МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
КРИВОРІЗЬКИЙ ДЕРЖАВНИЙ ПЕДАГОГІЧНИЙ УНІВЕРСИТЕТ

Фізико-математичний факультет

Кафедра інформатики та прикладної математики

«Допущено до захисту»
Завідувач кафедри
_____ _____.
(підпис) (прізвище, ініціали)
«___» _____ 2024 р.

Реєстраційний № _____
«___» _____ 2024 р.

ПЕРСОНАЛІЗОВАНЕ НАВЧАННЯ ЛІЦЕЇСТІВ ОСНОВ ПРОГРАМУВАННЯ

Кваліфікаційна робота студента групи Ім-23
ступінь вищої освіти «магістр»
спеціальності
014 Середня освіта (Інформатика)
Фурси Олександра Володимировича
Керівник доцент,
кандидат педагогічних наук Шокалюк Світлана Вікторівна

Оцінка:
Національна шкала _____
Шкала ECTS ____ Кількість балів _____
Голова ЕК: _____
(підпис) (прізвище та ініціали)
Члени ЕК _____
(підпис) (прізвище та ініціали)
_____ _____
(підпис) (прізвище та ініціали)
_____ _____
(підпис) (прізвище та ініціали)

Кривий Ріг – 2024

ЗАПЕВНЕННЯ

Я, Фурса Олександр Володимирович, розумію і підтримую політику Криворізького державного педагогічного університету з академічної доброчесності. Запевняю, що ця кваліфікаційна робота виконана самостійно, не містить академічного плагіату, фабрикації, фальсифікації. Я не надавав і не одержував недозволену допомогу під час підготовки цієї роботи. Використання ідей, результатів і текстів інших авторів мають покликання на відповідне джерело.

Із чинним Положенням про запобігання та виявлення академічного плагіату в роботах здобувачів вищої освіти Криворізького державного педагогічного університету ознайомлений. Чітко усвідомлюю, що в разі виявлення у кваліфікаційній роботі порушення академічної доброчесності робота не допускається до захисту або оцінюється незадовільно.

ЗМІСТ

ВСТУП

У сучасному динамічному світі інформаційних технологій система освіти постійно еволюціонує, трансформуючись відповідно до викликів цифрової епохи. Особливого значення набуває персоналізація освітнього процесу, яка дозволяє розкрити індивідуальний потенціал кожного учня, створити умови для його особистісного та професійного зростання .

Програмування як ключова компетентність сучасної молоді потребує інноваційних підходів до навчання, здатних не лише передавати технічні знання, а й розвивати креативне мислення, здатність до нестандартних рішень та неперервного професійного вдосконалення. Мова Python, завдяки своїй простоті та водночас потужності, стає оптимальним інструментом для опанування основ програмування учнями 10-11 класів [5, 6].

Актуальність дослідження зумовлена низкою визначальних чинників:

- стрімкою цифровізацією всіх сфер життєдіяльності;
- необхідністю підготовки молодих фахівців у галузі інформаційних технологій;
- потребою в індивідуальному підході до навчання програмування.

Проведений аналіз наукової літератури та освітньої практики виявив низку суттєвих суперечностей [1-9]:

- між зростаючим попитом ринку праці на високопрофесійних програмістів та реальним рівнем підготовки випускників ліцеїв;
- потенціалом персоналізованого навчання та недосконалістю існуючих методичних інструментів його реалізації;
- необхідністю впровадження індивідуальних освітніх траєкторій та обмеженістю традиційних педагогічних технологій.

Мета дослідження полягає в теоретичному обґрунтуванні та розробці методичних засад персоналізованого навчання основ програмування мовою Python учнів 10-11 класів.

Об'єкт дослідження: навчання інформатики ліцеїстів.

Предмет дослідження: методичні засади персоналізованого навчання ліцеїстів основ програмування мовою Python.

Завдання дослідження:

1. Узагальнити та систематизувати теоретичні засади персоналізованого навчання програмування.

2. Дослідити сучасні підходи до навчання програмування мовою Python.

3. Вивчити можливості освітніх платформ для персоналізованого навчання.

4. Розробити методичні рекомендації впровадження персоналізованого навчання ліцеїстів основ програмування.

Методи дослідження:

- аналіз науково-методичної літератури з проблем персоналізованого навчання програмування Python;
- систематизація педагогічного досвіду персоналізованого навчання програмуванню;
- вивчення та узагальнення інформаційно-освітніх платформ для персоналізованого навчання Python;
- теоретичне обґрунтування методичних підходів до персоналізованого навчання програмування.

Практичне значення роботи полягає в:

- розробці курсу персоналізованого вивчення Python;
- створенні онлайн-платформи з практичними рекомендаціями для вчителів з розробки персоналізованих навчальних блоків Python.

РОЗДІЛ 1. ТЕОРЕТИЧНІ ОСНОВИ ПЕРСОНАЛІЗОВАНОГО НАВЧАННЯ ПРОГРАМУВАННЯ ЛІЦЕЇСТІВ

1.1. Психологічні аспекти персоналізованого навчання

1.1.1. Вікові особливості учнів 10-11 класів

Учні 10-11 класів (15-17 років) перебувають у періоді ранньої юності, який характеризується завершенням фізичного дозрівання організму, формуванням світогляду та професійним самовизначенням.

Основні психологічні особливості учнів цього віку [10-13]:

1. Когнітивний розвиток: формування абстрактно-логічного мислення, розвиток здатності до складних аналітичних операцій, покращення концентрації та стійкості уваги, вдосконалення навичок систематизації та класифікації інформації.

2. Емоційно-вольова сфера: підвищення самоконтролю та саморегуляції, формування стійкої самооцінки, розвиток емоційної стійкості, зростання вольових якостей та цілеспрямованості.

3. Мотиваційна сфера: формування стійких пізнавальних інтересів, розвиток професійної спрямованості, посилення мотивації до самоосвіти, зростання потреби у самореалізації.

У контексті навчання програмування ці особливості проявляються наступним чином:

1. Здатність до абстрактного мислення дозволяє: краще розуміти алгоритмічні конструкції, засвоювати принципи об'єктно-орієнтованого програмування, створювати складні програмні проекти.

2. Розвинена саморегуляція сприяє: самостійному плануванню навчальної діяльності, подоланню труднощів при налагодженні програм, довготривалій роботі над програмними проектами.

3. Професійна спрямованість впливає на: вибір спеціалізованих курсів програмування, участь у олімпіадах та конкурсах, професійне самовизначення в ІТ-сфері.

Врахування цих особливостей при організації персоналізованого навчання програмування передбачає:

1. Методичні аспекти: використання проблемних завдань, впровадження проектної діяльності, забезпечення професійної спрямованості навчання.

2. Організаційні аспекти: надання можливості вибору рівня складності завдань, створення умов для самостійної роботи, забезпечення індивідуального темпу навчання.

3. Мотиваційні аспекти: зв'язок навчального матеріалу з майбутньою професією, створення ситуацій успіху, підтримка ініціативи та творчості.

1.1.2. Когнітивні стилі навчання

Когнітивні стилі навчання при вивченні програмування Python – це індивідуальні способи сприйняття, обробки та засвоєння інформації, які впливають на те, як людина опановує навички програмування [14-16].

Актуальність використання когнітивних стилів при вивченні Python обумовлена кількома факторами:

1. *Персоналізація навчання:* врахування індивідуальних особливостей сприйняття дозволяє створювати більш ефективні навчальні програми, школярі краще засвоюють матеріал, коли він подається у спосіб, що відповідає їхньому когнітивному стилю.

2. *Підвищення ефективності навчання:* розуміння власного когнітивного стилю допомагає школярам обирати оптимальні методи навчання, викладачі можуть адаптувати подачу матеріалу під різні стилі навчання.

3. *Практичне застосування:* Python має широкий спектр застосувань, від веб-розробки до аналізу даних, різні сфери застосування можуть вимагати різних підходів до навчання.

4. *Зниження бар'єру входження:* врахування когнітивних стилів допомагає новачкам легше увійти в програмування, зменшується стрес та фрустрація при вивченні нового матеріалу.

Основні когнітивні стилі навчання та їх врахування:

1. *Візуальний стиль*

Перевага візуальних образів, потреба в наочності, важливість графічного представлення інформації, схильність до візуального планування.

Методи роботи: використання блок-схем алгоритмів, візуалізація програмних конструкцій, графічне моделювання процесів, інтерактивні середовища розробки.

2. Аудіальний стиль

Сприйняття через слухові образи, важливість вербального пояснення, перевага усних дискусій, схильність до проговорювання коду.

Методи навчання: аудіо-лекції та подкасти, групові обговорення, парне програмування, вербалізація алгоритмів.

3. Кінестетичний стиль

Навчання через практичний досвід, потреба в активній діяльності, важливість тактильного досвіду, перевага інтерактивних форм роботи.

Практичні методи: навчання через створення проєктів, інтерактивні тренажери, практичні експерименти з кодом, рольові ігри та симуляції.

Методи діагностики когнітивних стилів: психологічне тестування, спостереження за навчальною діяльністю, аналіз результатів виконання завдань, самооцінка учнів.

Адаптація навчального процесу включає: розробку різнорівневих завдань (базовий, середній, високий, творчий рівні), створення мультимодального контенту (текстові матеріали, відеоуроки, інтерактивні презентації, практичні завдання).

1.1.3. Мотивація до вивчення програмування

Мотивація до вивчення програмування - це комплекс внутрішніх і зовнішніх факторів, що спонукають людину до активного освоєння програмування та визначають спрямованість і стійкість цієї діяльності [29].

Сутність мотивації до програмування – це психологічний механізм, що керує навчальною діяльністю, сукупність спонукань до вивчення мов програмування, система факторів, що визначають активність у навчанні, процес формування та підтримки інтересу до програмування, внутрішні стимули до саморозвитку [30].

Структура мотивації [31]:

а) Внутрішні мотиви: особистий інтерес до технологій, прагнення до самореалізації, бажання створювати власні програми, допитливість та прагнення до пізнання, задоволення від процесу програмування.

б) Зовнішні мотиви: перспективи працевлаштування, високий рівень заробітної плати, соціальний престиж професії, вимоги ринку праці, вплив оточення.

Компоненти мотивації [30]:

а) Когнітивний: розуміння важливості програмування, усвідомлення перспектив розвитку, знання про можливості застосування, розуміння власних цілей, оцінка своїх можливостей.

б) Емоційний: інтерес до процесу навчання, задоволення від досягнень, позитивне ставлення до програмування, емоційна залученість, ентузіазм у навчанні.

в) Поведінковий: активна участь у навчанні, регулярна практика, пошук додаткової інформації, участь у проєктах, взаємодія з іншими програмістами.

Фактори впливу на мотивацію [31]:

а) Особистісні: попередній досвід, індивідуальні здібності, рівень самооцінки, цілі та амбіції, інтереси та схильності.

б) Соціальні: підтримка родини, вплив однолітків, роль вчителя, освітнє середовище, суспільні стереотипи.

в) Організаційні: якість навчальних матеріалів, доступність ресурсів, методи навчання, технічне забезпечення, режим навчання.

Характеристики ефективної мотивації: стійкість і тривалість, позитивна спрямованість, дієвість, усвідомленість, самостійність.

Способи підтримки мотивації [31]:

а) Педагогічні: індивідуальний підхід, практична спрямованість, інтерактивні методи, створення ситуації успіху, постійний зворотний зв'язок.

б) Психологічні: розвиток впевненості, подолання страхів, формування позитивного мислення, робота з самооцінкою, емоційна підтримка.

Результати сформованої мотивації:

а) Навчальні: висока успішність, стійкий інтерес до програмування, глибоке розуміння матеріалу, активна самоосвіта, якісне виконання завдань.

б) Особистісні: професійне самовизначення, розвиток здібностей, формування компетенцій, підвищення самооцінки, особистісне зростання.

Методи підвищення мотивації: демонстрація успішних прикладів та історій, створення проєктів актуальних для учнів, регулярний зворотний зв'язок, система заохочень та винагород, участь у конкурсах та хакатонах.

Практичні аспекти: зв'язок з реальними життєвими ситуаціями, можливість застосування знань на практиці, створення власного портфоліо проєктів, участь у стажуваннях та практиках, спілкування в мережі з професійною спільнотою.

Довгострокова мотивація: формування чіткого бачення професійного майбутнього, постановка реалістичних цілей, планування кар'єрного розвитку, безперервне навчання та вдосконалення, участь у професійних конференціях та заходах.

1.2. Педагогічні основи персоналізованого навчання

Персоналізоване навчання програмування – це педагогічний підхід, який передбачає організацію навчального процесу з урахуванням індивідуальних особливостей, потреб, інтересів та темпу навчання кожного учня, спрямований на максимальну адаптацію освітнього процесу до особистісних характеристик старшокласників [32].

Педагогічні основи персоналізованого навчання [32-35]:

1. Принципи персоналізованого навчання: індивідуальний темп навчання, врахування особистих інтересів та цілей, гнучкість освітньої траєкторії, активна роль учня у навчальному процесі, постійний моніторинг прогресу.
2. Методологічні засади: компетентнісний підхід, особистісно-орієнтоване навчання, адаптивність навчальних матеріалів, диференціація завдань, інтеграція сучасних технологій.
3. Організаційні аспекти: гнучкий розклад навчання, варіативність форм роботи, модульна структура навчання, можливість вибору рівня складності, індивідуальні консультації.
4. Педагогічні технології: змішане навчання (Blended Learning), проєктна діяльність, адаптивне навчання, перевернутий клас (Flipped Classroom), мікронавчання.
5. Роль вчителя: фасилітатор навчального процесу, тьютор, ментор, розробник індивідуальних траєкторій, аналітик освітніх результатів.
6. Система оцінювання: формувальне оцінювання, індивідуальний прогрес, самооцінювання, портфоліо досягнень, гнучкі критерії оцінювання.
7. Технологічна підтримка: системи управління навчанням (LMS), адаптивні платформи, інструменти аналітики, цифрові освітні ресурси, засоби комунікації.
8. Результати впровадження: підвищення мотивації, розвиток самостійності, формування індивідуального стилю навчання, покращення освітніх результатів, розвиток метакогнітивних навичок.
9. Виклики та шляхи їх подолання: ресурсне забезпечення, підготовка педагогів, баланс між персоналізацією та стандартизацією, масштабування індивідуального підходу, моніторинг ефективності.

10. Перспективи розвитку: інтеграція штучного інтелекту, розширення адаптивних можливостей, персоналізація на основі big data, розвиток систем рекомендацій, удосконалення інструментів аналітики.

Принципи персоналізованого навчання

Принципи персоналізованого навчання основ програмування мовою Python – це фундаментальні положення, що визначають зміст, організаційні форми та методи навчального процесу з урахуванням специфіки вивчення Python та індивідуальних особливостей учнів [44].

Дидактичні принципи персоналізованого навчання основ програмування мовою Python [45, 47, 48]:

1. Принцип індивідуалізації: адаптація навчального процесу до індивідуальних особливостей кожного учня.

Приклади завдань: різнорівневі завдання з теми "Цикли": від базового while для підрахунку суми до складних вкладених циклів; вибір проєктів за інтересами: створення гри для геймерів, аналіз даних для аналітиків; індивідуальний темп вивчення функцій: від простих до рекурсивних.

2. Принцип активності та самостійності: стимулювання самостійної навчальної діяльності учнів.

Приклади завдань: самостійне створення калькулятора з базових операцій; челендж "Напиши свою першу програму за тиждень"; дослідницькі проєкти з використанням бібліотек Python.

3. Принцип систематичності: логічна послідовність подачі матеріалу та формування навичок.

Приклади завдань: послідовність: змінні → оператори → умови → цикли → функції; кожне нове поняття базується на попередньому; регулярні практичні завдання з поступовим ускладненням.

4. Принцип наочності: візуалізація навчального матеріалу та процесів.

Приклади завдань: візуалізація алгоритмів через блок-схеми; демонстрація роботи циклів на анімаціях; інтерактивні приклади роботи з turtle графікою.

5. Принцип практичності: зв'язок теорії з реальними задачами.

Приклади завдань: створення телеграм-бота для розкладу уроків; програма обліку особистих фінансів; автоматизація рутинних задач на комп'ютері.

6. Принцип доступності: відповідність складності матеріалу рівню підготовки.

Приклади завдань: початок з простих print("Hello, World!"); поступовий перехід від простих типів даних до складних; декомпозиція складних задач на прості кроки.

7. Принцип мотивації: підтримка інтересу до навчання через досяжні цілі.

Приклади завдань: система досягнень за виконані завдання; створення власних ігор на Python; участь у хакатонах та конкурсах.

8. Принцип зворотного зв'язку: постійний моніторинг розуміння та прогресу.

Приклади завдань: автоматична перевірка коду через тести; код-рев'ю від викладача; взаємне оцінювання проєктів учнями.

9. Принцип технологічності: використання сучасних інструментів навчання.

Приклади завдань: робота в PyCharm Educational [53]; використання GitHub [61] для проєктів; онлайн-платформи для практики (Google Classroom + Colab [62,63], Moodle + Python плагіни [64], Eolymp [65]).

10. Принцип рефлексії: аналіз власного прогресу та помилок.

Приклади завдань: ведення щоденника програміста; документування власного коду; аналіз складності та оптимізації рішень.

11. Принцип колаборації: взаємодія між учнями та з викладачем.

Приклади завдань: парне програмування; групові проєкти з розподілом ролей; спільне обговорення рішень задач.

12. Принцип адаптивності: гнучке налаштування навчального процесу.

Приклади завдань: вибір часу та темпу навчання; можливість повернення до складних тем; додаткові матеріали за потреби.

13. Принцип результативності: орієнтація на конкретні навчальні результати.

Приклади завдань: створення портфоліо проєктів; виконання тестових завдань різної складності; демонстрація власних програмних рішень.

14. Принцип інтеграції: зв'язок програмування з іншими предметами.

Приклади завдань: розв'язання математичних задач на Python; створення програм для фізичних розрахунків; обробка текстів для мовних предметів.

15. Принцип відкритості: доступ до різноманітних навчальних ресурсів.

Приклади завдань: використання відкритих освітніх ресурсів; участь у онлайн-курсах; доступ до документації Python та спільноти.

Ці дидактичні принципи забезпечують [48]:

1. Ефективність навчання: досягнення навчальних цілей, оптимальне використання ресурсів, високі результати навчання, стійкість отриманих знань, розвиток практичних навичок.

2. Якість освітнього процесу: системність навчання, послідовність викладення, доступність матеріалу, практична спрямованість, індивідуальний підхід.

3. Розвиток особистості: формування компетентностей, розвиток самостійності, підвищення мотивації, особистісне зростання, професійне становлення.

1.2.2. Методи персоналізації навчання

Методи персоналізації навчання Python – це система педагогічних підходів для адаптації освітнього процесу програмування під індивідуальні особливості старшокласників [19].

Методи персоналізації навчання Python [17-19, 26, 32]:

1. Діагностичні методи: тестування базових знань програмування, визначення рівня алгоритмічного мислення, аналіз досвіду роботи з іншими мовами програмування, оцінка математичної підготовки.

2. Адаптивні методи: градація складності завдань з програмування, персоналізовані траєкторії вивчення Python, адаптивне тестування знання синтаксису та конструкцій, індивідуальний темп освоєння нових тем.

3. Інтерактивні методи: розробка міні-проєктів на Python, парне програмування, командна розробка застосунків, хакатони та код-марафони.

4. Цифрові методи: онлайн-платформи для навчання Python, інтерактивні середовища розробки, Jupyter Notebooks [57] для експериментів, автоматична перевірка коду.

5. Методи індивідуального супроводу: код-рев'ю учнівських проєктів, менторство з боку досвідчених програмістів, індивідуальні консультації з складних тем, допомога у виборі спеціалізації в Python.

6. Методи самостійної роботи: персональні проєкти на Python, робота з документацією та туторіалами, створення власних бібліотек, участь у відкритих проєктах.

7. Методи мотивації: рейтинг успішності розв'язання задач, змагання з програмування, система досягнень за опановані теми, публікація найкращих проєктів.

8. Технологічні методи: автоматична перевірка стилю коду, аналітика прогресу навчання, рекомендації щодо вивчення нових тем, інтелектуальні підказки при написанні коду.

Перелічені вище методики забезпечують персоніфікований підхід до навчання програмування на Python, що дозволяє кожному студенту розвиватися відповідно до своїх індивідуальних особливостей та потреб.

ВИСНОВКИ ДО РОЗДІЛУ 1

Персоналізація навчання програмування мовою Python – це сучасний підхід до організації освітнього процесу, який допомагає адаптувати навчання під індивідуальні особливості, потреби та здібності кожного учня. Цей метод передбачає глибоке розуміння психологічних особливостей кожного учня та застосування індивідуальних дидактичних принципів навчання.

У розділі 1 описані ключові психологічні аспекти, дидактичні принципи та методи персоналізованого навчання, які формують структуру навчання та забезпечують всебічний розвиток учня.

Ефективність персоналізованого навчання забезпечується комплексним підходом, який включає діагностику рівня підготовки, гнучку адаптацію навчального матеріалу, індивідуальну підтримку учня, постійну мотивацію до вивчення програмування та використання сучасних технологічних інструментів. Такий підхід дозволяє створити унікальне освітнє середовище, що максимально відповідає потребам кожного учня.

Головна мета персоналізованого навчання Python – не просто передати технічні знання, а розкрити потенціал кожного учня, сформувати стійкий інтерес до програмування та саморозвитку.

РОЗДІЛ 2. МЕТОДИЧНА СИСТЕМА ПЕРСОНАЛІЗОВАНОГО НАВЧАННЯ ПРОГРАМУВАННЯ МОВОЮ PYTHON

2.1. Аналіз сучасних підходів до навчання програмування

Сучасні підходи до навчання програмування Python – це комплексні методики та стратегії викладання, які поєднують теоретичні знання з практичним досвідом, використовують інтерактивні технології та враховують актуальні потреби IT-індустрії для ефективного формування програмістських компетенцій.

Сучасні підходи до навчання програмування Python [17-19, 50, 51]:

1. Проєктно-орієнтоване навчання Python: розробка реальних проєктів з веб-додатками, аналізом даних, машинним навчанням, практичні кейси з використанням бібліотек numpy, pandas, Django, командні проєкти з імітацією робочого процесу в IT-компанії, розвиток навичок вирішення комплексних завдань через практику.

2. Адаптивне навчання Python: персоналізовані навчальні траєкторії залежно від рівня підготовки, автоматизована система оцінювання прогресу, підбір індивідуальних завдань різної складності, використання штучного інтелекту для адаптації контенту.

3. Гейміфікація вивчення Python [27]: інтерактивні coding-квести та челенджі, система нагород та рейтингування, симуляційні ігрові сценарії программування для підтримки мотивації.

4. Інтерактивні платформи для Python: CodeCademy Python Track [54], Coursera Python Specialization [55], PyCharm Educational Edition [53], HackerRank Python challenges [52], реальне кодування безпосередньо в браузері.

5. Візуальне програмування Python: використання Blockly [56] для вивчення базових концепцій, Graphical Python Tutor [58] для розуміння алгоритмів, Jupyter Notebook [57] з інтерактивною візуалізацією, анімаційне представлення структур даних та алгоритмів.

6. Парне програмування на Python [59]: Code Review в групах, спільне розв'язання алгоритмічних задач, взаємне навчання та обмін досвідом, практика комунікативних навичок в контексті розробки.

7. Мікронавчання Python [60]: короткі відео-уроки (5-15 хвилин), концентровані практичні завдання, мобільні додатки для вивчення, Telegram боти з щоденними завданнями.

Завдяки таким підходам навчання програмування стає більш ефективним і цікавим. Практичні завдання, індивідуальний підхід і можливість працювати в своєму темпі дозволяють глибше зануритися в матеріал і краще зрозуміти, як все працює.

2.1.1. Огляд існуючих методик навчання Python

Методики навчання Python – це структуровані підходи та педагогічні стратегії, спрямовані на ефективне засвоєння мови програмування Python, які враховують її синтаксичні особливості, широкі можливості застосування та потреби сучасного ринку праці [17].

Методики навчання Python [6, 17-19, 26, 66]:

1. Традиційні методики

Класичний лекційно-практичний формат з покроковим вивченням синтаксису, виконанням типових завдань та роботою з підручниками.

2. Онлайн-платформи

Інтерактивні уроки з відеоматеріалами, автоматичною перевіркою та форумами підтримки, представлені на платформах Coursera, Udemy, DataCamp, Python.org.

3. Проєктний підхід

Створення веб-додатків, аналіз даних, розробка ігор та автоматизація, що дозволяє отримати практичний досвід, сформувати портфоліо та підвищити мотивацію.

4. Методика "від простого до складного"

Поетапне вивчення базового синтаксису, структур даних, функцій, ООП та фреймворків, що дозволяє ефективно засвоювати матеріал.

5. Інтерактивні середовища

Використання Jupyter Notebooks, Google Colab, Repl.it для візуалізації, експериментування та спільної роботи.

6. Гейміфіковані платформи

Платформи CheckiO, CodeCombat, PyCharm Edu, що включають рівні складності, досягнення та змагальні елементи для підвищення мотивації.

7. Спеціалізовані напрямки

Вивчення Python для конкретних сфер, таких як Data Science [67], Machine Learning [68] з фокусом на практичних кейсах та актуальних інструментах.

8. Методика Code Review [69]

Аналіз коду, отримання зворотного зв'язку, вивчення кращих практик та робота над помилками для вдосконалення навичок.

9. Метод занурення

Інтенсивні курси, хакатони та групові проєкти, що дозволяють повністю зануритись у процес навчання.

10. Адаптивне навчання

Індивідуальний темп, персоналізовані завдання, відстеження прогресу та корекція програми для максимальної ефективності.

Ефективність кожної методики залежить від рівня підготовки учнів, цілей навчання, доступного часу та технічних можливостей.

Рекомендації:

- для початківців – інтерактивні платформи та базові проєкти;
- для середнього рівня – проєктний підхід та спеціалізація;
- для просунутого рівня – складні проєкти та Code Review.

2.1.2. Міжнародний досвід персоналізації навчання програмування

Міжнародний досвід персоналізації навчання програмування – це сукупність перевірених освітніх практик, методологій та інструментів, що використовуються у різних країнах світу для створення індивідуалізованих траєкторій навчання програмуванню з урахуванням особливостей кожного учня та глобальних тенденцій розвитку IT-індустрії [70, 71].

Ключові тренди персоналізації [43, 72]:

- AI-рекомендації з аналізом патернів навчання, прогнозуванням складнощів, підбором оптимального контенту, персоналізованими підказками, адаптивні навчальні системи з динамічним регулюванням складності, індивідуальним темпом навчання, автоматичною адаптацією контенту, інтелектуальним повторенням, аналітика навчання з відстеженням прогресу, виявленням прогалин, аналізом ефективності, предиктивною аналітикою, віртуальні лабораторії з інтерактивними середовищами, симуляціями реальних проєктів, безпечним експериментуванням, миттєвим фідбеком.
- Компетентнісно-орієнтоване навчання з чіткими критеріями майстерності, гнучким графіком навчання, фокусом на практичних навичках, індивідуальним прогресом, навчання для досягнення майстерності з послідовним опануванням тем, глибоким розумінням матеріалу, повторенням до досягнення майстерності, індивідуальним темпом, проєктно-орієнтоване

навчання з реальними проєктами, практичним застосуванням знань, командною роботою, портфоліо досягнень, навчання у парах з взаємним навчанням, код-ревю, парним програмуванням, груповими проєктами.

- Постійне оцінювання з моніторингом, швидким фідбеком, адаптивними тестами, автоматичною перевіркою коду, портфоліо-орієнтоване оцінювання зі збором виконаних проєктів, демонстрацією прогресу, реальними кейсами, професійним портфоліо, перевірка навичок через технічні співбесіди, сертифікаційні екзамени, практичні завдання, оцінку soft skills, реальні проєкти з співпрацею з компаніями, стажуванням, внесками до open source, хакатонами.
- Підтримка учнів через персональні консультації, онлайн-консультації з 24/7 підтримкою, відео-консультації, чат-підтримка, групи підтримки з групами взаємодопомоги, форумами обговорень, роботу в мережі.

2.1.3. Вибір оптимальних інструментів та середовищ розробки

Вибір оптимальних інструментів та середовищ розробки при персоналізованому вивченні Python у 10-11 класах ліцеїв – це процес ретельного аналізу та порівняння різноманітних програмних засобів, платформ та технологій, доступних для використання в процесі навчання та розробки на мові Python, з метою визначення найбільш підходящих та ефективних рішень, що відповідають специфічним потребам та особливостям учнів старших класів ліцеїв [33].

Цей вибір враховує такі фактори:

1. Простота встановлення та налаштування інструментів, інтуїтивно зрозумілий інтерфейс для учнів, мінімальні системні вимоги для ефективної роботи.

2. Наявність навчальних матеріалів, туторіалів, прикладів коду, інтеграція з освітніми платформами, віртуальними середовищами, можливості для практичної роботи, експериментування, відпрацювання навичок.

3. Можливості налаштування інтерфейсу та робочих процесів під індивідуальні потреби учнів, адаптивний контент та зворотний зв'язок, що враховує рівень знань кожного учня, інструменти для відстеження прогресу, аналізу помилок, надання рекомендацій.

4. Захист персональних даних учнів, можливість безпечного експериментування та роботи з даними.

5. Можливість взаємодії з іншими інструментами, використовуваними в навчальному процесі, адаптація до різних операційних систем, пристроїв, рівнів доступу.

Розглянемо оптимальні інструменти та середовища розробки для персоналізованого вивчення Python у 10-11 класах ліцеїв:

1) Google Classroom [62]+ *Colab* [63, 85-94]

1. Переваги платформи

1.1. Google Workspace for Education надає повністю безкоштовний доступ до всіх інструментів, включаючи необмежене хмарне сховище, професійні інструменти для дистанційного навчання та розширені функції адміністрування для освітніх закладів.

1.2. Інтуїтивно зрозумілий інтерфейс не потребує спеціального навчання, має адаптивний дизайн для всіх пристроїв, швидкий старт роботи та автоматичне оновлення без необхідності технічного обслуговування.

1.3. Повна синхронізація з Gmail, Google Drive, Calendar, Meet та іншими сервісами забезпечує єдиний цифровий простір для навчання, комунікації та співпраці між учнями та вчителями.

1.4. Забезпечує одночасну роботу кількох учнів над одним проєктом у реальному часі, можливість коментування та обговорення коду, демонстрацію екрану та відеозв'язок для групових консультацій.

1.5. Постійна синхронізація всіх змін у хмарному сховищі, версіонування документів та можливість відновлення попередніх версій забезпечують надійне збереження навчальних матеріалів та учнівських робіт.

2. Процес налаштування

2.1. Організація віртуальних класів включає налаштування тем, створення категорій завдань, встановлення термінів виконання, додавання учнів через електронні адреси або коди доступу, призначення співвикладачів.

2.2. Підключення Colab notebooks через створення шаблонів notebooks для різних тем програмування, налаштування автоматичного копіювання для кожного учня, встановлення необхідних бібліотек та середовища виконання.

2.3. Гнучка система прав доступу дозволяє керувати можливостями перегляду та редагування матеріалів, встановлювати обмеження на копіювання, налаштовувати видимість коментарів та оцінок.

2.4. Структурування навчального контенту відбувається за допомогою тем, модулів та розділів, з можливістю планування публікацій, створення

оголошень, додавання різнотипних матеріалів (відео, презентації, документи) та інтерактивних завдань.

2.5. Система оцінювання включає створення рубрик оцінювання, налаштування шкали балів, автоматичну перевірку тестів, можливість надання зворотного зв'язку через коментарі, експорт оцінок та генерацію звітів про успішність.

3. Особливості використання для навчання Python

3.1. Інтерактивне середовище розробки Colab надає повноцінне середовище для написання та виконання Python-коду з підтримкою всіх сучасних бібліотек, візуалізацією даних, можливістю використання GPU та TPU для складних обчислень.

3.2. Адаптивне навчання дозволяє створювати персоналізовані навчальні траєкторії через систему різнорівневих завдань, додаткових матеріалів та індивідуальних проєктів, з можливістю відстеження прогресу кожного учня.

3.3. Забезпечує детальну статистику активності учнів, відстеження виконання завдань, автоматичне нагадування про дедлайни, аналіз типових помилок та труднощів у навчанні.

3.4. Вбудовані інструменти для індивідуальних та групових консультацій, можливість швидкого зворотного зв'язку, обговорення коду в коментарях, створення FAQ та бази знань для типових питань.

3.5. Підтримка групових проєктів через спільні notebook'и, система контролю версій, можливість розподілу завдань між учасниками команди, презентація результатів через вбудовані інструменти візуалізації.

2) Moodle + Python плагіни [64, 95-98]

1. Переваги платформи

1.1. Платформа повністю безкоштовна, має відкритий код з можливістю встановлення на власний сервер, гнучкою модифікацією під потреби закладу, прозорою системою безпеки та незалежністю від зовнішніх провайдерів.

1.2. Дозволяє створювати індивідуальні траєкторії навчання, включає адаптивне тестування, різнорівневі завдання, умовний доступ до матеріалів та автоматизацію навчального процесу.

1.3. Наявна активна українська спільнота, регулярні оновлення, готові рішення типових проблем, можливість обміну досвідом між викладачами та документація українською мовою.

1.4. Включає CodeRunner для автоматичної перевірки коду, Virtual Programming Lab (VPL), Python-based Assessment, інтеграцію з PyCharm Edu та Jupyter Notebook.

2. Процес налаштування

2.1. Встановлення Python-плагінів – налаштування безпечного середовища виконання коду з обмеженням доступу до системних ресурсів, встановлення лімітів пам'яті та часу виконання, створення системи тестових випадків.

2.2. Конфігурація середовища передбачає встановлення Python інтерпретатора, налаштування віртуальних середовищ, конфігурацію sandbox для безпечного виконання, встановлення необхідних бібліотек та налаштування лімітів ресурсів.

2.3. Автоматична перевірка включає unit-тести для завдань, перевірку стилю коду згідно PEP 8, систему антиплагіату, відстеження часу виконання та аналіз ефективності коду.

2.4. Інтеграція з IDE забезпечує налаштування PyCharm Educational, Visual Studio Code, інтеграцію з Jupyter notebooks, підтримку веб-IDE та синхронізацію з git-репозиторіями.

3. Адаптація під персоналізоване навчання

3.1. Організація матеріалу за рівнями складності: базовий (змінні, умови, цикли), середній (функції, списки, словники) та просунутий (класи, декоратори, генератори), з відповідними вимогами до мінімального балу для переходу між рівнями.

3.2. Комбінує автоматичне тестування коду, перевірку алгоритмічної складності, оцінку якості документації, peer review від однокласників та комплексні проєктні роботи.

3.3. Забезпечує динамічне регулювання складності, індивідуальний темп навчання, надання додаткових матеріалів, альтернативних пояснень та інтерактивних підказок.

3.4. Відстежує завершені теми, оцінки, витрачений час, рівень складності та бали досягнень, автоматично коригуючи рівень складності на основі середнього балу.

3.5. Комунікація та зворотній зв'язок реалізується через автоматичні повідомлення про прогрес, консультації з викладачем, форуми для обговорення, групові проєкти та систему досягнень і бейджів.

3) *Eolymp* [65, 99-104]

1. Переваги платформи

1.1. Платформа розроблена спеціально для навчання програмування та підготовки до олімпіад, має зручний інтерфейс для роботи з кодом, підтримує множину мов програмування включно з Python, та надає потужну систему тестування розв'язків.

1.2. Велика база алгоритмічних задач містить понад 10000 задач різного рівня складності, від найпростіших до олімпіадних, з детальними описами, прикладами вхідних та вихідних даних, та методичними вказівками для розв'язання, систематизованих за темами та рівнями складності.

1.3. Забезпечує швидку перевірку коду на великій кількості тестів, надає детальний звіт про проходження кожного тесту, включаючи час виконання та використану пам'ять, допомагає знаходити помилки та оптимізувати рішення.

1.4. Повністю локалізована українською мовою платформа з методичними матеріалами, умовами задач, системою допомоги та підтримкою користувачів, що робить її доступною для українських школярів та вчителів.

1.5. Включає спеціальні розділи для підготовки до олімпіад різних рівнів, архів задач з минулих змагань, можливість участі в онлайн-змаганнях та тренувальних турах, систему рейтингу та відзнак.

2. Процес налаштування

2.1. Процес реєстрації включає створення облікових записів для учнів, налаштування профілів, визначення початкового рівня підготовки, формування індивідуальних траєкторій навчання та встановлення цілей для кожного учня.

2.2. Організація учнів у групи за рівнем підготовки, налаштування спільного доступу до матеріалів, створення окремих змагань та завдань для кожної групи, моніторинг групової активності та прогресу.

2.3. Формування тематичних добірок задач відповідно до навчальної програми, створення послідовності завдань зростаючої складності, підбір задач для відпрацювання конкретних алгоритмів та прийомів програмування.

2.4. Відстеження успішності розв'язання задач, аналіз типових помилок, перегляд статистики спроб та успішних розв'язків, оцінка часу виконання та ефективноті коду, формування рекомендацій щодо покращення результатів.

2.5. Можливість створення власних змагань різних форматів, встановлення часових обмежень, вибір задач різної складності, налаштування системи оцінювання, проведення турнірів та марафонів програмування.

3. Методика використання

3.1. Структурування матеріалу за темами алгоритмів та структур даних, поступове ускладнення завдань, комбінування теоретичного матеріалу з практичними задачами, регулярні змагання для підтримки мотивації.

3.2. Адаптація складності задач до рівня кожного учня, надання додаткових підказок та пояснень, можливість консультацій з викладачем, формування індивідуальних планів підготовки.

3.3. Організація командних змагань, обговорення розв'язків у групах, спільний розбір складних задач, змагання між класами та школами, формування навичок командної роботи.

3.4. Спеціальні добірки олімпіадних задач, тренування в умовах обмеженого часу, розбір типових олімпіадних прийомів, участь у відбіркових турах та пробних олімпіадах.

3.5. Система рейтингів та досягнень, відзначення успішних розв'язків, змагальний елемент між учнями та групами, публічне визнання досягнень, формування портфоліо успіхів.

Вибір оптимальних інструментів та середовищ розробки для персоналізованого вивчення Python у 10-11 класах ліцеїв повинен базуватися на ретельному аналізі потреб учнів, наявних ресурсів та поточних тенденцій розвитку. Використання комбінації різних інструментів, що доповнюють один одного, може забезпечити найбільш ефективне та персоналізоване навчання.

2.2. Розробка персоналізованого навчального контенту

Розробка персоналізованого навчального контенту – це процес створення навчальних матеріалів, які адаптовані до індивідуальних потреб, інтересів, рівня знань та стилю навчання кожного учня. Ця концепція базується на ідеї, що учні мають різні здібності, переваги та темпи навчання, і тому використання універсального підходу до всіх може бути неефективним [32,73].

Основні аспекти розробки персоналізованого навчального контенту [32, 34, 35, 73 - 75]:

1. Оцінка потреб учнів: Збір інформації про поточний рівень знань, інтереси, стилі навчання, цілі та труднощі учнів для формування персоналізованого підходу.

2. Адаптація змісту: Створення навчальних матеріалів, які відповідають потребам та рівню кожного учня, включаючи різні формати (текст, відео, інтерактивні вправи тощо) та різні рівні складності.

3. Гнучкість та вибір: Надання учням вибору та контролю над тим, що і як вони вивчають, дозволяючи їм обирати теми, формати та темп навчання, що найкраще відповідає їхнім потребам.

4. Зворотний зв'язок та моніторинг: Відстеження прогресу учнів, надання своєчасного зворотного зв'язку та коригування навчального контенту та підходу при необхідності.

5. Технологічна підтримка: Використання освітніх технологій та програмного забезпечення для полегшення створення, доставки та моніторингу персоналізованого навчального контенту.

Класифікація етапів персоналізованого підходу до вивчення Python [76]:

1. Початкова діагностика:

Рівень знань

- Попереднє тестування для визначення поточного рівня знань учнів з предмету або теми.
- Виявлення прогалин та сильних сторін у знаннях.
- Оцінка готовності до вивчення нового матеріалу.

Стиль навчання:

- Визначення переважаючого стилю навчання кожного учня (візуальний, аудіальний, кінестетичний тощо).
- Врахування індивідуальних переваг при підборі форматів навчальних матеріалів.

Цілі та мотивація:

- Збір інформації про очікування, цілі та мотивацію учнів щодо вивчення предмету.
- Виявлення чинників, що мотивують або демотивують учнів.
- Врахування цілей для створення релевантного та привабливого контенту.

Технічний бекграунд:

- Оцінка рівня технічних навичок та досвіду учнів.
- Визначення потреби в додатковій технічній підтримці або навчанні.

2. Збір даних:

Тестування:

- Проведення діагностичних тестів, вікторин або завдань для оцінки поточного рівня знань.
- Використання тестів з різними форматами запитань та рівнями складності.

Опитування:

- Розробка анкет або опитувальників для збору інформації про стилі навчання, інтереси, мотивацію та труднощі учнів.
- Використання закритих та відкритих запитань.

Спостереження:

- Моніторинг та аналіз поведінки й залученості учнів під час навчального процесу.
- Фіксування реакцій, зауважень та труднощів при взаємодії з навчальним контентом.

Аналіз активності:

- Збирання та аналіз даних про взаємодію учнів з навчальними ресурсами та системами (перегляди, завантаження, виконання завдань тощо).
- Виявлення зразків та тенденцій, що вказують на інтереси та проблемні області.

Зібрана інформація допоможе створити повний профіль кожного учня та забезпечити персоналізований підхід до розробки навчального контенту, форматів та методів викладання.

З метою забезпечення персоналізованого підходу до розробки навчального контенту з вивчення Python у 10-11 класах ліцеїв, нами розроблено структуру профілю учня. Структура повного профілю може включати таку інформацію:

1. Рівень знань Python включає початківців без попереднього досвіду програмування, учнів з базовим рівнем знань Python або іншої мови програмування, просунутий рівень з солідним досвідом програмування на Python.

2. Стилі навчання охоплюють візуальний (сприйняття через зображення), аудіальний (усні пояснення), кінестетичний (практичні вправи), читацький (текстові матеріали).

3. Цілі та мотивація вивчення Python пов'язані з розвитком навичок програмування, створенням власних проектів, підготовкою до олімпіад, загальним інтересом до ІТ-технологій.

4. Технічний бекграунд може включати базові навички програмування, досвід роботи з апаратним забезпеченням, основи веб-розробки, навички роботи з базами даних та системами контролю версій.

5. Переважаючі формати навчання складаються з текстових матеріалів, відео-уроків, інтерактивних онлайн-курсів, практичних завдань та проектів.

6. Темп навчання диференціюється на повільний, середній та швидкий, залежно від здатності учня засвоювати матеріал.

7. Додаткові потреби та обмеження включають необхідність супроводу, технічні обмеження, особливі вимоги до навчального процесу.

Створивши повний профіль для кожного учня на основі зібраних даних, можна адаптувати навчальний контент, формати, темп викладання та додаткову підтримку відповідно до індивідуальних потреб. Це забезпечить більш ефективне засвоєння матеріалу, підвищить мотивацію учнів та загалом покращить якість навчання Python у 10-11 класах ліцеїв.

Від детального вивчення індивідуальних характеристик учня ми переходимо до розробленої нами концепції створення персоналізованого навчального блоку Python, який адаптивно вибудовує освітню траєкторію, максимально враховуючи особливості кожного школяра.

Персоналізований навчальний блок Python – це структурована навчальна одиниця з вивчення мови Python, що адаптується під індивідуальні потреби та рівень учня, включає теоретичні та практичні компоненти, має багаторівневу структуру складності, забезпечує інтерактивне навчання та зворотний зв'язок, передбачає різні траєкторії навчання, містить систему оцінювання та рефлексії, пов'язує навчальний матеріал з практичним застосуванням, підтримує професійний розвиток учня [76].

Персоналізований навчальний блок Python:

1. Вступна частина

1.1. Загальна інформація: тема з Python, базова ідея блоку, проблемне питання, тривалість, цільова аудиторія, технічні вимоги, формат навчання.

1.2. Цілі та актуальність: навчальні цілі для базового, середнього та високого рівнів, актуальність теми з реальними прикладами застосування, мультимедійні матеріали, зв'язок з ІТ-індустрією, статистика ринку праці, історії успіху відомих програмістів, приклади проєктів.

1.3. Вхідний контроль: попереднє оцінювання знань, виявлення потенційних складнощів, мотиваційне завдання, план роботи з типовими

помилками, адаптивне тестування, діагностика стилю навчання, визначення цілей розвитку.

1.4. Структурна схема блоку: візуальна карта знань, зв'язки з попередніми темами, схема опорних сигналів, нейронна карта понять, інтерактивна дорожня карта, діаграма компетенцій, FAQ-секція.

2. Частина базового рівня

2.1. Теоретична частина: основні поняття та концепції, інтерактивні матеріали, довідкові ресурси, приклади коду, інтерактивний глосарій, відеоуроки, анімовані пояснення.

2.2. Практична частина: базові вправи, робота з готовим кодом, завдання на розуміння, самоперевірка, інтерактивний простір програмування, система підказок, автоматична перевірка коду.

3. Частина середнього рівня

3.1. Розширені концепції: поглиблена теорія, складніші приклади, типові алгоритми, шаблони програмування, оптимізація коду, робота з документацією, кращі світові практики.

3.2. Практичний блок: самостійне написання програм, робота з бібліотеками, налагоджування та оптимізація, проміжне тестування, огляд коду сесії, парне програмування, міні-проєкти.

4. Частина поглибленого рівня

4.1. Просунуті концепції: експертні техніки, оптимізація коду, інтеграція з іншими технологіями, професійні практики, архітектурні патерни, безпека коду, масштабування рішень.

4.2. Проєктна діяльність: індивідуальні проєкти, групові завдання, творчі завдання, дослідницька робота, Open source контрибуції, хакатони, менторські програми.

5. Індивідуальні траєкторії

5.1. Варіанти маршрутів: швидкий маршрут, стандартний маршрут, підтримуючий маршрут, професійний маршрут, дослідницький маршрут, практико-орієнтований маршрут, командний маршрут.

5.2. Адаптивні елементи: додаткові матеріали, альтернативні завдання, варіативні проєкти, персоналізовані рекомендації, система трекінгу прогресу, адаптивне тестування, гнучкий графік навчання.

6. Оцінювання та рефлексія

6.1. Інструменти оцінювання: автотести, огляд коду, захист проєктів, підсумкове тестування, метрики якості коду, рецензування колегами, оцінювання портфоліо.

6.2. Рефлексія та аналіз: самооцінка, взаємооцінювання, аналіз прогресу, портфоліо робіт, щоденник навчання, метрики успішності, зворотній зв'язок.

7. Підсумковий блок

7.1. Систематизація знань: підсумкова схема знань, практичне застосування, професійні перспективи, сертифікація, портфоліо проєктів, резюме досягнень, рекомендації з розвитку.

7.2. Перехід до наступного блоку: попередній огляд наступної теми, необхідні знання, рекомендації з підготовки, план переходу, додаткові ресурси, кар'єрні можливості.

2.2.1. Структурування навчального матеріалу за рівнями складності

Структурування навчального матеріалу за рівнями складності для вивчення Python у 10-11 класах ліцеїв передбачає організацію та подання навчального контенту з урахуванням поступового нарощування складності, що відповідає рівню підготовки та темпу засвоєння матеріалу кожним учнем.

Структурування навчального матеріалу за рівнями складності для вивчення Python у 10-11 класах ліцеїв [77, 78]:

1. Початковий рівень (A1)

- Пояснення: Знайомство з базовими концепціями мови Python, включаючи синтаксис, змінні, типи даних, прості операції та умовні конструкції.
- Базові концепції: синтаксис Python, змінні та типи даних, прості операції, умовні конструкції.
- Практичні завдання: калькулятор, конвертер величин, прості ігри, робота з рядками.

2. Базовий рівень (A2)

- Пояснення: Вивчення основних структур мови, таких як цикли, списки, кортежі, функції та робота з файлами. Реалізація простих проєктів.
- Основні структури: цикли, списки та кортежі, функції, робота з файлами.
- Проєкти: текстові ігри, обробка даних, простий парсинг, базові алгоритми.

3. Середній рівень (B1)

- Пояснення: Опанування більш просунутих концепцій, включаючи словники, множини, обробку винятків, модулі та пакети, а також основи об'єктно-орієнтованого програмування. Практика з CLI додатками, роботою з API та базами даних.
- Просунуті концепції: словники та множини, обробка винятків, модулі та пакети, ООП основи.
- Практика: CLI додатки, робота з API, бази даних SQLite, автоматизація.

4. Високий рівень (B2)

- Пояснення: Вивчення складних тем, таких як декоратори, генератори, регулярні вирази та багатопоточність. Реалізація більш складних проєктів, включаючи веб-скрапінг, GUI додатки, REST API та тестування.
- Складні теми: декоратори, генератори, регулярні вирази, багатопоточність.
- Проєктні роботи: веб-скрапінг, GUI додатки, REST API, тестування.

5. Професійний рівень (C1)

- Пояснення: Вивчення експертних тем, таких як асинхронне програмування, метапрограмування, оптимізація та архітектура. Реалізація складних проєктів, включаючи веб-фреймворки, Data Science, мікросервіси та DevOps практики.
- Експертні теми: асинхронне програмування, метапрограмування, оптимізація, архітектура.
- Складні проєкти: веб-фреймворки, Data Science, мікросервіси, DevOps практики.

6. Експертний рівень (C2)

- Пояснення: Вивчення спеціалізованих областей, таких як Machine Learning, Distributed Systems, Security та Performance. Реалізація професійних проєктів, включаючи Enterprise додатки, AI системи, Cloud рішення та системну архітектуру.
- Спеціалізовані області: Machine Learning, Distributed Systems, Security, Performance.
- Професійні проєкти: Enterprise додатки, AI системи, Cloud рішення, системна архітектура.

Структурований підхід до вивчення мови програмування Python, який охоплює такі ключові аспекти:

1. Критерії переходу між рівнями:

- Технічні навички: визначають рівень опанування учнями практичних аспектів програмування, таких як якість коду, розуміння концепцій, вирішення проблем, оптимізація.
- Soft skills: характеризують набуті учнями надпредметні навички, важливі для успішного програмування, такі як самостійність, документування, командна робота, умінь проводити огляд коду.

2. Методичні матеріали:

- За рівнями: передбачають використання різних типів навчальних ресурсів (підручники, відеокурси, практикуми, референси) для забезпечення послідовного вивчення матеріалу.
- Додаткові ресурси: включають довідкові, теоретичні та практичні матеріали (документація, статті, приклади коду), а також можливість долучення до спільноти.

3. Система оцінювання:

- Критерії: визначають показники, за якими оцінюється прогрес учнів, зокрема виконання завдань, якість коду, розуміння теорії, практичне застосування.
- Методи контролю: передбачають різноманітні форми оцінювання, такі як тестування, захист проєктів, код-ревю, співбесіди.

4. Адаптивні елементи:

- Підтримка навчання: включає додаткові пояснення, альтернативні шляхи, практичні приклади, менторство для забезпечення індивідуального підходу.
- Корекція траєкторії: передбачає аналіз прогресу, виявлення прогалин, надання індивідуальних завдань, повторення матеріалу для коригування навчальної траєкторії.

5. Практичні рекомендації:

- Для вчителів: спрямовані на ефективне управління навчальним процесом, зокрема моніторинг прогресу, адаптацію матеріалів, підтримку мотивації, застосування індивідуального підходу.
- Для учнів: акцентують на важливості регулярної практики, роботи над проєктами, участі у спільноті, самоосвіти.

2.2.2. Створення адаптивних завдань та проєктів

Адаптивні завдання та проєкти при вивченні мови програмування Python – це комплекс навчальних завдань та практичних проєктів, спрямованих на забезпечення персоналізованого підходу до навчання, врахування індивідуальних особливостей і потреб учнів, а також поступове підвищення рівня складності відповідно до їхнього прогресу.

Методологія створення адаптивних завдань та персоналізованих навчальних траєкторій з Python для учнів 10-11 класів [38, 40, 73, 79, 80]:

1. *Процес побудови персоналізованої траєкторії:*

Етап діагностики:

Проведення вхідного комплексного тестування, глибокий аналіз потенціалу учня, створення деталізованого освітнього профілю, виявлення індивідуальних особливостей та здібностей.

Формування траєкторії:

Ретельний вибір оптимального освітнього маршруту, підбір персональних навчальних завдань, встановлення індивідуального темпу навчання, врахування особистісних характеристик учня.

2. *Механізми персоналізації:*

Динамічне налаштування складності матеріалу, індивідуалізація стилю подачі інформації, диференціація типів навчальних завдань, гнучке регулювання швидкості навчання, персоналізація формату практичних робіт.

3. *Типи індивідуальних траєкторій:*

За рівнем підготовки:

Створення траєкторії для абсолютних початківців, розробка базового маршруту для учнів з мінімальним досвідом, формування просунутого треку для підготовлених програмістів, конструювання дослідницької траєкторії для обдарованих учнів.

За професійним спрямуванням:

Побудова траєкторії для веб-розробників, створення маршруту для майбутніх аналітиків даних, розробка освітньої лінії для дослідників штучного інтелекту, формування напряму для фахівців з кібербезпеки, конструювання траєкторії для спеціалістів з автоматизації.

4. *Структура адаптивних завдань:*

Розробка глибокого теоретичного блоку, створення практичних випробувань різної складності, впровадження інтелектуальної системи

підказок, забезпечення варіативності завдань, розвиток механізму негайного зворотного зв'язку.

5. *Технологія створення завдань:*

Проведення глибокої діагностики вхідного рівня підготовки, чітке формулювання освітньої мети, побудова завдань з багаторівневою структурою складності, розробка комплексної системи оцінювання, створення адаптивного мультимедійного контенту, підготовка детальних методичних рекомендацій.

6. *Інструменти реалізації:*

Впровадження сучасних адаптивних платформ навчання, інтеграція систем штучного інтелекту для аналізу прогресу, використання автоматизованих систем тестування, створення інтерактивних освітніх середовищ.

7. *Система моніторингу:*

Проведення глибокої вхідної діагностики, здійснення об'єктивного проміжного оцінювання, реалізація підсумкової комплексної атестації, безперервний трекінг індивідуального прогресу учня.

8. *Методологічні принципи:*

Максимальна індивідуалізація навчального процесу, реалізація випереджаючого навчання, забезпечення практико-орієнтованості освіти, створення умов для неперервного розвитку, підтримка варіативності освітнього контенту.

Очікувані результати створення адаптивних завдань та персоналізованих навчальних траєкторій:

1. Результати на рівні освітнього процесу включають персоналізацію навчання з індивідуальним підходом до кожного учня, врахуванням особистих здібностей та гнучкою адаптацією освітнього контенту.

2. Підвищення мотивації та ефективності виражається у зростанні пізнавального інтересу до програмування, покращенні якості засвоєння матеріалу та скороченні часу на опанування складних тем.

3. Розвиток компетентностей передбачає формування професійних навичок програмування, розвиток критичного та алгоритмічного мислення, набуття практичного досвіду проєктної діяльності.

4. Результати особистісного розвитку охоплюють виявлення та підтримку обдарованості, професійне самовизначення, формування впевненості в собі та розвиток здатності до самонавчання.

5. Стратегічні освітні результати полягають у підготовці конкурентоспроможних кадрів, скороченні розриву між шкільною освітою та ринком праці, створенні сучасного освітнього середовища.

2.2.3. Розробка системи оцінювання та зворотного зв'язку

Система оцінювання та зворотного зв'язку при вивченні мови програмування Python – це комплекс методів і прийомів, спрямованих на постійне відстеження прогресу учнів, надання своєчасного та конструктивного зворотного зв'язку, а також об'єктивне оцінювання рівня їхніх знань, умінь та навичок [81-84].

Система оцінювання та зворотного зв'язку для персоналізованого навчання Python у 10-11 класах:

1. *Концептуальна модель оцінювання:* комплексна діагностика індивідуального прогресу учня, максимально об'єктивне вимірювання рівня компетентностей, забезпечення неперервного зворотного зв'язку, мотивація до постійного розвитку.

2. *Структура оцінювання:* багаторівнева система оцінки включає вхідний діагностичний рівень (0-20 балів), поточний моніторинг прогресу (0-40 балів), підсумковий рівень компетентності (0-40 балів).

3. *Компоненти оцінювання:* кількісні показники містять тестування теоретичних знань, практичні завдання з кодування, проєктну роботу, самостійні розробки, участь у додаткових активностях. Якісні показники включають креативність вирішення завдань, складність реалізованих проєктів, швидкість навчання, здатність до самостійного пошуку рішень.

4. *Технологія оцінювання:* використовує автоматизовану систему перевірки коду, інтерактивні онлайн-тести, експертну оцінку викладача, взаємооцінювання проєктів, самооцінювання учня.

Таблиця 1

Розподіл балів по видах робіт

Вид роботи	**Максимальна кількість балів**	**Критерії оцінювання**
Теоретичне тестування	20	Повнота знань, точність відповідей, системне мислення
Практичні завдання з програмування	15	Коректність коду, ефективність алгоритму, дотримання стандартів
Контрольні роботи	15	Складність вирішених завдань, самостійність виконання
Індивідуальні проєкти	20	Креативність, практична значущість, презентація
Домашні завдання	10	Регулярність, повнота виконання, ініціативність
Додаткові активності	10	Участь в олімпіадах, конкурсах
Самостійні дослідження	10	Глибина опрацювання теми, науковий підхід
Підсумковий проєкт	40	Комплексність, інноваційність, професійний рівень

5. *Критерії оцінювання:* теоретичний блок оцінює розуміння базових концепцій мови, знання синтаксису, вміння пояснювати алгоритми. Практичний блок враховує написання коректного та ефективного коду, дотримання стандартів програмування, вирішення складних завдань. Проєктний блок оцінює складність проєкту, самостійність розробки, практичну значущість та презентацію.

6. *Механізм зворотного зв'язку:* передбачає щотижневий аналіз прогресу, персоналізовані рекомендації, корекцію навчальної траєкторії, психолого-педагогічний супровід.

7. *Технологічне забезпечення:* включає особистий кабінет учня, автоматизовану систему аналітики, чат-бот технічної підтримки, інтерактивні освітні матеріали.

8. *Мотиваційний компонент:* система заохочення складається з нарахування додаткових балів, цифрових сертифікатів, рейтингової таблиці, можливості позаконкурсного вступу.

9. *Результативність системи:* очікувані показники включають підвищення мотивації на 70%, якість знань до 85%, збільшення швидкості навчання на 40%, практичну готовність до професійної діяльності.

10. *Додаткові переваги:* гнучка адаптація під індивідуальні особливості, неперервний моніторинг, випереджаюче навчання, формування soft skills.

11. *Підсумкова оцінка* формується як інтегральний показник: 90-100 балів – високий рівень (відмінно), 75-89 балів – достатній рівень (добре), 60-74 балів – середній рівень (задовільно), менше 60 балів – потребує додаткової підготовки.

Таблиця 2

Шкала переведення інтегрального показника в оцінки 12-бальної системи

Бали за інтегральним показником	**Оцінка за 12-бальною шкалою**	**Рівень навчальних досягнень**
96-100	12	Творчий, особливо високий
90-95	11	Творчий, високий
85-89	10	Високий
80-84	9	Достатній, високий
75-79	8	Достатній, середній
70-74	7	Достатній
65-69	6	Середній
60-64	5	Середній, достатній
55-59	4	Середній, низький
50-54	3	Початковий
35-49	2	Підготовчий
20-34	1	Низький
Менше 20	Не атестований	Потребує повної додаткової підготовки

Характеристика рівнів:

1-3 бали: учень розпізнає окремі об'єкти, елементи, має фрагментарні уявлення про предмет вивчення.

4-6 балів: відтворення незначної частини навчального матеріалу, володіння базовими знаннями.

7-9 балів: володіння основними поняттями, здатність виконувати завдання за зразком.

10-12 балів: системні знання, креативне мислення, здатність до самостійного творчого опрацювання матеріалу.

Додаткові рекомендації:

- Для отримання високої оцінки необхідно набрати більше 85 балів.
- Критичний мінімум для позитивної атестації - 60 балів.

- Індивідуальний план підтримки формується для учнів, що набрали менше 60 балів.

Технологічне забезпечення персоналізованого навчання

Технологічне забезпечення персоналізованого навчання Python – це комплексна інтегрована система технологічних рішень, що включає [76, 80, 83]:

1. Компоненти системи: особистий електронний кабінет учня, автоматизована система аналітики навчальних досягнень, інтерактивне навчальне середовище, система підтримки та консультування, модуль неперервного моніторингу прогресу.

2. Технологічні інструменти: хмарні платформи дистанційного навчання, штучний інтелект для персоналізації контенту, адаптивні навчальні алгоритми, система контролю та перевірки коду, інтерактивні навчальні симулятори.

3. Функціональні можливості: формування індивідуальної траєкторії навчання, діагностика рівня підготовки, автоматична генерація навчальних завдань, миттєвий зворотний зв'язок, накопичення портфоліо учня.

4. Технічні характеристики: крос-платформність, висока швидкодія, кросбраузерність, адаптивний дизайн, безпека даних.

5. Ключові переваги: індивідуальний підхід, гнучкість навчання, об'єктивність оцінювання, постійний моніторинг, мотивація до самонавчання.

2.3.1. Вибір освітньої платформи

За результатами опитування вчителів інформатики ліцеїв м. Кривого Рогу та інших міст України, було виявлено три найбільш ефективні платформи для навчання програмування Python. Розглянемо їх у порядку популярності та ефективності використання.

1. Google Classroom + Google Colab

Лідер серед освітніх платформ, який отримав найвищі оцінки від педагогів. Вчителі відзначають, що ця комбінація забезпечує найкращий баланс між функціональністю та простотою використання.

Причини вибору платформи вчителями: відсутність необхідності встановлення додаткового програмного забезпечення, звичний інтерфейс Google сервісів для учнів, можливість швидко ділитися матеріалами та надавати зворотний зв'язок, безперебійна робота навіть при слабкому інтернет-з'єднанні, зручність організації дистанційного навчання.

Учні особливо цінують можливість працювати з будь-якого пристрою та автоматичне збереження всіх змін. Це значно спрощує процес навчання та робить його більш гнучким.

2. Moodle + Python плагіни

Друге місце посідає Moodle з відповідними плагінами для Python. Платформа особливо популярна серед навчальних закладів, які мають власні сервери та технічну підтримку.

Вчителі відзначають низку важливих переваг персоналізованого навчання: повний контроль над навчальним процесом, розширені можливості для створення тестів, детальну аналітику успішності, можливість інтеграції з іншими системами навчального закладу. Такий комплексний підхід дозволяє викладачам більш ефективно управляти освітнім процесом, точніше оцінювати прогрес кожного учня та забезпечувати індивідуальний підхід до навчання.

Moodle дозволяє створювати складні навчальні курси з різними видами активностей та автоматизованою перевіркою завдань.

3. Eolymp

Третє місце посідає спеціалізована платформа Eolymp, яка особливо цінується вчителями, що готують учнів до олімпіад з програмування. Платформа має низку важливих переваг: величезну базу задач різного рівня складності, автоматичну перевірку розв'язків, можливість організації змагань, підтримку української мови. Ці особливості роблять Eolymp надзвичайно корисним інструментом для викладачів інформатики та учнів, які прагнуть вдосконалити свої навички програмування та підготуватися до олімпіадних випробувань.

Eolymp незамінний для підготовки до олімпіад та для учнів, які прагнуть поглибити свої знання в програмуванні.

Особливості впровадження в навчальний процес

Більшість опитаних вчителів використовують комбінацію платформ:

- Google Classroom + Colab як основну платформу для всього класу.
- Eolymp для додаткових завдань та роботи з обдарованими учнями.
- Moodle для шкіл з розвиненою IT-інфраструктурою.

Рекомендації щодо вибору платформи

На основі досвіду вчителів, можна виділити такі критерії вибору:

1. Для звичайних класів: Google Classroom + Colab.

2. Для спеціалізованих класів з поглибленим вивченням інформатики: комбінація всіх трьох платформ.

3. Для підготовки до олімпіад: Eolymp як додатковий ресурс.

Вибір платформи залежить від технічних можливостей школи, рівня підготовки учнів, цілей навчання, наявності технічної підтримки. Кожен з цих факторів відіграє важливу роль у прийнятті рішення про впровадження освітньої платформи, адже необхідно враховувати не лише технічні аспекти, але й освітній потенціал конкретного навчального середовища для ефективного навчання програмуванню.

Найоптимальнішим рішенням, за відгуками вчителів, є використання Google Classroom + Colab як базової платформи з поступовим додаванням інших інструментів залежно від потреб класу та окремих учнів.

2.3.2. Аналіз можливостей автоматизованої перевірки коду

Аналіз можливостей автоматизованої перевірки коду на рекомендованих платформах навчання:

1. Google Classroom + Colab

Методи перевірки: вбудовані assert-перевірки, unit-тести через unittest/pytest, автоматична перевірка стилю коду (PEP 8), перевірка через порівняння виводу.

Переваги автоперевірки: миттєвий фідбек учням, можливість багаторазових спроб, прозорість критеріїв, інтеграція з системою оцінювання.

Обмеження: обмежені можливості перевірки складних алгоритмів, потреба в ручному створенні тестів, відсутність перевірки ефективності коду.

2. Moodle + Python плагіни

Основні інструменти: Virtual Programming Lab (VPL), CodeRunner, Adaptive Quiz.

Можливості VPL: автоматичне тестування, перевірка плагіату, обмеження ресурсів, детальна звітність.

Переваги: розширені можливості тестування, безпека виконання коду, детальна аналітика, гнучке оцінювання.

Обмеження: складність налаштування, потреба в технічній підтримці, ресурсомісткість.

3. Eolymp

Особливості системи: готові тестові набори, автоматична перевірка на різних наборах даних, перевірка часу виконання, перевірка використання пам'яті.

Типи перевірок: порівняння з еталоном, перевірка форматування виводу, тестування граничних випадків, стрес-тестування.

Таблиця 3

Порівняння можливостей автоматизованої перевірки коду на рекомендованих платформах навчання

Характеристика	**G.Classroom+Colab**	**Moodle+VPL**	**Eolymp**
Автотести	Базові	Розширені	Повні
Перевірка ефективності	Обмежена	Середня	Повна
Плагіат	Ні	Так	Частково
Статистика	Базова	Детальна	Середня
Персоналізація	Висока	Висока	Низька

Рекомендації щодо вибору:

1. Для початкового навчання: використання Google Classroom + Colab з фокусом на розумінні основ та простотою використання.

2. Для поглибленого вивчення: впровадження Moodle + VPL з більшими можливостями для персоналізації та детальнішою аналітикою.

3. Для підготовки до олімпіад: застосування Eolymp з фокусом на ефективності та змагальним елементом.

2.3.3. Впровадження елементів адаптивного навчання

Впровадження елементів адаптивного навчання:

1) з використанням Google Classroom + Google Colab

- Архітектура інтеграційного рішення являє собою комплексну технологічну інфраструктуру, що включає Google Classroom для управління навчальним процесом, Google Colab як середовище виконання коду та інтерактивного навчання, GitHub як репозиторій навчальних матеріалів та Google Forms для діагностичного тестування.

- Методологія впровадження складається з послідовних етапів. На діагностичному етапі створюються форми Google для вхідного тестування, здійснюється оцінка базового рівня володіння Python та формування

індивідуального профілю учня. Персоналізація траєкторії передбачає автоматичне налаштування складності завдань, генерацію персоналізованих завдань та динамічну адаптацію контенту.

- Технічна реалізація включає використання Google Classroom для створення курсів з Python, розподілу учнів за групами та моніторингу активності. Google Colab забезпечує роботу з інтерактивними зошитами, що містять персональні завдання, негайну перевірку коду та вбудовану систему зворотного зв'язку.
- Система моніторингу та аналітики дозволяє відстежувати прогрес кожного учня, збирати статистику виконання завдань та коригувати траєкторію навчання. Ключовими перевагами такого підходу є індивідуальний підхід, гнучка адаптація складності, підтримка різних стилів навчання та мотивація через персоналізацію.
- Технічні вимоги включають використання Python 3.7+, бібліотек pandas та numpy, наявність облікового запису Google Workspace та базових навичок роботи з Colab. Рекомендації з впровадження передбачають пілотний запуск в одному класі, поступову інтеграцію, підготовку вчителів та забезпечення технічної підтримки.
- Такий комплексний підхід дозволяє створити ефективне персоналізоване освітнє середовище для вивчення Python, яке адаптується під індивідуальні потреби та можливості кожного учня.

2) *з використанням Moodle + Python-плагінів* включає комплексний підхід до організації освітнього процесу.

- Архітектура інтеграційного рішення являє собою комплексну систему на базі платформи Moodle, яка виступає централізованим середовищем навчання та забезпечує повний цикл управління освітнім контентом, обліком прогресу учнів, комунікацією та моніторингом результатів.
- Діагностичний блок системи включає вхідне комплексне тестування рівня підготовки, формування індивідуального профілю учня та виявлення освітніх потреб і здібностей. Механізми адаптації передбачають динамічне налаштування складності навчальних модулів, персоналізацію темпу навчання та індивідуальний підбір навчальних траєкторій.
- Структура плагінів для Moodle складається з декількох ключових компонентів: плагіну діагностики рівня підготовки, який забезпечує

автоматизовану систему вхідного тестування; адаптивного модуля контролю знань з динамічною зміною складності завдань; трекінгу навчального прогресу для моніторингу індивідуальних досягнень; плагіну персоналізації контенту, що враховує індивідуальний стиль навчання; та профорієнтаційного модуля для діагностики професійних нахилів.

- Алгоритм впровадження включає підготовчий етап з аналізом матеріально-технічної бази та навчанням викладацького складу, технічну реалізацію з встановленням Moodle та розробкою Python-плагінів, а також запуск пілотного проєкту для апробації та корекції системи.
- Очікувані результати такої персоналізованої системи навчання включають підвищення мотивації учнів, індивідуалізацію навчального процесу, розвиток практичних навичок програмування та професійну орієнтацію в IT-галузі.
- Технічні вимоги передбачають використання Moodle 3.9+, Python 3.7+, додаткових бібліотек для аналізу даних та стабільного інтернет-з'єднання. Рекомендована структура Python-плагінів включає модуль діагностики, систему адаптивного оцінювання, аналітичний блок прогресу, генератор персоналізованих завдань та інтерфейс комунікації з учнем.
- Додаткові переваги системи – її гнучкість, масштабованість, можливість постійного вдосконалення та підтримка різних освітніх стратегій. Водночас існують і певні обмеження, зокрема необхідність технічної підтримки, потреба в постійному оновленні контенту та залежність від технічного забезпечення школи.
- Такий комплексний підхід дозволяє створити сучасне персоналізоване освітнє середовище, яке максимально адаптується під індивідуальні потреби та можливості кожного учня.

3) *з використанням на платформі Eolymp* передбачає комплексний підхід до організації освітнього процесу з використанням потужного функціоналу онлайн-платформи для програмування.

- Технологічні переваги Eolymp: платформа є спеціалізованою для навчання програмуванню, має вбудовану систему тестування та перевірки коду, пропонує завдання різної складності, підтримує популярні мови програмування, забезпечує статистичний облік прогресу учнів.

- Компоненти персоналізації: діагностичний блок включає вхідне тестування рівня підготовки, автоматичний розподіл за рівнями складності, первинну оцінку програмістських навичок. Механізми адаптації передбачають динамічне налаштування складності завдань, побудову індивідуальної траєкторії навчання, персоналізований підбір контенту.
- Структура впровадження: профіль учня передбачає створення індивідуального облікового запису, прив'язку до навчального закладу/класу, формування персонального портфоліо. Рівні складності завдань включають початковий, середній, просунутий та дослідницький рівні. Траєкторія навчання забезпечує послідовне проходження модулів, відкриття нових завдань після успішного виконання, накопичення балів та досягнень.
- Система мотивації містить рейтингову систему, нарахування балів за складність, створення індивідуального портфоліо, визначення прогресу та досягнень. Методичне забезпечення включає розробку навчально-методичного комплексу, підготовку завдань різної складності, створення методичних рекомендацій, підготовку додаткових матеріалів.
- Алгоритм впровадження: підготовчий етап включає реєстрацію навчального закладу, створення групових класів, налаштування індивідуальних профілів, методичну підготовку викладачів. Технічна реалізація передбачає створення групових курсів, налаштування індивідуальних траєкторій, підготовку діагностичних матеріалів, інтеграцію з шкільною документацією.
- Навчальний процес охоплює проведення вхідного тестування, розподіл за рівнями складності, персоналізацію навчальних завдань, моніторинг індивідуального прогресу.
- Очікувані результати: підвищення мотивації до вивчення Python, індивідуалізація навчального процесу, розвиток практичних навичок програмування, професійна орієнтація в IT-галузі, формування алгоритмічного мислення.
- Технічні вимоги включають наявність персональних комп'ютерів, стабільне інтернет-з'єднання, базові навички роботи з комп'ютером, реєстрацію на платформі Eolymp.
- Додаткові переваги передбачають можливість дистанційного навчання, постійне оновлення контенту, міжнародний рівень завдань, підготовку до олімпіад з програмування.

- Обмеження та виклики стосуються необхідності технічної підтримки, потреби в методичному супроводі, залежності від технічного забезпечення.
- Рекомендована структура впровадження включає діагностику початкового рівня, формування індивідуальної траєкторії, послідовне виконання завдань, моніторинг прогресу, корекцію навчальної траєкторії.

2.4. Методичні рекомендації щодо впровадження

Загальні принципи впровадження:

- На діагностичному етапі здійснюється проведення вхідного тестування рівня підготовки, виявлення індивідуальних освітніх потреб, діагностика пізнавальних стилів учнів та визначення їхніх професійних інтересів.
- Методологічні підходи передбачають реалізацію особистісно-орієнтованого навчання, розробку індивідуальної траєкторії розвитку, забезпечення адаптивності освітнього контенту та практико-орієнтовану спрямованість навчального процесу.

Методичні рекомендації щодо впровадження персоналізованого навчання Python у 10-11 класах ліцеїв на рекомендованих платформах:

1) Google Classroom + Colab:

Організаційно-методичне забезпечення:

1. Підготовка вчителів:
- Курси підвищення кваліфікації з digital-педагогіки.
- Навчання роботі з Google Classroom.
- Опанування інструментарію Google Colab.
- Вивчення методик персоналізованого навчання.

2. Технічне налаштування:
- Створення корпоративних облікових записів.
- Формування навчальних груп.
- Налаштування спільного доступу до матеріалів.
- Інтеграція додаткових освітніх сервісів.

3. Структура навчального контенту:
- Модульна побудова курсу.
- Різнорівневі завдання.
- Інтерактивні навчальні матеріали.
- Відеоінструкції та демонстраційні приклади.

4. Механізми персоналізації:

- Динамічна зміна складності завдань.
- Адаптивний підбір навчальних кейсів.
- Врахування індивідуального темпу навчання.
- Підтримка різних стилів сприйняття інформації.

5. Система оцінювання:

- Накопичувальна бальна система.
- Критеріальне оцінювання.
- Формувальне оцінювання.
- Портфоліо навчальних досягнень.

2) Moodle + Python плагіни:

Методичні рекомендації:

1. Підготовка платформи:

- Встановлення Moodle 3.9+.
- Розробка custom Python-плагінів.
- Налаштування системи адаптивного навчання.
- Інтеграція аналітичних модулів.

2. Функціональні можливості:

- Automatic діагностика рівня підготовки.
- Генерація індивідуальних траєкторій.
- Моніторинг навчального прогресу.
- Адаптивний контроль знань.

3. Технічні вимоги:

- Стабільне інтернет-з'єднання.
- Серверне обладнання.
- Підтримка Python 3.7+.
- Наявність додаткових бібліотек для аналізу даних.

3) EOLYMP:

Методичні рекомендації:

1. Організація навчального процесу:

- Реєстрація навчального закладу.
- Створення групових класів.
- Формування індивідуальних траєкторій.
- Налаштування систем моніторингу.

2. Особливості платформи:

- Використання наявних завдань.

- Налаштування рівнів складності.
- Система рейтингування.
- Накопичення індивідуального портфоліо.

Спільні методичні рекомендації:

1. Психолого-педагогічний супровід:
- Підтримка мотивації учнів.
- Врахування індивідуальних особливостей.
- Створення ситуації успіху.
- Розвиток soft skills.

2. Профорієнтаційний компонент:
- Діагностика професійних нахилів.
- Знайомство з ІТ-професіями.
- Практичні кейси з реальних проєктів.
- Співпраця з ІТ-компаніями.

3. Очікувані результати:
- Підвищення якості навчання.
- Розвиток алгоритмічного мислення.
- Формування практичних навичок програмування.
- Професійна орієнтація в ІТ-галузі.

Курс "Персоналізоване вивчення Python"

Технологічне забезпечення курсу:

• Google Classroom (управління курсом).

• Google Colab (інтерактивне середовище програмування).

• Eolymp (інтерактивне середовище програмування з базу алгоритмічних задач різної складності та підтримкою багатьох мов програмування).

Нами запропонована така структура курсу "Персоналізованого вивчення Python":

1. Вступний модуль (2 тижні, 0-20 балів)

І. *Діагностичне тестування*

• Вхідний рівень знань: онлайн-тестування з базових концепцій програмування; анкетування для виявлення попереднього досвіду в програмуванні, освітніх цілей, сфер зацікавленості в ІТ; діагностична бесіда.

- Виявлення індивідуальних освітніх траєкторій: формування персонального профілю навчання; побудова індивідуального навчального маршруту; встановлення персональних освітніх цілей.

II. *Знайомство з платформою*

1) Реєстрація та налаштування особистого кабінету

- Реєстрація в Google Classroom.
- Налаштування профілю на Eolymp.
- Під'єднання облікового запису Google Colab.

2) Огляд навчальних матеріалів

- Структура курсу.
- Навігація по платформах.
- Огляд інструментів.
- Ознайомлення з системою оцінювання.
- Інструкції з використання платформ.

III. *Базова орієнтація*

1) Мотиваційна бесіда

- Презентація можливостей Python.
- Огляд кар'єрних траєкторій.
- Встановлення групових та індивідуальних очікувань.

2) Формування індивідуального плану навчання

- Визначення короткострокових цілей.
- Узгодження темпу навчання.
- Попередній вибір напряму спеціалізації.

Очікувані результати вступного модуля:

– Діагностичні результати: визначено початковий рівень знань кожного учня, складено індивідуальний профіль компетенцій, сформовано персональну освітню траєкторію.

– Технічна готовність: створено та налаштовано всі необхідні облікові записи (Google Classroom, Google Colab, Eolymp), освоєно базові навички роботи з навчальними платформами, досягнуто розуміння системи навігації та пошуку матеріалів.

– Організаційна підготовка: учні отримали чітке розуміння структури та вимог курсу, ознайомилися з системою оцінювання, засвоїли правила та процедури навчального процесу.

– Мотиваційна складова: сформовано чітке розуміння перспектив вивчення Python, визначено особисті цілі навчання, обрано попередній напрям спеціалізації.

– Планування: складено індивідуальний план навчання, встановлено реалістичні короткострокові цілі, узгоджено оптимальний темп навчання.

2. Основний модуль (24 тижні, 0-40 балів)

Блок 1: Вступ до Python та базове програмування (6 тижнів)

1) Підготовчий модуль (1 тиждень): налаштування середовища розробки, знайомство з Python, базові принципи програмування, вступ до алгоритмічного мислення.

2) Основи мови Python (2 тижні): вивчення змінних та типів даних, опанування базових операцій та виразів, вивчення введення/виведення даних, виконання практичних міні-проєктів.

3) Умовні конструкції та логічне програмування (2 тижні): дослідження операторів порівняння, вивчення розгалужень (if-elif-else), опанування логічних операцій, розв'язання простих алгоритмічних задач.

4) Блок 1 присвячений вступу до Python, де школярі опановують базове програмування через створення простих консольних ігор "Вгадай число", математичних тренажерів для вивчення таблиці множення, генераторів випадкових паролів та нескладних калькуляторів. Застосовуються ігрові методи навчання, розв'язання задач середньої складності, щотижневі міні-завдання та групова робота.

Блок 2: Циклічні конструкції та робота з колекціями (6 тижнів)

1) Цикли та ітерації (2 тижні): вивчення циклів for та while, опанування вкладених циклів, дослідження управління циклами через break та continue, знайомство з генераторами списків.

2) Колекції даних (2 тижні): опрацювання списків, кортежів, множин, словників, вивчення методів роботи з різними типами колекцій.

3) Просунуті техніки роботи з колекціями (1 тиждень): вивчення абстракції списків, опанування функціональних методів, освоєння маніпуляцій з колекціями.

4) Блок 2 розширює можливості через вивчення циклічних конструкцій, де учні розробляють додатки для особистого щоденника, простий облік кишенькових витрат, генератори випадкових цитат та нескладні вікторини.

Методологія включає колективне розв'язання простих алгоритмічних задач, навчальні міні-проєкти та парну роботу.

Блок 3: Функціональне програмування (6 тижнів)

1) Основи функцій (2 тижні): вивчення оголошення та виклику функцій, опанування роботи з параметрами та аргументами, дослідження області видимості змінних, знайомство з документуванням функцій.

2) Просунуте функціональне програмування (2 тижні): опрацювання рекурсії, вивчення lambda-функцій, дослідження функцій вищого порядку, опанування декораторів.

3) Модульність та абстракція (1 тиждень): вивчення принципів модульності, створення власних модулів, огляд вбудованих модулів Python.

4) Блок 3 зосереджений на функціональному програмуванні, де школярі створюють персональні списки справ, генератори привітань, простий аналізатор текстів та нескладні інтерактивні вправи. Робота відбувається через творчі завдання, невеликі групові проєкти та навчальні ігри.

Блок 4: Об'єктно-орієнтоване програмування (6 тижнів)

1) Базові принципи ООП (2 тижні): вивчення класів та об'єктів, опанування конструкторів, дослідження методів класів, знайомство з інкапсуляцією.

2) Наслідування та поліморфізм (2 тижні): опрацювання базових та похідних класів, вивчення множинного наслідування, дослідження перевизначення методів, знайомство з абстрактними класами.

3) Поглиблене вивчення ООП (1 тиждень): вивчення статичних та класових методів, порівняння composition та inheritance, огляд патернів проєктування, поглиблена робота з класами.

4) Блок 4 присвячений основам об'єктно-орієнтованого програмування з розробкою навчальних симуляторів, додатків для обліку шкільних оцінок, нескладних ігрових додатків та програм для особистого використання. Методи включають командну роботу в групах, парне програмування та створення навчальних проєктів.

Блок 5: Робота з даними та зовнішніми джерелами (6 тижнів)

1) Файлові операції (2 тижні): вивчення читання та запису файлів, опанування роботи з текстовими файлами, дослідження форматів CSV та JSON, опрацювання обробки виключних ситуацій.

2) Робота з базами даних (2 тижні): опанування основ SQLite, вивчення взаємодії Python з базами даних, дослідження принципів ORM, розгляд практичних кейсів.

3) Робота з зовнішніми API (1 тиждень): вивчення HTTP-запитів, опанування парсингу даних, дослідження взаємодії з зовнішніми сервісами, знайомство з бібліотеками requests та json.

4) Блок 5 досліджує роботу з даними через створення простих додатків для перевірки прогнозу погоди, конвертації валют, обліку особистих досягнень та нескладних інформаційних додатків. Особливістю навчання є творчий підхід, практична спрямованість, створення власних міні-проєктів та поступове ускладнення завдань. Додаткові методи навчання передбачають індивідуальний супровід, створення власного портфоліо проєктів, заохочення креативності та підтримку інтересу до програмування.

Очікувані результати основного модуля:

- Блок 1 (Вступ до Python та базове програмування): учні оволодівають базовими навичками програмування, створюють прості консольні ігри, вивчають основні конструкції мови Python, розвивають алгоритмічне мислення через розв'язання практичних завдань.
- Блок 2 (Циклічні конструкції та робота з колекціями): учні опановують циклічні структури, вивчають різні типи колекцій, розробляють додатки для обліку особистих даних, генерують випадкові послідовності, поглиблюють навички маніпуляції з даними.
- Блок 3 (Функціональне програмування): школярі вивчають принципи створення та використання функцій, опановують рекурсію, lambda-функції, декоратори, розвивають навички модульного програмування, створюють різноманітні утилітарні додатки.
- Блок 4 (Об'єктно-орієнтоване програмування): учні вивчають основи ООП, створюють класи, досліджують наслідування та поліморфізм, розробляють складніші додатки з використанням об'єктно-орієнтованого підходу, вивчають базові патерни проєктування.
- Блок 5 (Робота з даними та зовнішніми джерелами): учні опановують навички роботи з файлами, базами даних, вивчають принципи взаємодії з зовнішніми API, створюють додатки для обробки та аналізу даних, розвивають практичні навички роботи з інформаційними системами.

- Загальні очікувані результати: учні набувають комплексних навичок програмування, розвивають логічне та алгоритмічне мислення, опановують різні парадигми та підходи до розробки програмного забезпечення, створюють власне портфоліо проєктів, підвищують рівень технічної креативності.

3. Підсумковий модуль (4 тижні, 0-40 балів)

Це завершальний етап навчального курсу з програмування для школярів старшої школи, який має на меті комплексну оцінку набутих навичок та підготовку до подальшого професійного розвитку.

Структура модуля включає комплексний проєкт, підсумкову атестацію та формування портфоліо. Комплексний проєкт передбачає демонстрацію самостійності, креативності та технічних навичок у розробці власного програмного продукту.

Варіанти проєктів охоплюють різноманітні напрямки:

- Соціально-корисні додатки для волонтерської допомоги громаді, системи обліку екологічних ініціатив, платформи обміну навчальними матеріалами, додатки для підтримки людей з особливими потребами.
- Освітні проєкти включають інтерактивні навчальні додатки з математики, системи тестування знань, додатки для вивчення іноземної мови, віртуальні помічники для підготовки до ЗНО.
- Ігрові проєкти представлені освітніми RPG-іграми з вивчення історії, інтелектуальними вікторинами, симуляторами професійної діяльності, навчальними квестами з програмування.
- Допоміжні застосунки: персональні трекери корисних звичок, додатки для планування часу, калькулятори для абітурієнтів, системи моніторингу шкільної успішності.

Етапи роботи над проєктом включають формування ідеї та концепції, проєктування та розробку, тестування та налагодження, підготовку презентації. Критерії оцінювання проєкту враховують оригінальність ідеї, технічну складність, якість коду, дизайн та юзабіліті, презентацію проєкту, захист та пояснення технічних рішень.

Підсумкова атестація передбачає підсумкове тестування з основ програмування, оцінку теоретичних знань, перевірку практичних компетентностей, аналіз рівня логічного мислення, виявлення прогалин у знаннях.

Формування портфоліо включає систематизацію всіх навчальних проєктів, структурований опис виконаних робіт, рефлексію про особистий прогрес, аналіз отриманих навичок, формування індивідуальної траєкторії розвитку, розробку плану подальшого вивчення програмування.

Очікувані результати підсумкового модуля:

- Комплексний проєкт: учні демонструють здатність самостійно розробляти повноцінні програмні продукти, застосовувати набуті технічні навички, втілювати креативні ідеї в практичні додатки з високим соціальним або освітнім потенціалом.
- Соціально-корисні додатки: школярі створюють інноваційні рішення для supporting громадських ініціатив, розвивають навички соціально орієнтованого програмування, демонструють розуміння суспільних потреб через технологічні інструменти.
- Освітні проєкти: учні розробляють інтерактивні навчальні додатки, показують здатність перетворювати складні освітні концепції в зручні та зрозумілі програмні рішення, застосовують міжпредметні знання.
- Ігрові проєкти: учні експериментують з ігровими механіками, демонструють креативність у поєднанні навчальних цілей з інтерактивним геймінгом, розвивають навички проєктування складних програмних систем.
- Допоміжні застосунки: школярі створюють практичні інструменти для особистої ефективності, показують уміння розв'язувати повсякденні задачі через програмування, розвивають навички user-friendly дизайну.
- Підсумкова атестація: учні підтверджують комплексні знання з програмування, демонструють системне мислення, здатність розв'язувати складні алгоритмічні завдання, показують глибину технічної підготовки.
- Портфоліо: учні систематизують власний навчальний досвід, презентують індивідуальну траєкторію професійного розвитку, формують усвідомлене бачення майбутньої кар'єри в ІТ-галузі, розкривають потенціал особистісного зростання.
- Загальні результати: школярі набувають комплексних професійних компетентностей, розвивають критичне мислення, демонструють здатність до самостійної технологічної творчості, формують впевненість у власних професійних можливостях.

PythonEdu: онлайн-платформа з практичними рекомендаціями для вчителів з розробки персоналізованих навчальних блоків

Розроблена нами онлайн-платформа PythonEdu є простим методологічним рішенням для вчителів інформатики, спрямованим на створення індивідуальних навчальних траєкторій у вивченні програмування Python. Платформа пропонує простий підхід до адаптації навчальних матеріалів відповідно до індивідуальних особливостей, рівня підготовки та пізнавальних інтересів кожного учня.

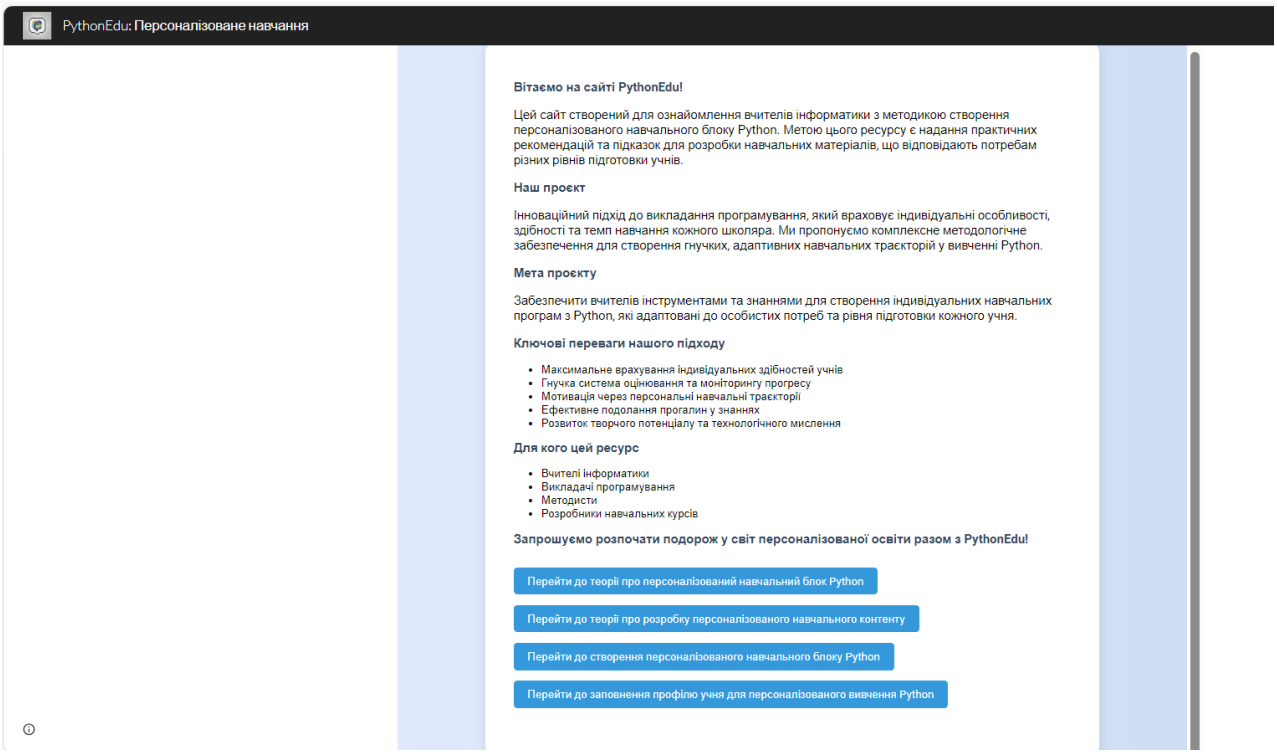

Рис. 1. Головна сторінка онлайн-платформа PythonEdu

Функціональні можливості платформи (рис. 1) включають:

1. Теоретичний блок :

- Методичні матеріали з розробки персоналізованого навчального контенту(рис. 2).
- Рекомендації зі створення персоналізованих навчальних блоків Python.

2. Інтерактивний редактор з розширеним функціоналом:

- Інтуїтивно зрозумілий інтерфейс для заповнення профілю учня (рис. 3).
- Гнучкі налаштування персоналізованого навчального блоку.
- Підтримка введення даних у текстовому та HTML-форматах.

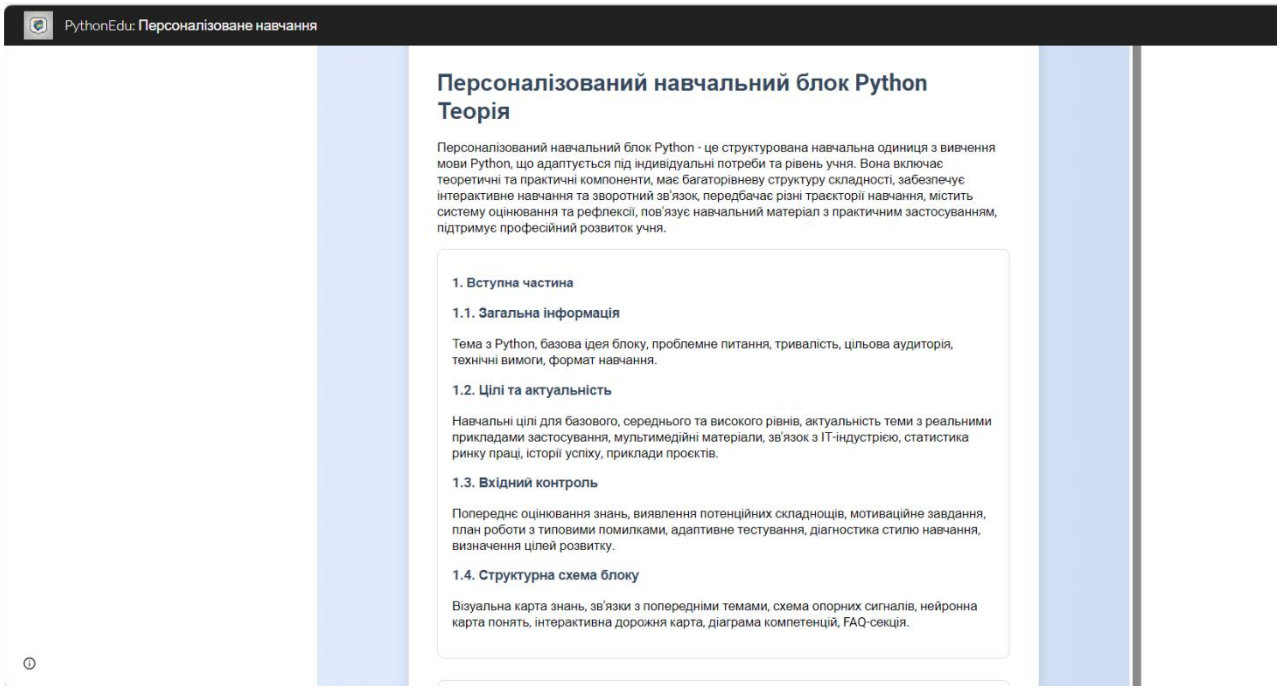

Рис. 2. Методичні матеріали з розробки персоналізованого навчального блоку Python

Редактор платформи PythonEdu забезпечує:

- Швидке створення індивідуальних навчальних матеріалів.
- Гнучку систему налаштування контенту.
- Зручність у використанні.
- Миттєве збереження результатів роботи.

Результатом роботи в редакторі є повноцінна веб-сторінка, яка може бути легко інтегрована в освітній процес та слугувати персоналізованим навчальним інструментом для конкретного учня.

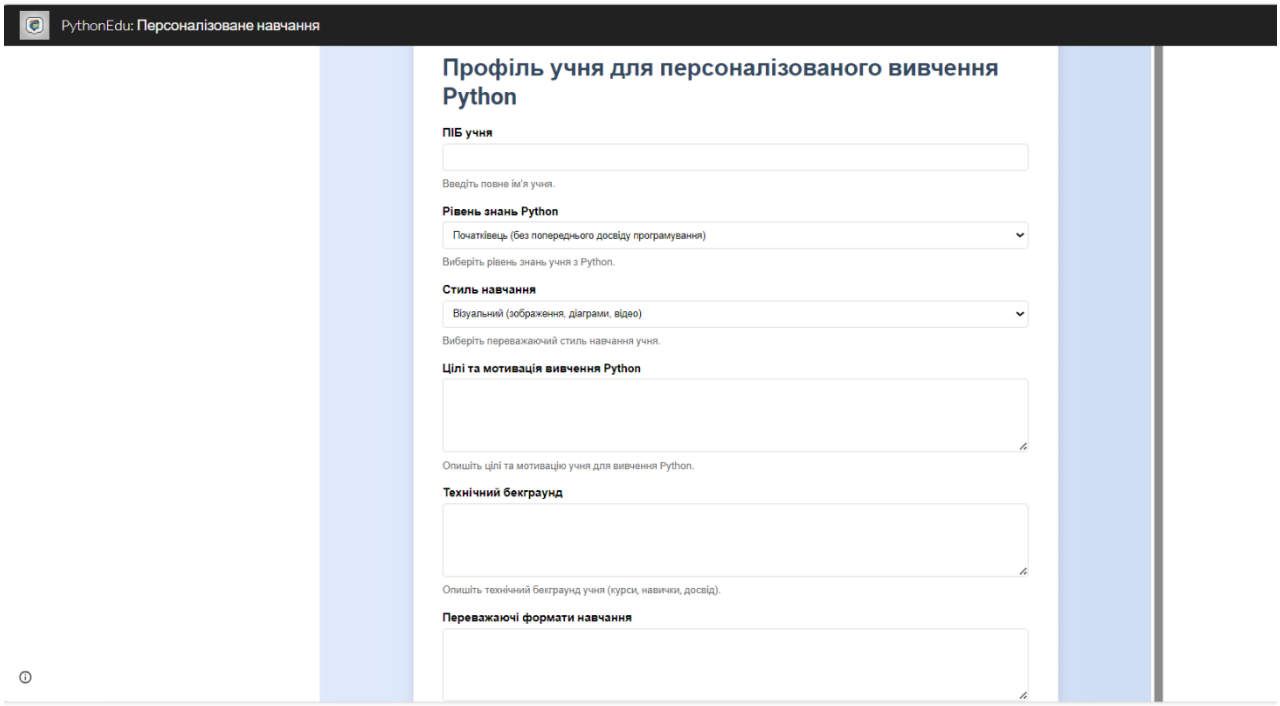

Рис. 3. Інтерактивний редактор для заповнення профілю учня

ВИСНОВКИ ДО РОЗДІЛУ 2

У розділі 2 проаналізовано методичну систему персоналізованого навчання програмування Python для старшокласників.

Проаналізовано технологічне забезпечення з використанням хмарних платформ, штучного інтелекту та інтерактивних симуляторів для підтримки індивідуальних траєкторій навчання.

Вивчено різні освітні платформи: Google Classroom, Moodle, Eolymp для ефективного навчання програмування.

Розглянуто можливості автоматизованої перевірки коду та розроблено методичні рекомендації для впровадження персоналізованого навчання Python.

Створено структуру курсу персоналізованого вивчення Python. Створено комплексний підхід до формування індивідуальних навчальних траєкторій, що враховує різні рівні підготовки та пізнавальні інтереси учнів.

Створено онлайн-платформу з практичними рекомендаціями для вчителів. Розроблено методичний інструментарій для ефективної розробки персоналізованих навчальних блоків Python, що забезпечує гнучкість та адаптивність освітнього процесу.

ВИСНОВКИ

У кваліфікаційній роботі представлено теоретичне узагальнення та практичне вирішення проблеми персоналізації навчання Python для старшокласників.

Теоретичні результати включають концептуальне розуміння персоналізації навчання як інноваційного підходу, який враховує індивідуальні траєкторії та пізнавальні інтереси учнів.

Розроблено концептуальну модель навчання, що поєднує особистісно-орієнтований, диференційований та компетентнісний підходи. Визначено ключові принципи персоналізації: індивідуальний підхід, гнучкість траєкторії, активне включення учня та діагностика стартового рівня.

Методичні результати представляють систему навчання Python з діагностичними, адаптивними та інтерактивними методами.

Обґрунтовано вибір освітніх платформ: Google Classroom, Moodle та Eolymp для різних рівнів підготовки.

Наукова новизна полягає в розробці методичної системи персоналізації навчання Python для старшокласників. Практичне значення – створення методичних рекомендацій та онлайн-інструментарію для вчителів з проєктування індивідуальних освітніх маршрутів.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Дущенко О. Сучасний стан цифрової трансформації освіти. Physical and Mathematical Education. 2021. Т. 28, № 2. С. 40–45. URL: https://doi.org/10.31110/2413-1571-2021-028-2-007.

2. Биков В. Ю. Цифрова трансформація суспільства і розвиток комп'ютерно-технологічної платформи освіти і науки України. Інформаційно-цифровий освітній простір України: трансформаційні процеси і перспективи розвитку : матеріали методол. семінару НАПН України (м. Київ, 4 квіт. 2019 р.). 2019. С. 20–25.

3. Биков В. Ю. Суспільство знань і освіта 4.0. Освіта для майбутнього у світлі викликів XXI століття (польська, edukacja w kontekście zmian cywilizacyjnych). Bydgoszcz : Widawnictwo Uniwersytetu Kazimierza Wielkiego, 2017. С. 30–45. URL: https://lib.iitta.gov.ua/708567/1/%D0%91%D0%B8%D0%BA%D0%BE%D0%B2%20%D0%92_%D1%81%D1%82%D0%B0%D1%82%D1%82%D1%8F2017.pdf

4. Shemshack A., Spector J. M. A systematic literature review of personalized learning terms. Smart Learning Environments. 2020. Vol. 7, no. 1. URL: https://doi.org/10.1186/s40561-020-00140-9.

5. Брюховецький А., Остапенко Л. Програмування як складова формування цифрової компетентності сучасного школяра. Інноваційні педагогічні технології в цифровій школі : зб. тез доп. учасників VI Міжнар. наук.-практ. конф. молод. учених, м. Харків, 15–16 трав. 2024 р. / Харків. нац. пед. ун-т ім. Г. С. Сковороди та ін. ; упор.: Н. Пономарьова, Н. Олефіренко, В. Андрієвська. Харків, 2024. С. 154–156.

6. Python in Education. Python in Education. URL: https://education.python.org/.

7. Іванова О. С., Єрмакова С. С., Ковальова О. О. Освіта в умовах цифрової трансформації суспільства. Наукові записки Міжнародного гуманітарного університету. 2024. № 40. С. 99–103. URL: https://doi.org/10.32782/2663-5682/2024/40/21.

8. Strutynska O. Transformation of education under conditions of digital society development: European experience and prospects for Ukraine. Scientific bulletin of South Ukrainian National Pedagogical University named after K. D. Ushynsky. 2020. Vol. 2020, no. 3 (132). P. 71–88. URL: https://doi.org/10.24195/2617-6688-2020-3-9 .

9. Nesterenko O. Computing Education & Technological Trends: a Systematic Review Study // Proceedings of the 13th International Scientific and Practical Programming Conference UkrPROG 2022, Kyiv, Ukraine, October 11-12, 2022. CEUR Workshop Proceedings. Vol. 3501. P. 101–112.

10. Токарева Н. М. Основи педагогічної психології: навч.-метод. посіб. Кривий Ріг, 2013. 223 с.

11. Калениченко Р. А. Педагогічна психологія: навч. посіб. Київ: КНУБА, 2023. 196 с.

12. Ушакова І. М. Вікова психологія: курс лекцій. Харків: НУЦЗУ, 2016. 123 с.

13. Вікова та педагогічна психологія (курс лекцій): навч. посіб. Київ: Центр навч. літератури, 2005. 128 с.

14. Сендер А. Когнітивні стилі як засіб інтенсифікації навчання майбутніх інженерів-програмістів. Молодь і ринок. 2021. № 5/184. URL: https://doi.org/10.24919/2308-4634.2020.

15. Бондар С. І. Когнітивний стиль як індивідуальна стратегія переробки інформації особистістю. Вісник Харківського Університету. Серія "Психологія". 2000. № 498. С. 13–17.

16. Пилявець Н. І., Потапчук Є. М. Сучасні підходи до визначення поняття «когнітивний стиль особистості». Габітус : наук. журн. Одеса, 2021. Вип. 26. С. 117–121.

17. Python як засіб навчання основ алгоритмізації у закладах загальної середньої освіти / Т. П. Кобильник та ін. Information Technologies and Learning Tools. 2022. Т. 89, № 3. С. 16–32. URL: https://doi.org/10.33407/itlt.v89i3.4896.

18. Lee Y., Cho J. The Influence of Python Programming Education for Raising Computational Thinking. International Journal of u- and e- Service, Science and Technology. 2017. Vol. 10, no. 8. P. 59–72.
URL: https://doi.org/10.14257/ijunesst.2017.10.8.06 .

19. Кобильник Т., Когут У., Жидик В. Методичні аспекти вивчення основ алгоритмізації і програмування мовою python у шкільному курсі інформатики у старших класах. Physical and Mathematical Education. 2021. Т. 31, № 5. С. 36–44. URL: https://doi.org/10.31110/2413-1571-2021-031-5-006.

20. Інформатика для 10-11 класів (профільне навчання). URL: https://mon.gov.ua/storage/app/media/zagalna%20serednya/programy-10-11-klas/2018-2019/01/10-11-profilniy-riven.docx.

21. Інформатика. Навчальна програма вибірково-обов'язкового предмету для учнів 10-11 класів загальноосвітніх навчальних закладів (рівень стандарту). URL: https://mon.gov.ua/storage/app/media/zagalna%20serednya/programy-10-11-klas/2018-2019/informatika-standart-10-11.docx.

22. Руденко В. Д., Речич Н. В., Потієнко В. О. Інформатика (профільний рівень): підруч. для 10 кл. закл. загал. серед. освіти. Харків: Вид-во «Ранок», 2019. 256 с.

23. Руденко В. Д., Речич Н. В., Потієнко В. О. Інформатика (профільний рівень): підруч. для 11 кл. закл. загал. серед. освіти. Харків: Вид-во «Ранок», 2019. 256 с.

24. Юрченко А. О., Семеніхіна О. В., Хворостіна Ю. В., Удовиченко О. М., Петренко С. І. Навчання програмувати в старшій школі крізь призму чинних навчальних програм. Фізико-математична освіта. 2019. Вип. 2, № 20. С. 48–55.

25. Semenykhina O. V., Rudenko Y. O. Проблеми навчання програмувати учнів старших класів та шляхи їх подолання. Information Technologies and Learning Tools. 2018. Т. 66, № 4. С. 54.
URL: https://doi.org/10.33407/itlt.v66i4.2149.

26. Руденко В., Жугастров О. Основи алгоритмізації та програмування мовою Python. Ранок, 2019. 192 с.

27. D Cuervo-Cely K., Restrepo-Calle F., J Ramírez-Echeverry J. Effect of Gamification on the Motivation of Computer Programming Students. Journal of Information Technology Education: Research. 2022. Vol. 21. P. 001–023. URL: https://doi.org/10.28945/4917.

28. Скрипченко О. В., Долинська Л. В., Огороднійчук З. В. та ін. Вікова і педагогічна психологія: навч. посіб. 2-ге вид. Київ: Каравела, 2008. 400 с.

29. Докучина Т. О. Мотивація навчання як запорука стимулювання учнів до досягнення успіху. Педагогічна освіта: теорія і практика: зб. наук. праць. Кам'янець-Подільський: ПП Зволейко Д. Г., 2011. Вип. 8. С. 32–37.

30. Занюк С. С. Психологія мотивації: навч. посіб. Київ: Либідь, 2002. 304 с.

31. Іванова Ю. Мотивація як чинник успішного формування навчально-пізнавальної діяльності учнів. Післядипломна освіта в Україні. 2016. № 1. С. 34-37.

32. Якубов С. В. Персоналізоване навчання у загальноосвітній школі. Основні визначення та шлях до впровадження. Директор школи. 2016. № 9. С. 59–72.

33. Нужин О. Теоретичні засади впровадження змішаної форми навчання у підготовці майбутніх іт-фахівців. Grail of Science. 2023. № 28. С. 399–405. URL: https://doi.org/10.36074/grail-of-science.09.06.2023.64.

34. Сікора Я. Б. Персоналізація як підхід до навчання майбутніх ІТ-фахівців. II International Scientific and Practical Conference «Modern Approaches to Problem Solving in Science and Technology» (15–17 листоп. 2023 р.). С. 338–340.

35. Локарєва Г. В., Бажміна Е. А. Персоналізація в освіті: управління студентами власною траєкторією навчання засобами цифрових технологій. Information Technologies and Learning Tools. 2021. Т. 86, № 6. С. 187–207. URL: https://doi.org/10.33407/itlt.v86i6.4103.

36. Інноваційні технології навчання в умовах модернізації сучасної освіти : монографія / за наук. ред. Л. З. Ребухи. Тернопіль: ЗУНУ, 2022. 143 с.

37. Кванденг Г. Переосмислення персоналізованої моделі навчання. Академічні студії. Серія «педагогіка». 2023. № 4. С. 117–121. URL: https://doi.org/10.52726/as.pedagogy/2022.4.17.

38. Дем’яненко В. М. Модель адаптивної навчальної системи інформаційного простору відкритої освіти. Information Technologies and Learning Tools. 2020. Т. 77, № 3. С. 27–38.
URL: https://doi.org/10.33407/itlt.v77i3.3603.

39. Tkachuk H. Model of realization of personalized learning of students of higher education institution. Engineering and Educational Technologies. 2021. Vol. 9, no. 3. P. 8–17. URL: https://doi.org/10.30929/2307-9770.2021.09.03.01.

40. Aeiad E., Meziane F. An adaptable and personalised E-learning system applied to computer science Programmes design. Educational Information Technologies. 2019. Vol. 24. P. 1485–1509. URL: https://doi.org/10.1007/s10639-018-9836-x.

41. Tkachuk H. Model of realization of personalized learning of students of higher education institution. Engineering and Educational Technologies. 2021. Vol. 9, no. 3. P. 8–17. URL: https://doi.org/10.30929/2307-9770.2021.09.03.01.

42. 10 Personalized Learning Examples for High School Students – Schools That Lead. Schools That Lead.
URL: https://www.schoolsthatlead.org/blog/personalized-learning-examples.

43. Personalization of E-Learning: Future Trends, Opportunities, and Challenges / M. Imran et al. International Journal of Interactive Mobile Technologies (iJIM). 2024. Vol. 18, no. 10. P. 4–18.
URL: https://doi.org/10.3991/ijim.v18i10.47053.

44. Мойсеюк Н. Є. Педагогіка: навч. посіб. 5-те вид., допов. і переробл. Київ, 2007. 656 с.

45. Максимюк С. П. Педагогіка: навч. посіб. Київ: Кондор, 2005. 667 с.

46. Реалізація індивідуалізації та персоналізації навчання засобами moodle / К. Осадча та ін. *Молодь і ринок*. 2021. № 1/187.
URL: https://doi.org/10.24919/2308-4634.2021.228274.

47. Effects and side effects of personal learning environments and personalized learning in formal education / X. Xu et al. Education and Information Technologies. 2024. URL: https://doi.org/10.1007/s10639-024-12685-0.

48. Botirova M. M. Didactic principles and rules of education. American Journal of Interdisciplinary Research and Development. 2024. Vol. 29. P. 75–79.
URL: https://www.ajird.journalspark.org/index.php/ajird/article/view/1203.

49. Реалізація диференційованого підходу при навчанні програмуванню мовою python здобувачів загальної середньої освіти / Н. Дегтярьова та ін. Modern Information Technologies and Innovation Methodologies of Education in Professional Training Methodology Theory Experience Problems. 2024. № 72. С. 53–60. URL: https://doi.org/10.31652/2412-1142-2024-72-53-61.

50. Saabith A., Fareez M, and Vinothraj T. Python current trend applications: an overview. International Journal of Advance Engineering and Research Development. 2019. Vol. 6, No. 10.

51. Rayhan Abu. The Rise of Python: A Survey of Recent Research. 2023. DOI: 10.13140/RG.2.2.27388.92809.

52. *HackerRank - Online Coding Tests and Technical Interviews*.
URL: https://www.hackerrank.com/domains/python.

53. JetBrains. Learn Python with PyCharm for Education. *JetBrains*.
URL: https://www.jetbrains.com/pycharm-edu/.

54. Python Courses & Tutorials | Codecademy. *Codecademy*.
URL: https://www.codecademy.com/catalog/language/python.

55. *Coursera*. URL: https://www.coursera.org/specializations/python.

56. Blockly | Google for Developers. Google for Developers. URL: https://developers.google.com/blockly.

57. Project Jupyter Documentation – Jupyter Documentation 4.1.1 alpha documentation. URL: https://docs.jupyter.org/.

58. Python Tutor code visualizer: Visualize code in Python, JavaScript, C, C++, and Java. Python Tutor - Python Online Compiler with Visual AI Help. URL: https://pythontutor.com/visualize.html.

59. Chong J., Plummer R., Leifer L., Klemmer S., Eris O., Toye G. Pair programming: when and why it works. Proceedings of the 17th Annual Workshop of the Psychology of Programming Interest Group, PPIG 2005, Brighton, UK, June 29 - July 1, 2005.

60. Integrating micro-learning content in traditional e-learning platforms / R. P. Díaz Redondo et al. Multimedia Tools and Applications. 2020. URL: https://doi.org/10.1007/s11042-020-09523-z.

61. Build and ship software on a single, collaborative platform. GitHub. URL: https://github.com/.

62. Classroom Management Tools & Resources - Google for Education. URL: https://classroom.google.com/.

63. Colab.google. URL: https://colab.google/.

64. Moodle.org. Moodle challenge. URL: https://moodle.org/?lang=uk.

65. Eolymp. URL: https://eolymp.org/.

66. Hamidillo Fayzillo ugli K. Python Learning Methodology: A Comprehensive Approach for Effective Skill Acquisition. European Journal of Education and Applied Psychology. 2023. P. 90–95.
URL: https://doi.org/10.29013/ejeap-23-2-90-95.

67. Python: Empowering Data Science Applications and Research / M. Ranjan et al. Journal of Operating Systems Development & Trends. 2023. URL: https://doi.org/10.37591/joosdt.v10i1.576.

68. Jawahar S., Kukunuri G., Gokuldev S., Jayaprakash S., Anandaram H., Manivasagan C., Thenmozhi M. An exploration of Python libraries in machine learning models for data science. 2023. DOI: 10.4018/978-1-6684-8696-2.ch001.

69. Çetin H. A., Doğan E., Tüzün E. A review of code reviewer recommendation studies: Challenges and future directions. Science of Computer

Programming. 2021. Vol. 208. P. 102652.
URL: https://doi.org/10.1016/j.scico.2021.102652.

70. Bhutoria A. Personalized education and artificial intelligence in United States, China, and India: A systematic Review using a Human-In-The-Loop model. Computers and Education: Artificial Intelligence. 2022. P. 100068. URL: https://doi.org/10.1016/j.caeai.2022.100068.

71. Personalized programming education: Using machine learning to boost learning performance based on students' personality traits / C.-H. Tseng et al. Cogent Education. 2023. Vol.10, no. 2.
URL: https://doi.org/10.1080/2331186x.2023.2245637.

72. A review of the Development Trend of Personalized learning Technologies and its Applications / R. M. Ambele et al. International Journal of Advances in Scientific Research and Engineering. 2022. Vol. 08, no. 11. P. 75–91. URL: https://doi.org/10.31695/ijasre.2022.8.11.9.

73. Er-Radi H., Aammou S., Jdidou A. Personalized learning through adaptive content modification. Conhecimento & Diversidade. 2023. Vol. 15, no. 39. P. 263–275. URL: https://doi.org/10.18316/rcd.v15i39.11153.

74. Harnessing AI for Education 4.0: Drivers of Personalized Learning / G. P. Barrera Castro et al. Electronic Journal of e-Learning. 2024. Vol. 22, no. 5. P. 01–14. URL: https://doi.org/10.34190/ejel.22.5.3467.

75. Enhancing personalized learning: AI-driven identification of learning styles and content modification strategies / M. K. H. Kanchon et al. International Journal of Cognitive Computing in Engineering. 2024.
URL: https://doi.org/10.1016/j.ijcce.2024.06.002.

76. Ho S.-B., Teh S.-K., Chai I., Tan C.-H., Chean S. L., Ahmad N. A. Assessing Python programming through personalised learning styles model. 2021. DOI: 10.1007/978-981-33-4069-5_13.

77. A Study of Difficulties in Teaching and Learning Programming: A Systematic Literature Review / R. Kadar et al. International Journal of Academic Research in Progressive Education and Development. 2021. Vol. 10, no. 3. URL: https://doi.org/10.6007/ijarped/v10-i3/11100.

78. A Study of Difficulties of Students in Learning Programming / N. Islam et al. Journal of Education & Social Sciences. 2019. Vol. 7, no. 2. P. 38–46. URL: https://doi.org/10.20547/jess0721907203.

79. Bilous V. Basic principles for developing an adaptive learning system. Open educational e-environment of modern university. 2019. Special edition. P. 23–31. URL: https://doi.org/10.28925/2414-0325.2019s3.

80. Lohr D., Berges M., Chugh A., Striewe M. Adaptive learning systems in programming education: a prototype for enhanced formative feedback. Proceedings of DELFI 2024. Gesellschaft für Informatik e.V. 2024. DOI: 10.18420/delfi2024_57.

81. Rocha H., Tedesco P., Costa E. On the use of feedback in learning computer programming by novices: a systematic literature mapping. Informatics in Education. 2022. Vol. 22. DOI: 10.15388/infedu.2023.09.

82. Automated Assessment of Computer Programming Practices: The 8-Years UNED Experience / D. Galan et al. IEEE Access. 2019. Vol. 7. P. 130113–130119. URL: https://doi.org/10.1109/access.2019.2938391.

83. Bjursten E.-L., Gumaelius L., Hartell E. Assessment practices in computer programming. 11th Biennial International Design and Technology Teacher's Association Research Conference (DATTArc), 7-10 Dec. 2022. Eds.: Adj. Prof. K. Seemann, Prof. P. J. Williams.

84. Mirmotahari O., Berg Y., Damşa C. Innovating assessment practices using automated feedback in software in computer science education. URL: https://ceur-ws.org/Vol-2128/scitech6.pdf.

85. Using a Personalized Learning Style and Google Classroom Technology to Bridge the Knowledge Gap on Computer Science / Z. Kopeyev et al. International Journal of Emerging Technologies in Learning *(iJET)*. 2020. Vol. 15, no. 02. P. 218. URL: https://doi.org/10.3991/ijet.v15i02.11602.

86. Teachers' Essential Guide to Google Classroom | Common Sense Education. URL: https://www.commonsense.org/education/articles/teachers-essential-guide-to-google-classroom.

87. Гуревич Р. С., Шахіна І. Ю., Подзигун О. А. Google Classroom as an effective tool of smart learning and monitoring of students' knowledge in vocational schools. Information Technologies and Learning Tools. 2020. Vol. 79, no. 5. P. 59–72. URL: https://doi.org/10.33407/itlt.v79i5.3651.

88. The use of Google classroom in the learning process / I. Ketut Sudarsana et al. Journal of Physics: Conference Series. 2019. Vol. 1175. P. 012165. URL: https://doi.org/10.1088/1742-6596/1175/1/012165.

89. Бондаренко С. Г., Шахновський А. М., Сангінова О. В. Досвід використання сервісу Google Classroom для дистанційного навчання //

Комп'ютерне моделювання і керування в техніці та технологіях КМКТТ-2021 : збірник наукових статей Дев'ятої міжнародної науково-практичної конференції, Київ, 12-14 травня 2021 р. Київ : КПІ ім. Ігоря Сікорського, 2021. С. 287–293.

90. Москаленко О. М., Федяй І. О., Бакуменко Т. К., Косенюк Г. В. Використання Google інструментів для освітнього процесу: Google Classroom як інноваційне рішення для дистанційного навчання. Академічні візії. 2023. № 19. URL: https://academy-vision.org/index.php/av/article/view/343.

91. Богачков Ю. М., Букач А. В., Ухань П. С. Комплексне застосування Google Classroom для створення варіативних дистанційних курсів. Інформаційні технології і засоби навчання. 2020. Т. 76, № 2.

92. Як налаштувати середовище виконання в Google Colab: Повний гід. URL: https://foxminded.ua/google-colab/.

93. Examples and tutorials on using Google Colab and Gradio to create online interactive student- learning modules / R. Ferreira et al. Computer Applications in Engineering Education. 2024. URL: https://doi.org/10.1002/cae.22729.

94. Naik P. G., Naik G. R., Patil M. B. Conceptualizing Python in Google COLAB. Research Gate, 2021. URL: https://www.researchgate.net/profile/Poornima-Naik/publication/357929808_Conceptualizing_Python_in_Google_COLAB/links/61e7ee675779d35951bca9d2/Conceptualizing-Python-in-Google-COLAB.pdf.

95. Moodle як основа системи дистанційного навчання та формування електронного освітнього середовища / Н. Зуєнко та ін. Перспективи та інновації науки. 2023. № 8(26). URL: https://doi.org/10.52058/2786-4952-2023-8(26)-122-135.

96. Chang Y.-C., Li J.-W., Huang D.-Y. A Personalized Learning Service Compatible with Moodle E-Learning Management System. Applied Sciences. 2022. Vol. 12, no. 7. P. 3562. URL: https://doi.org/10.3390/app12073562.

97. Papanikolaou K., Boubouka M. Personalised Learning Design in Moodle. 2020. DOI: 10.1109/ICALT49669.2020.00024. P. 57–61.

98. Vera A., González C. Educational Resource Recommender Systems Using Python and Moodle. 2022. DOI: 10.1007/978-3-031-10542-5_2. P. 15–30.

99. Махровська Н. А., Погромська Г. С. Застосування онлайн змагань з програмування в системі практичної підготовки студентів спеціальності "комп'ютерні науки". Information Technologies and Learning Tools. 2020. Т. 79, № 5. С. 260–275. URL: https://doi.org/10.33407/itlt.v79i5.3084.

100. Medvediev M. The use of E-olymp Internet Portal in Programming Competitions. Olympiads in Informatics. 2019. Vol. 13. P. 201–208. URL: https://doi.org/10.15388/ioi.2019.13.

101. The use of online coding platforms as additional distance tools in programming education / I. S. Zinovieva et al. Journal of Physics: Conference Series. 2021. Vol. 1840, no. 1. P. 012029. URL: https://doi.org/10.1088/1742-6596/1840/1/012029.

102. Жуковський С. С. E-olimp – педагогічний засіб дистанційної підготовки учнів та студентів до олімпіади з програмування. Інформаційні технології в освіті. 2010. № 8. С. 202–206.

103. Вакалюк Т. А. Використання інтернет-порталу E-olimp для проведення занять з програмування у вищих навчальних закладах. Інформаційні технології і засоби навчання. 2013. Т. 36, № 4.

104. Актуальні питання сучасної інформатики : тези доповідей II Всеукраїнської науково-практичної конференції з міжнародною участю "Сучасні інформаційні технології в освіті та науці", присвяченої 10-ій річниці функціонування Інтернет-порталу E-OLYMP (09-10 листопада 2017 р.) / за ред. Т. А. Вакалюк. Житомир : Вид О. О. Євенок, 2017. Вип. 5. 396 с.