МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЛЬВІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ВЕТЕРИНАРНОЇ МЕДИЦИНИ ТА БІОТЕХНОЛОГІЙ ІМЕНІ С.З. ҐЖИЦЬКОГО
Факультет механіки, енергетики та інформаційних технологій
Кафедра інформаційних технологій

ПОГОДЖЕНО
Гарант ОПП «Комп'ютерні науки»
Вадим ПТАШНИК
(ім'я та прізвище, підпис)
«27» серпня 2025 року

ЗАТВЕРДЖЕНО
Декан факультету механіки, енергетики та інформаційних технологій
Степан КОВАЛИШИН
(ім'я та прізвище, підпис)
«28» серпня 2025 року

РОБОЧА ПРОГРАМА НАВЧАЛЬНОЇ ДИСЦИПЛІНИ
«Об'єктно-орієнтоване програмування»
(код і назва навчальної дисципліни)

рівень вищої освіти перший (бакаларавський)
(назва освітнього рівня)
галузь знань 12 «Інформаційні технології»
(назва галузі знань)
спеціальність 122 «Комп'ютерні науки»
(назва спеціальності)
освітня програма «Комп'ютерні науки»
(назва)
вид дисципліни обов'язкова
(обов'язкова / за вибором)

2025–2026 навчальний рік

Робоча програма «Об'єктно-орієнтоване програмування»
(назва навчальної дисципліни)

Укладач: Татомир Андрій Володимирович, доцент, к.т.н., доцент кафедри інформаційних технологій
(вказати укладачів, їхні посади, наукові ступені та вчені звання)

Робочу програму схвалено на засіданні кафедри інформаційних технологій.

Протокол № 1 від 26.08.2025 року.

Завідувач кафедри _____ Тригуба А.М.
(підпис, ім'я та прізвище)

Погоджено навчально-методичною комісією спеціальності
«Комп'ютерні науки»
(назва спеціальності)

Протокол № 8/1 від «27.08.2025року»
Голова НМКС _____ Пташник В.В.
(підпис, ім'я та прізвище)

Схвалено рішенням навчально-методичної ради факультету ФМЕІТ
(назва факультету)

Протокол № 1 від «28.08.2025 року»
Голова НМРФ _____ Ковалишин С.Й.
(підпис, ім'я та прізвище)

Ухвалено вченою радою ФМЕІТ протокол №1 від «28.08.2025» р.

Опис навчальної дисципліни

Найменування показників	Всього годин	
	денна форма здобуття освіти	заочна форма здобуття освіти
Семестр	**4-5**	**4-5**
Кількість кредитів/годин	8/240	8/240
Усього годин аудиторної роботи	90	46
в т.ч.:		
● лекційні заняття, год.	30	22
● практичні заняття, год.	-	-
● лабораторні заняття, год.	60	24
● семінарські заняття, год.	–	–
Усього годин самостійної роботи	150	194
Форма контролю	залік, іспит, курсова	залік, іспит, курсова

Примітка.
Частка аудиторного навчального часу студента у відсотковому вимірі:
для денної форми навчання – 37,5%;
для заочної форми навчання – 19,1%.

1. Мета та завдання навчальної дисципліни

Метою вивчення освітньої компоненти «*Об'єктно-орієнтоване програмування*» є теоретична та практична підготовка здобувачів вищої освіти у напрямку формування комплексних знань і практичних навичок щодо принципів, методів і засобів створення, розгортання, підтримки сайтів та сучасних веб-орієнтованих інформаційних систем і застосунків. Навчальна дисципліна спрямована на опанування основних підходів та стандартів у розробці програмного, процесі навчання вони знайомляться з популярної об’єктно-орієнтованою мовою програмування Python, отримують знання про принципи об’єктно-орієнтованого програмування та вчаться організовувати програмний код відповідно до існуючих виробничих стандартів.

Завдання навчальної дисципліни передбачають:

1. Засвоєння основних концепцій об’єктно-орієнтованого програмування (класи, об’єкти, інкапсуляція, наслідування, поліморфізм).
2. Розробка програмного коду мовою Python із застосуванням принципів ООП для розв’язання прикладних завдань.
3. Ознайомлення з галузевими стандартами організації коду та підходами до розробки веб-застосунків і інформаційних систем.
4. Навчання основам роботи з системами контролю версій (наприклад, Git) та інтеграції індивідуальних результатів у спільні проєкти.
5. Вміння аналізувати вимоги, проєктувати програмні рішення та моделювати об’єктно-орієнтовані структури.
6. Ознайомлення з шаблонами проєктування, принципами SOLID та їхнім застосуванням у реальних проєктах.
7. Виконання практичних завдань і курсових проєктів, що передбачають створення веб-орієнтованих застосунків чи інформаційних систем.

Преквізити: освітня компонента «***Об'єктно-орієнтоване програмування***» є обов'язковою компонентою, яка забезпечує професійну підготовку здобувачів освітньо-професійної програми «Комп'ютерні науки» першого (бакалаврського) рівня вищої освіти. Вивчення дисципліни передбачає наявність систематичних та ґрунтовних знань із суміжних курсів першого (бакалаврського) рівня вищої освіти – «Основи інформаційних систем», «Алгоритми та структури даних», «Програмування».

Постреквізити: вивчення дисципліни «***Об'єктно-орієнтоване програмування***» створює підґрунтя для опанування наступних компонент освітньої програми, зокрема «Технологія розробки програмного забезпечення», «Моделювання систем», «Клієнт-серверне програмування», «Управління ІТ-проектами». Отримані знання та компетентності особливо важливі під час проходження практики, виконання тематичних курсових робіт та кваліфікаційної роботи, що сприяє формуванню професійних умінь у сфері розробки програмного забезпечення.

Результати навчання: У результаті вивчення дисципліни «Об'єктно-орієнтоване програмування (Python)» здобувачі розуміють основні принципи ООП — інкапсуляцію, наслідування, поліморфізм та абстракцію, уміють застосовувати їх для проєктування та реалізації програмних систем мовою Python. Вони набувають практичних навичок створення класів, об'єктів, методів, організації коду у вигляді модулів і пакетів, роботи з файлами та базами даних, використання бібліотек для серіалізації даних і взаємодії з API. Здобувачі здатні проектувати програмні рішення із застосуванням UML-діаграм, принципів SOLID та шаблонів проєктування, а також проводити тестування програмного забезпечення з використанням інструментів unit-тестування. Вони володіють навичками роботи з системами контролю версій, дотримуються стандартів написання коду, забезпечують якість, масштабованість і супровідність програмних продуктів. Завдяки цьому здобувачі готові до подальшого професійного розвитку та здатні застосовувати об'єктно-орієнтований підхід у різних сферах програмування.

Відповідно до освітньо-професійної програми «Комп'ютерні науки» вивчення навчальної дисципліни «*Об'єктно-орієнтоване програмування*» забезпечує набуття здобувачами таких компетентностей та програмних результатів навчання:

Інтегральна компетентність	Здатність розв'язувати складні спеціалізовані задачі та практичні проблеми у галузі комп'ютерних наук або у процесі навчання, що передбачає застосування теорій та методів інформаційних технологій і характеризується комплексністю та невизначеністю умов
Загальні компетентності	ЗК2. Здатність застосовувати знання у практичних ситуаціях.
Фахові (спеціальні) компетентності	СК3. Здатність до логічного мислення, побудови логічних висновків, використання формальних мов і моделей алгоритмічних обчислень, проектування, розроблення й аналізу алгоритмів, оцінювання їх ефективності та складності, розв'язності та нерозв'язності алгоритмічних проблем для адекватного моделювання предметних областей і створення програмних та інформаційних систем. СК8. Здатність проектувати та розробляти програмне забезпечення із застосуванням різних парадигм програмування: узагальненого, об'єктно-орієнтованого, функціонального, логічного, з відповідними моделями, методами й алгоритмами обчислень, структурами даних і механізмами управління.

Програмні результати навчання	ПРН5. Проектувати, розробляти та аналізувати алгоритми розв'язання обчислювальних та логічних задач, оцінювати ефективність та складність алгоритмів на основі застосування формальних моделей алгоритмів та обчислюваних функцій. ПРН9. Розробляти програмні моделі предметних середовищ, вибирати парадигму програмування з позицій зручності та якості застосування для реалізації методів та алгоритмів розв'язання задач в галузі комп'ютерних наук. ПРН14. Застосовувати знання методології та CASE-засобів проектування складних систем, методів структурного аналізу систем, об'єктно- орієнтованої методології проектування при розробці і дослідженні функціональних моделей організаційно-економічних і виробничо- технічних систем.

2. **Структура навчальної дисципліни**

Назви тем	Кількість годин											
	денна форма						заочна форма					
	усього	у тому числі					усього	у тому числі				
		л	п	лаб.	інд.	с. р.		л	п	лаб.	інд.	с.р.
1	2	3	4	5	6	7	8	9	10	11	12	13
	Рік підготовки 2 Семестр 4						Рік підготовки 2 Семестр 4					
Розділ 1. Об'єктно-орієнтоване програмування (частина 1)												
Тема 1.	9	2	-	2	-	5	9	2	-	-	-	7
Тема 2.	9	2	-	2	-	5	9	1	-	-	-	8
Тема 3.	13	2	-	6	-	5	13	1	-	2	-	10
Тема 4.	11	2	-	4	-	5	11	1	-	2	-	8
Тема 5.	11	2	-	4	-	5	11	1	-	2	-	8
Тема 6.	11	2		4		5	11	1	-	2		8
Тема 7.	13	2	-	5	-	6	13	2	-	2	-	9
Тема 8.	13	2	-	5	-	6	13	1	-	-	-	12
Разом за семестр I	90	16	-	32		42	90	10	-	10	-	70
	Рік підготовки 3 Семестр 5						Рік підготовки 3 Семестр 5					
Розділ 2. Об'єктно-орієнтоване програмування (частина 2)												
Тема 9.	16	2	-	8	-	6	16	2	-	2	-	12
Тема 10.	10	2	-	2	-	6	10	1	-	2	-	7
Тема 11.	10	2	-	2	-	6	10	2	-	2	-	6
Тема 12.	12	2	-	4	-	6	12	1	-	2	-	9
Тема 13.	12	2	-	4	-	6	12	2	-	2	-	8
Тема 14.	12	2	-	4	-	6	12	1	-	1	-	10
Тема 15.	9	1	-	2	-	6	9	1	-	2	-	6
Тема 16.	9	1		2		6	9	2		1		6
Іспит	30	-	-	-	-	30	30	-	-	-	-	30
Разом за	120	14	-	28	-	78	120	12	-	14	-	94

семестр II												
К.Р.	30	-	-	-	-	30	30	-	-	-	-	30
Усього годин	240	30	-	60	-	150	240	22	-	24	-	194

3. ЛЕКЦІЙНІ ЗАНЯТТЯ

№ з/п	Назви тем та їх короткий зміст	Кількість годин	
		ДФЗО	ЗФЗО
1	**Вступ** 1. Мета, структура і предмет курсу. 2. Парадигми і мови програмування. 3. Передумови виникнення ООП. Історичний процес розвитку підходів до програмування. Структуроване програмування. Процедурне програмування. Складність коду. Концепція об'єктно-орієнтованого програмування. 4. Переваги та недоліки ООП. Альтернативи ООП. Продовження і варіанти реалізації ООП у різних мовах	2	2
2	**Системи контролю версій програмного забезпечення** 1. Концепція версійности програмного коду. Відслідковування змін в коді. Пошук несправностей програмного забезпечення. 2. Система контролю версій Git. Основні команди та функціонал. 3. Розширене використання систем контролю версій. Робота з віддаленими репозиторіями. Колаборація.	2	1
3	**Загальний огляд об'єктно-орієнтованої мови програмування Python. Запуск і робота з Python** 1. Налаштування робочого середовища Python 3 для систем на базі Unix. Особливості роботи у Windows. Редактори коду та інтегровані середовища розробки. 2. Поняття компіляції та интерпретації. Динамічна типізація. 3. Змінні та оператори. 4. Базові та комплексні типи даних. Вбудовані типи даних у Python 3. Списки (Lists) та словники (Dictionaries). Розширення типів. Формат даних JSON. 5. Базові оператори для роботи з простими типами даних (математичні та логічні операції, робота зі стрічками). Приведення типів.	2	1
4	**Структурне і процедурне програмування**	2	1

	1. Складні алгоритми та програми. Історичні витоки появи структурного та процедурного програмування. 2. Конструкції для підтримки структурованого програмування. Лінійне виконання. Оператори розгалуження. Цикли в Python 3. Переваги та недоліки структурованого програмування. 3. Процедурне програмування. Функції та процедури. Оголошення та виконання функцій. Аргументи і параметри функцій. Рекурсія.		
5	**Принцип інкапсуляції даних** 1. Класифікація об'єктів. Стан об'єкта. Створення літералів об'єктів. 2. Взаємозв'язки між класами і об'єктами. Класи як типи даних. 3. Атрибути (властивості, поля) об'єктів. Змінні класу. 4. Оголошення методів. Створення об'єкта (class instance) за допомогою класу (конструктори і деструктори). Звертання класу до об'єкта через параметр "self". Методи об'єкта. Методи класу (параметр "cls"). Статичні методи. 5. Безпечне звертання до атрибутів. Концепція публічних, приватних та захищених полів об'єктів.	2	1
6	**Наслідування в ООП** 1. Взаємозв'язки між класами. Оголошення наслідування. 2. Перевикористання властивостей та методів (resolution order). Глобальний (вбудований) object. 3. Пряме та непряме наслідування. Функція super. 4. Множинне успадкування в Python. 5. Розуміння наслідування (функції help, isinstance, issubclass). 6. Формальне представлення наслідування. UML-діаграми класів. 7. Особливості прототипно-орієнтованих мов програмування.	2	1
7	**Поліморфізм** 1. Перевантаження методів. Способи організації поліморфізму. 2. "Магічні методи" в Python 3 та їх використання. Основні вбудовані оператори, що допускають перевизначення.	2	2

	3. Декоратори. Getter’и, setter’и та deleter’и.		
8	**Модулі та пакети** 1. Базове ядро Python 3. 2. Розширення функціональності за рахунок перевикористання стороннього коду. Імпорт пакетів. Роботи з PIP. Open source software. 3. Написання та підключення власних модулів.	2	1
Розділ 2. Об'єктно-орієнтоване програмування (частина 2)			
9	**Шаблони (патерни) проєктування програмного забезпечення** 1. Поняття шаблону проєктування. Передумови та історія появи. 2. Основні групи шаблонів (твірні, структурні, поведінкові). 3. Використання шаблонів у виробництві.	2	2
10	**Твірні шаблони проєктування** 1. Основні твірні шаблони. 2. Розгляд базового прикладу.	2	1
11	**Структурні шаблони проєктування.** 1. Основні структурні шаблони. 2. Розгляд базового прикладу.	2	2
12	**Поведінкові шаблони проєктування** 1. Основні поведінкові шаблони. 2. Розгляд базового прикладу.	2	1
13	**Тема 13. Принципи проєктування програмного забезпечення.** 1. Практичні та теоретичні принципи проєктування. DRY, AHA, KISS тощо. 2. Комплекс принципів проєктування SOLID. Принцип єдиної відповідальності. Принцип відкритості/закритості. Принцип підстановки Лісков. Принцип розділення інтерфейсу. Принцип інверсії залежностей.	2	2
14	**Якість коду. Рефакторинг** 1. Поняття якості коду. Складові якості коду (читабельність, розширюваність, підтримуваність, testability тощо). “Чистий” та “брудний” код. ”Code smells”. Досвід розробників програмного забезпечення (developer experience). 2. Статична перевірка коду.	2	1

	3. Тестування програмного забезпечення. Види тестування. Автоматичне тестування. Unit, integration та E2E тестування. 4. Процес рефакторингу. Тестування в процесі рефакторингу. Використання систем контролю версій. Техніки рефакторингу (перейменування, переміщення, спрощення або переписування тощо).		
15	**Переваги декларативного стилю програмування. Загальні відомості про функціональне програмування** 1. Відмінності між імперативним та декларативним стилем написання програм. Когнітивна складність коду. 2. Загальні відомості про лямбда-числення. 3. Основні положення функціонального програмування. 4. Відмінності між ООП та ФП. Імутабельність. Композиція замість наслідування. 5. Використання фукціонального програмування в Python 3. Лямбда-функції.	1	1
16	**Застосування ООП та інших парадигм програмування в реальному виробництві. Підведення підсумків** 1. Сучасні підходи до розробки програмного забезпечення. 2. Мультипарадигменна розробка.	1	2
Усього годин		30	22

4. ЛАБОРАТОРНІ ЗАНЯТТЯ

№ з/п	Назви тем та їх короткий зміст	Кількість годин	
		ДФЗО	ЗФЗО
Розділ 1. Об'єктно-орієнтоване програмування (частина 1)			
1	Вступне заняття. Налаштування середовища розробки Python3 і запуск програм	2	-
2	Вивчення простих типів даних і методів роботи з ними у Python 3	2	-
3	Основи структурного програмування в Python 3	6	2
4	Основи процедурного програмування в Python 3	4	2

5	Створення та використання класів	4	2
6	Змінні класу та об’єкта	4	2
7	Використання методів класу і статичних методів	5	2
8	Наслідування в об’єктно-орієнтованому програмуванні	5	-
Розділ 2. Об'єктно-орієнтоване програмування (частина 2)			
9	Поліморфізм в Python 3	8	2
10	Використання декораторів методів	2	2
11	Твірні шаблони проєктування	2	2
12	Структурні шаблони проєктування	4	2
13	Поведінкові шаблони проєктування	4	2
14	Принципи проєктування програмного забезпечення	4	1
15	Рефакторинг програмного забезпечення	2	2
16	Застосування ООП та інших парадигм програмування в реальному виробництві.	2	1
Усього годин		60	24

5. САМОСТІЙНА РОБОТА

№ з/п	Назви тем та їх короткий зміст	Кількість годин	
		ДФЗО	ЗФЗО
1	Вступне заняття. Налаштування середовища розробки Python3 і запуск програм	5	7
2	Вивчення простих типів даних і методів роботи з ними у Python 3	5	8
3	Основи структурного програмування в Python 3	5	10
4	Основи процедурного програмування в Python 3	5	8
5	Створення та використання класів	5	8
6	Змінні класу та об’єкта	5	8
7	Використання методів класу і статичних методів	6	9
8	Наслідування в об’єктно-орієнтованому програмуванні	6	12
Розділ 2. Об'єктно-орієнтоване програмування (частина 2)			
9	Поліморфізм в Python 3	6	12
10	Використання декораторів методів	6	7
11	Твірні шаблони проєктування	6	6
12	Структурні шаблони проєктування	6	9
13	Поведінкові шаблони проєктування	6	8
14	Принципи проєктування програмного забезпечення	6	10
15	Рефакторинг програмного забезпечення	6	6
16	Застосування ООП та інших парадигм програмування в реальному виробництві.	6	6
Підготовка до навчальних занять та контрольних заходів		30	30
К.Р.		30	30
Усього годин		150	194

Розподіл балів за виконання курсової роботи

Пояснювальна записка	Вибір методів та інструментів розробки для поставлених завдань	Інтерпретація отриманих результатів та формулювання обґрунтованих висновків	Презентація результатів дослідження та відповіді на запитання	Сума
до 20 балів	до 25 балів	до 35 балів	до 20 балів	100

6. ІНДИВІДУАЛЬНІ ЗАВДАННЯ

Мета курсової роботи полягає у закріпленні та розширенні теоретичних знань, отриманих під час вивчення дисципліни «Об'єктно-орієнтоване програмування», розвитку навичок самостійного проектування та розробки програм; оволодінні методами та інструментами розробки програмного забезпечення, вмінні застосовувати отримані знання для вирішення конкретних практичних завдань та розвитку вміння оформлення науково-дослідницької роботи.
Завдання курсової роботи з дисципліни «Об'єктно-орієнтоване програмування» включають:

- вибір та обґрунтування теми дослідження;
- огляд та аналіз літератури з обраної теми;
- формулювання мети та завдань дослідження;
- вибір та застосування інструментів розробки для вирішення поставлених завдань;
- отримання та інтерпретація результатів дослідження;
- формулювання висновків та рекомендацій;
- оформлення курсової роботи відповідно до встановлених вимог.

7. МЕТОДИ НАВЧАННЯ

Навчання з дисципліни «Об'єктно-орієнтоване програмування» здійснюється із застосуванням сучасних інтерактивних та практикоорієнтованих методів, що поєднують словесні (лекція, пояснення, дискусія), наочні (демонстрація, робота з мультимедійними матеріалами) та активні форми (групові проєкти, семінари-дискусії, моделювання ситуацій, аналіз кейсів). Використання методів мозкового штурму, проблемно-орієнтованих і дослідницьких підходів сприяє розвитку критичного та креативного мислення, уміння працювати в команді й приймати ефективні управлінські рішення. Ефективність забезпечується залученням сучасних цифрових інструментів, програмних засобів для планування й контролю, а також роботи з професійною літературою та науковими публікаціями.

8. МЕТОДИ КОНТРОЛЮ

Оцінювання результатів навчання студентів здійснюється проведенням поточного та підсумкового контролю.

Поточний контроль здійснюється під час практичних занять і має на меті перевірку рівня підготовленості студента до виконання відповідних завдань. Форми проведення поточного контролю – усне та письмове опитування, тестовий контроль.

Підсумковий контроль проводиться з метою оцінювання результатів навчання на завершальному етапі вивчення дисципліни. Підсумковий контроль здійснюється у формі екзамену.

9. КРИТЕРІЇ ОЦІНЮВАННЯ РЕЗУЛЬТАТІВ НАВЧАННЯ ЗДОБУВАЧІВ ВИЩОЇ ОСВІТИ

У 4 семестрі успішність студента оцінюється шляхом проведення поточного контролю.

Максимальна кількість балів з дисципліни «Об'єктно-орієнтоване програмування», яку може отримати студент протягом семестру за всі види роботи за результатами поточного оцінювання становить 100. Результати **поточного контролю** оцінюються за чотирибальною («2», «3», «4», «5») шкалою. В кінці семестру обчислюється середнє арифметичне значення (САЗ) усіх отриманих студентом оцінок з наступним переведенням його у сто бальну шкалу за формулою: **ПК = 20•САЗ.**

Поточний контроль здобувачів освіти заочної форми навчання під час настановчої та лабораторно-екзаменаційної сесії оцінюється максимум у 30 балів. Ще 70 балів відведено на виконання тематичної самостійної роботи (ТСР) у міжсесійний період за програмою курсу. Підсумкова оцінка розраховується за формулою: **ПК+ТСР**.

В залікову відомість студентові у графі «за національною шкалою» виставляється оцінка «зараховано/незараховано» відповідно до таблиці 1. Присутність студента при виставленні підсумкової оцінки необов’язкова, якщо ним виконані усі передбачені види робіт.

У 5 семестрі успішність студента оцінюється шляхом проведення поточного та підсумкового контролю.

Максимальна кількість балів з дисципліни «Об'єктно-орієнтоване програмування», яку може отримати студент протягом семестру за всі види роботи, становить 100, при цьому 50 балів за результатами поточного оцінювання, та 50 – за результатами екзаменаційного контролю.

Результати **поточного контролю** оцінюються за чотирибальною («2», «3», «4», «5») шкалою. В кінці семестру обчислюється середнє арифметичне значення (САЗ) усіх отриманих студентом оцінок з наступним переведенням його у 50-ти бальну шкалу за формулою:

ПК = 10•САЗ

Критерії поточного оцінювання знань студентів

Оцінка	Критерії оцінювання
5 («відмінно»)	У повному обсязі володіє навчальним матеріалом, вільно, самостійно та аргументовано його викладає, глибоко і всебічно розкриває зміст, використовуючи обов’язкову та додаткову літературу. Правильно вирішив 90% тестових завдань.
4 («добре»)	Достатньо повно володіє навчальним матеріалом, обґрунтовано його викладає, в основному розкриває зміст завдань, використовуючи обов’язкову літературу. При викладанні окремих питань не вистачає достатньої глибини та аргументації, допускаються несуттєві неточності й незначні помилки. Правильно вирішив більшість тестових завдань.
3 («задовільно»)	У цілому володіє навчальним матеріалом, викладає його основний зміст, але без глибокого всебічного аналізу, обґрунтування та аргументації, допускаючи окремі суттєві неточності та помилки. Правильно вирішив близько половини тестових завдань.
2 («незадовільно»)	Не в повному обсязі володіє навчальним матеріалом. Викладає матеріал фрагментарно та поверхово, без аргументації й обґрунтування, недостатньо розкриває зміст теоретичних і практичних завдань, допускає суттєві неточності. Правильно вирішив меншість тестових завдань.

Переведення підсумкових рейтингових оцінок з дисципліни, виражених у балах за 100-бальною шкалою, у оцінки за національною шкалою та шкалою ECTS

Таблиця 1 – **Шкала оцінювання: національна та ECTS**

Сума балів за всі види навчальної діяльності	Оцінка ECTS	Оцінка за національною шкалою	
		для екзамену, диференційованого заліку, курсового проєкту (роботи), практики	для заліку
90–100	**A**	відмінно	зараховано
82–89	**B**	добре	
74–81	**C**		
64–73	**D**	задовільно	
60–63	**E**		
35–59	**FX**	незадовільно з можливістю повторного складання	не зараховано з можливістю повторного складання
0–34	**F**	незадовільно з обов'язковим повторним вивченням дисципліни	не зараховано з обов'язковим повторним вивченням дисципліни

Максимальна кількість балів за курсову роботу (проект) становить 100. Критерії оцінювання курсових робіт (проектів) представлено у таблиці 1.

Захист курсових робіт (проектів) здійснюється перед комісією у складі 2-3 викладачів кафедри, у тому числі - керівника курсової роботи (проекту).

10. МЕТОДИЧНЕ ЗАБЕЗПЕЧЕННЯ

Підручники і навчальні посібники; інструктивно-методичні матеріали до практичних занять; індивідуальні навчально-дослідні завдання; контрольні роботи; текстові та електронні варіанти тестів для поточного контролю, методичні матеріали для організації самостійної роботи студентів, виконання індивідуальних завдань.

11. РЕКОМЕНДОВАНА ЛІТЕРАТУРА

Базова

1. Креневич А.П. Python у прикладах і задачах. Частина 2. Об'єктно-орієнтоване програмування. Навчальний посібник – К.: ВПЦ "Київський Університет", 2020. – 152 с.
2. Креневич А.П., 2020 рік 2. Креневич А.П. Методичні вказівки до лабораторних занять із дисципліни "Об'єктно-орієнтоване програмування" для студентів механіко-математичного факультету – К.: ВПЦ "Київський Університет", 2019. с. 33.
3. The Python Tutorial [Електронний ресурс] – Режим доступу до ресурсу: https:// docs. python. org /3/ tutorial/ index .html .
4. Chun, Wesley. Core python applications programming / Wesley J. Chun. — 3rd ed. Pearson Education, Inc., 2012. – 852p.
5. Jason R. Briggs. Python for kids. A Playful Introduction to Programming - No Starch Press, Inc., San Francisco, CA 2013. – 318p

Допоміжна

6. M. Lutz: Learning Python: Powerful Object-Oriented Programming, 5th ed. // O'Reilly Media, Inc., 2013.
7. D. Beazley, B.K. Jones: Python Cookbook: Recipes for Mastering Python 3, 3rd ed. // O'Reilly Media, Inc., 2013.
8. A. Martelli, A. Ravenscro, S. Holden: Python in a Nutshell: The Definitive Reference, 3rd ed. // O'Reilly Media, Inc., 2017.
9. B. Lubanovic: Introducing Python: Modern Computing in Simple Packages. // O'Reilly Media, Inc., 2015.
10. J. Hunt: A Beginners Guide to Python 3 Programming. // Springer, 2019.
11. J. Hunt: Advanced Guide to Python 3 Programming. // Springer, 2019

12. ІНФОРМАЦІЙНІ РЕСУРСИ

1. Віртуальне навчальне середовище ЛНУВМБ - https://moodle.lnup.edu.ua/?redirect=0.
2. Бібліотечно-інформаційні ресурси - книжковий фонд, періодика та фонди на електронних носіях бібліотеки ЛНУВМБ, державних органів науково-технічної інформації, наукових, науково-технічних бібліотек та інших наукових бібліотек України.

- Бібліотека Національного університету "Львівська політехніка" - 79013, Львів, вул. С. Бандери, 74;
- Бібліотека Інституту аграрної економіки НАН України - 01127, м. Київ, вул. Героїв Оборони, 10;
- Бібліотека Інституту регіональних досліджень НАН України - 70026, Львів, вул. Козельницька, 4;
- Бібліотека Львівського інституту менеджменту - 79601, Львів, пр. Чорновола, 57;
- Бібліотека Львівської комерційної академії - 79034, Львів, вул. Туган-Барановського, 10;
- Бібліотека Національного університету біоресурсів і природокористування України - 01127, м. Київ, вул. Героїв Оборони, 15;
- Львівська наукова бібліотека імені В. Стефаника НАН України – м. Львів, вул. В.Стефаника,
- Національна бібліотека України імені В.І.Вернадського – м.Київ, пр. 50-річчя Жовтня, 4.

3. Електронні інформаційні ресурси мережі інтернет з переліком сайтів:

- http://www.cprogramming.com
- https://www.w3schools.com/default.asp
- https://developer.mozilla.org/en-US/docs/Web/HTML