Россум [37]. Для розподіленої операційної системи Amoeba потрібний розширюваний скриптова мова, і Гвідо почав розробляти Python на дозвіллі, запозичивши деякі напрацювання для мови ABC (Гвідо брав участь в розробці цієї мови, орієнтованого на навчання програмування). У лютому 1991 року Гвідо опублікував вихідний текст в групі новин alt.sources [43]. З самого початку Python проектувався як об'єктно-орієнтована мова.

Гвідо ван Россум назвав мову на честь популярного британського комедійного телешоу 1970-х «Літаючий цирк Монті Пайтона» [44], оскільки автор був прихильником цього телешоу, як і багато інших розробники того

часу, а в самому шоу простежувалася якась паралель зі світом комп'ютерної техніки [20].

Наявність дружелюбного, чуйного спільноти користувачів вважається, поряд з дизайнерської інтуїцією Гвідо, одним з факторів успіху Python. Розвиток мови відбувається згідно чітко регламентованим процесу створення, обговорення, відбору та реалізації документів PEP (англ. Python Enhancement Proposal) - пропозицій щодо розвитку Python [45].

3 грудня 2008 [46], після тривалого тестування, вийшла перша версія Python 3000 (або Python 3.0, також використовується скорочення Py3k). В Python 3000 усунені багато недоліків архітектури з максимально можливим (але не повним) збереженням сумісності зі старими версіями Python.

Дата закінчення терміну підтримки Python 2.7 спочатку була встановлена на 2015 рік, а потім перенесена на 2020 рік з побоювання, що велика частина існуючого коду не може бути легко перенесена на Python 3 [47] [48]. Підтримка Python 2 була спрямована лише на вже існуючі проекти, нові проекти повинні були використовувати Python 3 [36]. Більше ніяких виправлень безпеки або інших поліпшень для Python 2.7 НЕ буде випущено [35] [49]. Із закінченням терміну служби Python 2.x підтримуються тільки Python 3.6.x і більш пізні версії [50].

Мова використовує динамічну типізацію разом з підрахунком посилань і циклічний збирач сміття для менеджменту пам'яті [51]. Також є динамічні дозволу імен (динамічне зв'язування), які пов'язують імена методів і змінних під час виконання програми.

Python пропонує підтримку функціонального програмування в традиціях Лиспа. Так, в Python є функції filter, map і reduce; також з Лиспа були запозичені поняття характеристик списків, асоціативних масивів (словників), множин і генераторів списків [52]. Стандартна бібліотека містить два модулі (itertools і functools), що реалізують інструменти, запозичені з Haskell і Standard ML [53].

Розробники мови Python дотримуються певної філософії програмування, званої «The Zen of Python» («Дзен Пітона», або «Дзен Пайтона») [54]. Її текст видається інтерпретатором Python по команді import this (працює один раз за сесію). Автором цієї філософії вважається Тім Петерс (Tim Peters).

Замість того, щоб вбудувати в ядро Python всю функціональність мови, він був спроектований таким чином, щоб бути легко розширюваним. Це зробило мову популярним засобом додавання програмованих інтерфейсів до існуючих додатків. Бачення Гвідо Ван Россум маленького ядра з великою стандартної бібліотекою і легко розширюваним інтерпретатором виникало з негативного досвіду розробки мови ABC, який дотримувався протилежної підходу [56].

Python прагне до більш простому, менш громіздкому синтаксису і граматики, надаючи розробникам вибір в їх методології кодування. На відміну від девізу Perl «є кілька способів зробити це», Python дотримується філософії «повинен існувати один - і, бажано, тільки один - очевидний спосіб зробити це» [57]. Алекс Мартеллі [en], член Python Software Foundation, і автор книг по Python пише, що «Описувати щось як" розумне "не рахується компліментом в культурі Python») [58].

Розробники Python прагнуть уникнути передчасної оптимізації і відкидають патчі до некритичним частинам еталонної реалізації CPython, які могли б запропонувати незначне збільшення швидкості за рахунок зрозумілості коду [59]. Проте є способи підвищення продуктивності. Якщо в програмі є вузькі місця, пов'язані з виконанням ресурсоємних операцій на центральному процесорі, але не пов'язані з використанням операцій введення-виведення, то підвищити продуктивність можливо за рахунок трансляції програми за допомогою Cython в мову Сі і подальшої компіляції [60]. Вимогливі до обчислювальних ресурсів частині програми також можна переписувати на мову Сі і підключати як окремі бібліотеки з прив'язками до Python [34].

Важлива мета розробників Python - робити його забавним для використання. Це знайшло своє відображення в назві мови, яке він дав на честь Монті Пайтона [44]. Також це відображено в іноді грайливому підході до навчальних програм і довідкових матеріалів, таким як приклади програм з документацій, які використовують назви spam і eggs замість використовуються в документації безлічі інших мов foo і bar [61] [62].

Python портований і працює майже на всіх відомих платформах - від КПК до мейнфреймів. Існують порти під Microsoft Windows, практично під всі варіанти UNIX (включаючи FreeBSD і Linux), Android [63], Plan 9, Mac OS і macOS, iPhone OS (iOS) 2.0 і вище, iPadOS, Palm OS, OS / 2, Amiga , HaikuOS, AS / 400, OS / 390, Windows Mobile і Symbian.

У міру старіння платформи її підтримка в основній гілці мови припиняється. Наприклад, з версії 2.6 припинена підтримка Windows 95, Windows 98 і Windows ME [64]. У версії 3.5 перестала підтримуватися Windows XP [65] У версії 3.9 перестала підтримуватися Windows Vista і Windows 7 [66].

При цьому, на відміну від багатьох портіруемость систем, для всіх основних платформ Python має підтримку характерних для даної платформи технологій (наприклад, Microsoft COM / DCOM). Більш того, існує спеціальна версія Python для віртуальної машини Java - Jython, що дозволяє інтерпретатору виконуватися на будь-якій системі, що підтримує Java, при цьому класи Java можуть безпосередньо використовуватися з Python і навіть бути написаними на Python. Також кілька проектів забезпечують інтеграцію з платформою Microsoft.NET, основні з яких - IronPython і Python.Net.

Ім'я (ідентифікатор) може починатися з літери будь-якого алфавіту в Юникоде будь-якого регістра або підкреслення, після чого в імені можна використовувати і цифри. Як ім'я не можна використовувати ключові слова (їх список можна дізнатися по import keyword; print (keyword.kwlist)) і небажано

перевизначати вбудовані імена. Імена, що починаються з символу підкреслення, мають спеціальне значення [74].

У кожній точці програми інтерпретатор має доступ до трьох просторів імен (тобто відображенням імен в об'єкти): локальному, глобальному і вбудованому.

Області видимості імен можуть бути вкладеними один в одного (всередині обумовленою функції видно імена з навколишнього блоку коду). На практиці з областями видимості і зв'язуванням імен пов'язано кілька правил «хорошого тону», про яких можна докладніше дізнатися з документації.

Об'єктно-орієнтоване програмування

Дизайн мови Python побудований навколо об'єктно-орієнтованої моделі програмування. Реалізація ООП в Python є добре продуманою, але разом з тим досить специфічною в порівнянні з іншими об'єктно-орієнтованими мовами. У мові все є об'єктами - або екземплярами класів, або екземплярами метаклассом. Винятком є базовий вбудований метаклассом type. Таким чином, класи насправді є екземплярами метаклассом, а похідні метакласи є екземплярами метаклассом type. Метакласи є частиною концепції метапрограмування і надають можливість управління успадкуванням класів, що дозволяє створювати абстрактні класи, реєструвати класи або додавати в них будь-якої програмний інтерфейс в рамках бібліотеки або фреймворка [22].

Класи за своєю суттю представляють план або опис того, як створити об'єкт, і зберігають в собі опис атрибутів об'єкта і методів для роботи з ним. Парадигма ООП грунтується на інкапсуляції, успадкування та поліморфізму [80]. Інкапсуляція в Python представлена можливістю зберігання публічних і прихованих атрибутів (полів) в об'єкті з наданням методів для роботи з ними [80], при цьому насправді всі атрибути є публічними, але для позначки прихованих атрибутів існує угода про іменування [81]. Спадкування дозволяє створювати похідні об'єкти без необхідності повторного написання коду, а

поліморфізм полягає в можливості перевизначення будь-яких методів об'єкта (в Python все методи є віртуальними [81]), а також в перевантаженні методів і операторів. Перевантаження методів в Python реалізується за рахунок можливості виклику одного і того ж методу з різним набором аргументів [80]. Особливістю Python є можливість модифікувати класи після їх оголошення, додаючи в них нові атрибути і методи [36], також можна модифікувати і самі об'єкти, в результаті чого класи можуть використовуватися як структури для зберігання довільних даних [81].

В Python підтримується множинне спадкування. Само по собі множинне спадкування є складним, і його реалізації стикаються з проблемами вирішення колізій імен між батьківськими класами і з можливим повторним успадкуванням від одного і того ж класу в ієрархії. В Python методи викликаються згідно з порядком дозволу методів (MRO), який заснований на алгоритмі C3-лінеаризації [82], в звичайних випадках при написанні програм не потрібно знати принцип роботи даного алгоритму, розуміння ж може знадобитися при створенні нетривіальних ієрархій класів [83].

Можливості та особливості, специфічні для Python:

- Спеціальні методи, що керують життєвим циклом об'єкта: конструктори, деструктори.
- Перевантаження операторів (всіх, окрім is, '.', '=' I символьних логічних).
- Властивості (імітація поля за допомогою функцій).
- Управління доступом до полів (емуляція полів і методів, частковий доступ, і т. П.).
- Методи для управління найбільш поширеними операціями (истинностное значення, len (), глибоке копіювання, сериализация, ітерація по об'єкту, ...).
- Повна інтроспекція.
- Класові і статичні методи, класові поля.

- Класи, вкладені в функції і класи.
- Можливість модифікувати об'єкти під час виконання програми.

Узагальнене програмування

Мови з підтримкою динамічний типізації та об'єктно-орієнтованого програмування зазвичай не розглядаються в рамках узагальненого програмування, оскільки завдання узагальненого програмування вирішуються за рахунок відсутності обмежень на типи даних [23] [24]. В Python узагальнене програмування зі строгою типізацією досягається використанням засобів мови спільно з зовнішніми аналізаторами коду [84], такими як Муру [85].

функціональне програмування

Незважаючи на те, що Python спочатку не замислювався як мова функціонального програмування [86], Python підтримує програмування в стилі функціонального програмування, зокрема [87]:

- функція є об'єктом,
- функції вищих порядків,
- рекурсія,
- фокус на роботу зі списками,
- аналог замикань,
- часткове застосування функції за допомогою методу partial (),
- можливість реалізації інших засобів на самій мові (наприклад, каррінг).

Однак, на відміну від більшості мов, безпосередньо орієнтованих на функціональне програмування, Python не є чистою мовою програмування і код не захищений від побічних ефектів [87] [88].

У стандартній бібліотеці Python існують спеціальні пакети operator і functools для функціонального програмування [86].

Модулі та пакети

Програмне забезпечення (додаток або бібліотека) на Python оформлюється у вигляді модулів, які в свою чергу можуть бути зібрані в пакети. Модулі

можуть розташовуватися як в каталогах, так і в ZIP-архівах. Модулі можуть бути двох типів за своїм походженням: модулі, написані на «чистому» Python, і модулі розширення (extension modules), написані на інших мовах програмування. Наприклад, в стандартній бібліотеці є «чистий» модуль pickle і його аналог на Сі: cPickle. Модуль оформляється у вигляді окремого файлу, а пакет - у вигляді окремого каталогу. Підключення модуля до програми здійснюється оператором import. Після імпорту модуль представлений окремим об'єктом, що дає доступ до простору імен модуля. В ході виконання програми модуль можна перезавантажити функцією reload ().

Інтроспекція

Python підтримує повну интроспекцию часу виконання [90]. Це означає, що для будь-якого об'єкта можна отримати всю інформацію про його внутрішню структуру.

Застосування інтроспекції є важливою частиною того, що називають pythonic style, і широко застосовується в бібліотеках і фреймворками Python, таких як PyRO, PLY, Cherry, Django і ін., Значно заощаджуючи час використовує їх програміста.

Необхідні для інтроспекції дані зберігаються в спеціальних атрибутах. Так, наприклад, отримати всі атрибути більшості об'єктів можна зі спеціального атрибута - словника (або іншого об'єкта, який надає інтерфейс dict) __dict__

Ітератори

У програмах на Python широко використовуються ітератори. Цикл for може працювати як з послідовністю, так і з ітератором. Більшість колекцій надають ітератори, ітератори можуть також визначатися користувачем для власних об'єктів. Модуль itertools стандартної бібліотеки містить засоби роботи з ітераторами.

Генератори

Однією з цікавих можливостей мови є генератори - функції, що зберігають внутрішній стан: значення локальних змінних і поточну інструкцію (див. Також: співпрограми). Генератори можуть використовуватися як ітератори для структур даних і для ледачих обчислень. Наприклад: генератор чисел Фібоначчі.

При виклику генератора функція негайно повертає об'єкт-ітератор, який зберігає поточну точку виконання і стан локальних змінних функції. При запиті наступного значення (за допомогою методу next (), неявно викликається в циклі for) генератор продовжує виконання функції від попередньої точки зупинки до наступного оператора yield або return.

В Python 2.4 з'явилися генераторні вирази - вирази, що дають в результаті генератор.

Стандартна бібліотека

Багата стандартна бібліотека є однією з привабливих сторін Python. Тут є засоби для роботи з багатьма мережевими протоколами і форматами Інтернету, наприклад, модулі для написання HTTP-серверів і клієнтів, для розбору і створення поштових повідомлень, для роботи з XML і т. П. Набір модулів для роботи з операційною системою дозволяє писати крос -платформенние додатки. Існують модулі для роботи з регулярними виразами, текстовими кодуваннями, мультимедійними форматами, криптографічними протоколами, архівами, сериализации даних, підтримка юніт-тестування та ін.

Модулі розширення та програмні інтерфейси

Крім стандартної бібліотеки існує безліч бібліотек, які надають інтерфейс до всіх системних викликів на різних платформах; зокрема, на платформі Win32 підтримуються всі виклики Win32 API, а також COM в обсязі не меншому, ніж у Visual Basic або Delphi. Кількість прикладних бібліотек для Python в самих різних областях без перебільшення великий за обсягом (веб, бази даних,

обробка зображень, обробка тексту, чисельні методи, додатки операційної системи і т. Д.).

Для Python прийнята специфікація програмного інтерфейсу до баз даних DB-API 2 та розроблені відповідні цієї специфікації пакети для доступу до різних СУБД: Oracle, MySQL, PostgreSQL, Sybase, Firebird (Interbase), Informix, Microsoft SQL Server і SQLite. На платформі Windows доступ до БД можливий через ADO (ADOdb). Комерційний пакет mxODBC для доступу до СУБД через ODBC для платформ Windows і UNIX розроблений eGenix [96]. Для Python написано багато ORM (SQLObject, SQLAlchemy, Dejavu, Django), виконані програмні каркаси для розробки веб-додатків (Django, Pylons, Pyramid).

Бібліотека NumPy для роботи з багатовимірними масивами дозволяє іноді досягти продуктивності наукових розрахунків, порівнянної зі спеціалізованими пакетами. SciPy використовує NumPy і надає доступ до великого спектру математичних алгоритмів (матрична алгебра - BLAS рівнів 1-3, LAPACK, БПФ ...). Numarray [97] спеціально розроблений для операцій з великими обсягами наукових даних.

WSGI [98] - інтерфейс шлюзу з веб-сервером (Python Web Server Gateway Interface).

Вихоядчи зі сфери використання і величезного функціоналу мови Python, який був описаний вище, можна зробити висновок про його мультифунціональність. Саме через це дана мова програмування була обрана для виконання даної роботи.

3.1.2 Фреймворк Django

Django - це безкоштовний веб-фреймворк на основі Python, що відповідає архітектурному зразку model-template-views (MTV). [9] [10] Він підтримується Django Software Foundation (DSF), американською незалежною організацією, створеною як некомерційна організація 501 (c) (3).

Основна мета Django - полегшити створення складних веб-сайтів, керованих базами даних. Структура підкреслює багаторазовість та "підключеність" компонентів, менше коду, низьку зв'язок, швидкий розвиток та принцип "не повторюватися". [11] Python використовується всюди, навіть для налаштувань, файлів та моделей даних. Django також надає додатковий адміністративний інтерфейс створення, читання, оновлення та видалення, який генерується динамічно за допомогою самоаналізу та налаштовується за допомогою моделей адміністратора.

Деякі добре відомі сайти, які використовують Django, включають PBS, [12] Instagram, [13] Mozilla, [14] The Washington Times, [15] Disqus, [16] Bitbucket, [17] і Nextdoor. [18]

3.1.3 Розробка графічного інтерфейсу користувача. Початок розробки додатку тестування знаннь був проведений за допомогою розробки програмного коду, що зобразимо нижче:

```
«««from django.contrib import admin
from .models import questionnaireAnswers, questionnaireData
admin.site.register(questionnaireAnswers)
admin.site.register(questionnaireData)
from django.apps import AppConfig

class MainappConfig(AppConfig):
    name = 'mainApp'
    verbose_name = 'Опросник'
from django.forms import ModelForm
from django.contrib.auth.forms import UserCreationForm
from django import forms
from  django.contrib.auth.models import User

class CreateUserForm(UserCreationForm):
    class Meta:
        model = User
        fields = ['username', 'email', 'password1', 'password2']
from django.db import models
from django.contrib.auth.models import User

class questionnaireData(models.Model):
    questionnaire_name = models.TextField('Название опросника')
    questions_text = models.TextField('Вопросы')
    questions_answers = models.TextField('Ответы')

    def __str__(self):
        return self.questionnaire_name
```

```
    class Meta:
        verbose_name = 'Опрос'
        verbose_name_plural = 'Опросы'


class questionnaireAnswers(models.Model):
    questionnaire_name_fk = models.ForeignKey(questionnaireData, on_delete=models.CASCADE)
    user = models.ForeignKey(User, on_delete=models.CASCADE)
    answers = models.TextField('Ответы')

    def __str__(self):
        return self.user.username + "/" + self.questionnaire_name_fk.questionnaire_name

    class Meta:
        verbose_name = 'Ответ'
        verbose_name_plural = 'Ответы'
from django.urls import path

from . import views

urlpatterns = [
    path('', views.index, name='index'),
    path('register/', views.regist, name='register'),
    path('login/', views.loginPage, name='login'),
    path('logout/', views.logoutUser, name='logout'),
    path('questionnaire_page/<str:name>', views.questionnaire_page, name='questionnaire_page'),
from django.shortcuts import render, redirect
from django.contrib.auth import authenticate, login, logout
from django.contrib import messages
from .forms import CreateUserForm
from .models import questionnaireAnswers, questionnaireData
from django.contrib.auth.models import User
def loginPage(request):
    if request.method == 'POST':
        username = request.POST.get('username')
        password = request.POST.get('password')
        user = authenticate(request, username=username, password=password)

        if user is not None:
            login(request, user)
            return redirect('index')
        else:
            messages.info(request, 'Имя пользователя или пароль введены неверно')
    context = {}
    return render(request, 'login.html', context)


def regist(request):
    form = CreateUserForm
    if request.method == "POST":
        form = CreateUserForm(request.POST)
        if form.is_valid():
            form.save()
            return redirect('login')
    context = {'form': form}
    return render(request, 'register.html', context)


def logoutUser(request):
```

```
    logout(request)
    return redirect('login')»»
```

Після цього при початку запуску додатку необхідно зробити клік на запуск і з’являється вспливаюче вікно на рис.3.1.

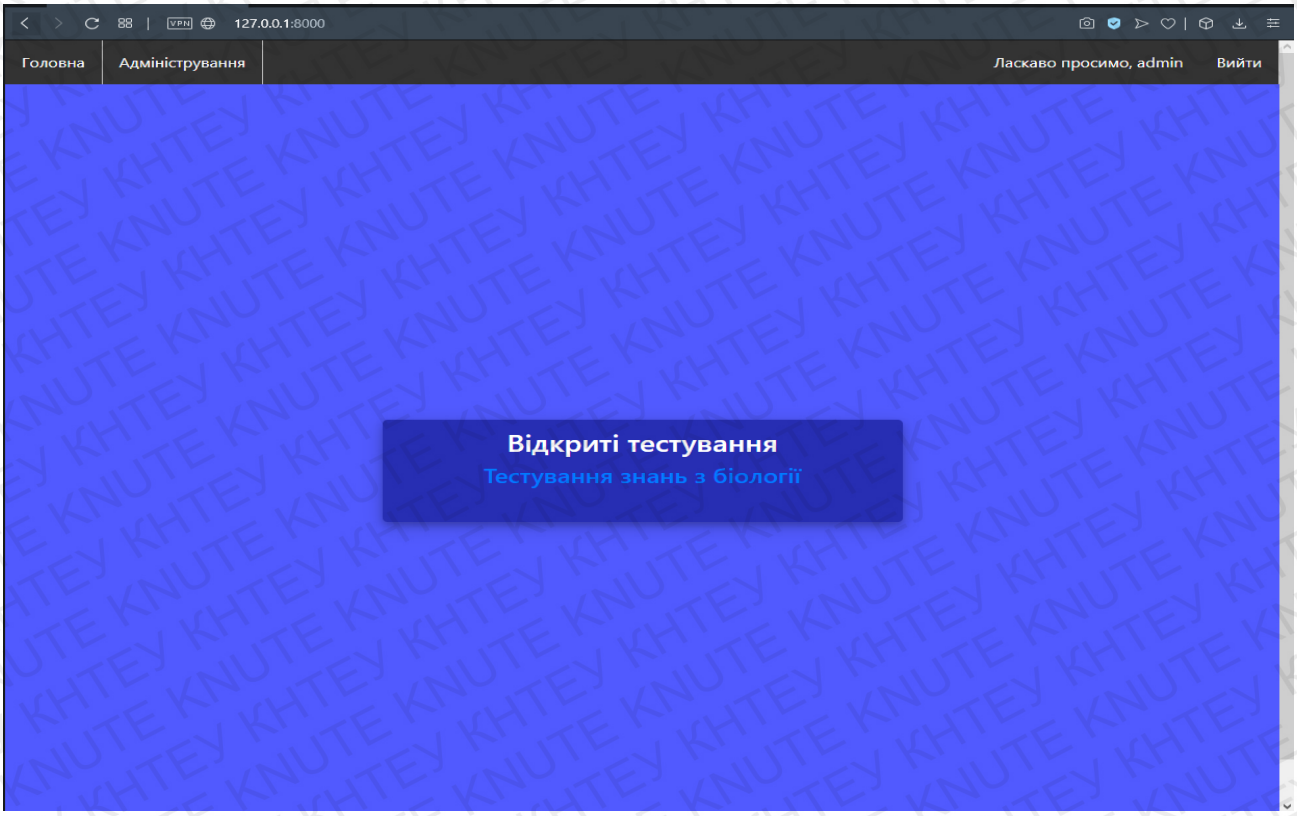

Рис. 3.1 Сторінка активації розробленого додатку тестування знань

Тепер у рис.3.2 відобразимо сторінку проведення опитування знань школярів з використанням коду:

```
def index(request):
    if request.user.is_authenticated:
        questionnaire_list = questionnaireData.objects.all()
        return render(request, 'index.html', {'questionnaire_list': questionnaire_list})
    else:
        return redirect('login')


def questionnaire_page(request, name):
    questionnaireCurr = questionnaireData.objects.get(questionnaire_name=name)
    questName = questionnaireCurr.questionnaire_name
    questions = questionnaireCurr.questions_text.split('/')
    tempAns = questionnaireCurr.questions_answers.split('/')
    tempQA = []
    for i in range(len(questions)):
        tempQA.append(questions[i] + '*' + tempAns[i])
```

```
    quest_ans = []
    for i in range(len(tempQA)):
        quest_ans.append(tempQA[i].split('*'))

    answers = []
    if request.method == 'POST':
        for i in range(1, len(quest_ans) + 1):
            answers.append(request.POST.get('quest' + str(i)))
        addAnswersToDB(answers, questName, request)
        return redirect('index')
    return render(request, 'questionnaire_page.html', {'questName': questName, 'quest_ans': quest_ans})


def addAnswersToDB(answers, name, req):
    quest = questionnaireData.objects.get(questionnaire_name=name)
    strAnsw = ""
    for str in answers:
        strAnsw += str + "\n"
    answ = questionnaireAnswers(questionnaire_name_fk=quest, user=req.user, answers=strAnsw)
    answ.save()
<!DOCTYPE html>
<html lang = "ru">
  <head>
          <meta charset="utf-8">

    <title>
          Тестирования
    </title>
```

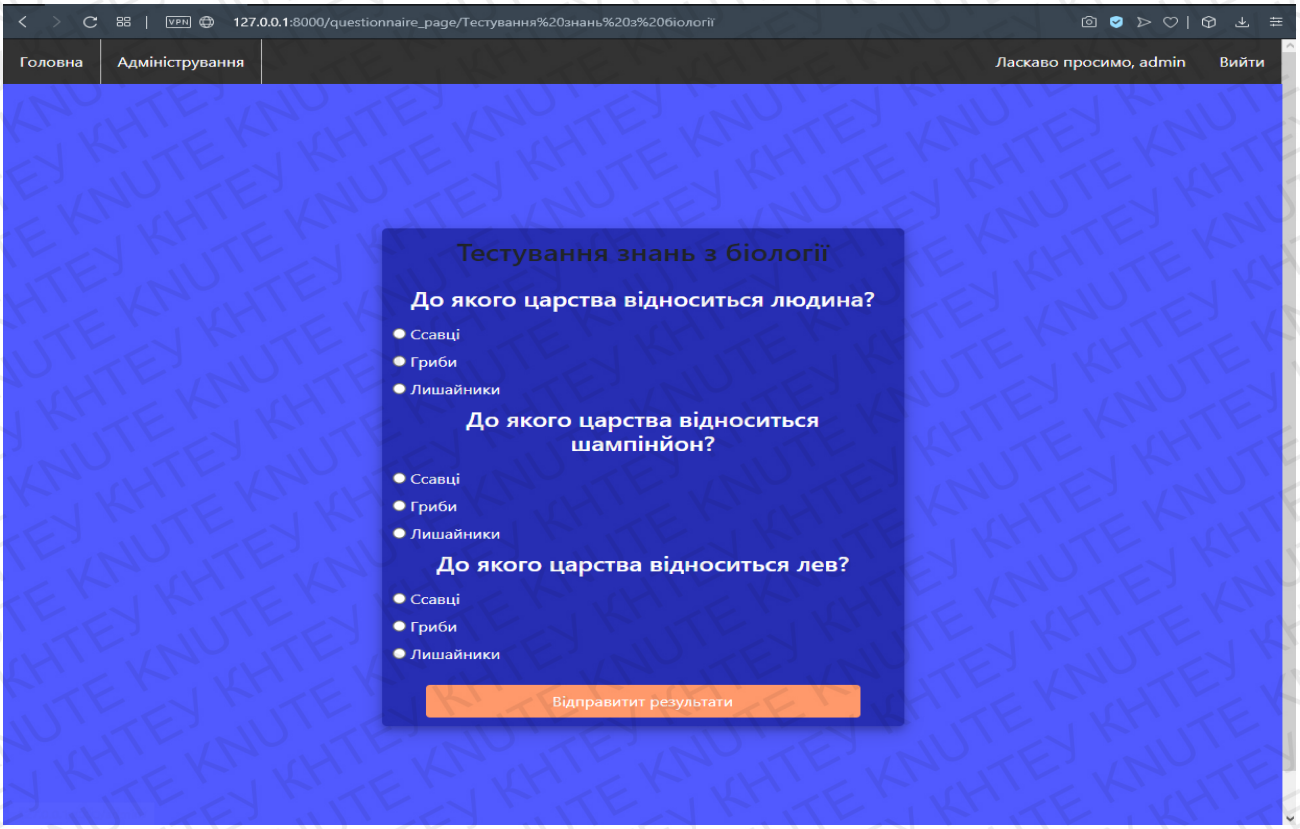

Рис. 3.2 Тестування знань студентів

ВИСНОВКИ

В ході проведеного дослідження на тему розробки інтерактивного додатку тестування знань було виявлено проблематику недостатньої кількості зручного програмного забезпечення для школярів і студентів з метою перевірки знань. Поставлен мета роботи була виконана з допомогою впровадження інтерактивного додатку знань із випливаючими функціональними блоками при авторизації з використанням мови програмування.

У першому розділі випускної кваліфікаційної роботи було узагальнено визначення тестування знань, що характеризується як інструментарій, щодо визначення поглибленості знань школярів та студентів за допомогою системи тестових завдань, стандартизованої процедури проведення, обробки та аналізу результатів. Було сформовано класифікацію проведення знань за наступними ознаками:

- за програмними цілями - інформаційні, діагностичні, навчальні, мотиваційні, атестаційні;
- за процедурою створення - стандартизовані, що не стандартизовані;
- за способом формування завдань - детерміновані, стохастичні, динамічні;
- за технологією проведення - паперові, в тому числі паперові з використанням оптичного розпізнавання, натурні, з використанням спеціальної апаратури, комп'ютерні;
- за формою завдань - закритого типу, відкритого типу, встановлення відповідності, упорядкування послідовності;
- по наявності зворотного зв'язку - традиційні і адаптивні

У розділі 2 було акцентовано увагу на:

Аналізі та виборі тематики тестування

Форматі анкети в Інтернеті

Довжині анкети

У розділі 3 було сформовано перелік рекомендацій і розробку інтерактивного додатку знань для студентів за рахунок програмного забезпечення. Апробація розробленого сайту допоможе покращити освітній процес в Україні. Функціонал роботи додатку тестування знань полягав у:

Використанні діаграми класів використання для користувача, що складається з послідовного схематичного алгоритму.

Після цього сформувалась діаграма класів із використанням алгоритму MainApp.

Апробація інтерактивного додатку тестування знань розпочинається із випливаючого вікна авторизації, де необхідно зареєструватись, або автологін.

Під час кліку на кнопку реєстрації вспливаюче вікно складається з чотирьох блоків для заповнення даних.

Після цього у випливаючому вікні зображено приклад тестувань, що були сформовані під час розробки, до прикладу візуально присутня графа тестування знань по біології.

Результат роботи інтерактивного додатку тестування знань зображено у вигляді рамки із випливаючим вікном запрограмованої дії, у якій надані варіанти відповіді і умови тестування знань.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. https://docs.python.org/3/license.html
2. Python 3.8.10, 3.9.5, and 3.10.0b1 are now available (англ.) — 2021.
3. Historique et licence
4. https://docs.python.org/3/library/py_compile.html
5. https://docs.python.org/3/faq/windows.html#is-a-pyd-file-the-same-as-a-dll
6. https://www.python.org/dev/peps/pep-0488/
7. https://docs.python.org/3/using/windows.html
8. https://docs.python.org/3/library/zipapp.html
9. https://www.python.org/dev/peps/pep-0484/
10. https://impythonist.wordpress.com/2014/02/16/open-heart-with-guido-van-rosuuma-lost-interview-of-python-creator-part2/
11. Why was Python created in the first place? — Python Software Foundation.
12. Classes The Python Tutorial — Python Software Foundation.
13. An Introduction to Python for UNIX/C Programmers
14. Functional Programming HOWTO
15. https://www.python.org/downloads/
16. Мария «Mifrill» Нефедова, Создатели языков программирования: Они такие разные, но кодинг их объединяет, Хакер № 09/08 (117). Дата обращения: 1 декабря 2012. Архивировано 2 июля 2013 года.
17. Прохоренок Н., Дронов В. Введение // Python 3. Самое необходимое, 2-е изд.. — БХВ-Петербург, 2019. — С. 11. — 608 с. — ISBN 9785977539944.
18. Yogesh Rana. Python: Simple though an Important Programming language (англ.) // International Research Journal of Engineering and Technology (IRJET). — 2019. — 2 February (vol. 06, iss. 2). — P. 1856—1858. — ISSN 2395-0056. Архивировано 11 февраля 2021 года.

19. SkipMontanaro. Why is Python a dynamic language and also a strongly typed language - Python Wiki (англ.). wiki.python.org (24 February 2012). Дата обращения: 14 марта 2021. Архивировано 14 марта 2021 года.
20. Mark Lutz. A Python Q&A Session (англ.). Learning Python, 3rd Edition [Book]. O'Reilly Media, Inc. (2007). Дата обращения: 11 февраля 2021. Архивировано 8 февраля 2021 года.
21. Python Introduction | (англ.). Python Education. Google Developers (20 August 2018). Дата обращения: 21 февраля 2021. Архивировано 4 декабря 2020 года.
22. Satwik Kansal. Metaprogramming in Python (англ.). IBM (5 April 2018). Дата обращения: 14 апреля 2021. Архивировано 27 февраля 2021 года.
23. Alexandre Bergel, Lorenzo Bettini. Generic Programming in Pharo (англ.) // Software and Data Technologies / José Cordeiro, Slimane Hammoudi, Marten van Sinderen. — Berlin, Heidelberg: Springer, 2013. — P. 66–79. — ISBN 978-3-642-45404-2. — doi:10.1007/978-3-642-45404-2_5. Архивировано 13 февраля 2021 года.
24. R. Peschke, K. Nishimura, G. Varner. ARGG-HDL: A High Level Python BasedObject-Oriented HDL Framework (англ.) // IEEE Transactions on Nuclear Science : pre-print. — 2020. — October. — arXiv:011.02626v1.
25. Steven F. Lott. Aspect-oriented programming (англ.). Mastering Object-Oriented Python - Second Edition. Packt Publishing (2019). Дата обращения: 21 февраля 2021. Архивировано 21 февраля 2021 года.
26. Arne Bachmann, Henning Bergmeyer, Andreas Schreiber. Evaluation of aspect-oriented frameworks in Python for extending a project with provenance documentation features (англ.) // The Python Papers. — 2011. — Vol. 6, iss. 3. — P. 1–18. — ISSN 1834-3147. Архивировано 22 апреля 2018 года.
27. Steven Cooper. Data Science from Scratch: The #1 Data Science Guide for Everything A Data Scientist Needs to Know: Python, Linear Algebra, Statistics,

Coding, Applications, Neural Networks, and Decision Trees. — Roland Bind, 2018. — 126 с.

28. Reuven M. Lerner. Multiprocessing in Python (англ.). Linux Journal (16 April 2018). Дата обращения: 14 февраля 2021. Архивировано 14 февраля 2021 года.
29. David Beazley, Brian K. Jones. 10. Modules and Packages - Python Cookbook, 3rd Edition [Book] (англ.). Python Cookbook, 3rd Edition. O'Reilly Media, Inc. (2013). Дата обращения: 21 февраля 2021. Архивировано 21 февраля 2021 года.
30. About Python. Дата обращения: 7 августа 2007. Архивировано 11 августа 2007 года.
31. PythonImplementations - Python Wiki (англ.). wiki.python.org (21 July 2020). Дата обращения: 17 февраля 2021. Архивировано 11 ноября 2020 года.
32. History and License (англ.). Python. Дата обращения: 21 мая 2021. Архивировано 5 декабря 2016 года.
33. Mostafa Chandra Krintz, C. Cascaval, D. Edelsohn, P. Nagpurkar, P. Wu. Understanding the Potential of Interpreter-based Optimizations for Python (англ.) // UCSB Technical Report. — 2010. — 11 August. Архивировано 23 февраля 2021 года.
34. J. Akeret, L. Gamper, A. Amara, A. Refregier. HOPE: A Python just-in-time compiler for astrophysical computations (англ.) // Astronomy and Computing. — 2015. — 1 April (vol. 10). — P. 1–8. — ISSN 2213-1337. — doi:10.1016/j.ascom.2014.12.001. — arXiv:1410.4345v2. Архивировано 15 февраля 2021 года.
35. PEP 373 -- Python 2.7 Release Schedule (англ.) (23 March 2014). Дата обращения: 7 марта 2021. Архивировано 25 февраля 2021 года.
36. Berk Ekmekci, Charles E. McAnany, Cameron Mura. An Introduction to Programming for Bioscientists: A Python-Based Primer (англ.) // PLOS

Computational Biology. — 2016. — 6 July (vol. 12, iss. 6). — P. e1004867. — ISSN 1553-7358. — doi:10.1371/journal.pcbi.1004867. — PMID 27271528. Архивировано 16 февраля 2021 года.

37. Kalyani Adawadkar. Python Programming - Applications and Future (англ.) // International Journal of Advance Engineering and Research Development. — 2017. — April (iss. SIEICON-2017). — P. 1—4. — ISSN 2348-447. Архивировано 15 июля 2020 года.
38. Ethan Bommarito, Michael James Bommarito. An Empirical Analysis of the Python Package Index (PyPI) (англ.) // Social Science Research Network. — Rochester, NY: Social Science Research Network, 2019. — 25 July. — doi:10.2139/ssrn.3426281. — arXiv:arXiv:1907.11073v2.
39. Pratik Desai. Python Programming for Arduino. — Packt Publishing Ltd, 2015. — С. 8. — 400 с. — ISBN 978-1-78328-594-5.
40. Sebastian Bassi. A Primer on Python for Life Science Researchers (англ.) // PLOS Computational Biology. — 2007. — 30 November (vol. 3, iss. 11). — P. e199. — ISSN 1553-7358. — doi:10.1371/journal.pcbi.0030199. Архивировано 13 марта 2021 года.
41. TIOBE Index (англ.). tiobe.com. TIOBE - The Software Quality Company. Дата обращения: 5 апреля 2021. Архивировано 6 апреля 2021 года.
42. Python | TIOBE - The Software Quality Company. www.tiobe.com. Дата обращения: 13 февраля 2021. Архивировано 6 февраля 2021 года.
43. Архивированная копия (недоступная ссылка). Дата обращения: 1 июня 2009. Архивировано 17 февраля 2016 года.
44. General Python FAQ. Python v2.7.3 documentation. Docs.python.org. Дата обращения: 4 июня 2020. Архивировано 24 октября 2012 года.
45. Index of Python Enhancement Proposals (PEPs). Дата обращения: 28 января 2007. Архивировано 28 января 2007 года.

46. Python 3.0 Release. Дата обращения: 1 июня 2009. Архивировано 2 июня 2009 года.
47. PEP 373 -- Python 2.7 Release Schedule. python.org. Дата обращения: 9 января 2017. Архивировано 19 мая 2020 года.
48. PEP 466 -- Network Security Enhancements for Python 2.7.x. python.org. Дата обращения: 9 января 2017. Архивировано 4 июня 2020 года.
49. Sunsetting Python 2 (англ.). Python.org. Дата обращения: 22 сентября 2019. Архивировано 12 января 2020 года.
50. Python Developer's Guide — Python Developer's Guide. devguide.python.org. Дата обращения: 17 декабря 2019. Архивировано 9 ноября 2020 года.
51. Extending and Embedding the Python Interpreter: Reference Counts (англ.). Docs.python.org. — «Since Python makes heavy use of malloc() and free(), it needs a strategy to avoid memory leaks as well as the use of freed memory. The chosen method is called reference counting.». Дата обращения: 5 июня 2020. Архивировано 18 октября 2012 года.
52. Hettinger, Raymond PEP 289 – Generator Expressions. Python Enhancement Proposals. Python Software Foundation (30 January 2002). Дата обращения: 19 февраля 2012. Архивировано 14 июня 2020 года.
53. 6.5 itertools – Functions creating iterators for efficient looping. Docs.python.org. Дата обращения: 22 ноября 2016. Архивировано 14 июня 2020 года.
54. PEP 20 — The Zen of Python. Дата обращения: 23 сентября 2005. Архивировано 17 июля 2005 года.
55. Бейдер Дэн. Чистый Python. Тонкости программирования для профи. — "Издательский дом ""Питер""", 2018. — С. 64—65. — 288 с. — ISBN 978-5-4461-0803-9.

56. Venners, Bill The Making of Python. Artima Developer. Artima (13 January 2003). Дата обращения: 22 марта 2007. Архивировано 1 сентября 2016 года.
57. Peters, Tim PEP 20 – The Zen of Python. Python Enhancement Proposals. Python Software Foundation (19 August 2004). Дата обращения: 24 ноября 2008. Архивировано 26 декабря 2018 года.
58. Alex Martelli, Anna Ravenscroft, David Ascher. Python Cookbook, 2nd Edition. — O'Reilly Media, 2005. — P. 230. — ISBN 978-0-596-00797-3. Архивная копия от 23 февраля 2020 на Wayback Machine
59. Python Culture (недоступная ссылка). ebeab (January 21, 2014). Архивировано 30 января 2014 года.
60. Mark Summerfield. Python in Practice: Create Better Programs Using Concurrency, Libraries, and Patterns. — Addison-Wesley, 2013-08-20. — C. 201. — 326 с. — ISBN 978-0-13-337323-3.
61. 15 Ways Python Is a Powerful Force on the Web (недоступная ссылка). Дата обращения: 28 декабря 2020. Архивировано 11 мая 2019 года.
62. 8.18. pprint — Data pretty printer — Python 3.8.3 documentation. docs.python.org. Дата обращения: 28 декабря 2020. Архивировано 22 января 2021 года.
63. Python on Android (англ.) (недоступная ссылка). www.damonkohler.com. Дата обращения: 19 декабря 2009. Архивировано 28 января 2011 года.
64. Port-Specific Changes: Windows (англ.) (недоступная ссылка). Python v2.6.1 documentation. What’s New in Python 2.6. Python Software Foundation. Дата обращения: 11 декабря 2008. Архивировано 28 января 2011 года.
65. Using Python on Windows — Python 3.5.9 documentation (англ.). Python Documentation. Python Software Foundation. Дата обращения: 8 июня 2020. Архивировано 15 октября 2020 года.

66. Drop support of Windows Vista and 7 in Python 3.9 (англ.). Дата обращения: 10 января 2021. Архивировано 4 ноября 2020 года.
67. 15. Floating Point Arithmetic: Issues and Limitations — Python 3.8.3 documentation. docs.python.org. — «Almost all machines today (November 2000) use IEEE-754 floating point arithmetic, and almost all platforms map Python floats to IEEE-754 “double precision”.». Дата обращения: 6 июня 2020. Архивировано 6 июня 2020 года.
68. Moshe Zadka, Guido van Rossum. PEP 237 – Unifying Long Integers and Integers. Python Enhancement Proposals. Python Software Foundation (11 March 2001). Дата обращения: 24 сентября 2011. Архивировано 28 мая 2020 года.
69. Built-in Types. Дата обращения: 3 октября 2019. Архивировано 14 июня 2020 года.
70. Рамальо, 2016, pp. 52—54.
71. Fredrik Lundh. Call By Object (англ.) (недоступная ссылка). effbot.org. Дата обращения: 31 мая 2014. Архивировано 23 ноября 2019 года.
72. Ошибка в сносках?: Неверный тег <ref>; для сносок foreword не указан текст
73. 2.3.2. Reserved classes of identifiers. Python documentation (18 октября 2009). Архивировано 28 января 2011 года.
74. …целостность больших проектов на Python строится на двух вещах: тесты и doc-строка. Дата обращения: 31 октября 2008. Архивировано 21 октября 2008 года.
75. Steve D. Jost. Structured Programming Details. IT 211, DePaul University (2019). Дата обращения: 17 февраля 2021. Архивировано 29 апреля 2020 года.

76. PyDBC: method preconditions, method postconditions and class invariants for Python. Дата обращения: 24 сентября 2011. Архивировано 23 ноября 2019 года.
77. Contracts for Python. Дата обращения: 24 сентября 2011. Архивировано 15 июня 2020 года.
78. PyDatalog. Дата обращения: 22 июля 2012. Архивировано 13 июня 2020 года.
79. Object-oriented programming in Python (англ.). IBM Developer. ibm.com (20 October 2020). Дата обращения: 11 марта 2021. Архивировано 11 марта 2021 года.
8. Classes (англ.). Python 3.9.2 documentation. docs.python.org. Дата обращения: 14 марта 2021. Архивировано 14 марта 2021 года.
80. Fawzi Albalooshi, Amjad Mahmood. A Comparative Study on the Effect of Multiple Inheritance Mechanism in Java, C++, and Python on Complexity and Reusability of Code (англ.) // International Journal of Advanced Computer Science and Applications (IJACSA). — 2017. — Vol. 8, iss. 6. — ISSN 2156-5570. — doi:10.14569/IJACSA.2017.080614. Архивировано 10 июля 2020 года.
81. Michele Simionato. The Python 2.3 Method Resolution Order (англ.). Python.org. Дата обращения: 14 марта 2021. Архивировано 14 марта 2021 года.
82. PEP 484 -- Type Hints (англ.). Python.org (24 September 2014). Дата обращения: 13 февраля 2021. Архивировано 9 февраля 2021 года.
83. Jukka Lehtosalo. Generics (англ.). Mypy 0.800 documentation. Read the Docs (2016). Дата обращения: 13 февраля 2021. Архивировано 13 февраля 2021 года.
84. Рамальо, 2016, pp. 188—191.

85. David Mertz. Functional Programming in Python. — O'Reilly, 2015. — ISBN 978-1491928561.
86. Рамальо, 2016, p. 273.
87. Рамальо, 2016, pp. 613—708.
88. Патрик О'Брайен. Руководство по интроспекции на Python / Intersoft Lab.
89. Beazley, 2009, pp. 222—225.
90. Рамальо, 2016, pp. 214—246.
91. Рамальо, 2016, pp. 686—688.
92. 6.2. re — Regular expression operations — Python 3.5.1 documentation. Дата обращения: 11 мая 2016. Архивировано 18 июля 2018 года.
93. A.M. Kuchling. PEP 206 -- Python Advanced Library, Python.org (14.07.2000). Архивировано 5 мая 2021 года. Дата обращения 4 апреля 2021.
94. eGenix.com — Professional Python Software, Skills and Services. Дата обращения: 29 января 2007. Архивировано 28 января 2007 года.
95. numarray Home Page. Дата обращения: 5 февраля 2007. Архивировано 9 июня 2021 года.
96. PEP333. Дата обращения: 29 января 2007. Архивировано 9 июня 2021 года.
97. Pyste Documentation

ДОДАТКИ

Додаток А.

Діаграма використання програмного продукту