МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЖИТОМИРСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ ІМЕНІ ІВАНА ФРАНКА
ФІЗИКО-МАТЕМАТИЧНИЙ ФАКУЛЬТЕТ
КАФЕДРА КОМП'ЮТЕРНИХ НАУК ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

Реєстраційний № _____
Дата здачі _____

МЕТОДИКА ВИВЧЕННЯ ОБ'ЄКТНО-ОРІЄНТОВАНОГО ПРОГРАМУВАННЯ В КУРСІ ІНФОРМАТИКИ З ВИКОРИСТАННЯМ WEB-ФРЕЙМВОРКУ DJANGO

Дипломна робота здобувача вищої освіти
другого (магістерського) рівня вищої освіти
спеціальності 014 Середня освіта
предметної спеціальності 014.09 Середня освіта (інформатика)
освітньої програми Інформатика в закладах освіти
Махенька Ярослава Дмитровича

Науковий керівник:
доцент кафедри комп'ютерних наук та інформаційних технологій,
кандидат педагогічних наук,
Кривонос Олександр Миколайович

Рекомендовано до захисту
рішенням кафедри комп'ютерних наук та інформаційних технологій
Протокол №___ від «__» _____ 2022 р.

Зав. кафедри _____ _____
(підпис) *(Власне ім'я ПРІЗВИЩЕ)*

Житомир – 2022

Дата захисту _____

Результат захисту	Оцінка		
	за університетською шкалою	за 100 бальною шкалою	ЄКТС

Голова ЕК _____ _____
(підпис) (Власне ім’я ПРІЗВИЩЕ)

Члени ЕК _____ _____
(підпис) (Власне ім’я ПРІЗВИЩЕ)

_____ _____
(підпис) (Власне ім’я ПРІЗВИЩЕ)

_____ _____
(підпис) (Власне ім’я ПРІЗВИЩЕ)

Секретар ЕК _____ _____
(підпис) (Власне ім’я ПРІЗВИЩЕ)

ЗМІСТ

ВСТУП

Актуальність. В зв’язку з стрімким розвитком новітніх технологій, в програмуванні все частіше можна зустріти саме об’єктно-орієнтовний підхід до програмування. У зв’язку з цими тенденціями в шкільному курсі інформатики 10-11 класу починають вивчати таку тему, як «Парадигми та технології програмування», після вивчення якого учні мають вміти проектувати об’єктно-орієнтовану архітектуру програмних рішень, створювати об’єктно-орієнтовані програмні рішення, тощо. Оскільки даний курс тільки починає впроваджуватись у шкільну програму, то зустрічається він лише в програмах профільних класів, де інформатика вивчається з початкової школи. Хоч і зрозуміло, що навчання, яке містить в собі вивчення даної теми, буде якіснішим, вчителі намагаються оминати даний вибірковий блок через проблематику підбору методів щодо вивчення об’єктно-орієнтованого програмування та складності подачі матеріалу.

Вчителю важливо знати: основи об’єктно-орієнтованого програмування, методи його навчання та викладання, основні проблеми, з якими стикаються учні при вивченні об’єктно-орієнтованого програмування, способи подачі матеріалу та знати за допомогою яких середовищ програмування можна викладати ООП у школі. Незважаючи на всі труднощі, з якими стикається вчитель, курс має переваги такі, як: учні розвивають об’єктно-орієнтоване мислення та абстрактне уявлення, у учнів з’являється усвідомлення майбутньої професії, адже багато випускників обирають ІТ спеціальності в якості фаху, починає активніше працювати критичне мислення та пристосування до різних умов.

Стає зрозумілим, що тема «Парадигми та технології програмування» має вивчатись і в класах рівня стандарту, а найактуальніша парадигма – об’єктно-орієнтоване програмування.

Мета – дослідити вивчення об’єктно-орієнтованого програмування в школі та розробити онлайн ресурс для вивчення об’єктно-орієнтованого програмування засобами мови програмування Python, фреймворк Django.

Для досягнення даної мети потрібно поставити такі **завдання**:

1. Дослідити теоретичні засади вивчення об’єктно-орієнтовано програмування.
2. Проаналізувати середовища програмування, за допомогою яких можна вивчати ООП.
3. Дослідити основні поняття мови програмування Python та фреймворку Django.
4. Створити онлайн ресурс для вивчення ООП засобами мови програмування Python, фреймворк Django.

Об’єкт дослідження – методика вивчення об’єктно -орієнтованого програмування.

Предмет дослідження – елементи шкільного курсу інформатики що стосуються об’єктно орієнтованого програмування.

Методи дослідження:

- порівняльний,
- експериментальний,
- структурний,
- описовий.

Апробація роботи: участь у IV Всеукраїнському відкритому науково-практичному онлайн-форумі «ІННОВАЦІЙНІ ТРАНСФОРМАЦІЇ В СУЧАСНІЙ ОСВІТІ: ВИКЛИКИ, РЕАЛІЇ, СТРАТЕГІЇ» (27 жовтня 2022 р.), участь у VII Міжнародній науково-практичній студентській конференції «Професійна іншомовна підготовка в умовах глобальних комунікативних потреб» (3 листопада 2022р.)., участь у VII Всеукраїнській науково-практичній конференції з міжнародною участю “Сучасні інформаційні технології в освіті та науці” (17-18 листопада 2022 р.).

Публікація: результати дослідження висвітлено у 3 матеріалах конференцій.

Структура і обсяг роботи: робота складається зі вступу, трьох розділів, висновків до розділів, висновків, списку використаних джерел та літератури. Загальний обсяг – 47 сторінок, основна частина розміщена на сторінках. Загальна кількість використаних джерел – 17. Кількість рисунків – 36.

РОЗДІЛ 1. ТЕОРЕТИЧНІ АСПЕКТИ ВИВЧЕННЯ ПРОГРАМУВАННЯ В КУРСІ ЗАГАЛЬНООСВІТНЬОЇ ШКОЛИ

1.1 Будова шкільного курсу інформатики 10-11 класів

Шкільний курс інформатики в старшій школі побудований з метою продовження формування у учнів інформаційної культури і грамотності. Оскільки інформатика в 10-11 класах є продовженням шкільного курсу, який вивчався раніше, учні, у процесі навчання, мають розширити знання з деяких тем, що вивчались в молодшій та середній школах, а також засвоїти нові знання, що допоможуть у формуванні інформаційної культури та базової компетентності у інформаційно-комунікаційних технологіях.

Проте, оскільки шкільний курс може формуватись по різному (учні починають вчити інформатику в 2, або в 5 класі), у старшій школі з'являється розгалуження у рівнях навчання: у профільному рівні діти поглиблено вивчають деякі теми, та захоплюють більше тем, у рівні стандарту, відповідно, теми вивчаються на базовому рівні.

Рівень стандарту, за навчальним планом, розрахований на 105 годин, при цьому 35 з них складає модуль, що є обов'язковим для вивчення, а решта годин займає вибірковий блок з переліком тем, які учні можуть обрати за власними вподобаннями.

Аналізуючи навчальну програму, можна виділити основні завдання (рис.1):

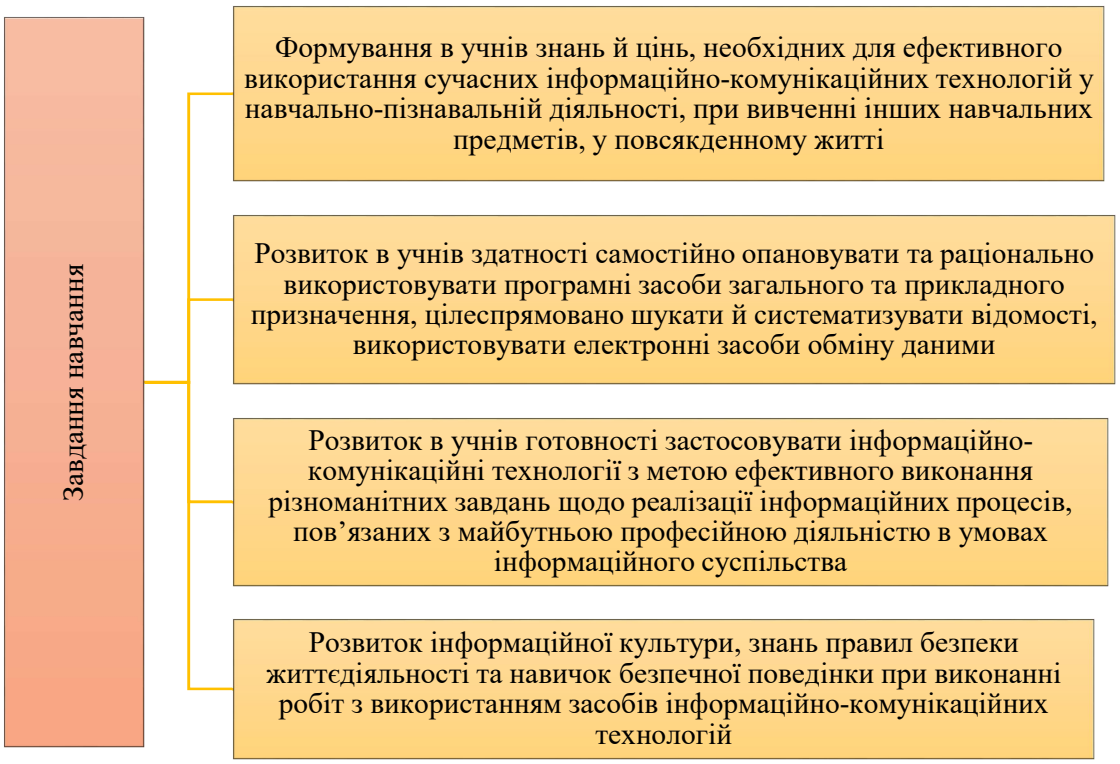

Рисунок 1.1 Завдання навчання інформатики в старшій школі

Програма для старшої школи єдина на 10 і 11 клас, вона складається з базового та вибіркового модулів. Основний модуль включає кілька обов'язкових тем, що розраховані на 35 годин, завдяки вивченню яких у учнів формуються предметні та ключові компетентності використання ІКТ на рівні визначеному чинним Державним стандартом базової і повної загальної середньої освіти[1]. Модуль включає в собі такі загальні теми, як:

- «Інформаційні технології в суспільстві» - передбачає розгляд тем пов'язаних з інформацією, інформаційними технологіями, безпеки в інтернеті, інтернет-маркетинг та інтернет-банкінг, штучний інтелект тощо.
- «Моделі і моделювання. Аналіз та візуалізація даних» - включає вивчення комп'ютерного моделювання, інфографіки, розв'язання задач та рівнянь тощо.
- «Системи керування тазами даних» - вивчається поняття бази даних та СБД, її функції та можливості.

- «Мультимедійні та гіпертекстові документи» - тема пов’язана з технологіями опрацювання мультимедійних даних, поняттям гіпертекстового документа та можливостями роботи з ним.

В свою чергу вибірковий модуль учитель добирає в залежності від багатьох факторів (профіль навчального закладу, запити, індивідуальні інтереси і здібності учнів, регіональні особливості, матеріально-технічної бази школи та можливість використовувати те, чи інше програмне забезпечення).

В складі вибіркового модуля є 10 тем (рис. 1.2), що розраховані на різну кількість годин, тому, при складанні плану, вчитель має прорахувати аби в сумі з обов’язковим блоком отримати 105 годин.

Графічний дизайн (35 годин)	Графічний дизайн як засіб візуальної комунікації
	Растрова графіка
	Основи композиції та дизайну
	Векторна графіка
	Графічний дизайн у поліграфії
Комп'ютерна анімація (35 годин)	Основи анімації
	Анімація в редакторі растрової графіки
	Векторна анімація
Тривимірне моделювання (35 годин)	Тривимірна графіка
	Створення простих тривимірних об'єктів
	Створення та редагування тривимірних об'ктів неправильної форми
	Матеріали і текстури
	Тривимірна анімація
	Візуалізація та рендеринг

Рисунок 1.2 Теми вибіркового модуля (частина 1)

Математичні основи інформатики (35 годин	Системи числення
	Подання даних у комп'ютері
	Математична логіка
	Основи теорії інформації
Інформаційна безпека (17 годин)	Основи безпеки інформаційних технологій
	Забезпечення безпеки інформаційних технологій
	Забезпечення безпеки комп'ютерних систем і мереж
Веб-технології (35 годин)	Напрямки та інструменти веб-дизайну
	Проектування та верстка веб-сторінок
	Графіка та мультимедіа для веб-середовища
	Веб-програмування
	Основи дизайну та просування веб-сайту
Основи електронного документообігу (17 годин)	Документи та документообіг
	Технічні та програмні засоби обробки документів та інформації
	Електронний документообіг
Бази даних 35 годин	Проектування моделі бази даних
	Створення реляційної бази даних

Рисунок 1.3 теми вибіркового блоку (частина 2)

Комп'ютерні технології опрацювання звукової інформації (35 годин)	Звукова інформація в комп'ютерних системах
	Програмне забезпечення для роботи зі звуком
	Способи та засоби обробки звукової інформації
	Публікація звукової інформації
Креативне програмування (35 годин)	Цифрове мистецтво та творчість
	Графічні побудови та взаємодії
	Функції
	Об'єкти та класи
	Мультимедіа
	Інтерфейс програмного продукту

Рисунок 1.4 Теми вибіркового блоку (частина 3)

Профільний рівень. Порівняно з програмою рівня стандарту, у профільних класах за мету ставиться не лише розвинути загальне розуміння ІКТ та сформувати комп’ютерну грамотність. За мету ставиться навчити більш розширеним процесам :

- розвинути логічне, аналітичне мислення, а також основні види розумової діяльності, вміння використовувати індукцію, дедукцію, аналіз, синтез, робити висновки та узагальнення;
- розвинути теоретичну базу, що відноситься до процесів перетворення, передавання та використання інформації;
- розвити вміння розв’язувати задачі різної складності за допомогою ІКТ;
- підготувати учнів до олімпіад, конкурсів, науково-дослідницьких робіт тощо;
- вивчення інформатики на творчому рівні;

- створити зв’язки між інформатикою та іншими предметами, що є у учнів.[2]

Вчителям, які навчають дітей по програмі профільного рівня рекомендується використовувати різноманітні форми організації навчання для більш ефективного навчання, що дозволить мотивувати учнів та зацікавлювати у навчальному процесі.

Основною відмінністю від стандартного рівня є те, що в профільному вчитель самостійно визначає кількість навчальних годин на вивчення тем, а також їх порядок. Це є великою перевагою, адже завдяки цьому можна орієнтуватись на профіль закладу освіти, підлаштовуватись під учнів, проте зміст робочої програми змінювати не можна (рис. 1.5).

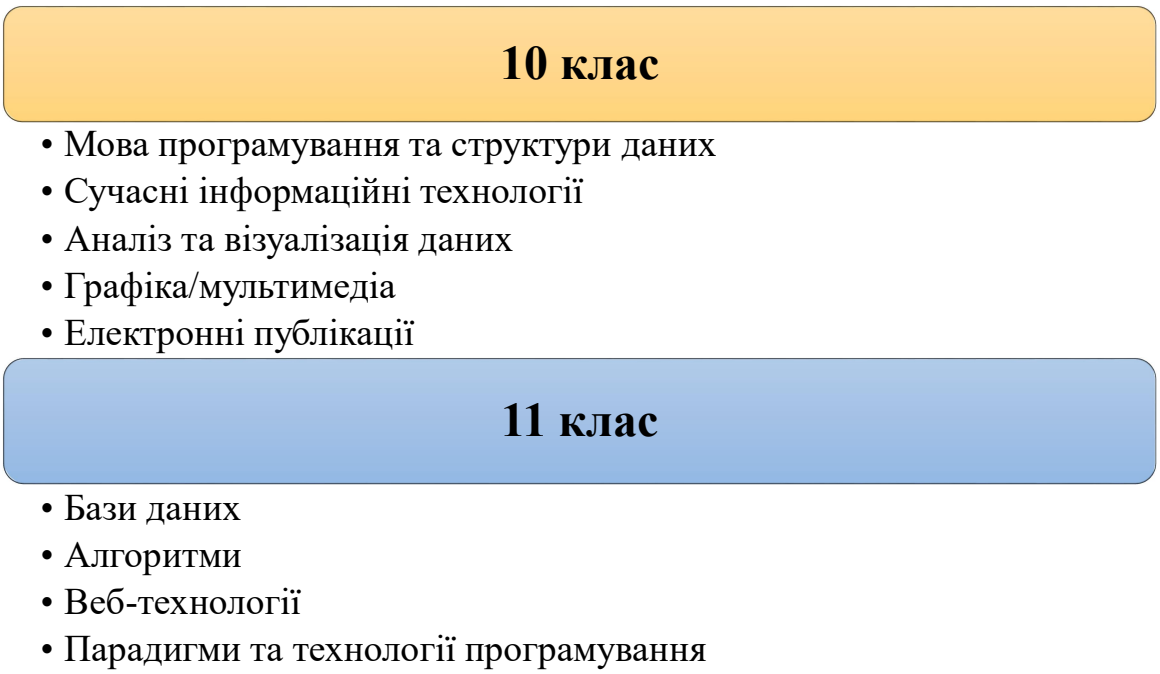

Рисунок 1.5 Перелік тем основних розділів для профільного рівня

Після аналізу рисунків стає зрозуміло, що вибіркові блоки більше пов’язані з графічною частиною інформатики, або з даними чи документами, а блоку, який буде пов’язаний з програмуванням практично немає, тому можна зробити висновок, що учні, що навчались за рівнем стандарту у 10-11 класах, після закінчення школи не можуть знати якусь мову програмування або розв’язувати складні завдання, які ставлять перед собою починаючі розробники/програмісти.[3]

1.2 ООП та його вивчення в шкільному курсі інформатики

В шкільному курсі інформатики основи об’єктно-орієнтованого програмування вивчаються в курсі «Парадигми та технології програмування».

Термін «об’єктно-орієнтоване програмування використовується для визначення багатьох речей, ключовий термін тут "об’єкт". Об’єкти – це сутності, які поєднуються властивості процедур і даних, оскільки вони виконують обчислення та збереження локального стану. Рівномірне використання предметів контрастує з використанням окремих процедур і даних у звичайне програмування. Об'єкт складається з двох елементів: даних і функцій (які називаються методами), що працюють з даними. Наприклад, об'єкти – це рядки, а дані – літери, з яких складається рядок. Це стосується не лише рядків, а й цілих чисел, десяткових чисел та навіть функцій.[4]

Об’єкти – це екземпляри класів, які можуть бути пов’язані один з одним. це пояснюється в книзі «Python 3 Object Oriented Programing» (автор D. Phillips) [5] на простому прикладі про апельсини та яблука: і яблука, і апельсини – об’єкти, але прийнято, що їх порівнювати не можна. Якщо уявити, що створюється програма інвентаризації для фруктової ферми і, для прикладу, уявити що яблука зберігаються в бочках, а апельсини – в кошиках, тобто в нас з’являється вже чотири об’єкта: яблука, апельсини, кошики та бочки. В об’єктно-орієнтованому моделюванні типи об’єктів називають класом. Отже, з технічної точки зору ми маємо чотири класи об’єктів.

Класи описують об’єкти, вони схожі в моделюванні для створення об’єкта. Взаємозв’язок цих чотирьох класів можна описати за допомогою діаграми класів уніфікованої мови моделювання UML (рис. 1.6):

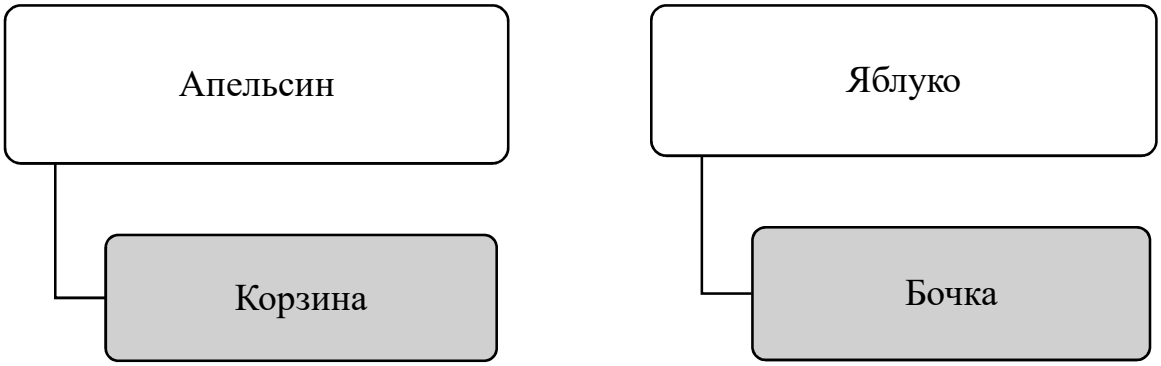

Рисунок 1.6 Взаємозв'язок класів

Ця діаграма на простому прикладі показує, що апельсин асоціюється з кошиком і що яблуко – з бочкою. Асоціація – найпростіший спосіб зв’язати два класи. На початковій діаграмі випливає асоціація що одне яблуко йду в одну бочку, але потрібно зазначити, що це не вірно, адже в одну бочку йде багато яблук, тому покращена схема матиме такий вигляд (рис. 1.7):

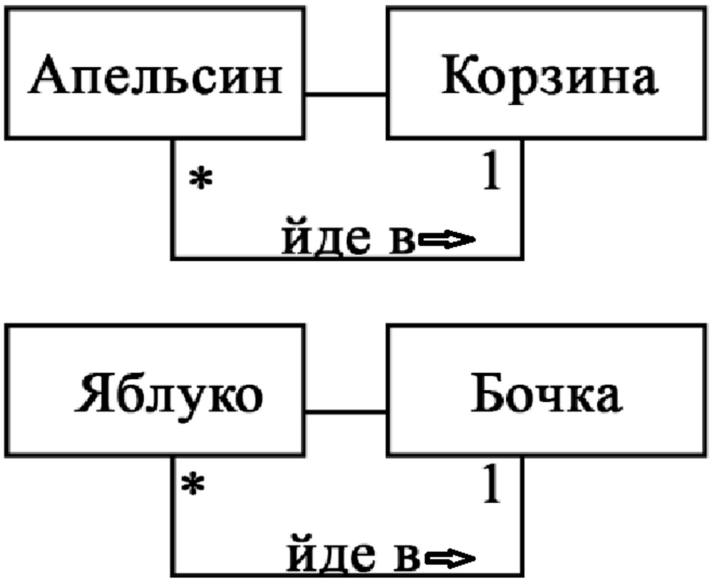

Рисунок 1.7 Покращена схема

На цій діаграмі показано, що апельсини йдуть у кошики, а маленька стрілка показує, що куди йде. Він також повідомляє нам про множинність (кількість цього об’єкта, який можна використовувати в асоціації) з обох

сторін відносини. Один кошик може вмістити багато (позначених *) апельсин. Будь-який апельсин може помістится рівно в один кошик.

Можна легко сплутати, на якій стороні стосунків йде множинність. Множинність - це кількість об'єктів цього класу, які можуть бути пов'язані з будь-яким об'єктом на іншому кінці асоціації. Оскільки яблуко входить у асоціацію бочок, читаючи зліва направо, багато екземплярів класу яблуко (тобто багато яблук) можуть входити до будь-якої бочки. Якщо читати справа наліво, з будь-яким яблуком можна пов’язати рівно одну бочку.

Клас об’єктів може визначати конкретні характеристики (у апельсинів є вага), тобто даний клас має певні атрибути (їх часто називають властивостями), вони не повинні бути унікальними.

Об'єктно-орієнтований підхід до програмування базується на маніпулюванні об'єктами. Це означає, що розвиток логіки програми досягається шляхом визначення класів різних об'єктів та використовуючи взаємодію об'єктів.

Щодо об’єктно-орієнтованої парадигми, то вона має такі принципи:

- дані структуруються в вигляді об’єктів, кожен з яких має певний тип (належить до якогось класу);
- класи – це результат формалізації задачі, що вирішується та виділення головних її аспектів;
- всередині об’єкта інкапсулюється логіка роботи з відносною до неї інформацією;
- об’єкти в програмі взаємодіють один з одним, обмінюються запитами та відповідями;
- об’єкти одного типу відповідають на одні і ті ж запити;
- об’єкти можуть організуватись в більш складні структури, наприклад, можуть мати інші об’єкти чи наслідувати їх.

Перші спроби викладання об’єктно-орієнтованого програмування в школі довели, що учні, які до цього вивчали програмування хоча б на базову

рівні гірше засвоюють ООП, через зовсім інший процес мислення, ніж ті, які спочатку зрозуміли основи ООП, а потім вже почали розвивати знання з програмування. Це явище назвали «зміною парадигми» і почало зустрічатись в джерелах все частіше.

Тож стає зрозумілим, що при введені в шкільну програму такої теми як «Парадигми програмування» в шкільний курс інформатики, потрібно завчасно вводити в курс інформатики вивчення об’єктно-орієнтованого програмування до вивчення основ програмування. Цього твердження притримуються певні вчителі та науковці. Так, вчені, що досліджували це поняття, у ході експериментів довели, що « взаємодія з об’єктами з самого початку допомогла учням побудувати конкретне розуміння та забезпечила відповідні концептуальні моделі. Ті, хто вивчає об’єктно-орієнтоване програмування, можуть легко зосередитися на поняттях зв’язків класів і об’єктів замість того, щоб зосереджуватися на фактах структурного програмування» [6].

Проте існує і зворотне твердження, що послідовність вивчення немає ніякого значення. Так у результаті експериментальних досліджень двох вчених було оголошено: «Головний результат дослідження говорить, що немає різниці між тим, вивчати спочатку об’єктно-орієнтоване програмування, а потім основи програмування (OOP-first), чи навпаки – спочатку основи програмування, а потім ООП (OOP-later) – це не впливає на результат навчання».[7]

Варто також згадати експериментальні дослідження що показують відмінності в розумінні програмних текстів об’єктно-орієнтованого стилю. В результаті даного експерименту було виявлено різкий контраст між ментальними репрезентаціями імперативних і об'єктно-орієнтованих програм. У той час як розуміння імперативних програм було в цілому кращим, ніж у об’єктно-орієнтованих програм, уявлення імперативних програм були зосереджені на знаннях програмного рівня. З іншого боку,

ментальні уявлення об’єктно-орієнтованих програм більше зосереджені на знаннях рівня домену.[8]

В результаті, стає очевидним що об'єктно-орієнтоване програмування точно повинне вивчатись в обов’язковому модулі програми, при цьому немає значення чи буде передувати йому вивчення основ програмування, адже, аналізуючи проведенні дослідження на дану тему, стає зрозумілим, що об’єктно-орієнтоване програмування розширює можливості мислення при будь-яких умовах його вивчення.

В темі «Парадигми та технології програмування» закладено, що учні, після вивчення даного курсу вміли:

- укладати документацію вимог проекту за результатами дослідження потреб предметної галузі;
- будувати модель задачі або проекту за допомогою візуальних засобів моделювання;
- проектувати об’єктну-орієнтовану архітектуру програмних рішень на основі моделей даних та процесів;
- проектувати інтерфейс користувача програмного продукту;
- створювати об’єктно-орієнтовані програмні рішення;
- описувати апаратне забезпечення для реалізації програмного проекту;
- реалізувати прототипи проектів на основі розробленої архітектури;
- користуватись системами контролю версій у процесі розробки програмного забезпечення.

Всі ці пункти наявні в програмі профільного рівня як діяльнісна складова теми. Але є також і знаннєва, до якої входять: «знати основні етапи та методології розробки програмного забезпечення, а також програмні інструменти підтримки цієї діяльності» та «пояснювати основні принципи

побудови моделі задачі».

У темі реалізуються такі підтеми (рис 1.7):

Рисунок 1.8 Підтеми, що розглядають у темі "Парадигми та технології програмування"

1.3 Середовища програмування які доцільно використовувати при вивчення ООП в школі

Серед мов, які підтримують реалізацію об’єктно-орієнтованого програмування і є повністю об’єктно-орієнтованими можна виділити: Java,

C#, C++, Python, Ruby і Objective-C, ActionScript 3, Swift, Vala.[9]

Тому, виходячи з того, яку мову програмування вивчають або вивчатимуть учні, потрібно обирати певне середовище програмування.

Мова Java. Java – одна з простих мов програмування, яка в більшості використовується для розробки корпоративних, мобільних та веб-додатків. [10] Для програмування на даній мові потрібно використовувати інтегроване середовище розробки (IDE), їх є декілька найпопулярніших:

- Eclipse – це безкоштовна IDE для розробників JAVA, в основі написання якої є також JAVA.
- IntelliJ – також безкоштовне для спільноти середовище програмування Java IDE, яка використовується для розробки додатків Android, програмування Java, Java SE, Groovy и Scala. Проте, аби мати розширені функції – потрібно отримати ліцензію.
- NetBeans – офіційне середовище програмування для Java 8, яка може створювати додатки з набору програмних компонентів – модулів.

Мова C#. Данна мова одна з популярних мов програмування, що замислювалася як альтернатива мові JAVA, але була реалізована як самостійна мова програмування. C# викладають в більшості технічних вузів світу. Найпопулярнішим середовищем програмування для мови C# є Visual Studio. Оскільки і мова програмування і дане середовище створили у компанії Microsoft, доцільно використовувати саме Visual Studio у вивченні мови програмування C#.[11]

Мова програмування C++. Оскільки обрана мова програмування має дуже багато середовищ програмування, варто перечислити лише популярні та практичні, такими є:

- Microsoft Visual Studio – середовище розробки від Microsoft, яке в більшості використовується для написання додатків.
- Codeblocks – відкрите середовище IDE для C і C++. Головна перевага - кросплатформеність.

- Dev-C++ – безкоштовне інтегроване середовище програмування з відкритим кодом, написана в Delphi для Windows. Оскільки це проста IDE, то для встановлення програми потрібно лише кілька хвилин. [12]

Мова програмування Python. Дана мова програмування дуже популярна серед розробників завдяки тому, що за допомогою Python можна швидше та ефективніше інтегрувати системи, при роботі з нею відразу можна побачити збільшення продуктивності роботи, при цьому зменшуються затрати на обслуговування.[13]

Популярні середовища програмування які використовуються для Python:

- PyCharm – найпопулярніша з IDE, які використовується як новачками, так і професійними розробниками. Це середовище адаптована під Python, тому її не потрібно довго налаштовувати, основною перевагою є доступність, адже PyCharm хоч і має платну версію для компаній, проте є безкоштовна версія для тих, хто вчиться чи використовує для особистого користування не в компанії.
- Spyder – це безкоштовна IDE з відкритим кодом для Python з важливим нюансом: вона розрахована на розробку у сфері Data Science. У ній є багато інструментів та оптимізації для роботи з даними, але для інших завдань вона підходить не так добре.[14]
- IDLE – середовище програмування, яке по замовчуванню встановлюється разом з Python. Воно написане цією ж мовою, має всі можливості інтегрованого середовища, проте може використовуватись тільки на початку вивчення мови, адже не має додаткових функцій.[15]

1.4 Висновки до 1 розділу

Проаналізувавши інформацію, що подана в 1 розділі дипломної роботи стає зрозумілим, що об’єктно-орієнтоване програмування розвиває такі якості як абстрактне мислення, аналітичне мислення та вміння впоратись з задачею будь-якого рівня складності. Завдяки проаналізованим дослідженням

можна зробити висновок, що не має значення у послідовності вивчення ООП з основами програмування, а важливо лише присутність розгляду обох тем при вивченні програмування.

У процесі аналізу середовищ програмування, які доцільно використовувати при вивченні ООП, зроблено висновки, що в залежності від мови програмування, яка була обрана для вивчення об’єктно-орієнтованого програмування вчителем, або учнями, потрібно обирати зручне середовище програмування, опираючись на такі фактори, як: мова програмування, можливості освітнього закладу придбати ліцензію, функціонал IDE та зручність у використанні як для новачків, так і для професійних програмістів.

РОЗДІЛ 2. МОВА ПРОГРАМУВАНЯ PYTHON ТА ФРЕЙВОРК DJANGO

2.1 Загальні відомості про мову. Об’єкти та класи в Python.

Оскільки для вивчення об’єктно-орієнтованого програмування можна обрати будь-яку з мов, було обрано мову програмування Python, як засіб вивчення ООП. Адже Python це популярна мова програмування, за допомогою неї багато розробників створюють навчальні програми, створюють додатки та веб-додатки (в тому числі і освітні), вирішують різного рівня складності задачі, а також створюють ігри.

Загалом Python – легка в навчанні та потужна мова програмування. Вона має ефективні високо рівневі структури даних і простий підхід до об’єктно-орієнтованого програмування, а її елегантний синтаксис, динамічна типізація та інтерпретований характер роблять її ідеальною мовою для написання сценаріїв та швидкої розробки додатків в багатьох сферах і на більшості платформ.

Інтерпретатор Python і велика стандартна бібліотека є у вільному доступі для всіх основних платформ на веб-сайті Python (https://www.python.org/) [15] та можуть вільно розповсюджуватись. На цьому ж сайті є дистрибутиви та вказівники на інші безкоштовні модулі Python, програми та інструменти, додаткову документацію.

Причинами, чому обирають саме Python є його **переваги** (рис. 1.8):

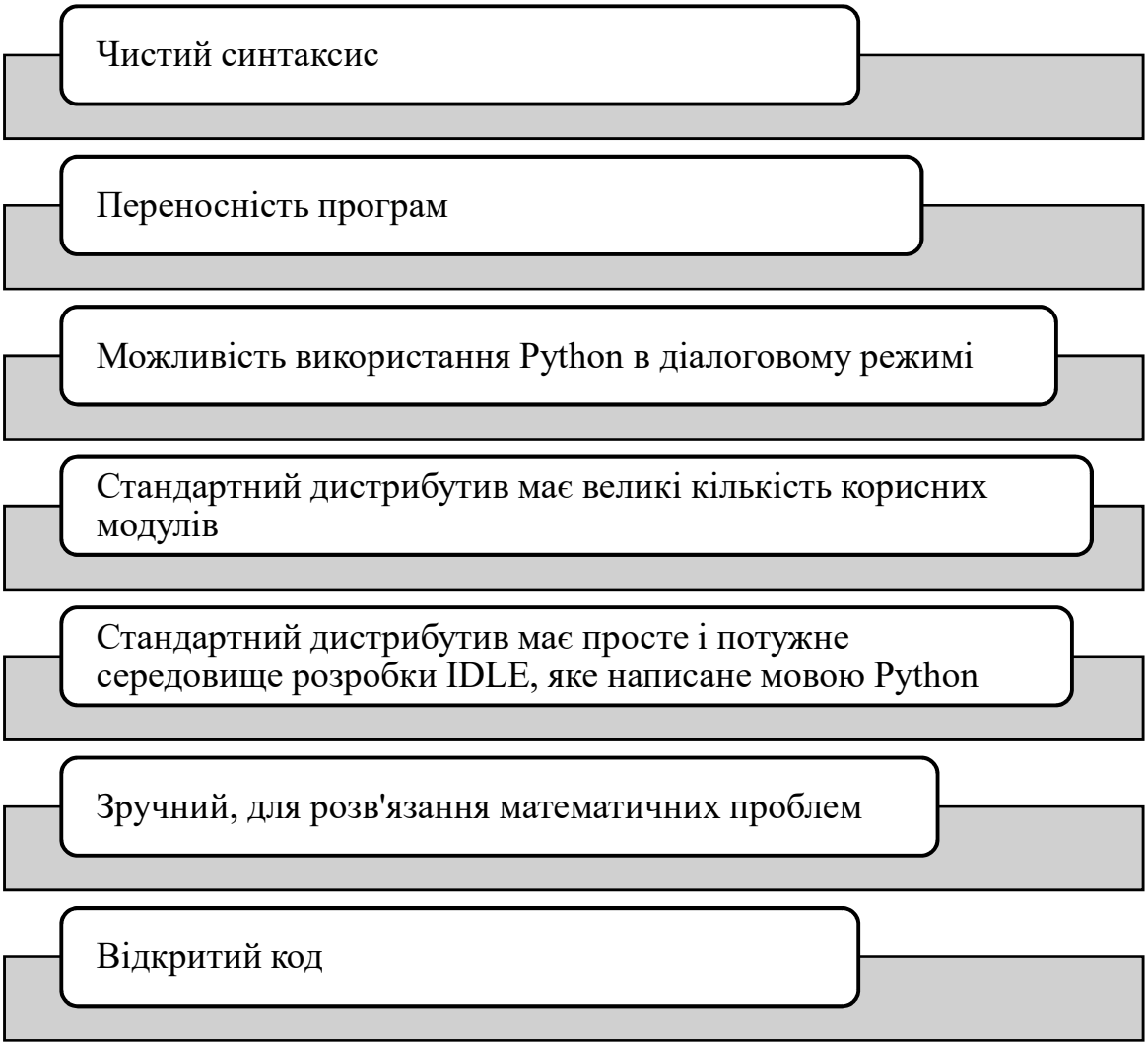

Рисунок 2.1 Переваги Python

Оскільки Python є максимально простою та зручною мовою програмування, то програма, яка пишеться найпершою на будь-якій мові програмування «Hello, world!» займає лише один рядок, тому початок вивчення об’єктно-орієнтованого програмування не є важким (рис. 2.2):

```
class MyFirstClass:
    pass
```

Рисунок 2.2 Створення класу на мові Python

На рисунку видно, що для створення класу потрібно написати команду class, потім сказати назву класу, а за рядком визначення класу слідує вміст класу. Як і в інших конструкціях Python, для розмежування класів

використовується відступ, а не дужки (як в багатьох інших мовах). В даному випадку замість вмісту класу стоїть ключове слово pass, щоб вказати що жодних дій виконувати з даним класом не потрібно.

Об’єкт – екземпляр класу. Оголошений клас – опис об’єкта, тобто йому не виділяється пам’ять.

Наприклад, аби клас та його об’єкти, потрібно створити конкретний клас (в нашому випадку «student»), його об’єктами можуть бути: ім’я, вік, стать тощо (рис. 2.3).

```
1   class Student(object):
2       specias = "Студент"
3
4       def __init__(self, name, age):
5           self.name = name
6           self.age = age
7
8
9   Katya = Student("Катерина", 15)
10  Toma = Student("Тамара", 16)
11
12  print(f'{Katya.name} - {Katya.__class__.specias}')
13  print(f'{Toma.name} також {Toma.__class__.specias}')
14
15  print(f'{Katya.name} - {Katya.age}-річна учениця')
16  print(f'{Toma.name} - {Toma.age}-річна учениця')
17
```

Рисунок 2.3 Елементарний клас у мові Python

Як видно з рисунка, було створено клас «Student» , після чого оголошено атрибути (характеристика об’єкта) такі, як ім’я та вік. Вони оголошені всередині класу за допомогою методу `__init__` (метод – ініціалізатор, який запускається відразу після створення об’єкта. Після цього створено екземпляри класу `Student`: `Katya` та `Toma`, вони є посиланнями на нові об’єкти. А в рядках 12-16 наведено приклад отримання доступу до

атрибутів класу та атрибутів екземпляру та виведення їх елементів за допомогою форматованого рядка (`f'{Katya.name} - {Katya.__class__.specias}'`) , де `Katya.name` – це отримання доступу до атрибуту класу, а `__class__.specias` – отримання доступу до атрибутів екземпляру. Після виконання програми отримано такий результат (рис.2.4):

```
Катерина - Студент
Тамара також Студент
Катерина - 15-річна учениця
Тамара - 16-річна учениця
```

Рисунок 2.4 Результат виконання програми

Функціями, оголошеними всередині тіла класу, що визначають поведінку об’єкта називають *методами* (рис. 2.5).

```
class Student(object):
    specias = "Студент"

    def __init__(self, name, age):
        self.name = name
        self.age = age

    def draw(self, picture):
        return f"{self.name} малює {picture}"

    def dance(self):
        return f"{self.name} танцює"

Katya = Student("Катерина", 15)
Toma = Student("Тамара", 16)
print(Katya.dance())
print(Toma.draw("ілюстрацію"))
```

Рисунок 2.5 Програма, що містить методи

На поданому рисунку створені методи `dance` і `draw`, які є методами екземпляру, адже вони викликаються об’єктами (`Katya`). Результати виконання даної програми: (рис. 2.6).

```
Катерина танцює
Тамара малює ілюстрацію
```

Рисунок 2.6 Результат виконання програми

Спосіб створення класу називається **наслідуванням** (рис.2.7). Це означає, що функціональність нового класу успадковується від вже існуючого класу. Створюваний, або новий, клас називається похідним (дочірнім), а існуючий – базовим (батьківським).

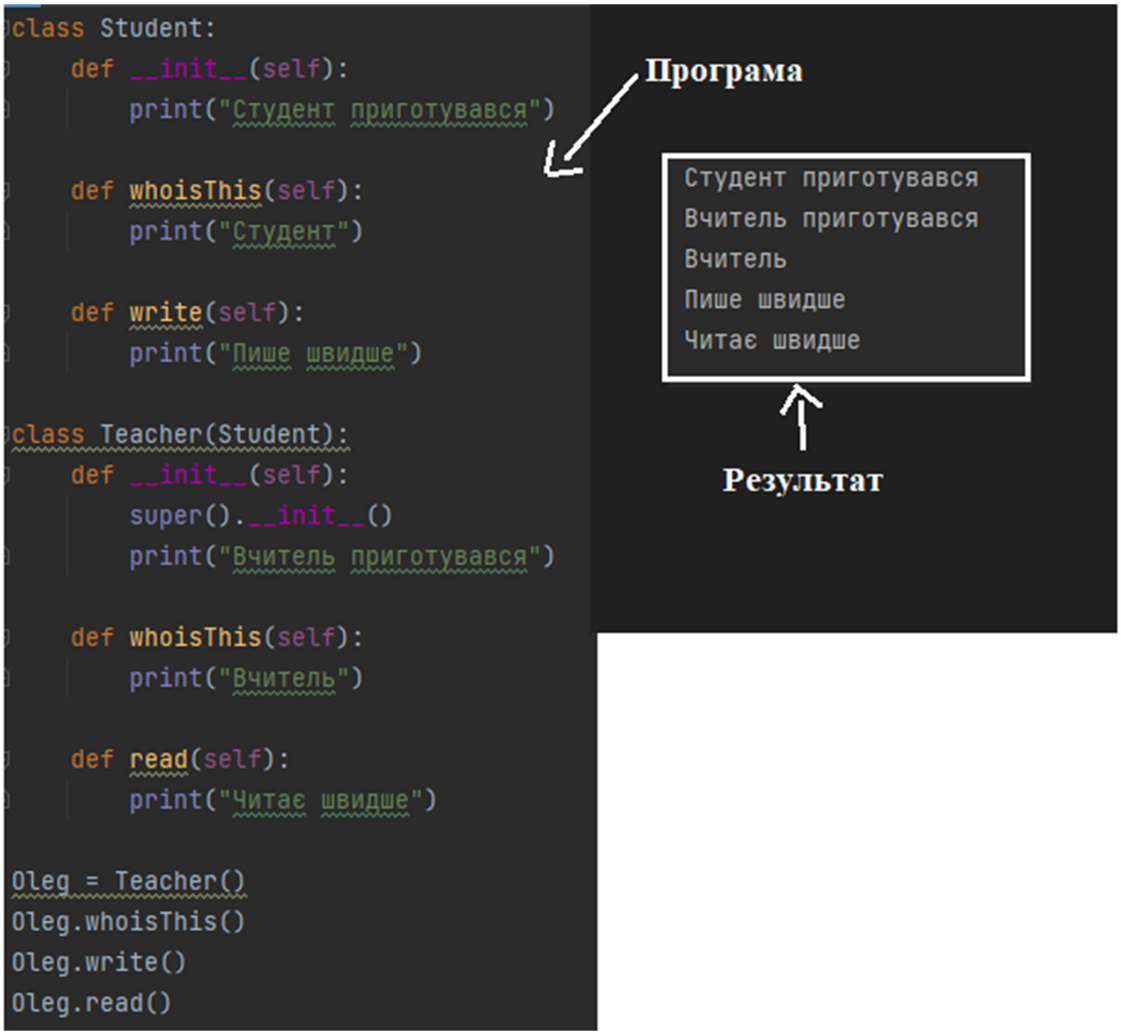

Рисунок 2.7 Наслідування класів та результат виконання програми

Як видно на рисунку, в програмі ми створили два класи – `Student` (батьківський) та `Teacher` (дочірній). Дочірній клас успадковує функції батьківського, це можна побачити на прикладі методу `write()`.

Але при цьому і дочірній клас змінює функціонал батьківського, це

видно на прикладі методу `whoisThis()`. Також, функціонал батьківського класу розширюється методом `read()`.

Також, аби запускати метод `__init__()` батьківського класу всередині дочірнього, використовуються функція `super()` всередині методу `__init__()`.

Інкапсуляція - обмеження доступу до методів та змінних, за рахунок якого можна запобігти модифікації даних. Приватні атрибути вирізняються одинарним або подвійним нижнім підкресленням. Наприклад: (рис. 2.8).

```
class Computer:

    def __init__(self):
        self.__maxprice = 900

    def sell(self):
        print("Ціна : {}".format(self.__maxprice))

    def setMaxPrice(self, price):
        self.__maxprice = price

c = Computer()
c.sell()

c.__maxprice = 1000
c.sell()

c.setMaxPrice(1000)
c.sell()
```

Результат

```
Ціна : 900
Ціна : 900
Ціна : 1000
```

Рисунок 2.8 Програма, що використовує інкапсуляцію

У виконуваній програмі оголошений клас `Computer`. Після чого, використаний метод `__init__()` для збереження максимальної ціни. З результату видно, що при спробі змінити ціну – змін не відбувається, адже Python сприймає `__maxprice` приватним атрибутом. Тому, для зміни ціни потрібно використовувати спеціальну функцію – `setMaxPrice()`, вона прийматиме ціну як параметр.

Поліморфізм – особливість об’єктно-орієнтованого програмування, що дозволяє використовувати одну функцію для різних форм (типів даних).

Наприклад, завдання: розфарбувати фігуру. Форма фігури може бути будь-якою: ромб, квадрат, трикутник, коло. Одним і тим самим методом

можна розфарбувати фігури, незважаючи на їх форму. Такий принцип і називається поліморфізмом (рис.2.9).

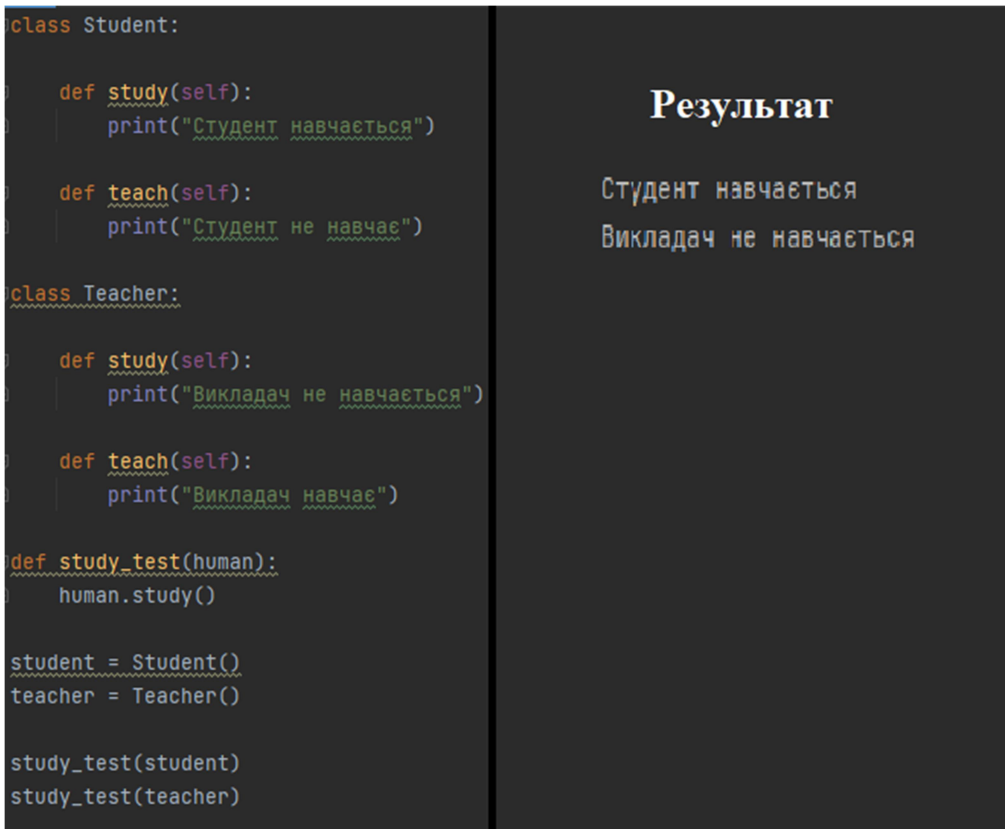

Рисунок 2.9 Поліморфізм

В цій програмі оголошено два класи: `Student` та `Teacher`. В кожному з них описаний загальний метод `study()` , проте функції в них різні. Для використання поліморфізму був створений загальний інтерфейс – функція `study_test()`. В якості аргументу вона приймає будь-який об'єкт, після чого відбувається виклик його власного методу `study()`.

2.2 Можливості використання фреймворку.

Веб-фреймворк Django був винайдений для того, щоб відповідати стислим термінам редакції новин та задовольняти жорсткі вимоги досвідчений веб-розробників. За допомогою Django можливо перевести веб-додатки від концепції до запуску за лічені години. Django бере на себе більшу частину клопоту веб-розробки, тому можна зосередитися на написанні свого додатка без потреби винаходити колесо. Це безкоштовно та з відкритим кодом.

Django можна використовувати і без бази даних, адже він подається з об’єктно-реляційним картографом, у якому можна описати макет своєї бази даних у коді Python. Проте, синтаксис моделі даних пропонує багато різноманітних способів представлення ваших моделей – наразі він вирішував багаторічні проблеми зі схемою бази даних. Ось короткий приклад (рис. 2.10):

```
from django.db import models

class Reporter(models.Model):
    full_name = models.CharField(max_length=70)

    def __str__(self):
        return self.full_name

class Article(models.Model):
    pub_date = models.DateField()
    headline = models.CharField(max_length=200)
    content = models.TextField()
    reporter = models.ForeignKey(Reporter, on_delete=models.CASCADE)

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

Рисунок 2.10 Вирішення проблеми зі схемою бази даних

У даному фрагменті коду створюється дві моделі `Reporter`, яка містить повне ім’я репортера та `Article`, яка містить дату публікації, заголовок, контент та репортера, що посилається на першу модель.

Після виконання даного фрагменту коду, що знаходиться у файлі models.py, потрібно запустити утиліти командного рядка Django, аби автоматично створити таблиці бази даних.

Команда `makemigrations` переглядає всі доступні моделі та створює міграції для тих таблиць, яких ще не існує. А команда `migrate` запускає міграцію та створює таблиці для бази даних, а також надає широкий контроль схеми.

При цьому, можна користуватись безкоштовним та

багатофункціональним Python API для доступу до даних, вона створюється автоматично, без генерації коду (Додаток 1).

Після визначення моделей Django може автоматично може створити професійний готовий адміністративний інтерфейс – веб-сайт, в якому аутентифіковані користувачі можуть додавати, змінювати та видаляти об'єкти (рис.2.11).

```
from django.db import models

class Article(models.Model):
    pub_date = models.DateField()
    headline = models.CharField(max_length=200)
    content = models.TextField()
    reporter = models.ForeignKey(Reporter,
on_delete=models.CASCADE)
```

Рисунок 2.11 Фрагмент коду у файлі models.py

Тому, у файлі `models.py` буде створена модель `Article`, де оголошені його об'єкти.

А от у файлі під назвою `admin.py`, буде знаходитись фрагмент коду, що означатиме створення та оголошення моделей:

```
    from django.contrib import admin

from . import models

admin.site.register(models.Article)
```

Сенс даних фрагментів коду полягає в тому, що сайт редагується співробітниками, або клієнтом, і для цього не обов'язково створювати серверні інтерфейси для керування вмістом. Типовим робочим процесом у створенні програм Django є створення моделей і якнайшвидший запуск сайтів адміністратора, щоб співробітники (або клієнти) могли почати заповнювати дані. Потім розроблятиметься спосіб представлення даних для громадськості.

Чиста, елегантна схема URL-адреси є важливою деталлю високоякісної веб-програми. Django заохочує красивий дизайн URL-адрес і не вкладає в URL-адреси ніяких дрібниць, як-от .php чи .asp. Щоб розробити URL-адреси для програми, треба створити модуль Python під назвою URLconf. Зміст програми, містить зіставлення між шаблонами URL-адрес і функціями зворотного виклику Python. URLconfs також служить для відокремлення URL-адрес від коду Python. Ось як може виглядати URLconf для Reporter:

```
    from django.urls import path

from . import views

urlpatterns = [
    path('articles/<int:year>/',
views.year_archive),
    path('articles/<int:year>/<int:month>/',
views.month_archive),

path('articles/<int:year>/<int:month>/<int:pk>/',
views.article_detail),
]
```

Наведений вище код відображає URL-шляхи до функцій зворотного виклику Python («подання»). Рядки шляху використовують теги параметрів для «захоплення» значень з URL-адрес. Коли користувач надсилає запит на сторінку, Django проходить по порядку кожен шлях і зупиняється на першому, який відповідає запитаній URL-адресі. (Якщо жоден із них не збігається, Django викликає спеціальне представлення 404.) Це неймовірно швидко, оскільки шляхи компілюються у регулярні вирази під час завантаження. Як тільки один із шаблонів URL збігається, Django викликає задане представлення, яке є функцією Python. Кожному представленню передається об’єкт запиту, який містить метадані запиту, і значення, зафіксовані в шаблоні. Наприклад, якщо користувач запросив URL “/articles/2005/05/39323/”, Django викличе функцію news.views.article_detail(request, year=2005, month=5, pk=39323).

Кожне представлення відповідає за виконання однієї з двох речей:

повернення об’єкта HttpResponse, що містить вміст запитуваної сторінки, або виклик виняткової ситуації, наприклад Http404. Решта залежить від розробника. Зазвичай перегляд отримує дані відповідно до параметрів, завантажує шаблон і відображає шаблон із отриманими даними. Ось приклад перегляду year_archive зверху (рис.2.12):

```
from django.shortcuts import render

from .models import Article

def year_archive(request, year):
    a_list =
Article.objects.filter(pub_date__year=year)
    context = {'year': year, 'article_list': a_list}
    return render(request, 'news/year_archive.html',
context)
```

Рисунок 2.12 Фрагмент коду, що знаходиться у файлі views.py

У цьому прикладі використовується система шаблонів Django, яка має кілька потужних функцій, але намагається залишатися достатньо простою для використання непрограмістами.

Наведений вище код завантажує шаблон news/year_archive.html.

Django має шлях пошуку шаблонів, який дозволяє мінімізувати надмірність серед шаблонів. У налаштуваннях Django ви вказуєте список каталогів для перевірки шаблонів за допомогою DIRS. Якщо шаблон не існує в першому каталозі, він перевіряє другий і так далі.

Припустимо, знайдено шаблон news/year_archive.html. Ось як це може виглядати (рис. 2.13):

```
{% extends "base.html" %}

{% block title %}Articles for {{ year }}{% endblock %}

{% block content %}
<h1>Articles for {{ year }}</h1>

{% for article in article_list %}
    <p>{{ article.headline }}</p>
    <p>By {{ article.reporter.full_name }}</p>
    <p>Published {{ article.pub_date|date:"F j, Y"
}}</p>
{% endfor %}
{% endblock %}
```

Рисунок 2.13 Створення власних шаблонів

Фрагмент коду, що знаходиться вище має зберігатись у файлі за шляхом: mysite/news/templates/news/year_archive.html.

Змінні оточені подвійними фігурними дужками. {{ article.headline }} означає «Вивести значення атрибута заголовка статті». Але крапки використовуються не лише для пошуку атрибутів. Вони також можуть виконувати пошук за ключем словника, пошук індексу та виклики функцій.

Примітка: {{ article.pub_date|date:"F j, Y" }} використовує “конвеєр” у стилі Unix (символ “|”). Це називається фільтром шаблону, і це спосіб фільтрувати значення змінної. У цьому випадку фільтр дати форматує об’єкт Python datetime у заданому форматі (як це можна знайти у функції дати PHP).

Можна об’єднати безліч фільтрів, а також можна написати спеціальні шаблони фільтрів та користувацькі теги шаблону, які запускають нестандартний код Python за лаштунками.

Нарешті, Django використовує концепцію «успадкування шаблонів». Ось що робить {% extends "base.html" %}. Це означає певні дії: «Спочатку завантажити шаблон під назвою «база», який визначив групу блоків, і заповнити блоки наступними блоками». Це дозволяє значно скоротити надмірність у шаблонах: кожен шаблон має визначати лише те, що є

унікальним для цього шаблону.

Ось як може виглядати шаблон «base.html», включаючи використання статичних файлів (рис. 2.14):

```
{% load static %}
<html>
<head>
    <title>{% block title %}{% endblock %}</title>
</head>
<body>
    <img src="{% static 'images/sitelogo.png' %}"
alt="Logo">
    {% block content %}{% endblock %}
</body>
</html>
```

Рисунок 2.14 Шаблон «base.html»

Він визначає зовнішній вигляд сайту (з логотипом сайту) і забезпечує «діри» для заповнення дочірніх шаблонів. Це означає, що редизайн сайту можна здійснити, змінивши один файл – базовий шаблон. Це також дозволяє створювати кілька версій сайту з різними базовими шаблонами, одночасно використовуючи дочірні шаблони. Творці Django використали цю техніку для створення разюче різних мобільних версій сайтів, створивши лише новий базовий шаблон. Зауважте, що не обов'язково використовувати систему шаблонів Django, якщо перевага віддається іншій системі. Хоча система шаблонів Django особливо добре інтегрована з моделлю Django, ніщо не змушує використовувати її. З цього приводу також не потрібно використовувати API бази даних Django. Є можливість використовувати інший рівень абстракції бази даних, читати XML-файли, читати файли з диска або все, що завгодно. Кожна частина Django – моделі, перегляди, шаблони – відокремлена від наступної.

Ще кілька корисних функцій:

- фреймворк кешування, який інтегрується з memcached або іншими бекендами;

- структура синдикації, яка дозволяє створювати канали RSS і Atom, написавши невеликий клас Python;
- більше привабливих автоматично згенерованих функцій адміністратора.

2.3 Висновки до 2 розділу

З огляду на пункти 2.1 та 2.2 стає зрозуміло, що використання мови програмування Python та фреймворку Django є очевидним: Python – об’єктно-орієнтована мова програмування, що має великий функціонал, простий синтаксис та велику популярність серед розробників. А Django – найкращий помічний для розробника, що при спрощеній роботі програміста, сам створює великий функціонал. А великою перевагою Django є система адміністрування, що не вимагає додаткових дій з боку розробника.

Об’єктно-орієнтоване програмування спрощує розуміння програми і робить її ефективнішою, а поліморфізм надає загальний інтерфейс різним об’єктам, що робить програму ефективнішою. Також стає зрозуміло, що всі класи – загальні, їх код можна використовувати повторно, а всі дані знаходяться під захистом завдяки абстракції.

РОЗДІЛ 3. СТВОРЕННЯ ОНЛАЙН РЕСУРСУ ДЛЯ ВИВЧЕННЯ ООП НА БАЗІ МОВИ ПРОГРАМУВАННЯ PYTHON ТА ФРЕЙВОРКУ DJANGO

Розробка веб-додатку передбачає створення ресурсу, що слугуватиме помічником вчителя та учнів у вивченні будь-якого предмету, зокрема орієнтуючись на вивчення об’єктно-орієнтованого програмування засобами Python. Функціонал сайту передбачає додавання, редагування та створення матеріалів, що прикріплені до певного розділу та теми з будь-якого предмету, відстеження прогресу учня у освоєнні нового матеріалу та можливість прикріплення готових матеріалів, що подані у форматі pdf, для ознайомлення з ними.

3.1 Опис структури проєкту та створених моделей.

Для створення проєкту була обрана мова програмування Python та фреймворк Django, а середовищем програмування у якому створювався проєкт був обраний PyCharm, за причини переваг у зручності та інтерфейсі. При створенні проєкту утворюється певна структура у вигляді папок та файлів, яка відображається в середовищі програмування (рис. 3.1).

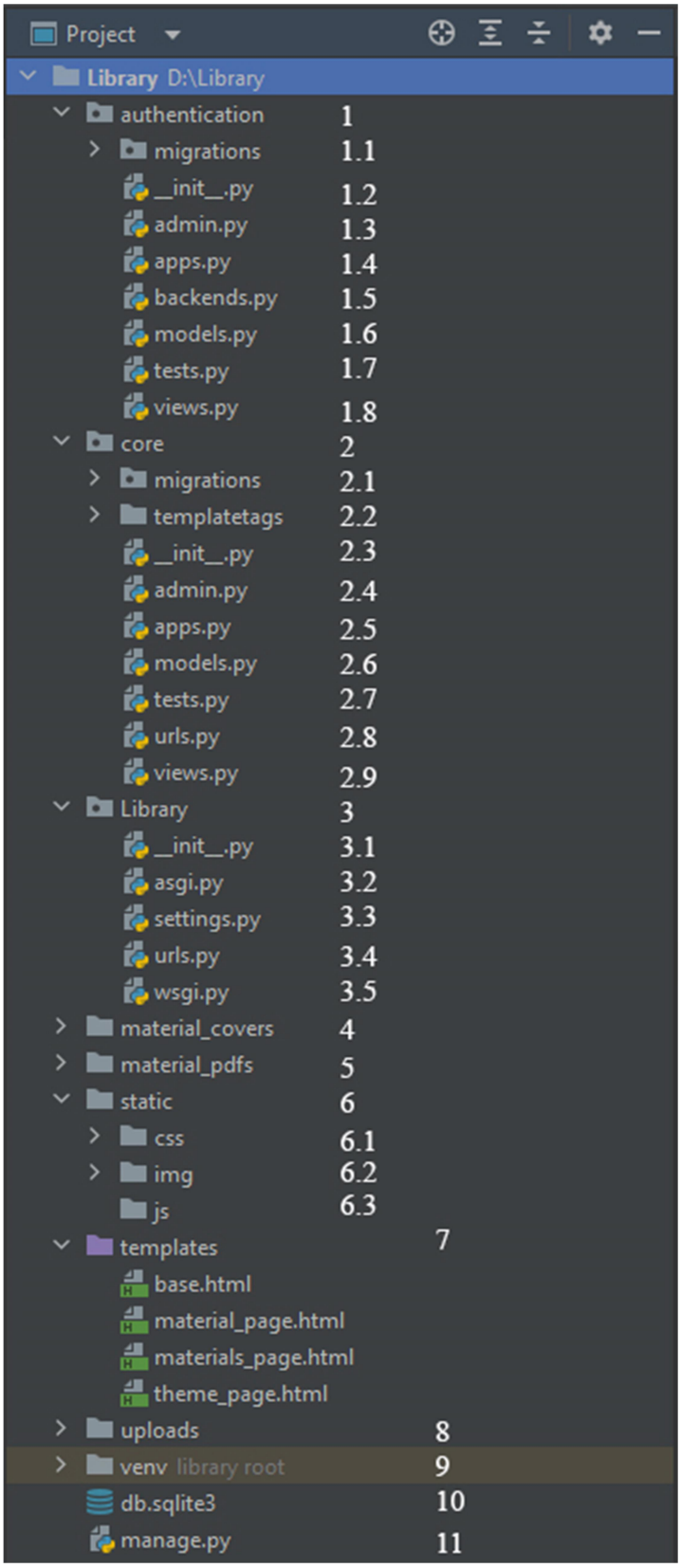

Рисунок 3.1 Структура проєкту

На поданому рисунку, що відображає структуру створеного проєкту, кожен файл та папка відповідають за певний функціонал:

1. Каталог authentification – створений додаток, у якому перевизначається стандартна модель користувача Django.
   1.1. Каталог migrations – каталог міграцій створеного додатку, який проєктує базу даних.
   1.2. Файл __init__.py – порожній файл для того, щоб Django та Python розпізнавали папку як модуль Python і дозволяє використовувати його об'єкти всередині інших частин проекту.
   1.3. Файл admin.py – використовується для реєстрації моделей в панелі адміністратора Django.
   1.4. Файл apps.py – містить в собі базові налаштування додатку такі, як назва додатку в панелі адміністратора.
   1.5. Файл backends.py – код, який знаходиться у файлі, перевизначає логіку поведінки базового користувача Django.
   1.6. Файл models.py – описує в собі моделі, які в подальшому проектуватимуть таблиці в базі даних.
   1.7. Файл tests.py – файл для тестування додатку.
   1.8. Файл views.py – відповідає за логіку поведінки сторінок сайту, за їх генерацію та дані, що будуть на них відображатись, логіку кнопок тощо.
2. Каталог core – створений додаток, у якому міститься загальна логіка функціоналу сайту.
   2.1. Каталог migrations – каталог міграцій створеного додатку, який проєктує базу даних.
   2.2. Каталог templatetags – містить в собі, створені розробником, теги, які використовуються в шаблонах.
   2.3. – 2.7. – Файли повторюються та мають схожу логіку, але для різних додатків.
   2.8. Файл urls.py – відповідає за реєстрацію сторінок у вигляді посилань.

3. Каталог Library – створений автоматично каталог проєкту, який містить в собі глобальні налаштування.
   3.1. Файл __init__.py – наявний для ініціалізації додатку в фреймворку.
   3.2. Файл asgi.py є наступником WSGI, призначеним для забезпечення стандартного інтерфейсу між асинхронними веб-серверами, фреймворками та програмами Python.
   3.3. Файл settings.py містить у собі всі налаштування проекту, він реєструє додатки, задає розміщення статичних файлів, налаштування бази даних тощо.
   3.4. Файл urls.py – задає асоціації url адрес з уявленнями. Незважаючи на те, що цей файл може містити всі налаштування URL, зазвичай його ділять на частини, по одній на додаток.
   3.5. Файл wsgi.py – використовується для налагодження зв'язку між вашим програмою Django і веб-сервером.
4. Каталог material_covers – каталог, що містить обкладинки для візуалізації створених матеріалів.
5. Каталог material_pdfs – папка, в якій розміщуються та зберігаються копії матеріалів у форматі pdf.
6. У каталозі static розміщуються статичні матеріали.
   6.1. Каталог css – містить файли стилів, які відповідають за зовнішній вигляд веб-сторінки.
   6.2. Каталог img - містить статичні зображення, що використовуються на сайті, наприклад, логотип сайту.
   6.3. Каталог js – містить статичні файли скриптів.
7. Каталог templates – може містити безліч користувацький шаблонів сторінок.
8. Каталог uploads – містить файли, що завантажує користувач при створенні матеріалу.

9. Каталог venv – модуль venv підтримує створення легких «віртуальних середовищ», кожне з власним незалежним набором пакетів Python, встановлених у каталогах сайту. Віртуальне середовище створюється поверх існуючої інсталяції Python, відомої як «базова» Python віртуального середовища, і за бажанням може бути ізольовано від пакетів у базовому середовищі, тому доступні лише ті, які явно встановлено у віртуальному середовищі.
10. Файл db.sqlite3 – реляційна база даних SqLite.
11. Файл manage.py автоматично створюється у проєктах – утиліта командного рядка Django для виконання адміністративних задач.

Модель User – базова модель користувача (рис. 3.2).

```
class User(AbstractUser):
    username = None
    last_name = None
    email = models.EmailField('Електронна пошта', unique=True)
    second_name = models.CharField('Прізвище', max_length=55,
null=True)
    first_name = models.CharField('Ім\'я', max_length=55)
    third_name = models.CharField('По батькові', max_length=150)
    phone = models.CharField('Номер телефону', blank=True,
max_length=80,
                             help_text='Телефон може бачити тільки
адміністратор')
    birthday = models.DateField('Дата народження', blank=True,
null=True,
                                help_text='Дату народження може
бачити тільки адміністратор')
    restore_password_token = models.CharField('Токен для
відновлення паролю', max_length=100, blank=True)

    USERNAME_FIELD = 'email'
    REQUIRED_FIELDS = []

    objects = UserManager()
```

Рисунок 3.2 Модель User

Оскільки стандартна модель перевизначається, то поля username та last_name стають непотрібними, тому їх значення анулюється.

Поле email відповідає за електронну пошту користувача, для його подальшої авторизації у системі, поле є унікальним.

Поле second_name, first_name та third_name символьні поля, з максимальною довжиною у 55 символів, що відповідають прізвищу, імені та імені по-батькові користувача.

Поле phone також є символьним полем, довжина якого може бути не більше 80 символів відповідає номеру мобільного телефону користувача.

Поле birthday відповідає за дату народження користувача і може бути записано тільки у форматі дати.

Поле restore_password_token – поле для відновлення паролю, містить в собі рядок зі 100 випадкових символів.

Фрагмент коду, що має вигляд EmailField або CharField, або будь-яке …Field – це оголошення типу поля.

Модель Class. Данна модель (рис. 3.3) слугує допоміжним засобом сортування учнів по класам, а також вибором класу для вчителя. Містить лише одне поле – name, що дозволяє написати назву класу, наприклад «10А».

```
class Class(Model):
    name = CharField('Назва', max_length=10)

    def __str__(self):
        return f'{self.name}'

    class Meta:
        verbose_name = 'Клас'
        verbose_name_plural = 'Класи'
        ordering = ('name',)
```

Рисунок 3.3 Модель Class

Модель Teacher. У моделі визначається який з користувачів є викладачем за допомогою поля user, що утворює зв’язок з моделлю User «один-до-одного». А також, у полі classes утворюється зв’язок між вчителем і класом, у якому він може викладати (рис. 3.4).

```
class Teacher(Model):
    user = OneToOneField(User, CASCADE, verbose_name='Користувач')
    classes = ManyToManyField(Class, verbose_name='Класи',
blank=True)

    def __str__(self):
        return f'{self.user.get_full_name()}'

    class Meta:
        verbose_name = 'Вчитель'
        verbose_name_plural = 'Вчителі'
        ordering = ('user__second_name', 'user__first_name',
'user__third_name')
```

Рисунок 3.4 Модель Teacher

Модель Pupil. Створена за аналогією з попередньою моделлю, проте відповідає користувачам з роллю «Учень» (рис. 3.5).

```
class Pupil(Model):
    user = OneToOneField(User, CASCADE, verbose_name='Користувач')
    classes = ManyToManyField(Class, verbose_name='Класи',
blank=True)

    def __str__(self):
        return f'{self.user.get_full_name()}'

    class Meta:
        verbose_name = 'Учень'
        verbose_name_plural = 'Учні'
        ordering = ('user__second_name', 'user__first_name',
'user__third_name')
```

Рисунок 3.5 Модель Pupil

Модель Category. Відповідає за поділення матеріалів на групи, що відповідають певному предмету. Має лише одне поле назви (рис.3.6).

```
class Category(Model):
    name = CharField('Назва', max_length=128)

    def __str__(self):
        return f'{self.name}'

    class Meta:
        verbose_name = 'Категорія'
        verbose_name_plural = 'Категорії'
        ordering = ('name',)
```

Рисунок 3.6 Модель Category

Модель Material. Модель, що дозволяє додавати матеріал до теми (рис. 3.7).

```
class Material(Model):
    category = ForeignKey(Category, CASCADE,
verbose_name='Категорія', null=True)
    cover = ImageField('Обложка', upload_to='material_covers/',
null=True, blank=True)
    name = CharField('Назва', max_length=128)
    description = RichTextUploadingField('Опис')
    pdf = FileField('Готовий PDF файл матеріалу',
upload_to='material_pdfs/', null=True, blank=True)
    access_classes = ManyToManyField(Class, verbose_name='Доступний
для класів', blank=True)
        hide = BooleanField('Приховати матеріал', default=False)
    created_date = DateTimeField('Дата створення',
auto_now_add=True)
    creator = ForeignKey(Teacher, CASCADE, verbose_name='Автор',
null=True)

    def __str__(self):
        return f'{self.name} /
{self.created_date.strftime("%d.%m.%Y")}'


    class Meta:
        verbose_name = 'Матеріал'
        verbose_name_plural = 'Матеріали'
        ordering = ('-created_date', 'name')
```

Рисунок 3.7 Модель Material

Модель має певні поля, що посилаються на моделі з відповідною назвою: поле category – модель Category, поле access_classes – модель Class у зв'язку «багато до багатьох», а також поле creator утворює зв'язок з моделлю Teacher.

За допомогою полів cover можна додати обкладинку до матеріалу, name – давати назву доданому матеріалу, description – поле для короткого опису матеріалу, pdf – дозволяє додавати додатковий матеріал, а поле created_date автоматично додає дату створення матеріалу.

Модель Theme. Модель, що описує теми, які додаються до матеріалу.

```
class Theme(Model):
    material = ForeignKey(Material, CASCADE,
verbose_name='Матеріал')
    name = CharField('Назва', max_length=128)
    description = RichTextUploadingField('Опис')
    ordering = IntegerField('Номер по порядку', null=True,
blank=True)
    hide = BooleanField('Приховати матеріал', default=False)
    created_date = DateTimeField('Дата створення',
auto_now_add=True)

    def __str__(self):
        return f'{self.name} /
{self.created_date.strftime("%d.%m.%Y")}'

    def save(
            self, force_insert=False, force_update=False,
using=None, update_fields=None
    ):
        if not self.ordering and not self.id:
            self.ordering = self.material.theme_set.all().count() +
1
        super().save()

    class Meta:
        verbose_name = 'Тема'
        verbose_name_plural = 'Теми'
        ordering = ('created_date', 'name')
```

Рисунок 3.8 Модель Theme

Дана модель так само, як попередня має поле, що пов’язане з попередньо створеною моделлю та утворює з нею зв’язки: поле material – модель Material.

Поля name та created_date автоматично додають дату створення матеріалу та дають назву доданому темі відповідно. Поле ordering допомагає відсортувати теми, задавши їм порядковий номер. Поле hide дозволяє приховати тему, якщо в цьому є необхідність та має тип boolean.

Модель Progress. Модель має поля theme (пов’язана з відповідною моделлю), pupil (зв’язок з моделлю Pupil) та created_date та дозволяє відслідковувати прогрес учня у опрацювання матеріалу.

```
class Progress(Model):
    theme = ForeignKey(Theme, CASCADE, verbose_name='Тема')
    pupil = ForeignKey(Pupil, CASCADE, verbose_name='Учень')
    created_date = DateTimeField('Дата створення',
auto_now_add=True)

    def __str__(self):
        return f'{self.theme} / {self.pupil}'

    class Meta:
        verbose_name = 'Прогрес'
        verbose_name_plural = 'Прогрес'
        unique_together = ('theme', 'pupil',)
```

Рисунок 3.9 Модель Progress

3.2 Опис робочого інтерфейсу.

У результаті написання програми та описаних моделей утворюється сайт, що має власний графічний інтерфейс для зручності користувача.

У верхній частині головної сторінки сайту (рис. 3.10), розміщуються логотип, активні кнопки основних вкладок («Головна» та «Матеріали») та активна кнопка, що відображає авторизованого в систему користувача.

Основну частину сторінки займають матеріали, при цьому відображаються: обкладинка, тема, короткий опис, дата додання, автор, назва класу, для якого ці матеріали доступні та прогрес.

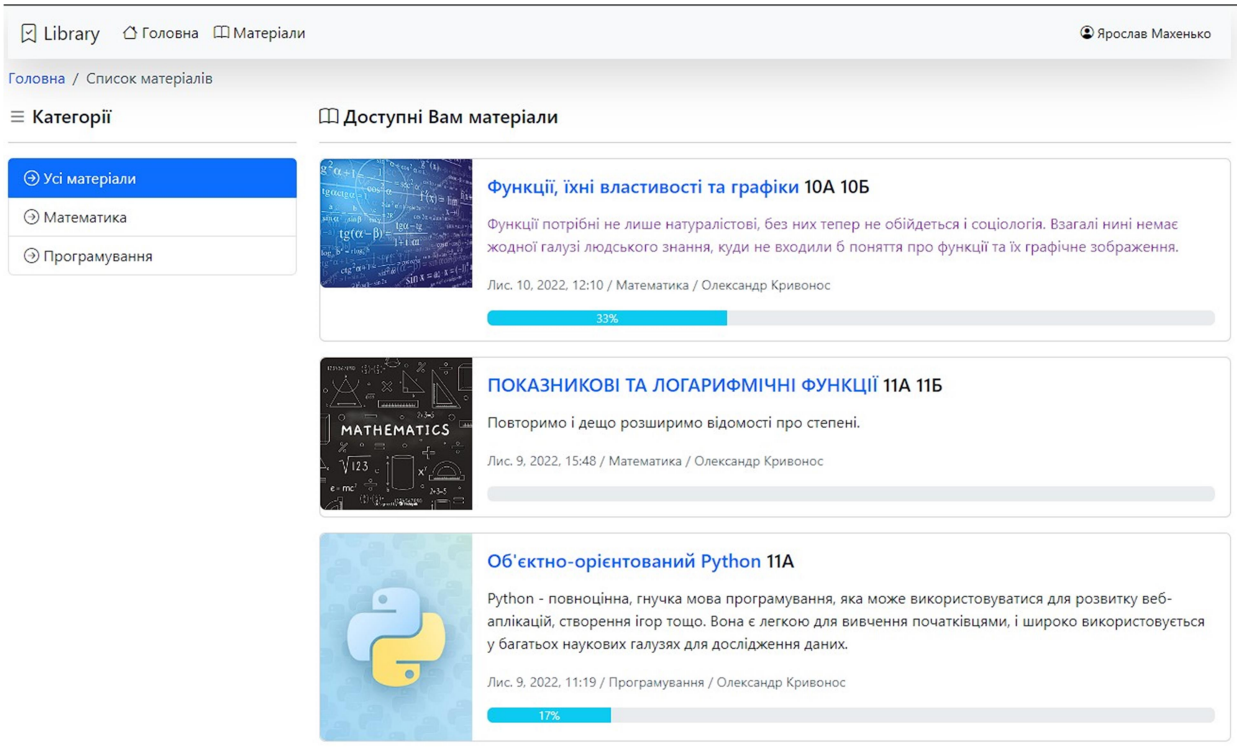

Рисунок 3.10 Головна сторінка сайту

Ліворуч від основної частини розміщується категоризація матеріалів по предметам (рис. 3.11).

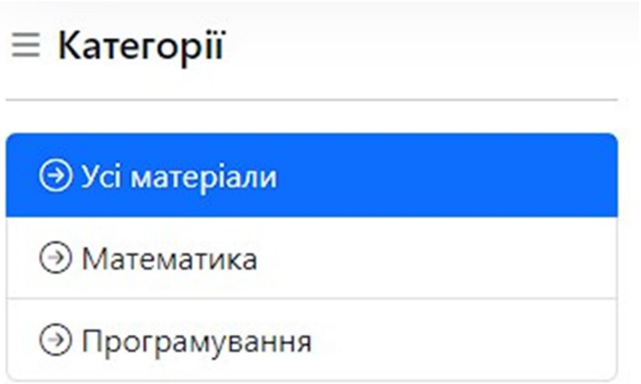

Рисунок 3.11 Категоризація матеріалів

Як видно з рисунку, на сайт можна додавати не лише матеріали до курсу «Інформатика», його використання передбачено для будь-якого предмету.

При виборі категорії матеріалів (тобто предмету), відкривається сторінка з темами (відображаються ліворуч), що відповідають обраному предмету (рис.3.12). А при виборі конкретної теми у основній частині

сторінки доступні матеріали теми для опрацювання де, після зображення короткої інформації, буде доступний повний матеріал з теми.

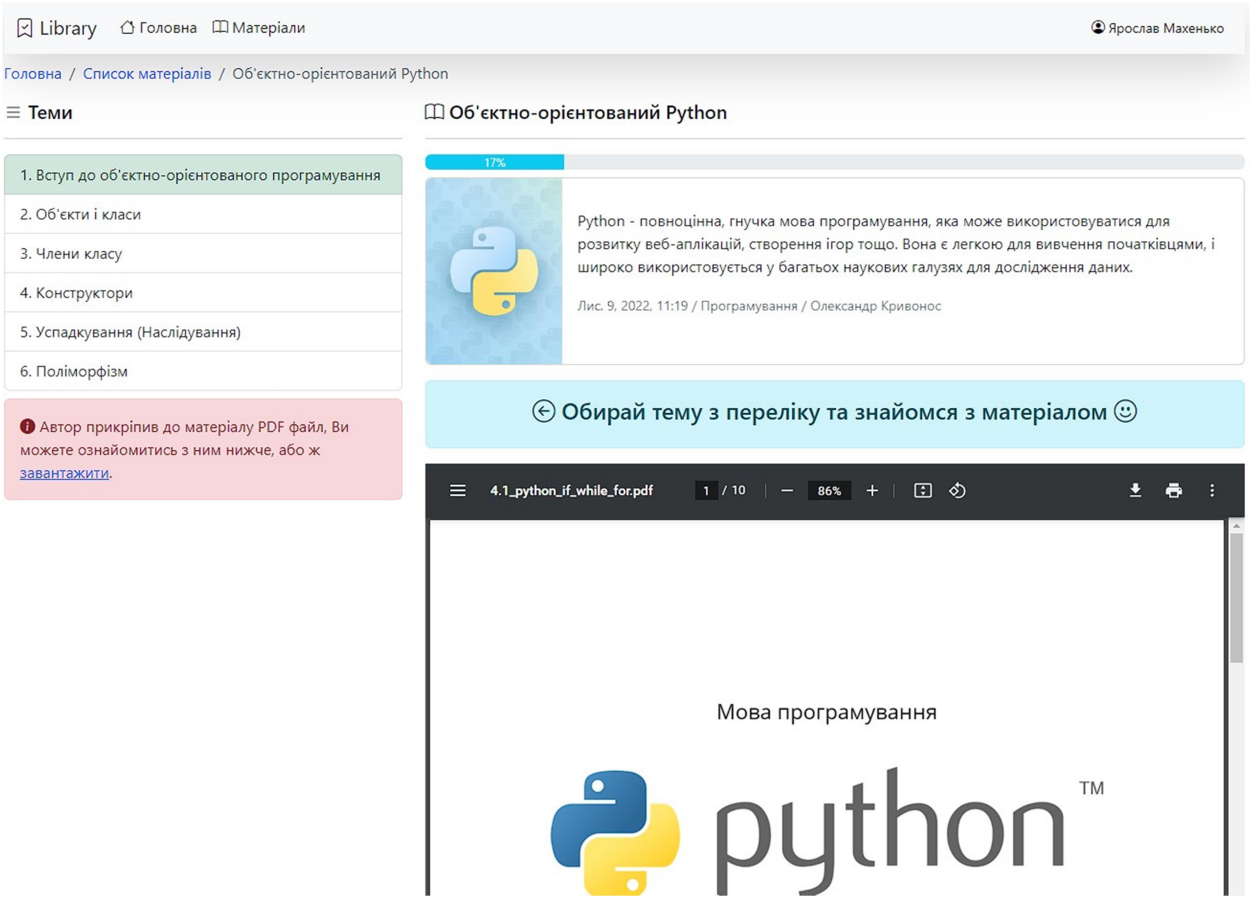

Рисунок 3.12 Сторінка матеріалу

На наступному зображенні (рис. 3.13) сторінка, що відображається після вибору підтеми:

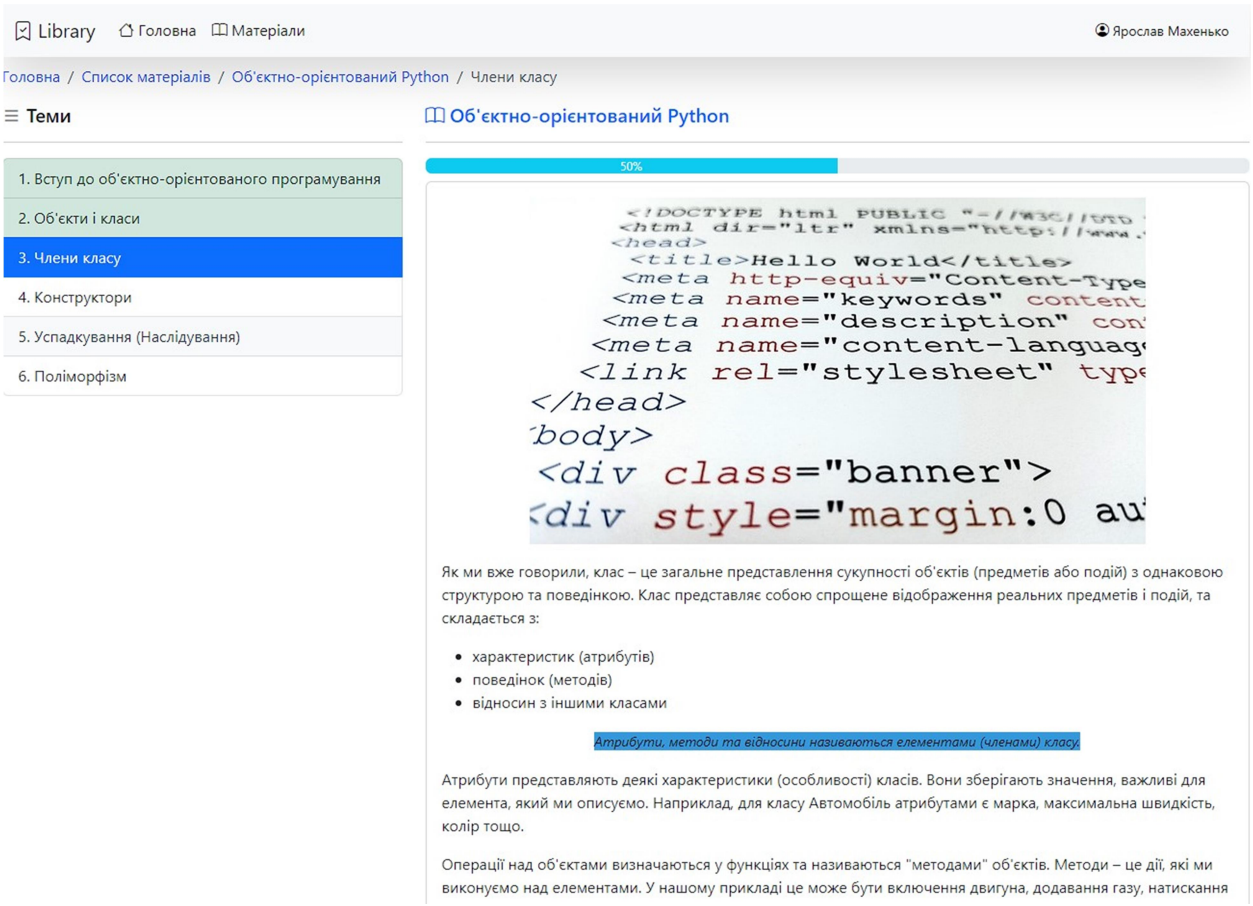

Рисунок 3.13 Тема матеріалу

На даній сторінці доступний повний матеріал, що вчитель додав. При цьому, після того як учень ознайомиться з ним та перейде до наступного – відсоток прогресу, що відображається над назвою підтеми, збільшиться – це дозволить відслідковувати активність учнів та рівень прогресу кожного з них.

Варто зауважити, що при додаванні матеріалу можна завантажити готовий pdf, аби не редагувати доданий текст у редакторі.

При додаванні матеріалів вчителем потрібно перейти на сторінку адміністратора, та натиснути кнопку «додати» біля пункту «матеріали» . Відкриється сторінка (рис. 3.14) де вчитель матиме можливість редагувати та форматувати текст матеріалів, вставляти рисунки, формули чи таблиці.

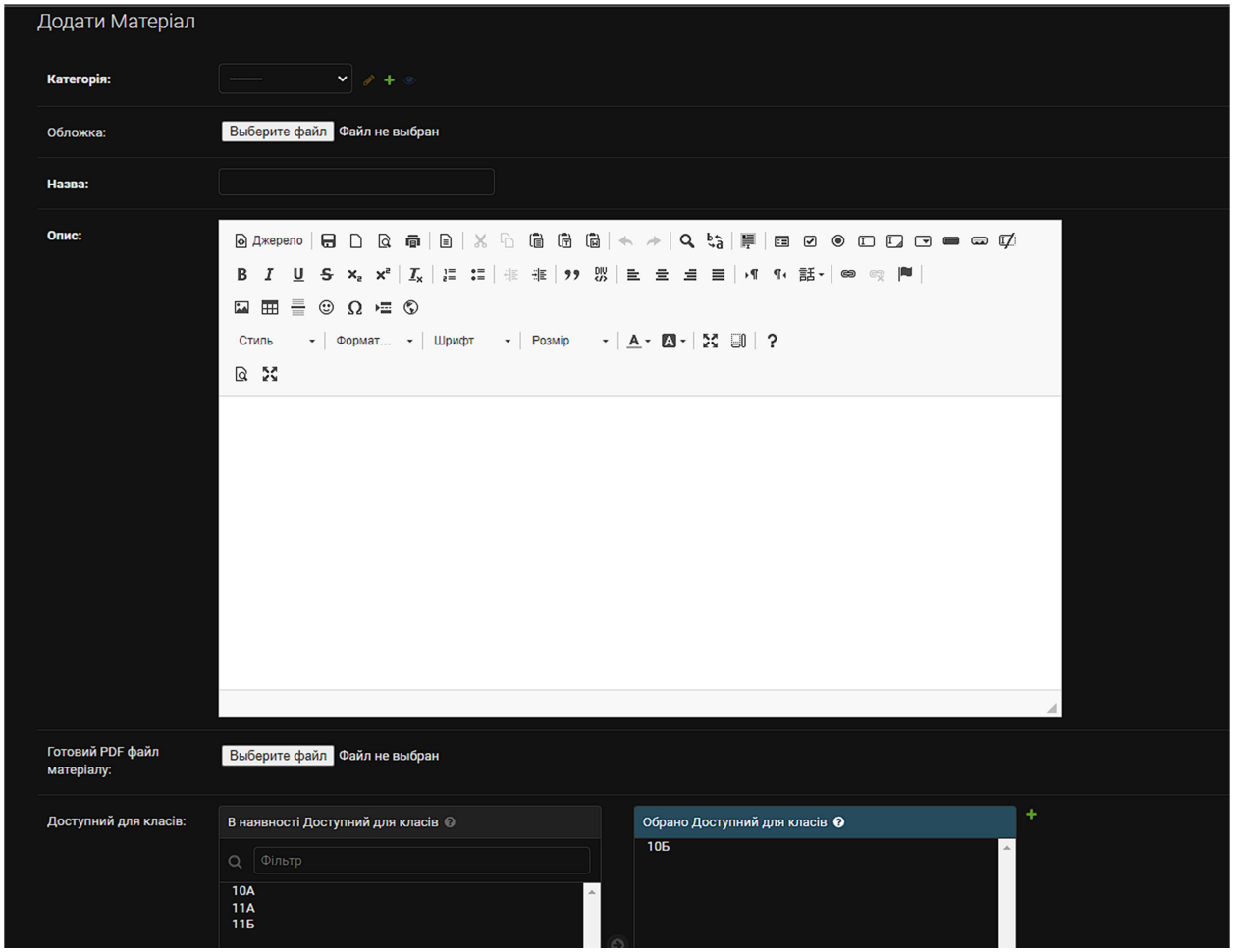

Рисунок 3.14 Додавання матеріалів

При додаванні матеріалу обов’язково потрібно обрати з переліку категорію, додати обгортку, вписати назву матеріалу та обрати клас, якому буде доступний перегляд даного матеріалу.

ВИСНОВКИ

У процесі створення дипломної роботи на тему: «Методика вивчення об’єктно-орієнтованого програмування в курсі інформатики з використанням web-фреймворку Django» були досліджені теоретичні засади вивчення об’єктно-орієнтованого програмування в шкільній програмі, проаналізовано навчальні програми для 10-11 класів рівня стандарту та профільного рівня на вміст тем, що вивчають об’єктно-орієнтоване програмування; проаналізовано середовища програмування, за допомогою яких можна та доцільно вивчати об’єктно-орієнтоване програмування у навчальних закладів орієнтуючись на мову програмування засобами якої буде відбуватись вивчення; досліджено основні поняття мови програмування Python і її особливості у об’єктно-орієнтованому програмуванні, а також досліджено фреймворк Django, її особливості підключення та використання; було створено онлайн ресурс для вивчення об’єктно – орієнтованого програмування засобами мови програмування Python та з використанням фреймворку Django. У процесі створення сайту були додані матеріали для вивчення об’єктно-орієнтованого програмування за допомогою Python, а також матеріали до іншої категорії для прикладу.

У процесі виконання кваліфікаційної роботи була досягнена поставлена мета – дослідити вивчення об’єктно-орієнтованого програмування в школі та розробити онлайн ресурс для вивчення об’єктно-орієнтованого програмування засобами мови програмування Python, фреймворк Django.

Апробація відбулась на: IV Всеукраїнському відкритому науково-практичному онлайн-форумі «ІННОВАЦІЙНІ ТРАНСФОРМАЦІЇ В СУЧАСНІЙ ОСВІТІ: ВИКЛИКИ, РЕАЛІЇ, СТРАТЕГІЇ» (27 жовтня 2022 р.), участь у VII Міжнародній науково-практичній студентській конференції «Професійна іншомовна підготовка в умовах глобальних комунікативних потреб» (3 листопада 2022р.)., участь у VII Всеукраїнській науково-

практичній конференції з міжнародною участю “Сучасні інформаційні технології в освіті та науці” (17-18 листопада 2022 р.).

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Навчальна програма рівня стандарту для 10-11 класів з інформатики, веб-сайт. URL: https://mon.gov.ua/ua/osvita/zagalna-serednya-osvita/navchalni-programi/navchalni-programi-dlya-10-11-klasiv (дата звернення: 01.10.2022).
2. Навчальна програма профільного рівня для 10-11 класів з інформатики, веб-сайт. URL: https://mon.gov.ua/ua/osvita/zagalna-serednya-osvita/navchalni-programi/navchalni-programi-dlya-10-11-klasiv (дата звернення: 01.10.2022).
3. Махенько Я., Стельмашенко Я. Структура шкільного курсу інформатики в старших класах. VII Всеукр. наук.-практ. конф. з міжн. уч. «Сучасні інформаційні технології в освіті та науці»,17-18 лист. 2022 р. Житомир: ЖДУ, 2022.
4. Махенько Я., Стельмашенко Я. Вивчення об’єктно-орієнтованого програмування в школі. IV Всеукр. наук.-практ. онл.-фор. «Іноваційні трансформації в сучасній освіті: виклики, реалії, стратегії», 27 жовт. 2022 р. Київ: ДНУ, 2022.
5. PHILLIPS, Dusty. Python 3 object oriented programming. Packt Publishing Ltd, 2010
6. The Effects of Objects-First and Objects-Late Methods on Achievements of OOP Learners. URL: https://www.scirp.org/html/23962.html?pagespeed=noscript (дата звернення: 01.10.2022).
7. Empirical comparison of objects-first and objects-later. URL: https://dl.acm.org/doi/pdf/10.1145/1584322.1584326?casa_token=xL2ji8lMSTUAAAAA:zivCJixx7uP2Q3K3ECycHzMuGBZpjNicBpHBhtvMSrTB4YsK0H-0B607TzZbMp1_SyriZWDwYp-D (дата звернення: 01.10.2022).

8. An Empirical Study of Novice Program Comprehension in the Imperative and Object-Oriented Styles. URL: https://dl.acm.org/doi/pdf/10.1145/266399.266411 (дата звернення: 01.10.2022).

9. Конспект лекції No 8-9-10Тема No 5. КОНЦЕПЦІЯ ОБ'ЄКТНО-ОРІЄНТОВАНОГО ПРОГРАМУВАННЯ URL: https://financial.lnu.edu.ua/wp-content/uploads/2017/09/lektsiia_OOP_8_9_10_tema-5.pdf (дата звернення: 03.10.2022).

10. J. Gosling, B. Joy, G. Steele, G. Brachda. The Java Language Specification, 2nd Edition URL: https://docs.oracle.com/javase/specs/jls/se19/jls19.pdf (дата звернення: 03.10.2022).

11. Microsoft. The modern, innovative, open-source programming language for building all your apps. URL: https://dotnet.microsoft.com/en-us/languages/csharp (дата звернення: 04.10.2022).

10. Editors' Report -- Programming Languages -- C++ URL: https://web.archive.org/web/20200204081709/http://open-std.org/jtc1/sc22/wg21/docs/papers/2017/n4661.html (дата звернення: 04.10.2022).

13. Офіційна сторінка Python URL: https://www.python.org/ (дата звернення: 03.10.2022).

14. Python 3.10.1 is available URL: https://web.archive.org/web/20220106181415/https://blog.python.org/2021/12/python-3101-is-available.html (дата звернення: 02.10.2022).

15. The Python tutorial, URL: https://docs.python.org/3/tutorial/ (дата звернення: 01.10.2022).

16. Django, URL: https://www.djangoproject.com/ (дата звернення: 05.10.2022).

17. Інформатика 10-11 клас/ Ривкінд Й. Я. та ін. Київ: Генеза, 2018, 144 с.

ДОДАТКИ

Додаток 1

```
# No reporters are in the system yet.
>>> Reporter.objects.all()
<QuerySet []>

# Create a new Reporter.
>>> r = Reporter(full_name='John Smith')

# Save the object into the database. You have to call save() explicitly.
>>> r.save()

# Now it has an ID.
>>> r.id
1

# Now the new reporter is in the database.
>>> Reporter.objects.all()
<QuerySet [<Reporter: John Smith>]>

# Fields are represented as attributes on the Python object.
>>> r.full_name
'John Smith'

# Django provides a rich database lookup API.
>>> Reporter.objects.get(id=1)
<Reporter: John Smith>
>>> Reporter.objects.get(full_name__startswith='John')
<Reporter: John Smith>
>>> Reporter.objects.get(full_name__contains='mith')
<Reporter: John Smith>
>>> Reporter.objects.get(id=2)
Traceback (most recent call last):
    ...
DoesNotExist: Reporter matching query does not exist.

# Create an article.
>>> from datetime import date
>>> a = Article(pub_date=date.today(), headline='Django is cool',
...     content='Yeah.', reporter=r)
>>> a.save()

# Now the article is in the database.
>>> Article.objects.all()
<QuerySet [<Article: Django is cool>]>

# Article objects get API access to related Reporter objects.
>>> r = a.reporter
>>> r.full_name
'John Smith'

# And vice versa: Reporter objects get API access to Article objects.
>>> r.article_set.all()
<QuerySet [<Article: Django is cool>]>

# The API follows relationships as far as you need, performing efficient
# JOINs for you behind the scenes.
# This finds all articles by a reporter whose name starts with "John".
>>> Article.objects.filter(reporter__full_name__startswith='John')
<QuerySet [<Article: Django is cool>]>

# Change an object by altering its attributes and calling save().
```

```
>>> r.full_name = 'Billy Goat'
>>> r.save()

# Delete an object with delete().
>>> r.delete()
```

Додаток 2

```
# No reporters are in the system yet.
>>> Reporter.objects.all()
<QuerySet []>

# Create a new Reporter.
>>> r = Reporter(full_name='John Smith')

# Save the object into the database. You have to call save() explicitly.
>>> r.save()

# Now it has an ID.
>>> r.id
1

# Now the new reporter is in the database.
>>> Reporter.objects.all()
<QuerySet [<Reporter: John Smith>]>

# Fields are represented as attributes on the Python object.
>>> r.full_name
'John Smith'

# Django provides a rich database lookup API.
>>> Reporter.objects.get(id=1)
<Reporter: John Smith>
>>> Reporter.objects.get(full_name__startswith='John')
<Reporter: John Smith>
>>> Reporter.objects.get(full_name__contains='mith')
<Reporter: John Smith>
>>> Reporter.objects.get(id=2)
Traceback (most recent call last):
    ...
DoesNotExist: Reporter matching query does not exist.

# Create an article.
>>> from datetime import date
>>> a = Article(pub_date=date.today(), headline='Django is cool',
...     content='Yeah.', reporter=r)
>>> a.save()

# Now the article is in the database.
>>> Article.objects.all()
<QuerySet [<Article: Django is cool>]>

# Article objects get API access to related Reporter objects.
>>> r = a.reporter
>>> r.full_name
'John Smith'

# And vice versa: Reporter objects get API access to Article objects.
>>> r.article_set.all()
<QuerySet [<Article: Django is cool>]>

# The API follows relationships as far as you need, performing efficient
# JOINs for you behind the scenes.
# This finds all articles by a reporter whose name starts with "John".
>>> Article.objects.filter(reporter__full_name__startswith='John')
<QuerySet [<Article: Django is cool>]>

# Change an object by altering its attributes and calling save().
>>> r.full_name = 'Billy Goat'
>>> r.save()
```

```
# No reporters are in the system yet.
>>> Reporter.objects.all()
<QuerySet []>

# Create a new Reporter.
>>> r = Reporter(full_name='John Smith')

# Save the object into the database. You have to call save() explicitly.
>>> r.save()

# Now it has an ID.
>>> r.id
1

# Now the new reporter is in the database.
>>> Reporter.objects.all()
<QuerySet [<Reporter: John Smith>]>

# Fields are represented as attributes on the Python object.
>>> r.full_name
'John Smith'

# Django provides a rich database lookup API.
>>> Reporter.objects.get(id=1)
<Reporter: John Smith>
>>> Reporter.objects.get(full_name__startswith='John')
<Reporter: John Smith>
>>> Reporter.objects.get(full_name__contains='mith')
<Reporter: John Smith>
>>> Reporter.objects.get(id=2)
Traceback (most recent call last):
    ...
DoesNotExist: Reporter matching query does not exist.

# Create an article.
>>> from datetime import date
>>> a = Article(pub_date=date.today(), headline='Django is cool',
...     content='Yeah.', reporter=r)
>>> a.save()

# Now the article is in the database.
>>> Article.objects.all()
<QuerySet [<Article: Django is cool>]>

# Article objects get API access to related Reporter objects.
>>> r = a.reporter
>>> r.full_name
'John Smith'

# And vice versa: Reporter objects get API access to Article objects.
>>> r.article_set.all()
<QuerySet [<Article: Django is cool>]>

# The API follows relationships as far as you need, performing efficient
# JOINs for you behind the scenes.
# This finds all articles by a reporter whose name starts with "John".
>>> Article.objects.filter(reporter__full_name__startswith='John')
<QuerySet [<Article: Django is cool>]>

# Change an object by altering its attributes and calling save().
>>> r.full_name = 'Billy Goat'
>>> r.save()
```