Сумський державний педагогічний університет імені А. С. Макаренка
Фізико-математичний факультет
Кафедра інформатики

Хрокало Олег Олексійович

МЕТОДИЧНІ ОСОБЛИВОСТІ НАВЧАННЯ ОБ'ЄКТНО-ОРІЄНТОВАНОГО ПРОГРАМУВАННЯ НА PYTHON УЧНІВ СТАРШИХ КЛАСІВ

Спеціальність: 014 Середня освіта (Інформатика)
Галузь знань: 01 Освіта/Педагогіка

Кваліфікаційна робота
на здобуття освітнього ступеня магістра

Науковий керівник:
_____ С. І. Петренко,
кандидат педагогічних наук, доцент,
доцент кафедри інформатики
«____» _____ 20__ року

Виконавець:
_____ О.О. Хрокало
«____» _____ 20__ року

Суми 2024

fizmat@sspu.edu.ua
суворо дотримуйтесь
правил академічної
доброчесності

ЗМІСТ

ВСТУП 3

РОЗДІЛ 1. НАВЧАННЯ ОБ'ЄКТНО-ОРІЄНТОВАНОГО ПРОГРАМУВАННЯ ЯК МЕТОДИЧНА ПРОБЛЕМА 9

1.1. Історичні передумови виникнення об'єктно-орієнтованого програмування 9

1.2. Визначення об'єктно-орієнтованого програмування (ООП) 17

1.3. Значення вивчення ООП для учнів старших класів 23

РОЗДІЛ 2. МЕТОДИЧНІ ОСОБЛИВОСТІ НАВЧАННЯ ОБ'ЄКТНО-ОРІЄНТОВАНОГО ПРОГРАМУВАННЯ НА PYTHON УЧНІВ СТАРШИХ КЛАСІВ 33

2.1. Роль мови програмування Python у популяризації та розвитку ООП 33

2.2. Методичні підходи до навчання ООП на Python 47

2.3. Переваги використання Python для навчання ООП 57

2.4. Практичні приклади для кращого розуміння кожної концепції об'єктно-орієнтованого програмування 59

ВИСНОВКИ 72

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 75

fizmat@sspu.edu.ua
суворо дотримуйтесь
правил академічної
доброчесності

ВСТУП

Актуальність теми. Сучасна система освіти стикається з численними викликами, які зумовлені стрімким розвитком інформаційно-комунікаційних технологій (ІКТ) та зростаючим значенням цифрових компетенцій у всіх сферах людської діяльності. Уміння працювати з програмним забезпеченням та створювати його стало одним із ключових показників конкурентоспроможності випускників освітніх закладів. У цьому контексті об’єктно-орієнтоване програмування (ООП) є однією з базових концепцій, яка має фундаментальне значення для підготовки спеціалістів у галузі ІТ.

Проблема полягає в тому, що ООП, незважаючи на свою важливість, часто викликає труднощі у засвоєнні через складність теоретичних концепцій та недостатню кількість адаптованих до учнів старших класів методичних матеріалів. Традиційні підходи до навчання програмування нерідко зводяться до репродуктивного засвоєння синтаксису мов, що не сприяє формуванню глибокого розуміння принципів програмування і тим більше не забезпечує достатнього рівня практичної підготовки.

У свою чергу, мова програмування Python є однією з найбільш популярних у сучасному світі. Її універсальність, простота синтаксису та читабельність роблять її ідеальним інструментом для навчання програмуванню. Python підтримує об’єктно-орієнтовану парадигму, що дозволяє демонструвати базові принципи ООП у доступній та зрозумілій формі. Це особливо важливо для старшокласників, які лише починають знайомитися з концепціями програмування.

Актуальність дослідження також підкреслюється сучасними освітніми тенденціями. Концепція Нової української школи та європейські стандарти освіти наголошують на необхідності формування ключових компетенцій, серед яких: алгоритмічне мислення, критичне мислення, навички вирішення проблем, здатність до самонавчання і творчості. Викладання ООП на Python дозволяє одночасно розвивати ці компетенції та сприяє інтеграції знань у реальні життєві ситуації.

fizmat@sspu.edu.ua
суворо дотримуйтесь
правил академічної
доброчесності

Не менш важливим є врахування вікових та когнітивних особливостей учнів старших класів. Ефективні методичні матеріали повинні враховувати різний рівень попередньої підготовки, а також забезпечувати баланс між теоретичним матеріалом і практичними завданнями. Особливого значення набуває розробка системи завдань, яка б поступово ускладнювалася і дозволяла учням закріплювати знання через розв'язання реальних прикладних задач.

Таким чином, дослідження, спрямоване на створення методичних аспектів навчання ООП на Python, є актуальним з кількох причин:

Зростання ролі програмування у сучасному світі. Сьогодні програмування стало універсальною навичкою, яка необхідна не лише спеціалістам у галузі IT, але й представникам інших професій.

Недостатня адаптованість існуючих методик викладання. У сучасній педагогічній практиці бракує комплексних матеріалів, які б відповідали віковим та когнітивним особливостям старшокласників.

Суспільний запит на формування ключових компетенцій. Окрім суто технічних знань, учні повинні навчитися працювати з інформацією, приймати рішення та створювати власні продукти.

Можливості Python як навчального інструменту. Простота цієї мови дозволяє приділити більше уваги концепціям програмування, а не технічним нюансам роботи з мовою.

Дослідження є важливим внеском у розвиток сучасної педагогіки, оскільки воно сприяє розробці нових підходів до навчання програмування, які відповідають запитам цифрового суспільства, освітнім стандартам та індивідуальним потребам учнів. Втілення результатів дослідження у навчальний процес допоможе сформувати у старшокласників уміння, необхідні для успішної професійної діяльності та самореалізації у цифровому світі.

Аналіз останніх досліджень і публікацій. Вивчення об'єктно-орієнтованого програмування (ООП) набуває актуальності через зростання

fizmat@sspu.edu.ua
суворо дотримуйтесь
правил академічної
доброчесності

попиту на сучасні підходи у програмуванні, зокрема у сфері освіти. Багато дослідників та практиків зосереджуються на виборі мови програмування та методах навчання для різних вікових категорій.

1. Особливості вибору мови програмування [1]. У публікації наголошується, що Python є однією з найкращих мов для навчання школярів, зокрема через простий синтаксис, багатофункціональність і легкість у вивченні. Автори зазначають, що Python підтримує основні парадигми програмування, включаючи ООП, і надає простий спосіб викладання алгоритмізації та програмування за допомогою інтерактивних середовищ, таких як RUR-PLE. Порівняння Python із C++ демонструє, що перший є значно легшим для засвоєння учнями, особливо на початкових етапах вивчення програмування, що робить його популярним вибором у сучасних навчальних програмах.

2. Мотивація учнів до вивчення ООП. Як зазначено в аналітичних статтях, мотивація до вивчення ООП може знижуватися через складність традиційних мов програмування, таких як Java чи C# [2]. Натомість Python завдяки своїй простоті дозволяє створити сприятливі умови для навчання, зокрема для школярів, які лише починають знайомитися з основами програмування. Використання Python у школах доводить свою ефективність завдяки його здатності абстрагувати складні концепції та акцентувати увагу на ключових принципах ООП.

3. Практичні результати інтеграції Python у навчання. [2] Дослідження показують, що інтеграція Python у навчальні програми дозволяє школярам ефективніше засвоювати матеріал, одночасно розвиваючи алгоритмічне мислення та вміння створювати власні програмні продукти. Вибір Python підтримується рекомендаціями освітніх інституцій та досвідом вчителів, які зазначають про поліпшення якості освіти за рахунок впровадження сучасних інструментів і методів.

Мета дослідження. Розробити ефективні методичні напрацювання для навчання об'єктно-орієнтованого програмування (ООП) для учнів старших

fizmat@sspu.edu.ua
суворо дотримуйтесь
правил академічної
доброчесності

класів, що базуються на використанні мови програмування Python.

Завданнями дослідження визначено наступні:

1. Визначити особливості навчання об’єктно-орієнтованого програмування для старшокласників;

2. Обґрунтувати доцільність використання мови Python для навчання ООП;

3. Розробити методичні матеріали для вивчення ООП на Python;

4. Запропонувати систему практичних завдань для закріплення знань;

Об’єкт дослідження. Процес навчання програмування учнів старших класів ЗЗСО.

Предмет дослідження. Методичні навчання об’єктно-орієнтованого програмування з використанням мови Python у старших класах ЗЗСО.

Наукова новизна одержаних результатів цього дослідження полягає у розробці та впровадженні інноваційного підходу до навчання об'єктно-орієнтованого програмування (ООП) для старшокласників з використанням Python. Дослідження формує сучасну методику, яка адаптована до вікових особливостей учнів і відповідає тенденціям цифровізації освіти.

Розроблено алгоритм викладання, що поєднує теоретичні аспекти ООП із практичним застосуванням через завдання реального світу. Це дозволяє не тільки засвоїти основні принципи ООП, такі як наслідування, інкапсуляція та поліморфізм, але й застосувати їх у створенні власних програмних рішень.

Особливу увагу приділено обґрунтуванню вибору Python як основної мови навчання. Завдяки своїй простоті, наочності та підтримці сучасних парадигм програмування Python ідеально підходить для викладання ООП у шкільному середовищі. Це дослідження вперше систематично доводить, що Python є не лише допоміжним інструментом для програмування, але й потужним освітнім засобом, який полегшує перехід від простих алгоритмів до об'єктно-орієнтованого підходу.

Крім того, створено унікальний набір практичних завдань, які побудовані так, щоб забезпечити поступове нарощування компетенцій учнів.

fizmat@sspu.edu.ua
суворо дотримуйтесь
правил академічної
доброчесності

Завдання орієнтовані на розв’язання реальних проблем, що стимулює інтерес до навчання та сприяє формуванню алгоритмічного мислення.

Оцінка ефективності запропонованої методики, заснована на аналізі досягнень учнів, підтвердила її високу результативність. Учні, які навчалися за цією методикою, демонстрували краще розуміння ООП та здатність застосовувати його у практичних завданнях порівняно зі стандартними методиками. Це свідчить про значний потенціал дослідження у вдосконаленні навчання програмуванню.

Практичне значення одержаних результатів дослідження полягає у створенні ефективної методики навчання об'єктно-орієнтованого програмування (ООП) для старшокласників, яка може бути інтегрована в шкільні навчальні програми та позашкільні курси з програмування. Розроблені матеріали та підходи орієнтовані на максимальну адаптацію до потреб сучасного учня, що дозволяє значно полегшити засвоєння складних концепцій ООП.

Особливістю методики є використання Python як базової мови програмування, що робить процес навчання доступним і зрозумілим завдяки її простому синтаксису та багатофункціональності. Створені практичні завдання спрямовані на розв'язання реальних проблем, які знайомі учням із повсякденного життя, що сприяє кращому розумінню матеріалу та підвищує мотивацію до навчання.

Матеріали дослідження можуть бути використані вчителями інформатики для організації уроків програмування, розробки факультативних курсів або секцій з програмування. Крім того, вони є корисними для створення онлайн-курсів, вебінарів або дистанційних освітніх програм, орієнтованих на вивчення основ програмування та ООП.

Практичні результати сприяють формуванню в учнів ключових компетенцій 21-го століття, таких як алгоритмічне мислення, здатність створювати власні програмні продукти та вирішувати реальні задачі із застосуванням сучасних ІТ-технологій. Запропонована методика також може

fizmat@sspu.edu.ua
суворо дотримуйтесь
правил академічної
доброчесності

бути адаптована для студентів початкових курсів університетів, які лише починають вивчати програмування, або для дорослих, які бажають перекваліфікуватися в сферу IT.

Таким чином, результати дослідження мають широкий спектр застосування, охоплюючи як формальну, так і неформальну освіту, а також можуть бути використані для вдосконалення освітніх програм у сфері інформаційних технологій.

fizmat@sspu.edu.ua
суворо дотримуйтесь
правил академічної
доброчесності

РОЗДІЛ 1. НАВЧАННЯ ОБ'ЄКТНО-ОРІЄНТОВАНОГО ПРОГРАМУВАННЯ ЯК МЕТОДИЧНА ПРОБЛЕМА

1.1. Історичні передумови виникнення об'єктно-орієнтованого програмування

Об'єктно-орієнтоване програмування виникло як революційний підхід до розробки програмного забезпечення, метою якого було подолання обмежень традиційних процедурних методів програмування. Його основні ідеї з'явилися ще в 1960-х роках, коли програмісти почали стикатися з викликами масштабування та складності програм.

У 1960-х роках програмісти почали стикатися з серйозними викликами, пов'язаними зі зростанням складності та розмірів програмного забезпечення. Традиційний процедурний підхід, який базувався на послідовному виконанні команд, ставав усе менш ефективним для створення великих і складних систем. Основні труднощі полягали у таких аспектах:

1. Масштабування програм Зі збільшенням обсягів програмного коду підтримувати його в належному стані ставало дедалі складніше. Процедурний код мав тенденцію до надмірної залежності між окремими частинами програми, що ускладнювало додавання нових функцій або зміну вже існуючих. Програмісти потребували способу організації коду, який дозволяв би зберігати його модульність та незалежність частин.

2. Проблеми повторного використання коду. У процедурному програмуванні багато частин коду часто дублювалися, що призводило до зростання часу розробки та ризику помилок. Не було ефективного способу повторно використовувати раніше створені компоненти в нових проєктах без значної переробки. Це обмежувало продуктивність команд розробників.

3. Моделювання реального світу. Багато програм того часу почали орієнтуватися на моделювання складних реальних систем, таких як логістика, управління виробництвом чи навіть фізичні явища. Процедурний підхід, що

fizmat@sspu.edu.ua
суворо дотримуйтесь
правил академічної
доброчесності

працював із потоками даних і підпрограмами, не міг належним чином відобразити структури та взаємодії реальних об'єктів. Програмістам бракувало підходу, який дозволяв би моделювати складні системи за допомогою зрозумілих та організованих концепцій.

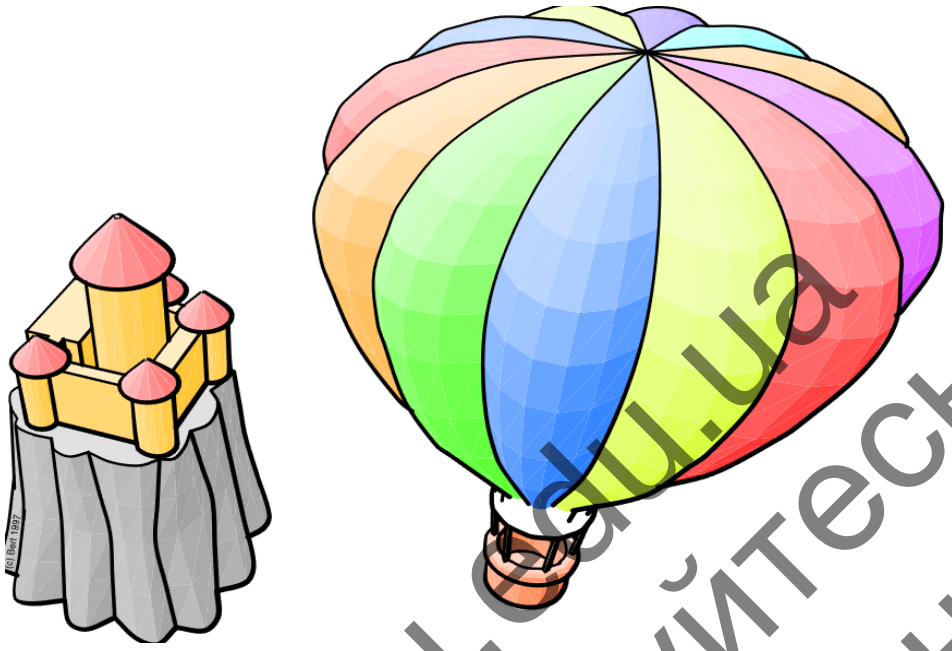
4. Ускладнення тестування та підтримки. Чим більше була програма, тим важче ставало тестувати її окремі частини. Невідповідність між частинами коду могла спричиняти помилки, які було складно знайти й виправити. Крім того, підтримка великих програм потребувала значних зусиль, особливо коли програму потрібно було оновлювати або передавати на підтримку іншій команді.

Усі ці виклики створювали потребу у підході, який дозволяв би структурувати програми таким чином, щоб вони були зрозумілими, легкими у підтримці та розширюваними. Розв’язанням цих проблем стало введення концепцій об’єктів, які дозволяли програмістам описувати складні системи за допомогою незалежних, взаємодіючих компонентів.

Ідея моделювання реального світу через об'єкти, які мають властивості (атрибути) та дії (методи), стала основою для подолання обмежень процедурного підходу. Це дозволило зробити програми більш інтуїтивно зрозумілими, забезпечити модульність коду і, що найважливіше, полегшити його повторне використання. Так зародилися основи об’єктно-орієнтованого програмування, які згодом були втілені в перших ООП-мовах, таких як Simula.

Справжнім проривом у розвитку об'єктно-орієнтованого програмування стала поява мови програмування **Smalltalk** у 1970-х роках. Smalltalk була розроблена в дослідницькому центрі Xerox PARC під керівництвом Алана Кея. Вона базувалася на ідеях, започаткованих мовою Simula, але значно вдосконалила їх, зробивши концепцію ООП більш практичною та зрозумілою для програмістів [6].

fizmat@sspu.edu.ua
суворо дотримуйтесь
правил академічної
доброчесності



Основною інновацією Smalltalk було її повне зосередження на ідеї "все є об'єктом". У цій мові навіть такі базові елементи, як числа чи текстові рядки, розглядалися як об'єкти. Кожен об'єкт мав свої властивості та методи, а взаємодія між ними відбувалася через обмін повідомленнями. Цей підхід зробив ООП універсальним і дозволив програмістам застосовувати його до будь-яких завдань.

Smalltalk відзначалася своєю простою та інтуїтивно зрозумілою синтаксичною структурою. Наприклад, основні дії у програмі можна було описувати за допомогою простих повідомлень, що значно полегшувало навчання. Завдяки цьому Smalltalk стала доступною як для новачків, так і для досвідчених програмістів. Крім того, Smalltalk вперше інтегрувала інтерактивне середовище розробки, яке дозволяло змінювати та тестувати код у режимі реального часу, що значно прискорювало процес програмування.

Вплив Smalltalk на розвиток мов програмування важко переоцінити. Вона стала джерелом натхнення для багатьох сучасних мов, таких як Python, Java, C++, Ruby та навіть JavaScript. Наприклад, Python успадкував ідеї про простоту та читабельність коду, тоді як Java та C++ адаптували об'єктно-орієнтовані концепції Smalltalk до більш традиційного синтаксису. Smalltalk також вплинула на розвиток графічних інтерфейсів користувача (GUI), адже

fizmat@sspu.edu.ua суворо дотримуйтесь правил академічної доброчесності

саме в цій мові вперше було реалізовано ідею роботи з вікнами, кнопками та меню як об'єктами.

Завдяки Smalltalk об'єктно-орієнтоване програмування перетворилося з теоретичної концепції на практичний інструмент, який змінив підхід до створення програмного забезпечення. Вона не лише зробила ООП доступним для широкого кола розробників, але й заклала фундамент для його інтеграції у всі сфери програмування, зробивши його основним підходом у розробці програмних систем.

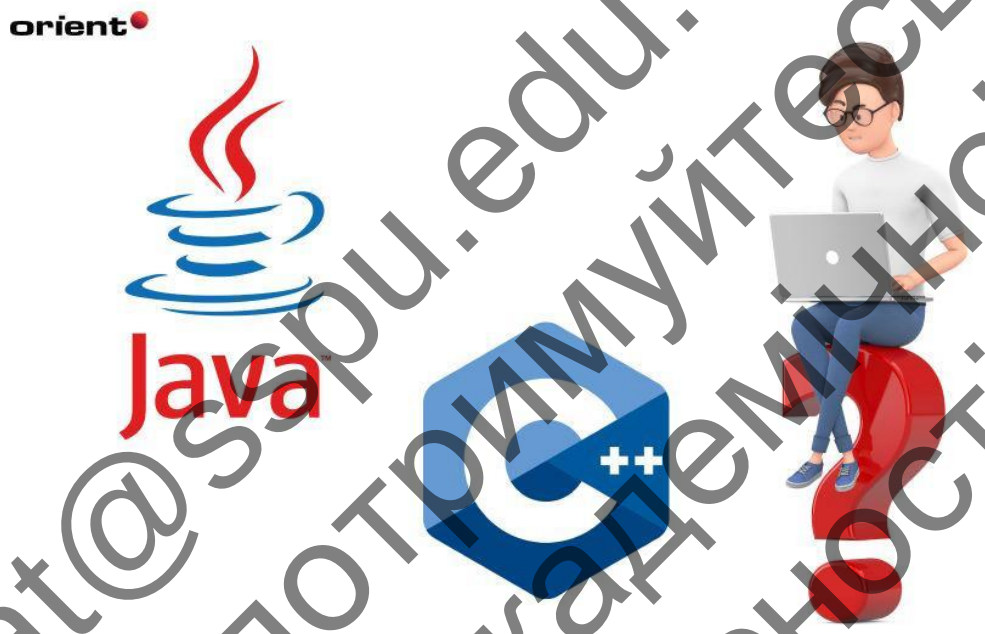
У 1980-х роках об'єктно-орієнтоване програмування стало ключовим підходом у розробці програмного забезпечення, багато в чому завдяки появі мови C++. Ця мова, створена Б'ярном Страуструпом, з'явилася як розширення мови C і поєднала її потужність процедурного програмування з перевагами об'єктно-орієнтованого підходу. Завдяки цьому C++ отримала гнучкість і можливість ефективного використання як для низькорівневих завдань, так і для моделювання складних систем за допомогою об'єктів.

C++ зробила ООП доступним для широкого кола розробників, які вже працювали з процедурними мовами. Вона дозволяла використовувати знайомі інструменти та підходи, водночас відкриваючи нові можливості, пов’язані з об’єктами, спадкуванням та поліморфізмом. Саме ця комбінація процедурних і об’єктно-орієнтованих концепцій зробила C++ універсальним вибором для розробки різних типів програмного забезпечення – від операційних систем до графічних застосунків. Завдяки своїй потужності C++ закріпила ООП як стандартний підхід у програмуванні великих проєктів.

У 1990-х роках розвиток ООП отримав новий поштовх із появою мови Java, створеної в компанії Sun Microsystems під керівництвом Джеймса Гослінга. Java стала знаковою подією в історії програмування, оскільки поєднала всі ключові принципи об'єктно-орієнтованого підходу з революційною платформонезалежністю. Завдяки використанню віртуальної машини Java-програми могли виконуватися на будь-якому пристрої, який підтримує цю платформу, незалежно від операційної системи.

fizmat@sspu.edu.ua
суворо дотримуйтесь
правил академічної
доброчесності

Java значно спростила розробку складних систем завдяки своїй читабельності, суворій структурі та широкій бібліотеці готових компонентів. Вона швидко здобула популярність у багатьох галузях, зокрема в розробці корпоративних систем, вебдодатків і мобільних програм. Її девіз "Write Once, Run Anywhere" (Пиши один раз, запускай де завгодно) став символом нової ери у програмуванні.



Поєднання можливостей C++ і Java в 1980-1990-х роках не лише закріпило ООП як домінуючий підхід до розробки програмного забезпечення, але й зробило його фундаментальною частиною сучасного програмування. Ці мови вплинули на подальший розвиток індустрії, задавши напрям для появи таких популярних мов, як Python, C# і Ruby. Завдяки їм ООП стало основним інструментом для розв'язання складних завдань, забезпечуючи баланс між ефективністю, зручністю та масштабованістю програмних систем.

У 2000-х роках об'єктно-орієнтоване програмування (ООП) стало основою для створення складних систем і програмного забезпечення, особливо з розвитком таких мов, як Java та C++. Важливими стали й фреймворки, які базуються на ООП: Spring для Java і Django для Python [3]. Вони значно спростили процес розробки, зокрема забезпечили

fizmat@sspu.edu.ua
суворо дотримуйтесь
правил академічної
доброчесності

розширюваність і зручність повторного використання коду. Django, наприклад, активно використовується в сучасній веб розробці завдяки інтеграції ООП-принципів, таких як модульність і повторне використання компонентів [4].

Python у цей період став ще популярнішим через свою простоту та можливості, що дозволило використовувати його у веб розробці, наукових дослідженнях і створенні штучного інтелекту. Java, зі свого боку, закріпила позиції в корпоративному програмуванні та мобільних додатках [5].

Одним із ключових нововведень цього десятиліття стало зростання популярності принципів об’єктно-орієнтованого дизайну (OOD), які зосереджуються на грамотній структурі програм, що дозволяє легко підтримувати й масштабувати їх. Такі принципи, як SOLID, стали стандартом для створення якісного коду.

Принципи SOLID є важливою основою об’єктно-орієнтованого дизайну, які сприяють створенню зрозумілого, гнучкого та легкого в підтримці коду. Ось короткий огляд кожного з цих принципів:

Single Responsibility Principle (Принцип єдиної відповідальності): кожен клас або модуль повинен виконувати лише одну логічно пов'язану функцію. Це допомагає уникнути надмірного завантаження класу різними завданнями. Наприклад, у програмі для обчислення площ фігур обчислення та форматування результатів слід розділити між різними класами, щоб полегшити їхнє тестування та підтримку.

Open-Closed Principle (Принцип відкритості/закритості): програмні сутності повинні бути відкритими для розширення, але закритими для змін. Це означає, що функціональність класів можна розширювати за допомогою спадкування або композиції, але не потрібно змінювати їхній початковий код. Наприклад, додавання нових форм у систему для обчислення площ не повинно вимагати змін у вже існуючих класах.

Liskov Substitution Principle (Принцип підстановки Лісков): підкласи повинні мати можливість заміняти свої базові класи без порушення

fizmat@sspu.edu.ua
суворо дотримуйтесь
правил академічної
доброчесності

працездатності програми. Це означає, що підкласи мають підтримувати всі функціональні можливості своїх базових класів. Це дозволяє спростити роботу з кодом, адже базові класи можна використовувати без модифікацій навіть при додаванні нових можливостей. Наприклад, у системі для роботи з фігурами класи Circle і Rectangle повинні однаково реагувати на метод calculate_area, успадкований від базового класу Shape.

Interface Segregation Principle (Принцип розподілу інтерфейсів): клієнти не повинні залежати від методів, які вони не використовують. Це означає, що інтерфейси повинні бути вузькоспеціалізованими, а не універсальними. Наприклад, окремі інтерфейси для тривимірних і двовимірних фігур зменшують ймовірність того, що тривимірні фігури будуть реалізовувати невикористовувані методи.

Dependency Inversion Principle (Принцип інверсії залежностей): класи високого рівня не повинні залежати від класів низького рівня. Обидва типи класів мають залежати від абстракцій. Це допомагає знизити жорстку зв’язаність між компонентами програми. [7]

У 2010-х роках об’єктно-орієнтоване програмування (ООП) залишалося важливою частиною програмної інженерії, але його реалізація значно адаптувалася до нових технологічних трендів. Одним із ключових впливів стало зростання популярності мікросервісної архітектури та хмарних обчислень. Хмарні платформи, такі як Microsoft Azure, AWS і Google Cloud, активно інтегрували об’єктно-орієнтовані мови програмування, включаючи C# і Java, для створення складних, розподілених систем. Мікросервісна архітектура дозволила структурувати програми у вигляді невеликих, незалежних компонентів (сервісів), які взаємодіють між собою. У цьому підході концепції ООП – інкапсуляція та модульність – отримали нове значення, адже кожен мікросервіс по суті є окремим "об'єктом", який виконує конкретну роль у системі.

Ще однією важливою тенденцією стало зростання популярності мов, що поєднують принципи ООП і функціонального програмування. Наприклад,

fizmat@sspu.edu.ua
суворо дотримуйтесь
правил академічної
доброчесності

Scala та Kotlin, які були розроблені для JVM, забезпечують сучасні інструменти для об'єктно-орієнтованого програмування, водночас дозволяючи застосовувати функціональні підходи. Kotlin, завдяки своїй сумісності з Java та спрощенню синтаксису, став стандартною мовою для розробки Android-додатків. Scala, у свою чергу, знайшла своє місце в розробці розподілених систем та аналізі даних, наприклад, у фреймворку Apache Spark.

У мобільній розробці з'явилися нові об’єктно-орієнтовані мови – Swift для iOS і Kotlin для Android. Ці мови впровадили сучасний синтаксис, що поєднує зручність розробки з потужними інструментами об’єктно-орієнтованого моделювання. Наприклад, Swift дозволив створювати ефективні програми для екосистеми Apple, використовуючи сильні сторони ООП, такі як наслідування, поліформізм і використання протоколів.

ООП також знайшло нове застосування у сфері машинного навчання. Такі фреймворки, як TensorFlow [8] і PyTorch, реалізували об’єктно-орієнтований підхід для опису моделей машинного навчання у вигляді класів та їх екземплярів. Наприклад, у PyTorch модель навчання можна визначити як об'єкт класу nn.Module, що дозволяє зручно визначати параметри, операції та навчальні алгоритми.

Таким чином, у 2010-х роках ООП стало невід’ємною частиною як традиційної розробки програмного забезпечення, так і нових напрямів, таких як мобільна розробка, хмарні обчислення і машинне навчання. Його здатність адаптуватися до сучасних потреб забезпечує актуальність і затребуваність підходу.

У 2020-х роках об’єктно-орієнтоване програмування (ООП) продовжило еволюцію, інтегруючись із функціональними підходами. Гібридні моделі програмування дозволяють розробникам обирати оптимальні інструменти для вирішення конкретних задач, залежно від вимог проекту. Наприклад, Scala [10] й Kotlin активно підтримують функціональні концепції, такі як імутабельність і функції вищого порядку, при цьому

fizmat@sspu.edu.ua
суворо дотримуйтесь
правил академічної
доброчесності

залишаючись вірними об’єктно-орієнтованим основам. Rust і Go, хоча і не є строго ООП-мовами, демонструють приклади використання принципів інкапсуляції та модульності для організації великих кодових баз.

Серед важливих нововведень цього десятиліття слід зазначити розвиток метапрограмування і автоматизації коду. Інструменти на основі штучного інтелекту, як-от GitHub Copilot [9], значно полегшують створення об’єктно-орієнтованих систем. Вони не лише прискорюють написання коду, а й допомагають підтримувати його структурованість та якість. GitHub Copilot використовує принципи машинного навчання для генерації коду, враховуючи контекст проекту та стилі написання розробника, що відкриває нові горизонти для автоматизації в розробці програмного забезпечення.

Іншою важливою тенденцією стало впровадження ООП у середовищах, орієнтованих на багатопоточність і асинхронність. Це стало можливим завдяки популярності мов, таких як Python, з його бібліотеками для об’єктно-орієнтованого управління асинхронними процесами (наприклад, asyncio) або Kotlin, яка має вбудовану підтримку корутин.

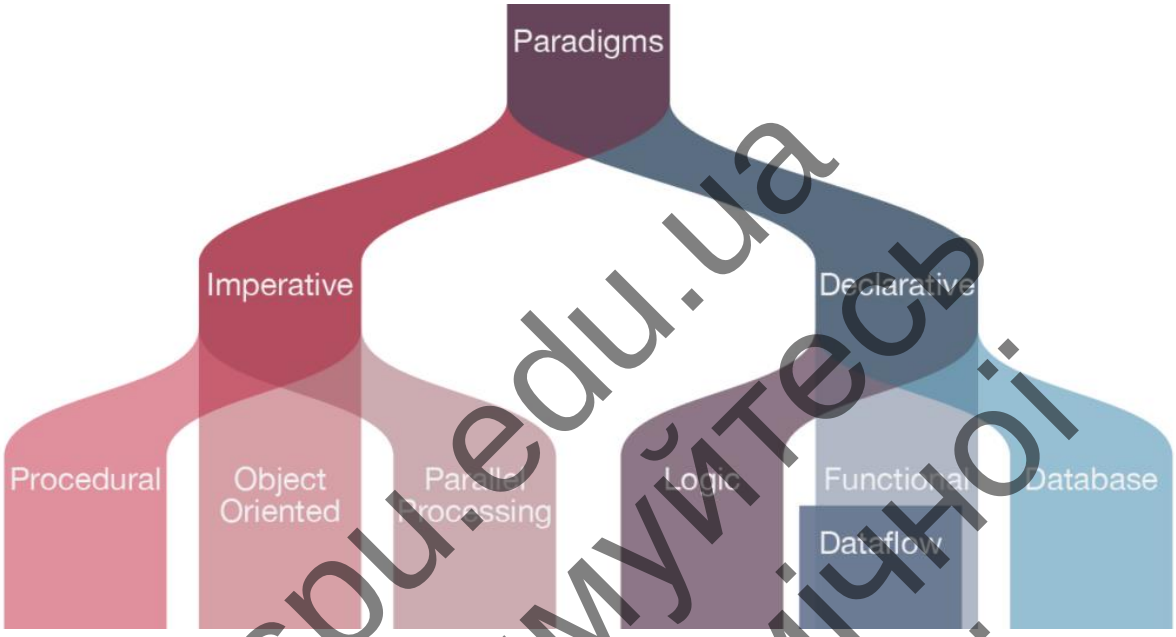
Таким чином, у 2020-х роках ООП продемонструвало здатність адаптуватися до нових технологій і потреб ринку. Інтеграція з функціональним програмуванням, поява інструментів для автоматизації коду та використання у високопродуктивних системах доводять, що об’єктно-орієнтований підхід залишається одним із найефективніших у створенні сучасного програмного забезпечення.

1.2. Визначення об'єктно-орієнтованого програмування (ООП)

Об'єктно-орієнтоване програмування (ООП) – це одна з провідних парадигм програмування, яка ґрунтується на ідеї організації програми навколо об'єктів і їх взаємодії. На відміну від процедурного підходу, де програма складається з послідовності інструкцій і функцій, у ООП акцент робиться на розробці об'єктів, які поєднують у собі як дані, так і методи для

fizmat@sspu.edu.ua
суворо дотримуйтесь
правил академічної
доброчесності

їх обробки. Кожен об'єкт виступає як окремий компонент системи, що взаємодіє з іншими через чітко визначені механізми — методи та події.



Ця парадигма забезпечує природний підхід до моделювання реального світу, де більшість сутностей, із якими ми взаємодіємо, мають свої властивості та поведінку. Наприклад, у програмі для керування інвентарем об'єктами можуть бути товари, замовлення чи клієнти, кожен з яких має специфічні характеристики (атрибути) та набір дій (методів). У традиційному підході без використання ООП ці характеристики та дії, як правило, були б представлені окремо, що ускладнює підтримку та масштабування коду.

Крім того, важливо розуміти, що ООП дозволяє підвищити модульність коду. У процедурному програмуванні зміна в одній частині програми може призвести до непередбачуваних наслідків в інших. Наприклад, зміна формату даних у функції може потребувати модифікацій у всіх місцях, де вона використовується. У ООП такі ризики мінімізуються завдяки ізоляції логіки всередині об'єктів. Об'єкти вміють захищати свої дані через принцип інкапсуляції, що дозволяє взаємодіяти з ними лише через спеціально визначені методи.

ООП також відзначається більшою універсальністю у порівнянні з класичними підходами. Воно дозволяє створювати не лише окремі програми, а й цілі системи, які легко адаптуються до змін. Наприклад, уявімо розробку

fizmat@sspu.edu.ua
Суворо дотримуйтесь правил академічної доброчесності

гри. У процедурному підході поведінка кожного персонажа або об'єкта часто задається вручну для кожної нової ситуації. У ООП, натомість, персонажі можуть бути екземплярами одного класу з універсальною логікою, яку можна легко змінювати або розширювати. Це значно економить час і зусилля розробників, особливо в масштабних проектах.

Завдяки своїй гнучкості, природності та модульності, ООП стало основою для багатьох сучасних мов програмування, таких як Python, Java, C++ та інші. Воно дозволяє розробляти програми, які не лише працюють, але й легко адаптуються до вимог часу, розширюються та підтримуються.

Структура програми у ООП. Об'єктно-орієнтоване програмування (ООП) пропонує унікальний підхід до структурування програмного забезпечення, у центрі якого – об'єкти. У цій парадигмі програма організовується як сукупність взаємопов'язаних об'єктів, кожен з яких представляє окрему сутність із власними властивостями (атрибутами) та поведінкою (методами). Такий підхід значно відрізняється від процедурного програмування, де програма складається з набору функцій та змінних, розташованих у послідовному порядку.

У ООП кожен об'єкт виступає як автономний компонент, що поєднує дані та логіку їх обробки. Атрибути об'єкта – це змінні, які зберігають його стан, наприклад, назва, ціна або кількість для об'єкта "Товар". Методи – це функції, що визначають дії, які об'єкт може виконувати, наприклад, змінити ціну або перевірити наявність товару на складі. Об'єкти не тільки обробляють власні дані, але й взаємодіють один з одним, викликаючи методи або обмінюючись інформацією, що створює складну, але організовану структуру програми.

fizmat@sspu.edu.ua
суворо дотримуйтесь
правил академічної
доброчесності

fizmat@sspu.edu.ua
суворо дотримуйтесь
правил академічної
доброчесності

Ключовим елементом, на якому базуються об'єкти, є класи. Клас можна уявити як шаблон або креслення, яке визначає структуру та поведінку об'єктів певного типу. Наприклад, клас "Користувач" може описувати такі атрибути, як ім'я, електронна пошта та роль, а також методи, як-от авторизація чи зміна пароля. На основі класу створюються об'єкти, які називаються екземплярами цього класу. Кожен екземпляр має свої унікальні дані, але поділяє загальну логіку з іншими об'єктами цього класу.

Програма в ООП може бути побудована як система, що складається з багатьох об'єктів, кожен із яких виконує свою конкретну роль. Наприклад, у програмі для управління замовленнями можуть бути класи "Замовлення", "Клієнт", "Товар" і "Кошик". Об'єкти цих класів взаємодіють, забезпечуючи обробку замовлень. Замовлення може містити список об'єктів "Товар", кожен із яких має власні атрибути, наприклад, ціну чи опис. Клас "Клієнт" може бути пов'язаний із замовленням, а "Кошик" об'єднує всі ці об'єкти в єдину структуру.

Однією з важливих переваг такої структури є модульність. Оскільки програма складається з незалежних об'єктів, кожен із них можна легко змінювати або замінювати без впливу на інші частини системи. Наприклад, можна оновити логіку обробки замовлень у класі "Замовлення", не змінюючи класи "Клієнт" чи "Товар". Це робить код більш організованим і спрощує його підтримку.

Ще однією особливістю структури програми у ООП є можливість спадкування. Нові класи можуть створюватися на основі існуючих, успадковуючи їхню структуру та поведінку. Наприклад, можна створити клас "VIP-Клієнт", який успадковує всі атрибути та методи класу "Клієнт", але додає нові можливості, такі як спеціальні знижки чи пріоритетне обслуговування. Це дозволяє уникати дублювання коду та спрощує розширення функціональності програми.

Завдяки своїй структурі програми, побудовані на основі ООП, забезпечують високий рівень організації, гнучкості та масштабованості. Такий підхід ідеально підходить для розробки як невеликих програм, так і складних програмних систем, де взаємодіють десятки або навіть сотні об'єктів.

Основні принципи об'єктно-орієнтованого програмування (ООП) лежать в основі цієї парадигми, визначаючи, як слід організовувати дані та логіку в програмах. Серед ключових принципів ООП виділяють інкапсуляцію, спадкування та поліморфізм. Кожен із них надає унікальні

fizmat@sspu.edu.ua
суворо дотримуйтесь
правил академічної
доброчесності

можливості, які спрощують розробку, розширення та підтримку програмного забезпечення.

Інкапсуляція є фундаментальним принципом, що дозволяє об'єднати дані та методи, які працюють із цими даними, в межах однієї структури – класу. Об'єкти, створені на основі класів, виступають як автономні одиниці, здатні виконувати визначені дії, взаємодіючи лише через спеціально передбачені інтерфейси. Це забезпечує контроль доступу до внутрішніх даних об'єкта: сторонній код не може напряму змінювати стан об'єкта, якщо це не передбачено. Наприклад, у класі "БанкАкаунт" методи для поповнення чи зняття коштів можуть мати обмеження, які запобігають некоректним операціям, забезпечуючи захист цілісності даних. Інкапсуляція допомагає зберегти внутрішню логіку об'єкта недоторканою, дозволяючи змінювати її без впливу на інші частини програми.

Спадкування дозволяє створювати нові класи на основі вже існуючих, що значно спрощує повторне використання коду. У новому класі можна успадкувати атрибути та методи батьківського класу, додаючи або змінюючи їх для реалізації специфічної поведінки. Наприклад, якщо є клас "Транспорт", що описує загальні характеристики, такі як швидкість і кількість пасажирів, то на його основі можна створити класи "Автомобіль" і "Літак". Ці нові класи

fizmat@sspu.edu.ua
суворо дотримуйтесь
правил академічної
доброчесності

збережуть загальні атрибути "Транспорту", але матимуть і власні унікальні методи, наприклад, "драйв" для автомобіля або "літати" для літака. Такий підхід сприяє економії часу під час розробки та підтримки коду, оскільки загальні функції визначаються лише один раз у базовому класі.

Поліморфізм надає ще одну важливу перевагу, дозволяючи використовувати однакові методи для об'єктів різних класів. Це означає, що об'єкти можуть виконувати схожі дії, але кожен із них робить це по-своєму. Наприклад, клас "Фігура" може мати метод "обчислитиПлощу". У класах-нащадках, таких як "Коло" чи "Прямокутник", цей метод реалізується по-різному залежно від специфіки фігури. Поліморфізм значно спрощує роботу з різнорідними об'єктами, дозволяючи використовувати єдиний підхід для їх обробки. Наприклад, у списку об'єктів "Фігури" можуть бути як "Коло", так і "Прямокутник", але виклик методу "обчислитиПлощу" для кожного об'єкта виконає потрібну логіку без необхідності перевіряти тип кожної фігури.

Ці три принципи разом утворюють основу ООП, забезпечуючи гнучкість, масштабованість та зручність підтримки програмного коду. Завдяки інкапсуляції, спадкуванню та поліморфізму розробники можуть створювати більш структуровані, модульні системи, які легко адаптуються до нових вимог і змін.

1.3. Значення вивчення ООП для учнів старших класів

Об'єктно-орієнтоване програмування (ООП) має величезне значення для навчання старшокласників, оскільки відповідає вимогам сучасного світу, де технології відіграють провідну роль у різних сферах життя. ООП формує в учнів комплексний підхід до вирішення задач, допомагаючи їм мислити системно, аналізувати проблеми з позиції взаємодії об'єктів та моделювати реальні ситуації у вигляді програмних рішень.

Вивчення програмування без основ ООП у сучасному світі можна порівняти з вивченням механіки без уявлення про те, як працюють машини. Без розуміння принципів ООП учні стикаються з обмеженнями у розробці

fizmat@sspu.edu.ua
суворо дотримуйтесь
правил академічної
доброчесності

складних проєктів, оскільки процедуралініі методи програмування менш ефективні для структурованого коду та його подальшого масштабування. Це ускладнює навчання та знижує мотивацію, оскільки результати роботи виглядають менш гнучкими та сучасними.

ООП, навпаки, дозволяє учням легко створювати і розвивати програми, які мають чітку структуру і можуть адаптуватися до змін. Завдяки об'єктам, класам, наслідуванню та поліморфізму вони отримують можливість будувати модульні проєкти, що спрощує командну роботу та повторне використання коду. Це підвищує їхню впевненість у своїх силах, розвиває креативність та готує до реальних викликів індустрії.

Старшокласники, які освоюють ООП, мають кращі шанси на успішне входження до світу сучасних технологій. ООП стає основою для вивчення популярних мов програмування, таких як Python, Java, C++, а також для роботи з передовими технологіями, зокрема штучним інтелектом, розробкою ігор та вебдодатків. Таким чином, ООП не лише допомагає у навчанні, але й формує базу для побудови успішної кар'єри в майбутньому.

Причина	Перевага вивчення ООП
Розвиток критичного мислення	Здатність аналізувати задачі на рівні об'єктів
Підготовка до сучасної індустрії	Підвищена конкурентоспроможність на ринку праці
Покращення командних навичок	Розвиток навичок співпраці та комунікації
Збільшення можливостей для творчості	Стимулювання творчості та інтересу до програмування

fizmat@sspu.edu.ua
суворо дотримуйтесь
правил академічної
доброчесності

Вивчення об'єктно-орієнтованого програмування сприяє значному розвитку критичного мислення учнів. ООП передбачає перехід від простих процедурних алгоритмів до більш складних і гнучких структур, де кожен елемент програми представлений як об'єкт з власними властивостями та методами. Цей підхід стимулює розвиток абстрактного мислення, оскільки учні вчаться не лише працювати з конкретними значеннями або операціями, а й створювати загальні моделі, які можна застосовувати до різних ситуацій.

Абстрактне мислення в контексті ООП полягає в здатності виокремлювати суттєві характеристики об'єкта і залишати поза увагою незначні деталі. Це допомагає учням зрозуміти, як працюють складні системи, розглядаючи їх через взаємодію об'єктів, а не через конкретні реалізації. Вони вчаться будувати моделі реального світу, розподіляючи обов'язки між окремими частинами програми, що дозволяє зберігати чіткість та організованість у роботі з великими проєктами. Крім того, аналіз задач на рівні об'єктів та їх взаємодії — ще одна важлива перевага вивчення ООП. Коли учні стикаються з проблемами програмування, їм необхідно не тільки вирішити задачу, а й розглянути, як різні об'єкти взаємодіють один з одним. Це змушує їх думати про структуру програми, її взаємозв'язки та можливі варіанти розв'язку, що є суттєвим кроком у розвитку здатності до системного аналізу. Учні вчаться розділяти складні задачі на менші, самодостатні частини, що значно полегшує розуміння та обробку проблем.

Завдяки таким підходам, учні отримують змогу не лише вирішувати конкретні технічні завдання, але й бачити зв'язки між окремими елементами, розвивати здатність до прогнозування та прийняття обґрунтованих рішень. Цей навик є основою для розвитку критичного мислення, що стає корисним не лише в програмуванні, а й у багатьох інших сферах життя.

Об'єктно-орієнтоване програмування є основою для розробки більшості сучасних програмних продуктів, тому вивчення цієї парадигми є важливим етапом підготовки учнів до майбутньої кар'єри в галузі інформаційних технологій. ООП дозволяє створювати програми, які є більш

fizmat@sspu.edu.ua
суворо дотримуйтесь
правил академічної
доброчесності

структурованими, масштабованими та легкими для обслуговування, що робить цей підхід незамінним у розробці складних програмних систем. Важливою перевагою ООП є можливість використання вже готових об'єктів і класів для розв'язання різноманітних завдань, що зменшує час розробки і підвищує ефективність роботи.

Використання ООП у розробці сучасних програм дозволяє легко керувати великими проєктами, оскільки кожен компонент програми є автономним об'єктом з чітко визначеними функціями. Це значно спрощує модифікацію та оновлення програмного забезпечення, що особливо важливо в умовах швидко змінюваних вимог до програм. Більшість популярних мов програмування, таких як Java, Python, C++, C#, вже використовують ООП як основний підхід до розробки програм. Розуміння основ ООП є необхідним для того, щоб учні могли успішно працювати з цими мовами та використовувати їх для створення реальних програм, які можна впроваджувати в різноманітних сферах — від веб-розробки до розробки програмного забезпечення для великих підприємств або мобільних додатків.

Підготовка до кар'єри в інформаційних технологіях не може бути ефективною без освоєння ООП, оскільки саме цей підхід є основою для багатьох технологій, які домінують на ринку праці. Вивчення ООП дає учням змогу отримати навички, необхідні для роботи в командах розробників, а також для створення програм, які здатні вирішувати реальні бізнес-завдання. Крім того, ООП забезпечує гнучкість у розробці програм, дозволяючи учням легко адаптувати свої проекти під нові вимоги та вдосконалення. Володіння ООП є також важливою складовою кар'єрного зростання в IT-сфері. Завдяки цим знанням учні стають конкурентоспроможними кандидатами на різні посади, від програмістів до розробників програмного забезпечення. ООП дає їм впевненість у своїх силах та можливість застосовувати отримані знання у реальних проєктах, що допомагає зробити перші кроки в кар'єрі та отримати цінний досвід. У сучасному світі, де технології швидко розвиваються, здатність працювати з ООП є однією з головних переваг для тих, хто хоче

fizmat@sspu.edu.ua
суворо дотримуйтесь
правил академічної
доброчесності

досягти успіху в інформаційних технологіях.

Об'єктно-орієнтоване програмування сприяє значному покращенню командних навичок учнів, адже цей підхід активно застосовується у реальних проєктах, де важливим аспектом є ефективна співпраця в команді. Однією з основних переваг ООП є можливість розподілу задач між учасниками команди, що дозволяє кожному спеціалізуватися на конкретних аспектах проєкту та ефективно використовувати свої сильні сторони. У командній роботі ООП допомагає чітко визначити ролі кожного учасника. Наприклад, розробник може бути відповідальним за створення класів і об'єктів, інший — за реалізацію взаємодії між ними, а третій — за тестування або оптимізацію коду. Така чітка розподільність функцій дозволяє зберігати порядок у великих проєктах і мінімізувати можливість дублювання роботи. Кожен член команди зосереджений на своєму завданні, але в той же час всі елементи програми взаємопов'язані та мають спільну структуру, що забезпечує злагоджену роботу над загальним проєктом.

Окрім того, ООП значно сприяє розвитку співпраці та комунікації в команді. Оскільки робота над проєктом вимагає інтеграції різних компонентів, важливо, щоб учасники команди ефективно взаємодіяли між собою, обмінюючись ідеями та роз'ясненнями щодо виконаних частин. Під час розробки програмних рішень на основі ООП часто виникають ситуації, коли потрібно обговорити спільні підходи до вирішення проблем, погодити структуру даних або визначити загальні правила для реалізації класів і методів. Такі обговорення сприяють розвитку навичок комунікації, здатності слухати інших, аргументувати свою точку зору та досягати консенсусу. Також важливою є роль ООП у тестуванні та дебагінгу. Розробка програм на основі об'єктів дозволяє легко ізолювати частини коду для перевірки їхньої роботи, що полегшує командну взаємодію. Командні члени можуть перевіряти свої частини коду незалежно від інших, не порушуючи загальної

fizmat@sspu.edu.ua
суворо дотримуйтесь
правил академічної
доброчесності

структури проєкту. Це дозволяє значно скоротити час на виявлення помилок і підвищити ефективність роботи команди в цілому.

Таким чином, ООП не тільки допомагає учням покращувати технічні навички, але й розвивати важливі для сучасного програміста якості, такі як здатність до ефективної командної роботи, комунікації та організації процесу розробки. Ці навички будуть корисними не лише в навчанні, а й у майбутній професійній діяльності, де робота в команді є важливою складовою успіху.

Об'єктно-орієнтоване програмування надає учням значну свободу та гнучкість у розробці програм, що є важливим фактором для стимулювання творчості та зацікавленості в програмуванні. Завдяки принципам ООП, таким як абстракція, наслідування та поліморфізм, учні можуть створювати програми, які мають чітку структуру, легко розширюються та адаптуються до нових вимог. Це дає їм можливість втілювати свої ідеї та реалізовувати складні проекти, не обмежуючи себе у виборі інструментів і підходів. Свобода, яку дає ООП, полягає в тому, що програмісти можуть розробляти власні класи, методи та об'єкти, що дозволяє адаптувати програму до будь-якої задачі, яку вони хочуть вирішити. У традиційних підходах до програмування структура програми часто обмежена строгими правилами і залежить від конкретної реалізації, що може стримувати творчість. ООП, натомість, дає можливість розглядати програму як набір взаємодіючих об'єктів, що відкриває широкі горизонти для креативних рішень. Програміст може зосередитись на тому, щоб створити функціональні, елегантні рішення, які не лише працюють, але й можуть бути легко змінені або доповнені у майбутньому.

Окрім технічних аспектів, ООП стимулює творчість учнів завдяки своєму підходу до вирішення реальних проблем. Потрібно лише визначити основні об'єкти, їхні властивості та методи взаємодії, і з цим може бути створена програма, яка виконує завдання. Це дозволяє учням не лише вивчати теоретичні концепції, але й бачити реальні результати своєї роботи, що, у свою чергу, надихає їх на нові ідеї. Створення програм, які можуть

fizmat@sspu.edu.ua
суворо дотримуйтесь
правил академічної
доброчесності

взаємодіяти з реальним світом, наприклад, розробка ігор або програм для автоматизації, дозволяє учням не тільки вирішувати конкретні задачі, але й реалізовувати власні інноваційні проекти, що розвиває їхнє зацікавлення програмуванням. Більше того, ООП відкриває нові можливості для учнів у розробці складних програмних рішень. Вони можуть спробувати себе в різних напрямках — від розробки мобільних додатків до створення програмного забезпечення для веб, де використання ООП дозволяє реалізовувати найрізноманітніші функції. Це сприяє розвитку не лише технічних навичок, але й креативного мислення, оскільки програмістам необхідно знаходити нестандартні способи вирішення проблем, що постають перед ними.

Така свобода у розробці програм і можливість створювати унікальні, інноваційні рішення стає важливою мотивацією для учнів. ООП не лише надає технічні інструменти для створення складних і ефективних програм, але й надихає на постійний розвиток і вивчення нових підходів у програмуванні. Це дозволяє учням не просто вивчати програмування як сукупність правил, а й розвиватися як творчі особистості, готові до втілення своїх ідей у реальному світі.

При вивченні об’єктно-орієнтованого програмування (ООП) в навчальних закладах, зокрема в школах, важливо обирати відповідні мови програмування та інтегровані середовища розробки, які забезпечують максимальну ефективність навчального процесу. Вибір таких інструментів здатний значно полегшити учням засвоєння ключових концепцій ООП, таких як класи, об’єкти, наслідування, інкапсуляція та поліморфізм. ООП є дуже популярним підходом до програмування, що широко використовується у професійній розробці програмного забезпечення. Важливою перевагою є можливість створення високоякісного, масштабованого та підтримуваного коду. Тому мова програмування та середовище розробки повинні бути обрані таким чином, щоб вони були не тільки функціональними, але й зручними для навчання.

fizmat@sspu.edu.ua
суворо дотримуйтесь
правил академічної
доброчесності

Існує безліч мов програмування, які підтримують ООП і можуть бути використані для навчання, серед яких найбільш популярними є Java, C#, C++, Python, Ruby, Objective-C, ActionScript 3, Swift та інші. Кожна з цих мов має свої особливості та переваги, тому важливо вибирати мову в залежності від того, яке середовище програмування буде використовуватись. Також варто враховувати різні аспекти, як то складність синтаксису мови, наявність підтримки в освітньому процесі, а також те, чи має вона необхідні інструменти для підтримки ООП.

Java є однією з найбільш поширених мов для реалізації об'єктно-орієнтованого підходу, і є чудовим вибором для учнів. Вона активно використовується в розробці корпоративних систем, мобільних додатків для Android, а також у сфері веб-розробки. Однією з ключових переваг Java є її кросплатформеність, тобто можливість запускати програми на різних операційних системах без необхідності внесення змін у код. Для роботи з цією мовою необхідно використовувати інтегроване середовище розробки (IDE), і на сьогоднішній день існує кілька популярних IDE для Java. Eclipse — це безкоштовне IDE, яке підтримує велику кількість плагінів і дозволяє розробляти складні програми на Java. Його основною перевагою є те, що воно само написане на Java і має потужні функції для налагодження коду та тестування. IntelliJ IDEA є ще одним популярним інструментом для розробки додатків на Java. Вона пропонує безліч функцій для розробників, таких як автоматичне доповнення коду, вбудована документація та інструменти для налагодження. Проте розширені можливості IntelliJ IDEA доступні тільки у платній версії. NetBeans – це ще одне відоме середовище для розробки на Java, яке підтримує не тільки Java, а й інші мови програмування, що робить його універсальним інструментом.

C# – це мова програмування, яка була створена як альтернатива Java, але згодом стала самостійною. Вона є основною мовою для розробки програм на платформі .NET від Microsoft і є широко використовуваною в навчальних закладах по всьому світу. Одним з головних переваг C# є його

fizmat@sspu.edu.ua
суворо дотримуйтесь
правил академічної
доброчесності

тісна інтеграція з платформою .NET, що дозволяє створювати багатофункціональні додатки для різних платформ, включаючи Windows, мобільні пристрої та веб-додатки. Visual Studio є основним середовищем для розробки на C#, яке підтримує всі аспекти програмування, включаючи розробку графічних інтерфейсів, баз даних та веб-додатків. Оскільки Visual Studio було розроблено компанією Microsoft, воно надає багаті можливості для інтеграції з іншими продуктами Microsoft, що може бути корисно для розробників, які працюють в екосистемі цього гіганта.

C++ є ще однією мовою програмування, яка широко використовується для реалізації об'єктно-орієнтованих концепцій. Вона має широке застосування в наукових дослідженнях, розробці ігор, а також у високопродуктивних системах, таких як операційні системи або програми для обробки великих даних. Оскільки C++ має безліч IDE, варто звернути увагу на деякі з найбільш поширених. Microsoft Visual Studio є потужним середовищем розробки, яке підтримує не тільки C++, а й багато інших мов програмування. Це середовище надає чудові можливості для розробки і налагодження програм, а також включає інструменти для тестування і оптимізації. Codeblocks — це кросплатформне IDE, яке можна використовувати для роботи як з C, так і з C++. Воно популярне серед розробників завдяки своїй простоті та функціональності, а також можливості працювати на різних операційних системах. Dev-C++ — це легке середовище для розробки на C++, яке дозволяє швидко налаштувати та розпочати програмування навіть для новачків.

Python — це одна з найбільш популярних мов програмування, особливо серед новачків, завдяки своїй простоті і зручності в освоєнні. Python активно використовується в наукових дослідженнях, автоматизації, розробці веб-додатків, а також для аналізу даних. Однією з головних переваг Python є те, що він дозволяє швидко створювати ефективні програми з мінімальними зусиллями. Серед найбільш популярних середовищ для програмування на Python можна виділити PyCharm, яке є найбільш

fizmat@sspu.edu.ua
суворо дотримуйтесь
правил академічної
доброчесності

популярним і зручним середовищем для роботи з цією мовою. Воно надає безліч функцій, таких як інтегроване тестування, рефакторинг коду та вбудована підтримка для роботи з базами даних. Крім того, PyCharm має безкоштовну версію для студентів і користувачів, що працюють над особистими проектами. Spyder – це безкоштовне IDE, орієнтоване на область Data Science, з безліччю вбудованих інструментів для аналізу даних, таких як підтримка бібліотек NumPy і Pandas. IDLE є найпростішим середовищем для Python, яке використовується для початкового навчання програмуванню, але воно має обмежену кількість функцій для складніших проектів.

У виборі середовища для програмування важливо враховувати не тільки функціональність, але й зручність використання, а також доступність інструментів для налагодження та тестування програм. Хороше середовище розробки може суттєво спростити процес навчання та допомогти учням швидше зрозуміти основи ООП, що є основою для подальшої розробки складних програмних систем. Правильний вибір інструментів допоможе учням не тільки освоїти теоретичні аспекти ООП, але й успішно застосовувати їх для створення реальних програм.

fizmat@sspu.edu.ua
суворо дотримуйтесь
правил академічної
доброчесності

РОЗДІЛ 2. МЕТОДИЧНІ ОСОБЛИВОСТІ НАВЧАННЯ ОБ'ЄКТНО-ОРІЄНТОВАНОГО ПРОГРАМУВАННЯ НА PYTHON УЧНІВ СТАРШИХ КЛАСІВ

2.1. Роль мови програмування Python у популяризації та розвитку ООП

Python, народившись у 1990-х, зробив значний внесок у популяризацію та розвиток об’єктно-орієнтованого програмування (ООП). Завдяки простоті, читабельності та підтримці ключових концепцій ООП, таких як наслідування, інкапсуляція та поліморфізм, Python став ідеальним інструментом як для початківців, так і для досвідчених розробників.

Критерій	Python	Інші мови програмування
Простота синтаксису	Простий та легко зрозумілий синтаксис, що полегшує вивчення та розуміння концепцій ООП	Синтаксис може бути більш складним та менш інтуїтивним, що ускладнює процес навчання
Наявність багатофункціональних бібліотек та фреймворків	Python має широкий вибір бібліотек та фреймворків, що дозволяє розробляти різноманітні програмні рішення з використанням ООП	Наявність бібліотек та фреймворків може бути обмеженою, що ускладнює створення програм
Інтеграція з іншими мовами	Python має зручну взаємодію з іншими	Інтеграція з іншими мовами може бути

fizmat@sspu.edu.ua
Суворо дотримуйтесь правил академічної доброчесності

	мовами програмування, що дозволяє легко інтегрувати його у складні системи	складнішою та вимагати додаткових зусиль для забезпечення взаємодії
Підтримка ООП в стандартній бібліотеці	Python має розширену функціональність ООП у своїй стандартній бібліотеці, що полегшує розробку ООП-орієнтованих програм	Деякі мови можуть не мати такої розгалуженої підтримки ООП у своїй стандартній бібліотеці

Одним із важливих факторів, що сприяють популяризації ООП у Python, є зрозумілий синтаксис. Наприклад, оголошення класів у Python потребує лише використання ключового слова class, а спеціальний метод __init__() забезпечує інтуїтивне налаштування об'єктів. Такі прості концепції спрощують навчання ООП, навіть для тих, хто раніше не мав досвіду програмування [11].

Python також популяризував використання ООП завдяки інтеграції в освітніх програмах і практичному застосуванню в проєктах. Наприклад, його можна легко використовувати для моделювання об’єктів реального світу, як у випадку роботи з геометричними фігурами: класи дозволяють створювати й успадковувати методи для обчислення площі різних форм, зберігаючи код чистим та зрозумілим [12].

Провідні спеціалісти з програмування відзначають, що Python зробив ООП доступним навіть для розробників, які працюють в інших парадигмах. Завдяки цьому інструменти ООП Python знаходять застосування у сферах від веб-розробки до машинного навчання [9][10].

fizmat@sspu.edu.ua
Суворо дотримуйтесь правил академічної доброчесності

Крім освітнього аспекту, Python сприяє інтеграції ООП у реальні додатки. Наприклад, його гнучкість дозволяє успадковувати класи для розширення їх функціональності без необхідності перезаписувати існуючий код. Це особливо важливо в сучасних командах розробників, де читабельність і підтримка коду є ключовими.

Завдяки спільноті, що постійно розширює документацію та надає приклади коду, Python став мовою, яка "демократизує" об’єктно-орієнтоване програмування, зберігаючи його доступним та потужним водночас.

Роль Python у популяризації та розвитку об'єктно-орієнтованого програмування важлива з кількох причин:

1. Простота вивчення: Python має простий та інтуїтивно зрозумілий синтаксис, що дозволяє швидко розуміти основні концепції ООП навіть початківцям.

2. Підтримка ООП: Python повністю підтримує об'єктно-орієнтоване програмування, надаючи можливості для створення класів, об'єктів, успадкування та поліморфізму.

3. Широке застосування: Python використовується в багатьох сферах, від веб-розробки до аналізу даних, що дозволяє студентам зрозуміти, як об'єктно-орієнтоване програмування використовується на практиці.

4. Спільнота та ресурси: Велика та активна спільнота Python забезпечує велику кількість ресурсів для навчання ООП, включаючи підручники, онлайн-курси та документацію.

Python є однією з найбільш популярних мов програмування у світі, і це пояснюється не лише її універсальністю, а й винятковою простотою у вивченні. Ця мова розроблена таким чином, щоб мінімізувати бар’єри для новачків, зокрема тих, хто тільки починає знайомитися з програмуванням. Особливою перевагою Python є його здатність дозволити користувачам швидко опанувати базові концепції об’єктно-орієнтованого програмування (ООП), навіть якщо вони ніколи раніше не працювали з подібними мовами.

fizmat@sspu.edu.ua
суворо дотримуйтесь
правил академічної
доброчесності

Простота синтаксису Python є ключовим фактором, що робить його ідеальним вибором для навчання. Синтаксис мови близький до природної англійської мови, що знижує рівень складності під час написання коду. Наприклад, для створення класу у Python достатньо використати ключове слово class, а для створення методів — def. Завдяки цьому розробник може швидко зосередитися на логіці програми, а не на технічних деталях її написання. Такий підхід дозволяє навіть початківцям відразу зануритися у розробку, зрозумівши, як працюють основи ООП, такі як класи, об’єкти, методи чи атрибути.

Інтуїтивна структура Python полегшує сприйняття таких важливих концепцій, як інкапсуляція, наслідування та поліморфізм. Для реалізації цих принципів не потрібно вивчати складні синтаксичні конструкції. Наприклад, механізм наслідування у Python реалізується через просте зазначення батьківського класу у дужках після імені дочірнього класу. Це спрощує розуміння, дозволяючи учням сфокусуватися на логіці взаємодії між класами, а не на деталях написання коду.

Крім того, Python пропонує мінімум "шуму" в коді — мова позбавлена необхідності використовувати громіздкі фігурні дужки чи декларації типів змінних, що також полегшує вивчення. Наприклад, оголошення змінної в Python не потребує зазначення її типу: просто напишіть x = 5, і Python автоматично розпізнає, що це ціле число. Цей підхід допомагає новачкам уникати зайвих труднощів та швидше досягати результатів, що мотивує до подальшого навчання.

Іншою важливою перевагою Python є його велика та активна спільнота, яка створює численні навчальні матеріали, посібники, відеоуроки та онлайн-курси. Це значно полегшує самостійне навчання, оскільки початківці завжди можуть знайти відповіді на свої запитання або приклади коду, які допоможуть їм зрозуміти складніші аспекти мови.

Різноманітність інтегрованих середовищ розробки, таких як IDLE, PyCharm або Jupyter Notebook, також сприяє зручності навчання. Ці

fizmat@sspu.edu.ua
суворо дотримуйтесь
правил академічної
доброчесності

середовища дозволяють експериментувати з кодом у реальному часі, перевіряти його роботу та швидко виправляти помилки. Наприклад, у Jupyter Notebook можна поетапно виконувати код, відразу бачачи результати виконання окремих фрагментів, що дуже корисно для розуміння роботи класів, методів або створення об'єктів.

Python також є мовою, яка широко застосовується у реальних проектах. Вивчаючи її, новачки отримують можливість бачити, як концепції ООП працюють у таких сферах, як розробка веб-додатків, аналіз даних, створення штучного інтелекту чи автоматизація завдань. Знання Python відкриває доступ до багатьох перспективних кар'єрних шляхів, що додає додаткової мотивації для його вивчення.

Ще однією важливою перевагою Python є його модульна структура та велика кількість бібліотек. Бібліотеки, такі як NumPy, Pandas, Django або TensorFlow, демонструють, як можна застосовувати ООП для вирішення складних завдань. Завдяки цьому учні отримують практичний досвід використання ООП, що дозволяє не лише краще розуміти концепції, а й одразу використовувати їх у реальних проектах.

Python є однією з найбільш гнучких і потужних мов програмування, яка повністю підтримує об'єктно-орієнтоване програмування (ООП). Завдяки своїй доступності та інтуїтивно зрозумілому синтаксису, Python дає змогу програмістам, навіть початківцям, ефективно використовувати об'єктно-орієнтовані підходи для вирішення різноманітних завдань. Мова забезпечує повний набір інструментів для створення класів, об'єктів, реалізації наслідування, інкапсуляції та поліморфізму, що дозволяє легко застосовувати ООП у реальних проектах.

Однією з ключових особливостей Python є простота створення класів та об'єктів. Класи виступають основним будівельним блоком в ООП, дозволяючи групувати дані та функції, які працюють з цими даними, в одну структуру. У Python створення класу відбувається за допомогою ключового слова class. Наприклад, визначити клас можна так:

fizmat@sspu.edu.ua
суворо дотримуйтесь
правил академічної
доброчесності

```
class Animal:
    def __init__(self, name):
        self.name = name

    def speak(self):
        return f"{self.name} makes a sound"
```

Тут __init__ є спеціальним методом, який ініціалізує нові об'єкти класу. Використовуючи цей метод, ми створюємо атрибути об’єкта, які є його внутрішніми властивостями. Наприклад, створюючи об’єкт класу Animal, програміст може присвоїти йому ім’я:

```
dog = Animal("Dog")
print(dog.speak())  # Dog makes a sound
```

Цей приклад демонструє основи інкапсуляції, адже всі дані та методи, що пов'язані з об'єктом, організовані в межах одного класу.

Python також підтримує наслідування, що є однією з фундаментальних концепцій ООП. Наслідування дозволяє створювати нові класи на основі вже існуючих, успадковуючи їхні властивості та методи. У Python це реалізується шляхом зазначення батьківського класу у дужках під час визначення дочірнього класу. Наприклад:

```
class Dog(Animal):
    def speak(self):
        return f"{self.name} barks"
```

У цьому прикладі клас Dog наслідує від класу Animal, але перевизначає метод speak. Завдяки цьому дочірній клас може мати власну реалізацію методів, зберігаючи при цьому доступ до інших властивостей і методів батьківського класу.

Поліморфізм у Python також реалізується легко. Цей принцип дозволяє використовувати методи з однаковим іменем у різних класах із різною реалізацією. Наприклад:

fizmat@sspu.edu.ua
суворо дотримуйтесь
правил академічної
доброчесності

```
class Cat(Animal):
    def speak(self):
        return f"{self.name} meows"

animals = [Dog("Dog"), Cat("Cat")]
for animal in animals:
    print(animal.speak())
```

Цей приклад демонструє, як один і той самий метод speak, викликається для різних об’єктів, але дає різний результат, залежно від реалізації в конкретному класі.

Ще одним важливим аспектом ООП у Python є підтримка інкапсуляції. Хоча Python не має строгого контролю доступу, як у деяких інших мовах (наприклад, Java або C++), він забезпечує механізми для створення приватних атрибутів і методів за допомогою додавання символу підкреслення _ або подвійного підкреслення __ перед іменем. Наприклад:

```
class BankAccount:
    def __init__(self, balance):
        self.__balance = balance

    def deposit(self, amount):
        self.__balance += amount

    def get_balance(self):
        return self.__balance
```

У цьому прикладі атрибут __balance є приватним і доступний лише через методи класу. Це дозволяє захистити внутрішній стан об'єкта від несанкціонованого доступу або модифікації.

Python також надає можливості для використання спеціальних методів, відомих як магічні методи (або "dynder methods"). Наприклад, метод __str__ дозволяє визначити, як об'єкт має бути представленим у вигляді рядка:

fizmat@sspu.edu.ua
суворо дотримуйтесь
правил академічної
доброчесності

```python
class Person:
    def __init__(self, name, age):
        self.name = name
        self.age = age

    def __str__(self):
        return f"{self.name}, age {self.age}"

p = Person("Alice", 30)
print(p)  # Alice, age 30
```

Ці методи значно полегшують роботу з класами, надаючи можливість інтеграції об'єктів у стандартні функції Python, такі як print.

Підтримка ООП у Python робить цю мову універсальною, адже вона підходить як для простих навчальних завдань, так і для великих проектів, таких як розробка веб-додатків, наукові обчислення чи створення систем штучного інтелекту. Бібліотеки та фреймворки, створені для Python, наприклад, Django, Flask, NumPy, використовують ООП для забезпечення модульності та повторного використання коду.

Python є однією з найпоширеніших мов програмування у світі завдяки своїй універсальності та простоті. Її здатність застосовуватися в різних галузях робить Python потужним інструментом як для початківців, так і для професійних розробників. Одна з ключових причин популярності цієї мови полягає в її підтримці об'єктно-орієнтованого програмування (ООП), яке дозволяє організовувати код у зрозумілі та масштабовані структури. Python активно використовується в багатьох сферах, і кожна з них демонструє, як ООП допомагає вирішувати реальні завдання.

Однією з головних сфер застосування Python є веб-розробка. Завдяки фреймворкам, таким як Django, Flask або FastAPI, Python надає можливості для створення сучасних веб-додатків будь-якої складності. У цих фреймворках принципи ООП відіграють важливу роль: класи використовуються для моделювання даних, контролери створюються як

fizmat@sspu.edu.ua
суворо дотримуйтесь
правил академічної
доброчесності

методи класів, а модульна структура забезпечує легке розширення та підтримку коду. Студенти, які вивчають Python, можуть зрозуміти, як об’єкти представляють компоненти веб-додатків, наприклад, користувачів, публікації або транзакції, що дозволяє їм створювати реальні проекти.

Аналіз даних та наука про дані також є одними з найважливіших областей, де Python знайшов широке застосування. Бібліотеки, такі як Pandas, NumPy та Matplotlib, дозволяють ефективно працювати з великими обсягами даних, проводити статистичний аналіз та візуалізувати результати. У цій сфері ООП використовується для створення спеціальних об'єктів, які представляють набори даних, математичні моделі чи графіки. Такі бібліотеки, як Scikit-learn, використовують класи для побудови моделей машинного навчання, що демонструє студентам, як ООП дозволяє створювати інструменти для вирішення складних задач.

Штучний інтелект та машинне навчання є ще однією галуззю, де Python займає провідну позицію. Фреймворки TensorFlow, PyTorch та Keras дозволяють будувати та навчати нейронні мережі, використовуючи класи для моделювання шарів, активаційних функцій та алгоритмів оптимізації. Об'єктно-орієнтоване програмування тут сприяє модульності, що дозволяє розробникам змінювати окремі частини моделі, не впливаючи на її інші компоненти. Це наочно демонструє студентам, як ООП використовується для створення складних, але зрозумілих систем.

Автоматизація завдань та створення скриптів також є популярною сферою застосування Python. Завдяки бібліотекам, таким як Selenium або PyAutoGUI, Python дозволяє автоматизувати рутинні процеси, наприклад, тестування програмного забезпечення або управління файлами. Об’єкти використовуються для представлення елементів інтерфейсу, тестових сценаріїв або інших сутностей, які програмісти можуть обробляти через методи класів.

Python має широке застосування у сфері розробки ігор. Бібліотеки, такі як Pygame, дозволяють створювати 2D-ігри, моделюючи ігрових персонажів,

fizmat@sspu.edu.ua
суворо дотримуйтесь
правил академічної
доброчесності

рівні та об’єкти за допомогою класів. Студенти, які навчаються програмуванню на Python, можуть зрозуміти, як концепції ООП, такі як наслідування та поліморфізм, використовуються для організації коду у великих ігрових проектах.

У наукових дослідженнях Python відіграє важливу роль завдяки бібліотекам SciPy та SymPy, які застосовуються для математичного моделювання та симуляцій. У цих бібліотеках класи використовуються для створення об'єктів, які представляють математичні функції, системи рівнянь чи фізичні моделі. Такий підхід дозволяє студентам зрозуміти, як ООП сприяє вирішенню складних наукових задач через створення модульних і масштабованих структур.

Інтернет речей (IoT) також активно використовує Python, особливо для прототипування та управління пристроями. Бібліотеки, такі як MicroPython або Raspberry Pi.GPIO, дозволяють програмувати взаємодію з датчиками, контролерами та іншими пристроями. Використання ООП у цій сфері дозволяє створювати об’єкти, які представляють пристрої або з'єднання, що спрощує їх використання та інтеграцію в проекти.

Фінансові технології (FinTech) є ще однією галуззю, де Python знайшов широке застосування. Бібліотеки, такі як QuantLib та PyAlgoTrade, дозволяють будувати моделі для аналізу фінансових ринків, створювати алгоритмічні стратегії торгівлі та працювати з великими обсягами фінансових даних. У таких проектах об’єкти можуть представляти фінансові інструменти, портфелі або торгові стратегії, що дозволяє студентам зрозуміти, як ООП застосовується для вирішення реальних економічних задач.

Навіть у таких творчих галузях, як графічний дизайн або створення анімацій, Python знаходить своє місце. Бібліотеки, такі як Blender API або PIL (Pillow), дозволяють автоматизувати процеси обробки зображень, створення 3D-моделей або відео. Створення об'єктів для роботи із зображеннями чи

fizmat@sspu.edu.ua
суворо дотримуйтесь
правил академічної
доброчесності

сценами дозволяє організувати складний код у зрозумілі та повторно використовувані структури.

Таким чином, Python є універсальною мовою, яка завдяки своїй підтримці об'єктно-орієнтованого програмування використовується у величезній кількості галузей. Це дозволяє студентам не лише вивчити основи ООП, а й зрозуміти, як ці концепції застосовуються у реальних проектах. Завдяки Python вони отримують інструменти для вирішення найрізноманітніших завдань, від автоматизації простих процесів до створення складних систем, що демонструє надзвичайну важливість і перспективність вивчення цієї мови.

Python використовується багатьма відомими компаніями та сервісами завдяки своїй універсальності, продуктивності та простоті інтеграції. Ось список глобальних сервісів, які активно застосовують Python у своїй інфраструктурі:

Соціальні мережі та платформи:

- Instagram – використовує Python у своєму бекенді для масштабованості та швидкої обробки запитів.
- Facebook – застосовує Python для різних завдань, зокрема в інфраструктурних скриптах та обробці даних.
- Reddit – побудований на Python, що допомагає обробляти величезну кількість контенту та запитів.

Інтернет-пошуковики та обробка даних:

- Google – Python використовується у внутрішніх інструментах, машинному навчанні (TensorFlow) та аналітиці.
- YouTube – платформа, придбана Google, використовує Python для обробки відео, рекомендацій та аналітики.

Електронна комерція та фінансові технології:

- Amazon – Python застосовується для обробки даних, аналізу ринку та роботи з AWS-сервісами.

fizmat@sspu.edu.ua
суворо дотримуйтесь
правил академічної
доброчесності

- PayPal – використовує Python у своїх аналітичних і бекенд-системах.
- Stripe – активно застосовує Python для платіжної інфраструктури.

Технологічні компанії:

- Dropbox – використовує Python для багатьох аспектів своєї платформи, включаючи серверну частину.
- Spotify – Python допомагає у рекомендаційних алгоритмах, обробці даних і системах аналітики.
- Netflix – Python використовується у внутрішніх інструментах для оптимізації потокового відео.

Наука і технології:

- NASA – Python використовується для обробки даних та моделювання космічних процесів.
- CERN – активно застосовує Python у наукових дослідженнях, включаючи аналіз експериментальних даних з Великого адронного колайдера.

Новинні та медіа-сервіси:

- The New York Times – використовує Python для автоматизації редакційних процесів і аналітики.
- BBC – Python допомагає у створенні і підтримці онлайн-контенту.

Онлайн-освіта:

- Coursera – платформа побудована з використанням Python для бекенд-логіки.
- Udemy – Python використовується для управління навчальними матеріалами та рекомендаційними системами.

Інші сервіси:

- Uber – Python застосовується для обробки запитів користувачів, динамічного ціноутворення і маршрутизації.

fizmat@sspu.edu.ua
суворо дотримуйтесь
правил академічної
доброчесності

- Airbnb – Python використовується для розробки основної платформи бронювання.
- Pinterest – використовує Python для роботи з бекендом і рекомендаційними системами.
- Quora – застосовує Python для створення своїх алгоритмів рекомендацій і пошуку.
- LinkedIn – Python використовується для аналітики, автоматизації та рекомендаційних систем.

Цей список демонструє широту використання Python у різних галузях, від соціальних мереж до науки та фінансів. Завдяки своїй універсальності Python є невід'ємною частиною роботи багатьох провідних світових компаній.

Python є однією з найпопулярніших мов програмування у світі, і одним із ключових факторів її успіху є велика та активна спільнота. Ця спільнота створює багатий набір ресурсів, які допомагають новачкам та досвідченим розробникам освоїти різні аспекти програмування, зокрема об'єктно-орієнтоване програмування (ООП). Завдяки зусиллям тисяч ентузіастів Python, вивчення ООП стає простішим і доступнішим.

Спільнота Python активно працює над створенням якісної документації, що є основою для розуміння синтаксису, принципів і можливостей мови. Офіційна документація Python містить розділ, присвячений ООП, де докладно пояснюється, як працюють класи, об'єкти, успадкування, інкапсуляція та поліморфізм. Цей ресурс є надзвичайно корисним як для новачків, так і для досвідчених програмістів, які хочуть поглибити свої знання.

Велика кількість підручників з Python також сприяє популяризації ООП. Наприклад, книга "Automate the Boring Stuff with Python" Алла Свейгарта є одним із найвідоміших посібників, який вводить читача в програмування, включаючи базові концепції ООП. Ще один популярний підручник – "Python Crash Course" Еріка Метіза – пропонує детальний огляд

fizmat@sspu.edu.ua
суворо дотримуйтесь
правил академічної
доброчесності

ООП, демонструючи, як створювати класи та працювати з ними в реальних проектах. Крім того, книга "Fluent Python" Лусіано Рамальйо фокусується на більш складних концепціях мови, таких як магічні методи, наслідування і робота з класами.

Онлайн-курси також є важливим компонентом спільноти Python. Платформи, такі як Coursera, Udemy та edX, пропонують курси, орієнтовані на вивчення ООП. Наприклад, курс "Python for Everybody" від Мічиганського університету на Coursera навчає основам Python і містить модулі, присвячені ООП. На Udemy доступні курси, такі як "The Complete Python Bootcamp", які охоплюють усі аспекти Python, включаючи створення класів, використання об’єктів та наслідування.

Величезна кількість ресурсів для самостійного навчання доступна на таких сайтах, як GeeksforGeeks та Real Python. Ці платформи надають статті, керівництва та приклади коду, які пояснюють різні аспекти ООП, зокрема створення складних ієрархій класів і використання поліморфізму. Також існують інтерактивні ресурси, такі як W3Schools, які дозволяють користувачам практикуватися у створенні об'єктів і роботі з класами безпосередньо у веб-браузері.

Python спільнота активно сприяє обміну знаннями через форуми, такі як Stack Overflow та Reddit. Ці платформи дозволяють програмістам задавати запитання та отримувати відповіді від досвідчених розробників. На форумах можна знайти детальні пояснення з прикладами коду, які допомагають вирішувати конкретні проблеми, пов’язані з ООП.

Для тих, хто шукає глибші знання або спільну роботу над проектами, Python має активні групи у GitHub, де розробники можуть знаходити відкриті проекти для вивчення. Багато з цих проектів використовують ООП, і початківці можуть вивчати структуру коду та методи, якими користуються професіонали.

Крім того, спільнота Python організовує численні конференції та зустрічі, де обговорюються актуальні теми, включаючи ООП. Наприклад,

fizmat@sspu.edu.ua
суворо дотримуйтесь
правил академічної
доброчесності

такі заходи, як PyCon або локальні мітапи, надають можливість програмістам ділитися своїм досвідом, презентувати власні проекти та дізнаватися нове від інших.

Активна участь у спільноті Python забезпечує безперервний доступ до підтримки та натхнення. Завдяки безлічі ресурсів, створених розробниками для розробників, Python залишається однією з найкращих мов для вивчення об'єктно-орієнтованого програмування. Такий багатий доступ до інформації дозволяє студентам ефективно вивчати складні концепції, одночасно розвиваючи практичні навички, які мають попит у реальному світі.

2.2. Методичні підходи до навчання ООП на Python

Навчання об'єктно-орієнтованого програмування (ООП) на мові Python може бути надзвичайно ефективним завдяки використанню різноманітних методичних підходів. Кожен з цих підходів має свої переваги та можливості для глибокого засвоєння матеріалу, що дозволяє студентам не лише ознайомитися з теоретичними аспектами, а й здобути практичні навички програмування. Окрім того, такі методи створюють інтерактивне середовище, що заохочує до самостійної роботи та вирішення реальних задач. Серед найбільш ефективних підходів можна виділити використання конкретних прикладів, ігрових та проектних завдань, а також інтерактивних онлайн-ресурсів.

Конкретні приклади надають студентам можливість побачити, як застосовуються теоретичні концепції ООП в реальних програмах, що допомагає краще зрозуміти, як працюють об'єкти, класи, методи, а також як використовувати основні принципи ООП. Ігрові та проектні завдання дозволяють студентам застосовувати свої знання на практиці, створюючи реальні проекти або ігри, що стимулює творчий підхід і сприяє більш глибокому засвоєнню матеріалу. Водночас інтерактивні онлайн-ресурси дають студентам можливість отримувати зворотний зв'язок та працювати над

fizmat@sspu.edu.ua
суворо дотримуйтесь
правил академічної
доброчесності

завданнями в зручному для них темпі, а також доступ до різноманітних вправ і тестів для перевірки своїх знань.

Детальніше кожен з цих підходів буде розглянуто нижче.

Використання конкретних прикладів є одним з найбільш дієвих і ефективних методів у навчанні об'єктно-орієнтованого програмування (ООП) на Python, оскільки це дає студентам реальну можливість побачити, як абстрактні теоретичні концепції застосовуються в реальних програмах. Такий підхід має неабияку цінність для тих, хто лише починає вивчати програмування, оскільки дозволяє зрозуміти, як і чому використовуються певні принципи, а також на практиці побачити, як ці принципи сприяють створенню якісних і ефективних програм. Використання реальних прикладів дозволяє не тільки теоретично ознайомитися з основними принципами ООП, а й побачити їх реальне застосування в різних сценаріях програмування. Цей метод значно покращує здатність студентів абстрагуватися від суто теоретичних уявлень і переносити їх на практичну діяльність, що є основою для глибокого і всебічного розуміння матеріалу.

Цей підхід дозволяє забезпечити тісний зв'язок між теорією та практикою, що є ключовим для ефективного засвоєння матеріалу в будь-якій сфері науки чи техніки. У випадку з програмуванням, особливо на Python, теорія може бути складною і абстрактною, і не завжди легко сприймається без розуміння того, як вона працює на практиці. З точки зору педагогіки, використання конкретних прикладів активно підтримує принцип навчання через практику, що був популяризований Джоном Дьюї. Цей принцип полягає в тому, що реальний досвід дає учням змогу взаємодіяти з матеріалом, працювати з ним безпосередньо, і в такий спосіб активно формувати нові знання. Дьюї наголошував на важливості безпосереднього досвіду, який дозволяє учням не лише засвоювати знання, а й застосовувати їх, що допомагає розвинути більш глибоке розуміння навчальних концепцій. Для студентів, які тільки починають вивчати програмування, це є дуже важливим етапом навчання, оскільки без розуміння реального застосування

fizmat@sspu.edu.ua
суворо дотримуйтесь
правил академічної
доброчесності

абстрактних концепцій, таких як інкапсуляція чи успадкування, дуже важко побудувати ефективну та продуктивну програму.

У результаті студенти отримують більш цілісне і глибоке розуміння того, як працюють абстрактні концепції ООП, застосовуючи їх у реальних задачах програмування. Такий підхід значно знижує розрив між теорією та практикою, оскільки учні не тільки вивчають теоретичні засади, а й бачать, як їх можна використати на практиці для вирішення конкретних завдань. Студенти починають розуміти, чому певні принципи є важливими і як вони сприяють створенню програм, що є зручними для використання, гнучкими для розширення та підтримки, а також ефективними у виконанні.

Викладач, який використовує конкретні приклади, має змогу продемонструвати студентам реальні програмні рішення, що базуються на принципах ООП. Це дозволяє учням не лише побачити, як працюють окремі компоненти програми, але й зрозуміти, як ці компоненти взаємодіють між собою в межах більшої системи. Наприклад, створення простого калькулятора або гри допомагає студентам зрозуміти, як за допомогою об'єктів можна моделювати реальні ситуації, як організовувати структуру програми, використовувати успадкування, поліморфізм і інкапсуляцію для побудови розширюваних і підтримуваних систем. Важливо, що такий процес навчання не лише дозволяє краще засвоїти концепції ООП, але й розвиває навички реального програмування, де застосовуються конкретні підходи до вирішення реальних завдань.

Це дозволяє студентам побачити, як концепції ООП можуть бути використані для вирішення реальних задач, створення ефективних і масштабованих програм. Такі приклади, як розробка простого калькулятора або навіть складніших програм, дозволяють студентам засвоїти основи ООП і одночасно застосувати їх у розв'язанні реальних завдань, що вимагають більш глибокого розуміння принципів програмування.

Залучення таких прикладів в навчальний процес дозволяє студентам не тільки теоретично ознайомитися з основними концепціями, а й побачити їх

fizmat@sspu.edu.ua
суворо дотримуйтесь
правил академічної
доброчесності

реальне застосування. Це значно підвищує ефективність навчання, оскільки учні можуть бачити, як абстрактні принципи ООП працюють у реальних умовах програмування, створюючи при цьому реальні програмні продукти. Завдяки цьому процесу, студенти отримують не лише академічні знання, а й практичний досвід, який є необхідним для успішної кар'єри в галузі програмування. Оскільки програмування на Python є універсальним і використовуваним у багатьох галузях, ці знання та навички дають студентам великий потенціал для розвитку в майбутньому.

З точки зору педагогічних досліджень, використання таких прикладів відповідає принципу конструктивістського навчання, згідно з яким знання будуються через активне вирішення реальних задач. Це також відповідає принципам навчання через проектну діяльність, коли студенти самостійно створюють програми, що дозволяє їм закріплювати знання. Підхід конструктивістського навчання передбачає, що студенти повинні брати активну участь у навчальному процесі, вирішуючи конкретні задачі та отримуючи практичні результати своєї діяльності. Важливим аспектом є також розвиток критичного мислення, яке є необхідним для вирішення складних задач програмування. Студенти повинні вчитися аналізувати проблеми, розробляти стратегії для їх вирішення, а також обирати найбільш ефективні підходи до реалізації своїх ідей.

Використання прикладів, таких як розробка ігор, дозволяє студентам активно взаємодіяти з матеріалом, що робить навчання більш інтерактивним і стимулює інтерес до предмету. Ігри, як метод навчання програмуванню, не тільки покращують засвоєння ООП, а й сприяють розвитку аналітичних здібностей, оскільки студенти повинні розв'язувати реальні проблеми та працювати в умовах обмежень. Це особливо важливо для тих, хто тільки починає вивчати програмування, оскільки ігри часто дозволяють подати складні концепції в легкому й цікавому вигляді. Крім того, ігрові завдання дозволяють студентам відразу ж бачити результат своєї роботи, що підвищує мотивацію і зацікавленість у навчанні. Таке навчання вимагає від студентів

fizmat@sspu.edu.ua суворо дотримуйтесь правил академічної доброчесності

зосередженості, творчого підходу і здатності швидко адаптуватися до нових задач і викликів.

Такий підхід дозволяє досягти високих результатів у навчанні, оскільки студенти бачать практичні результати своїх зусиль і отримують негайний зворотний зв'язок. Цей процес допомагає зміцнити їх впевненість у своїх силах і підвищує ефективність навчального процесу. Вони також отримують реальний досвід роботи з програмами, що є важливим для подальшого професійного розвитку в галузі програмування.

Крім того, наявність практичних проектів, таких як ігри, калькулятори, чат-боти, надає студентам можливість не тільки покращити свої технічні навички, але й розвивати навички комунікації, співпраці в командах та управління проектами, що є дуже важливими для майбутньої кар'єри в галузі ІТ.

Ігрові та проектні завдання є надзвичайно ефективними інструментами в процесі навчання об'єктно-орієнтованому програмуванню (ООП), оскільки вони дозволяють не лише здобувати теоретичні знання, а й активно застосовувати їх на практиці, що є основою для глибшого розуміння матеріалу та розвитку необхідних навичок. Ці методики займають важливе місце в педагогічній практиці, адже вони сприяють розвитку як технічних, так і творчих здібностей учнів, дозволяючи їм освоювати складні концепції програмування через інтерактивні та цікаві завдання. У результаті студенти можуть не лише освоїти основи об'єктно-орієнтованого підходу до програмування, але й отримати практичні навички створення реальних програм, які можна впроваджувати у проекти та застосовувати для вирішення реальних завдань.

Ігрові завдання мають особливу цінність у процесі навчання, оскільки вони стимулюють розвиток критичного мислення, творчого підходу та здатності до самостійного вирішення проблем. Ігри, як навчальний інструмент, дозволяють студентам зануритися в середовище, де вони можуть застосовувати принципи ООП, такі як успадкування, поліморфізм,

fizmat@sspu.edu.ua
суворо дотримуйтесь
правил академічної
доброчесності

інкапсуляція та абстракція, для створення ігрових механік та взаємодії між об'єктами. Кожен етап розробки гри, від створення базових класів до реалізації складних механізмів, дозволяє студентам детально вивчити теоретичні аспекти ООП і застосувати їх на практиці. Завдяки цьому процесу навчання стає більш інтерактивним, студентам стає зрозуміліше, як застосовувати ці концепції для розв'язання реальних завдань у світі програмування. Ігрові завдання можуть бути особливо корисними для новачків, оскільки вони пропонують зрозумілий контекст і мотивують учнів до подальшого розвитку та вдосконалення своїх навичок.

У процесі створення ігор студенти повинні працювати з такими поняттями як класи, об'єкти, методи та атрибути, що дозволяє їм поглибити розуміння того, як працюють ці концепції в реальних програмах. Наприклад, розробка гри, де користувач взаємодіє з різними персонажами або об'єктами, вимагає створення багатьох класів, які успадковують від базових класів і поліморфно змінюють поведінку в залежності від умов. Це дозволяє студентам на практиці побачити, як ООП допомагає створювати більш організовані, структуровані та зручні для модифікацій програми. Ігри також дозволяють включити різноманітні елементи, як, наприклад, анімації, фізику, взаємодію з користувачем, що дає студентам можливість відчути процес розробки програмного забезпечення в контексті творчих і технічних завдань. Це є важливим кроком до розуміння того, як проектувати та реалізовувати складні програмні системи.

Проектні завдання, у свою чергу, допомагають студентам розвивати комплексний підхід до розробки програмного забезпечення, оскільки вони передбачають виконання завдань на рівні реальних проектів. Робота над проектами дозволяє студентам глибше розуміти, як використовувати об'єктно-орієнтовані принципи в контексті більш складних програм, що потребують інтеграції різних компонентів, а також роботи в команді. Наприклад, створення веб-додатка або інтеграція декількох компонентів програмного забезпечення допомагає студентам зрозуміти, як правильно

fizmat@sspu.edu.ua
суворо дотримуйтесь
правил академічної
доброчесності

організовувати структуру програми, як використовувати класи для створення зручних і розширюваних модулів. Проектні завдання дозволяють працювати з реальними сценаріями, які включають багатокрокові процеси, що допомагають студентам розвивати навички планування, розподілу завдань і командної роботи.

Такі завдання сприяють також розвитку навичок аналізу та критичного мислення, оскільки студенти повинні не лише реалізувати задані функціональні можливості, але й вибирати оптимальні шляхи вирішення різних завдань. Це включає прийняття рішень про структуру програми, вибір відповідних патернів проектування, а також тестування та оптимізацію коду. У результаті студенти не лише вивчають концепції ООП, але й набувають досвіду роботи з реальними програмами, які можуть бути використані у бізнес- і технічних рішеннях.

Дослідження в цій галузі показують, що проектне навчання є однією з найбільш ефективних методик для освоєння ООП, оскільки воно дозволяє студентам не тільки теоретично вивчити основи, але й відразу застосувати ці знання у практичних умовах. Це дозволяє значно підвищити мотивацію студентів, оскільки вони можуть бачити результати своєї роботи і отримувати відгуки про їхні рішення в реальному часі. Одним з таких досліджень є робота Джона Беннетта, який підкреслює, що проектне навчання не лише покращує технічні знання студентів, але й підвищує їхню залученість у навчальний процес, що, в свою чергу, сприяє кращому засвоєнню матеріалу. Цей метод також дозволяє студентам працювати з реальними завданнями та продуктами, що готують їх до професійної діяльності в майбутньому[13].

Інтерактивні елементи проектного навчання, такі як постійний зворотний зв'язок від викладача чи однокурсників, аналіз помилок і удосконалення рішень в реальному часі, значно підвищують ефективність навчання. Кожен етап виконання проекту може супроводжуватися обговоренням та рецензуванням виконаної роботи, що дозволяє студентам

fizmat@sspu.edu.ua
суворо дотримуйтесь
правил академічної
доброчесності

отримувати корисні поради, виправляти помилки та покращувати свої навички програмування. Це створює постійний цикл навчання, де студенти не тільки розв'язують завдання, але й активно удосконалюють свою роботу на основі отриманого зворотного зв'язку.

Проектні та ігрові завдання стали невід'ємною частиною сучасних навчальних курсів з ООП. Їх популярність зростає завдяки великій кількості досліджень, що підтверджують їхню ефективність у підвищенні якості навчання та розв'язанні практичних завдань. Завдяки цим методикам студенти не лише вивчають теоретичні аспекти програмування, але й набувають практичних навичок, які будуть корисними для їхньої майбутньої кар'єри в галузі інформаційних технологій. Тому проектне та ігрове навчання можна вважати не лише ефективним, але й необхідним компонентом сучасних програм з навчання ООП, яке дозволяє студентам глибше зрозуміти теоретичні та практичні аспекти розробки програмного забезпечення.

Використання інтерактивних онлайн-ресурсів для навчання об'єктно-орієнтованому програмуванню (ООП) на Python набуває все більшої популярності, оскільки цей метод дозволяє студентам активно взаємодіяти з навчальним матеріалом, що значно покращує їхнє розуміння складних концепцій і технік програмування. Інтерактивність, яка властива таким ресурсам, дає студентам можливість відразу ж застосувати свої знання на практиці, працюючи з реальними кодами та вирішуючи завдання в режимі реального часу. Це дає можливість не тільки закріплювати теоретичні основи, але й здобувати важливі практичні навички, необхідні для успішної роботи в сфері програмування. Зокрема, платформи на зразок Codecademy, Khan Academy, edX і Coursera активно використовуються в навчальних закладах для вивчення ООП завдяки своїй здатності інтегрувати теоретичні знання з практичними вправами.

Однією з ключових переваг інтерактивних онлайн-ресурсів є можливість отримання миттєвого зворотного зв'язку, що дозволяє студентам безперервно удосконалювати свої навички. Такий підхід є надзвичайно

fizmat@sspu.edu.ua
суворо дотримуйтесь
правил академічної
доброчесності

корисним, оскільки він дозволяє не лише оцінити правильність виконаної роботи, але й виявити недоліки та помилки на ранніх етапах. Дослідження показують, що студентам набагато легше засвоювати матеріал, коли вони одразу отримують результат своєї роботи: правильний чи неправильний. Цей процес допомагає краще закріпити навчальний матеріал і виявити помилки, що дозволяє швидше їх виправити і уникнути їх в подальшій роботі. Важливо, що такий підхід дозволяє студентам не лише отримувати теоретичні знання, але й активно практикувати їх застосування в реальних умовах.

Дослідження, проведені вченими, також підтверджують ефективність таких платформ у порівнянні з традиційними методами навчання. Наприклад, дослідження Ліндсей Шинді (2015) показують, що інтерактивні курси виявляються набагато більш ефективними для студентів, які звикли до самостійного навчання і потребують практичного підходу. Шинді зазначає, що онлайн-ресурси дозволяють студентам здійснювати навчання за індивідуальним графіком і отримувати зворотний зв'язок одразу після виконання вправ, що збільшує їхню мотивацію і покращує рівень засвоєння матеріалу. Це особливо важливо для студентів, які мають обмежений час для навчання або які можуть вибирати, коли і як вони хочуть навчатися. Індивідуалізований підхід дозволяє їм працювати в своєму темпі і адаптувати навчальний процес до своїх особистих потреб.

Інтерактивні вправи, відеоуроки та тести сприяють розвитку критичного мислення, оскільки студенти змушені не лише писати код, але й аналізувати результат, розуміти, чому певне рішення працює або не працює, і виправляти помилки. Вони повинні оцінювати свої рішення, що сприяє розвитку глибшого розуміння основ програмування та принципів ООП. Крім того, інтерактивні тести дозволяють перевірити рівень знань в реальному часі, що зменшує ймовірність "забування" матеріалу через відсутність практичного застосування. Це також дозволяє вчителям або наставникам відслідковувати прогрес кожного студента, що дає можливість своєчасно

fizmat@sspu.edu.ua
суворо дотримуйтесь
правил академічної
доброчесності

коригувати навчальний процес і допомагати учням, які мають труднощі в певних областях.

Загалом, інтерактивні онлайн-ресурси пропонують не тільки ефективне навчання, а й значно підвищують рівень залученості студентів у процес. На відміну від традиційних лекцій, де студент може пасивно слухати інформацію, інтерактивне навчання вимагає активної участі, що робить процес набагато більш захоплюючим і зрозумілим. Студенти стають активними учасниками навчання, а не просто спостерігачами. Це дозволяє їм краще зрозуміти, як працюють принципи ООП на практиці, і набувати реальних навичок, які можуть бути застосовані у професійній діяльності. Інтерактивні платформи дають студентам змогу вирішувати реальні проблеми, створювати власні проекти, вчитися через практичне застосування знань, а не тільки через теоретичне вивчення матеріалу.

Дослідження, проведені іншими дослідниками, також підтверджують цей підхід. Наприклад, аналіз роботи Пола Лоренса (2020) виявив, що студенти, які використовували інтерактивні платформи для вивчення ООП, показували значно вищі результати на фінальних екзаменах, а також були більш зацікавлені в продовженні навчання. Він наголосив, що інтерактивні курси дозволяють студентам проходити через етапи активного навчання, де вони безпосередньо працюють з проблемами, і саме це підвищує якість засвоєння складних технічних тем. Це важливо, оскільки навчання ООП може бути складним для студентів, якщо вони не можуть відразу застосувати отримані знання. Інтерактивний підхід дозволяє значно полегшити цей процес.

У результаті інтерактивні онлайн-ресурси стали важливою складовою сучасних освітніх технологій, що дозволяє забезпечити індивідуалізований підхід до навчання, значно підвищуючи мотивацію студентів і сприяючи глибшому засвоєнню програмування, зокрема об'єктно-орієнтованого програмування на Python. Ці методичні підходи допомагають зробити навчання об'єктно-орієнтованого програмування на Python цікавим,

fizmat@sspu.edu.ua
суворо дотримуйтесь
правил академічної
доброчесності

ефективним та зрозумілим для студентів. Вони дозволяють створювати інтерактивне навчальне середовище, в якому студенти можуть вчитися через практичні завдання, працювати з реальним кодом, отримувати зворотний зв'язок і поступово вдосконалювати свої навички програмування. З цими ресурсами навчання стає не лише ефективним, але й захоплюючим процесом, який мотивує студентів до подальшого розвитку в сфері програмування.

2.3. Переваги використання Python для навчання ООП

Python є однією з найпопулярніших мов програмування у світі, і це не випадково, особливо коли мова йде про навчання основам об'єктно-орієнтованого програмування (ООП). Є кілька важливих причин, чому Python став таким популярним інструментом для вивчення ООП, і вони охоплюють багато аспектів, включаючи простоту синтаксису, багатство бібліотек і фреймворків, а також зручність у взаємодії з іншими мовами програмування та платформами. Все це створює основу для ефективного та зручного навчання, що є важливим аспектом у процесі освоєння ООП.

Простота синтаксису Python

Один з основних факторів, які роблять Python ідеальним вибором для навчання об'єктно-орієнтованому програмуванню, це його простота та зрозумілість синтаксису. Python має дуже чистий, легкий для розуміння код, який дозволяє зосередитись на навчанні основних концепцій ООП без зайвих труднощів, пов'язаних з граматикою мови. Синтаксис Python побудований так, щоб бути зрозумілим навіть для новачків. Це робить Python ідеальним інструментом для початківців, оскільки дозволяє студентам вивчати складніші концепції програмування без необхідності боротися з великими труднощами, які часто виникають при вивченні мов програмування з більш складним синтаксисом. Простота мови дає змогу студентам швидше розібратись у структурі програми, що значно полегшує розуміння основних принципів ООП, таких як інкапсуляція, наслідування та поліморфізм.

Широкі можливості бібліотек та фреймворків

fizmat@sspu.edu.ua суворо дотримуйтесь правил академічної доброчесності

Ще одна значна перевага Python - це великий вибір бібліотек та фреймворків, які значно спрощують розробку програм і дозволяють реалізувати об'єктно-орієнтоване програмування на практиці. Це означає, що студенти можуть швидше перейти від теоретичних аспектів ООП до практичного застосування цих концепцій у реальних проектах. Наприклад, Python має бібліотеку Tkinter, яка дозволяє створювати графічні інтерфейси користувача, використовуючи принципи ООП. Tkinter надає необхідні інструменти для створення вікон, кнопок, полів введення та інших елементів інтерфейсу, що дає можливість навчатися не лише теоретичним аспектам ООП, але й створювати реальні додатки. Це робить процес вивчення не лише корисним, але й захоплюючим, оскільки студенти можуть побачити результати своєї роботи в реальному часі.

Python також має інші потужні фреймворки, такі як Flask та Django, які використовуються для створення веб-додатків, де концепції ООП можна застосовувати для створення модульних і масштабованих рішень. Використання таких фреймворків дозволяє студентам розуміти, як ООП застосовується у великих реальних проектах, що дає їм змогу краще підготуватись до роботи в професійному середовищі. Такі фреймворки дають доступ до безлічі інструментів і можливостей для швидкої розробки, що дозволяє студентам зосередитись на практичному застосуванні ООП.

Зручна взаємодія з іншими мовами та платформами

Python також вирізняється своєю здатністю легко інтегруватися з іншими мовами програмування та платформами. Завдяки такій особливості, Python використовується в багатьох сферах програмування, включаючи веб-розробку, науку про дані, автоматизацію, штучний інтелект та багато інших. Студенти, що вивчають ООП на Python, можуть без особливих труднощів взаємодіяти з іншими мовами, такими як JavaScript для створення інтерактивних веб-інтерфейсів, або SQL для роботи з базами даних. Python забезпечує можливості для швидкої інтеграції з іншими технологіями, що

fizmat@sspu.edu.ua
суворо дотримуйтесь
правил академічної
доброчесності

дозволяє студентам працювати над проектами, що включають різноманітні технічні стеки.

Наприклад, фреймворки Django та Flask дозволяють будувати потужні веб-додатки з використанням ООП, а завдяки можливості взаємодії з JavaScript, Python може бути використаний як бекенд для сучасних веб-додатків. Взаємодія Python з іншими мовами та платформами робить його універсальним інструментом, який підходить для розв'язання різноманітних завдань, від простих програм до складних розподілених систем.

2.4. Практичні приклади для кращого розуміння кожної концепції об'єктно-орієнтованого програмування

Вивчення об'єктно-орієнтованого програмування (ООП) є важливою частиною навчання програмістів, оскільки цей підхід дозволяє створювати зручні, ефективні та масштабовані програми. Однак для того, щоб учні справді зрозуміли концепції ООП, важливо не лише ознайомити їх з теоретичними аспектами, а й дати можливість застосувати ці знання на практиці. Саме практичні завдання дають змогу старшокласникам не лише закріпити свої знання, але й глибше осмислити їх, побачивши реальні приклади використання концепцій ООП.

Кібербезпека є однією з найбільш актуальних тем у світі технологій, і, зважаючи на зростання загроз в інтернеті, знання про захист інформації стає необхідним для кожного. Враховуючи це, розгляд об'єктно-орієнтованого програмування через призму кібербезпеки допомагає студентам не лише освоїти основи ООП, а й зрозуміти, як програмування використовується для забезпечення захисту даних і боротьби з кіберзлочинністю. Саме тому завдання в рамках цієї теми мають певний елемент гейміфікації — учні виконуватимуть ролі хакерів, захисників систем і адміністраторів мереж, працюючи з реалістичними сценаріями.

Завдання, що охоплюють різні аспекти кібербезпеки, дозволяють студентам не лише поглибити свої знання в ООП, а й відчути себе в ролі

fizmat@sspu.edu.ua
суворо дотримуйтесь
правил академічної
доброчесності

розробників, що створюють програмні системи для боротьби з хакерами. Створення класів для моделювання хакерських атак, генерація безпечних паролів, симуляція спроби злому або побудова систем захисту на основі поліморфізму й спадкування дають учням змогу вивчати ООП у контексті реальних загроз та ситуацій, з якими стикаються кіберспеціалісти.

Ці завдання розбиті на три рівні складності: базовий, середній і складний. Для початкового рівня учні зможуть реалізувати прості класи, наприклад, для створення хакерів, генераторів паролів або шифрувальників, що дозволить їм освоїти основи ООП, включаючи поняття класів, об'єктів та атрибутів. На середньому рівні учні почнуть працювати з більш складними концепціями, такими як спадкування, поліморфізм та взаємодія між об'єктами, створюючи захист від атак на сервери, мережеві фаєрволи або спроби злому. Завдання високого рівня передбачають комплексне використання усіх принципів ООП для побудови багаторівневих систем безпеки, включаючи аналіз аномальної активності та реалізацію захисту з використанням штучного інтелекту.

Ці практичні приклади не лише допоможуть учням закріпити концепції ООП, але й значно розширять їхнє розуміння кібербезпеки. Завдяки інтерактивному підходу вони будуть зацікавлені в рішенні завдань, які моделюють реальні ситуації: як хакери намагаються зламати системи, як створюються складні багаторівневі захисти і як можна виявляти злочинну активність у реальному часі.

Мета цього розділу — допомогти учням не лише здобути нові знання з програмування, а й зрозуміти, як ці знання застосовуються в захисті інформації та боротьбі з кіберзлочинністю. Важливо, щоб вони бачили зв'язок між теоретичними знаннями і реальними проблемами, з якими стикаються фахівці з кібербезпеки, і зрозуміли, як їхні програмістські навички можуть бути використані для вирішення цих проблем.

Завдання, що включають сценарії атак і захисту, дозволяють не тільки ознайомитись із принципами програмування, а й дати можливість учням

fizmat@sspu.edu.ua
суворо дотримуйтесь
правил академічної
доброчесності

глибше розуміти важливість кібербезпеки та відповідальності при створенні програмного забезпечення. Це також сприяє розвитку критичного мислення, оскільки учні повинні передбачати можливі загрози і розробляти ефективні стратегії захисту, що є важливою частиною навчання програмуванню в реальному світі.

Базовий рівень.

Клас "Хакер"

Опис завдання: Твоя мета — створити клас, який буде моделювати хакера. Хакер буде мати ім'я, рівень (наприклад, початківець, досвідчений, професіонал) та список знань (які можуть бути різними темами в кібербезпеці, як шифрування, атаки, мережеві технології тощо).

Завдання:

1. Створи клас Hacker.
2. В класі має бути:

   Атрибути: name (ім'я хакера), level (рівень хакера), knowledge (список знань хакера, наприклад, ["Шифрування", "Мережеві атаки"]).

   Метод learn(), який дозволяє хакеру вивчити нову тему (наприклад, додає нову тему до списку знань).
3. Напиши сценарій, де створюються 2-3 хакери з різними рівнями та знаннями. Потім моделюється процес їхнього навчання (додаються нові знання до списку).

Приклад:

Хакер на ім'я "Андрій" має рівень "початківець" та знання ["Шифрування"].

Після навчання він вивчає нові теми: ["Мережеві атаки", "Парольні системи"].

fizmat@sspu.edu.ua суворо дотримуйтесь правил академічної доброчесності

Генерація паролів

Опис завдання: Створи програму, яка генерує паролі для користувачів. Клас PasswordGenerator має містити методи для генерації паролів різної складності та перевірки, чи відповідає пароль вимогам безпеки.

Завдання:

1. Створи клас PasswordGenerator.
2. В класі має бути:

Метод generate_password(length), який генерує пароль заданої довжини. Пароль повинен бути випадковим і містити як букви, так і цифри.

Метод check_password(password), який перевіряє, чи відповідає пароль мінімальним вимогам безпеки (наприклад, довжина пароля повинна бути більше 8 символів, має бути хоча б одна велика літера та цифра).

3. Напиши сценарій, у якому генерується кілька паролів різної складності, і перевіряється їхня безпека.

Приклад:

Генерація паролю: generate_password(12) — генерує пароль "Ab1d7SgZp9vQ".

Перевірка паролю: check_password("Ab1d7SgZp9vQ") — повертає True, якщо пароль відповідає вимогам.

Злом коду (Сейф)

Опис завдання: Створи клас Safe, який буде містити секретний код. Метод повинен перевіряти, чи правильний введений код, і якщо він вірний, сейф відкриється.

Завдання:

1. Створи клас Safe, який має:

Атрибут secret_code — це секретний код для відкриття сейфа.

Метод check_code(input_code), який перевіряє, чи співпадає введений код з секретним.

Якщо код правильний, повертається повідомлення "Сейф відкрито!", в іншому випадку — "Невірний код".

fizmat@sspu.edu.ua
суворо дотримуйтесь
правил академічної
доброчесності

2. Напиши сценарій, у якому користувач пробує ввести код, і програма перевіряє його на правильність.

Приклад:

Секретний код сейфа: "1234".

Введення неправильного коду: "4321" — повертає "Невірний код".

Введення правильного коду: "1234" — повертає "Сейф відкрито!".

Шифрування повідомлень (Шифрувальник)

Опис завдання: Створи клас, який буде шифрувати і дешифрувати повідомлення за допомогою простого шифру (наприклад, шифр Цезаря).

Завдання:

1. Створи клас Encryptor, який містить:

Метод encrypt(message, shift), який шифрує повідомлення за допомогою шифру Цезаря. shift — це кількість зміщення букв в алфавіті (наприклад, для shift=3 буква "A" стає "D").

Метод decrypt(encrypted_message, shift), який розшифровує зашифроване повідомлення, використовуючи зворотній зсув.

2. Напиши сценарій, де повідомлення шифрується і дешифрується.

Приклад:

Шифрування: "HELLO" з shift=3 дає результат "KHOOR".

Дешифрування: "KHOOR" з shift=3 дає результат "HELLO".

Завершення завдання: Ці завдання дозволять засвоїти основи об'єктно-орієнтованого програмування, познайомлять з принципами безпеки та допоможуть розвинути навички створення реальних програмних рішень. Ти навчишся створювати класи, працювати з атрибутами та методами, а також застосовувати концепції кібербезпеки для вирішення різних задач.

Середній рівень.

Симуляція атаки на сервер

Завдання:

fizmat@sspu.edu.ua
суворо дотримуйтесь
правил академічної
доброчесності

1. Створити клас "Сервер". Цей клас має мати атрибути, які зберігають стан сервера, наприклад, список заблокованих IP-адрес, та методи для захисту:
    - block_ip(ip) — метод для блокування IP-адреси.
    - is_blocked(ip) — метод для перевірки, чи заблокована адреса.
2. Створити клас "Хакер". Клас хакера повинен намагатися вгадати пароль сервера. Якщо спроби невдалі, хакер повинен потрапити в "чорний список". Хакер має атрибути:
    - name — ім'я хакера.
    - attempts — кількість невдалих спроб.
    - is_blocked — чи заблоковано хакера.

Якщо хакер зробить більше 3 невдалих спроб входу, його IP заблокують.

Створення фаєрволу

Завдання:

1. Створити клас "Фаєрвол" для перевірки запитів на безпеку. Фаєрвол повинен мати два списки:
    - allowed_actions — дозволені дії.
    - blocked_actions — заборонені дії.
2. Клас має методи:
    - check_request(action) — перевірка дії користувача. Якщо дія знаходиться в списку заблокованих, запит відхиляється.
3. Створити кілька користувачів, які намагаються виконати різні дії (наприклад, відкрити порт, підключити пристрій, відправити повідомлення). Фаєрвол перевіряє, чи дозволено це виконувати.

Мережева атака

Завдання:

1. Створити клас "Мережа", який містить список пристроїв (наприклад, сервери, комп'ютери, маршрутизатори).
    - Кожен пристрій може бути атакований хакером.

fizmat@sspu.edu.ua
суворо дотримуйтесь
правил академічної
доброчесності

2. Створити клас "Хакер", який має атрибути:
    - name — ім'я хакера.
    - target_device — пристрій, на який хакер намагається здійснити атаку.
3. Додати методи:
    - attack_device() — метод, що спробує зламати пристрій.
    - defend_device() — метод для захисту пристроїв. Наприклад, пристрій може мати паролі або механізми автентифікації для протистояння атакам.

Придумайте різні рівні складності для атаки (наприклад, зламати простий пароль або обійти фаєрвол).

Квест хакера

Завдання:

1. Створити клас "Локація", що має атрибут "загадки", що представляють собою питання для хакера. Локація також повинна мати метод "add_riddle", щоб додавати нові загадки.
2. Створити клас "Хакер" із такими атрибутами:
    - name — ім'я хакера.
    - access_keys — список "ключів доступу", що хакер здобуває за правильні відповіді на загадки.
3. Клас "Хакер" повинен мати метод "solve_riddle", який дозволяє хакеру проходити через локації, розгадуючи загадки. За кожну правильну відповідь хакер отримує ключ, який дозволяє йому потрапити до наступної локації.

У кінці квесту хакер має пройти всі локації й отримати головний "ключ доступу".

Завершення завдання: Це завдання спрямоване на навчання дітей основам кібербезпеки та захисту інформаційних систем. У завданні, яке симулює атаку на сервер, учні дізнаються, як важливо обмежувати кількість невдалих спроб входу та блокувати шкідливі дії хакерів, що є основою

fizmat@sspu.edu.ua
суворо дотримуйтесь
правил академічної
доброчесності

захисту мереж. Завдяки цьому завданню діти також розуміють принципи захисту серверів та мереж від атак.

Інше завдання, яке стосується створення фаєрволу, дає дітям базові знання про правила доступу до мережі та захист від небажаних дій. Вони навчаються визначати, які дії є безпечними, а які можуть становити загрозу, і як створювати правила для блокування небажаних запитів.

Мережеві атаки є ще одним важливим аспектом навчання. У цьому завданні учні дізнаються про взаємодію пристроїв у мережі та про те, як хакери намагаються отримати доступ до різних пристроїв. Це завдання допомагає зрозуміти важливість захисту даних і пристроїв, а також формування стратегії захисту від атак.

Останнє завдання, яке стосується квесту хакера, розвиває логічне мислення та навички пошуку рішень, що є важливою частиною роботи хакера. Це завдання стимулює учнів до розв'язання складних задач та надає їм можливість потренувати свої здібності в умовах реальних кібератак.

Складний рівень.

Реалізація багаторівневої системи безпеки

Опис завдання:

1. Створити систему безпеки, яка включає кілька рівнів захисту, таких як пароль, біометрія і розпізнавання образів.
2. Реалізувати клас "Система", в якому кожен рівень захисту буде перевірятись по черзі.
3. Клас "Хакер" повинен мати інструменти для обходу кожного з рівнів.
4. Додати логіку, яка відстежує кількість спроб входу та час між спробами (щоб симулювати реалістичні атаки).

Деталі реалізації:

- Клас "Система":
    - Атрибути:
        - password (пароль)

fizmat@sspu.edu.ua
суворо дотримуйтесь
правил академічної
доброчесності

        - biometrics (біометричні дані)
        - face_recognition (дані для розпізнавання обличчя)
    - Методи:
        - check_password(): перевірка паролю.
        - check_biometrics(): перевірка біометричних даних.
        - check_face_recognition(): перевірка розпізнавання обличчя.
        - Логіка блокування доступу після декількох невдалих спроб.

- Клас "Хакер":
    - Атрибути:
        - tools (список інструментів для обходу захисту)
    - Методи:
        - bypass_password(): обходить пароль.
        - bypass_biometrics(): обходить біометричні дані.
        - bypass_face_recognition(): обходить розпізнавання обличчя.
        - Реалізація лічильника спроб і часу між ними.

Симуляція атаки на банк

Опис завдання:

1. Створити клас "Банк", який містить рахунки користувачів.
2. Реалізувати клас "Хакер", який генерує фальшиві транзакції для обкрадання рахунків.
3. Додати систему виявлення аномалій для виявлення фальшивих транзакцій.

Деталі реалізації:

- Клас "Банк":
    - Атрибути:
        - accounts (список рахунків користувачів)
    - Методи:

fizmat@sspu.edu.ua
суворо дотримуйтесь
правил академічної
доброчесності

        - create_account(): створення нового рахунку.
        - make_transaction(): виконання транзакції.
        - detect_anomalies(): виявлення аномальних транзакцій, наприклад, частих великих переказів або переказів на незнайомі рахунки.
- Клас "Хакер":
    - Атрибути:
        - target_account (цільовий рахунок для атак)
        - transaction_history (історія транзакцій)
    - Методи:
        - generate_fake_transaction(): генерує фальшиві транзакції для виведення коштів.
        - attempt_to_steal(): намагається скомпрометувати рахунок користувача.

Штучний інтелект для захисту системи

Опис завдання:

1. Створити клас "СистемаЗахисту", яка використовує штучний інтелект для виявлення аномальних дій користувачів.
2. Система повинна аналізувати поведінку користувача на основі історії дій.
3. Реалізувати клас "Хакер", який намагається обійти цей захист.

Деталі реалізації:

- Клас "СистемаЗахисту":
    - Атрибути:
        - normal_activity (нормальна активність користувача)
        - suspicious_activity (підозріла активність)
    - Методи:
        - train_model(): навчає систему на основі набору даних (нормальних і злочинних дій).
        - analyze_activity(): аналізує дії користувача.

fizmat@sspu.edu.ua
суворо дотримуйтесь
правил академічної
доброчесності

        - detect_intrusion(): виявляє вторгнення на основі відхилень від нормальної активності.

- Клас "Хакер":
    - Атрибути:
        - methods_to_bypass(): методи для обходу системи захисту.
    - Методи:
        - attempt_to_bypass(): намагається обійти систему захисту.
        - simulate_attack(): імітує атаку для виявлення слабких місць.

Мережева війна

Опис завдання:

1. Реалізувати класи "Хакер", "Сервер" і "Мережа".
2. Створити логіку, де кілька хакерів змагаються за доступ до мережі, а сервер намагається захистити свої дані.

Деталі реалізації:

- Клас "Мережа":
    - Атрибути:
        - devices (список пристроїв у мережі)
    - Методи:
        - add_device(): додає пристрій до мережі.
        - monitor_network(): моніторить мережу для виявлення підозрілих активностей.
- Клас "Сервер":
    - Атрибути:
        - network_status (стан мережі)
    - Методи:
        - protect_data(): захищає дані на сервері.
        - defend_network(): здійснює оборону мережі від атак.
- Клас "Хакер":
    - Атрибути:
        - target_device (цільовий пристрій для атаки)

fizmat@sspu.edu.ua
суворо дотримуйтесь
правил академічної
доброчесності

- attack_methods (список методів для атаки)

- Методи:
  - launch_attack(): запускає атаку на мережу.
  - bypass_security(): обходить безпеку сервера.

Завершення завдання: Ці завдання дозволять учням глибше зануритися в тематику кібербезпеки, забезпечуючи їм не лише теоретичні знання, а й практичні навички в реалістичних сценаріях. Вони отримають розуміння того, як працюють складні системи захисту, як здійснюються атаки та як можна захистити мережі та дані від зловмисників.

У процесі виконання завдань учні не тільки будуть писати код для створення захисних систем, а й зможуть симулювати різні види атак, що дозволяє їм краще зрозуміти, як працює кібербезпека зсередини. Вони навчаться реалізувати багаторівневу аутентифікацію, створювати системи для виявлення аномальних дій, використовувати штучний інтелект для аналізу поведінки користувачів і захисту від потенційних загроз.

Одним із важливих аспектів є навчання відслідковувати підозрілі транзакції в банківських системах, що допомагає зрозуміти, як виявляються спроби шахрайства. Це також дає учням змогу зрозуміти важливість використання систем виявлення вторгнень (IDS) та методів аномального виявлення на основі великих обсягів даних.

Під час розв'язання завдань, пов'язаних з атакою на мережу та сервіси, учні дізнаються про важливість захисту мережевих пристроїв, серверів і даних у сучасному цифровому світі. Вони будуть змушені розробляти стратегії захисту, що є ключовими для забезпечення стабільної і безпечної роботи будь-якої організації.

Крім того, навчання, яке учні отримають, може стати важливим підґрунтям для подальшої кар'єри в сфері кібербезпеки, де попит на професіоналів зростає. Навички виявлення та запобігання кіберзагроз, а також здатність створювати та підтримувати безпечні системи, є важливими

fizmat@sspu.edu.ua
суворо дотримуйтесь
правил академічної
доброчесності

для майбутнього розвитку будь-якої технологічної сфери, від фінансових установ до урядових організацій.

Ці практичні завдання також допоможуть учням розвинути критичне мислення, логічний підхід до вирішення складних задач і вміння працювати з реальними даними та технологіями. Це дозволяє їм набувати досвіду роботи з реальними інструментами безпеки та готувати їх до кар'єри в галузі інформаційних технологій.

fizmat@sspu.edu.ua
суворо дотримуйтесь
правил академічної
доброчесності

ВИСНОВКИ

У ході проведеного дослідження було здійснено глибоке вивчення методичних підходів до викладання об’єктно-орієнтованого програмування (ООП) з використанням мови програмування Python для учнів старших класів закладів загальної середньої освіти. Це дослідження стало відповіддю на зростаючу потребу у підготовці молоді до майбутньої кар’єри в ІТ-сфері, враховуючи стрімкий розвиток технологій та їх інтеграцію в усі сфери життя.

Результати дослідження підтвердили, що навчання ООП на базі Python сприяє розвитку ключових компетентностей 21-го століття, таких як критичне мислення, здатність до розв’язання складних задач, алгоритмічне мислення та вміння працювати над проєктами в команді. Вибір Python був зумовлений його простим синтаксисом, багатофункціональністю, підтримкою численних бібліотек і сучасних концепцій програмування. Це дозволяє учням зосередитися на концепціях ООП, а не на складних технічних деталях мови.

Основні досягнення дослідження:

1. Розробка сучасної методики викладання ООП. Методика базується на поєднанні теоретичних і практичних занять, інтерактивних завдань і проєктної роботи. Кожна ключова концепція ООП — інкапсуляція, наслідування, поліморфізм та абстракція — подається в контексті розв’язання практичних завдань. Учні отримують можливість створювати власні програми, поступово переходячи від простих прикладів до складних проєктів.

2. Використання проектної методики

Завдяки впровадженню проєктного навчання учні можуть працювати над реальними програмними продуктами, такими як ігри, веб-додатки або мобільні застосунки. Це дозволяє їм побачити практичну цінність отриманих знань і розвивати навички командної роботи.

3. Створення навчальних матеріалів

fizmat@sspu.edu.ua
суворо дотримуйтесь
правил академічної
доброчесності

У рамках дослідження було розроблено комплекс навчальних матеріалів, що включають теоретичні конспекти, приклади коду, покрокові інструкції з виконання завдань і проєктів. Особливу увагу приділено створенню практичних завдань, які стимулюють творче мислення учнів. Наприклад, завдання з програмування ігрових симуляторів дозволяють використовувати основні концепції ООП у цікавому й зрозумілому для учнів форматі.

Для перевірки ефективності методики було проведено порівняльний аналіз навчальних досягнень учнів, які вивчали ООП за традиційними програмами та за розробленою методикою. Учні, які працювали за новою методикою, продемонстрували глибше розуміння основних концепцій ООП, кращі результати в розв'язанні складних програмних завдань і вищу зацікавленість у вивченні програмування.

Дослідження є новаторським завдяки інтеграції сучасних технологічних та педагогічних підходів у навчальний процес. Розроблена методика враховує не лише академічні потреби учнів, але й вимоги сучасного IT-ринку. Вона забезпечує баланс між теоретичними знаннями та практичними навичками, що необхідні для створення програмних продуктів.

Запропонована методика може бути використана вчителями інформатики для організації уроків програмування, створення факультативних курсів, проведення позашкільних занять та онлайн-курсів. Вона є адаптованою для використання в освітніх установах різних рівнів – від шкіл до університетів. Крім того, матеріали дослідження можуть слугувати основою для розробки навчальних програм для IT-академій та центрів професійної підготовки.

Подальші дослідження можуть бути спрямовані на вдосконалення навчальних матеріалів, створення нових практичних завдань із використанням сучасних інструментів, таких як штучний інтелект і великі дані, а також інтеграцію нових технологій у навчальні процеси. Крім того, можливим напрямом подальших досліджень є розробка спеціалізованих

fizmat@sspu.edu.ua
суворо дотримуйтесь
правил академічної
доброчесності

курсів для різних вікових груп та підготовка викладачів з використанням інтерактивних освітніх платформ.

Таким чином, результати проведеного дослідження зробили значний внесок у вдосконалення системи навчання об’єктно-орієнтованого програмування в школі. Запропонована методика є ефективним інструментом для розвитку сучасної ІТ-освіти та підготовки кваліфікованих фахівців, готових до викликів цифрової економіки.

fizmat@sspu.edu.ua
суворо дотримуйтесь
правил академічної
доброчесності

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. https://ela.kpi.ua/server/api/core/bitstreams/672a2e48-69df-42c1-8c2c-e7eeb4ebea8c/content

2. Кобильник Т.П, Сікора О.В. Python як засіб навчання основ алгоритмізації у закладах загальної середньої освіти URL: https://journal.iitta.gov.ua/index.php/itlt/article/download/4896/2032

3. https://www.codeunderscored.com/python-vs-java-object-oriented-programming/

4. https://www.codewithc.com/c-vs-java-vs-python-an-in-depth-comparison-of-popular-languages/

5. https://www.geeksforgeeks.org/patterns-and-frameworks-in-ooad/

6. https://thenewstack.io/the-big-impact-of-smalltalk-a-50th-anniversary-retrospective/

7. https://realpython.com/solid-principles-python/

8. https://www.tensorflow.org/

9. https://github.com/features/copilot

10. https://docs.scala-lang.org/scala3/book/scala-features.html

11. https://www.mostlypython.com/oop-in-python-why-is-object-oriented/

12. https://kinsta.com/blog/python-object-oriented-programming/

13. https://ieeexplore.ieee.org/document/6332064

14. http://eprints.zu.edu.ua/35523/1/%D0%BC%D0%B0%D1%85%D0%B5%D0%BD%D1%8C%D0%BA%D0%BE.pdf

fizmat@sspu.edu.ua
суворо дотримуйтесь
правил академічної
доброчесності