Національний технічний університет України
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

Кафедра прикладної математики

ОБ'ЄКТНО-ОРІЄНТОВАНЕ ПРОГРАМУВАННЯ

Робоча програма навчальної дисципліни (Силабус)

Реквізити навчальної дисципліни	
Рівень вищої освіти	***Перший (бакалаврський)***
Галузь знань	*11 Математика та статистика*
Спеціальність	*113 Прикладна математика*
Освітня програма	*Наука про дані та математичне моделювання*
Статус дисципліни	*Нормативна*
Форма навчання	*очна(денна)*
Рік підготовки, семестр	*2 курс, осінній семестр*
Обсяг дисципліни	*4,5 кредити*
Семестровий контроль/ контрольні заходи	*Залік/Контрольні (4) та лабораторні (5) роботи*
Розклад занять	*Лекції — 1 раз на тиждень (18 занять),* *практичні — 1 раз на тиждень перші 4 тижні навчання (4 заняття),* *лабораторні — 1 раз на тиждень після 4 тижнів навчання (14 занять)*
Мова викладання	*Українська/Англійська*
Інформація про керівників курсу / викладачів	Лектор: *Борисенко Павло Борисович, pavlo.borysenko@gmail.com* Практичні: *Борисенко Павло Борисович, pavlo.borysenko@gmail.com;* *Щьоголєв Максим Олегович, shchoholiev.maksym@gmail.com*
Розміщення курсу	Канали *#oop-python-course* та *#oop-python-labs* у кафедральному Slack Лекції: *https://www.youtube.com/playlist?list=PLjBFEyMhtyMtbeOBSCvcZubWqZS4ExQmN*

Програма навчальної дисципліни

1. Опис навчальної дисципліни, її мета, предмет вивчення та результати навчання

Об'єктно-орієнтоване програмування (ООП) є однією з двох — поруч із функціональним програмуванням — домінуючих парадигм у сучасному підході до проектування програмного забезпечення. Розуміння принципів об'єктно-орієнтованого дизайну дозволяє проектувати стабільніші та легко підтримувані застосунки.

В рамках цього курсу ми розглянемо основні принципи ООП, що знадобляться будь-кому, хто хоче успішно пройти інтерв'ю у IT-компанію, а у рамках лабораторних робіт у групах розробимо з нуля повноцінний працюючий проект, який зможе стати частиною вашого майбутнього портфоліо чи початком власного пет-проекту.

Курс «Об'єктно-орієнтоване програмування» є базовим курсом підготовки спеціалістів у сфері науки про дані та математичного моделювання.

Метою курсу є:

- *вивчення основних понять і концепцій об'єктно-орієнтованого програмування,*
- *конструкцій та синтаксису мови Python пов'язаних із ООП,*
- *принципів та підходів до об'єктно-орієнтованого дизайну програмного забезпечення.*

Предметом вивчення є об'єкти і класи, програмний код мовою Python та його синтаксичні конструкції для їх реалізації; патерни програмування.

Після засвоєння дисципліни студенти матимуть наступні:

- *компетентності:*
  - *здатність застосовувати знання у практичних ситуаціях;*
  - *здатність генерувати нові ідеї;*
  - *здатність до абстрактного мислення, аналізу та синтезу;*
  - *знання та розуміння предметної області та розуміння професійної діяльності;*
  - *здатність розробляти алгоритми та структури даних, програмні засоби та програмну документацію;*
  - *здатність проектувати бази даних, інформаційні системи і ресурси;*
  - *здатність використовувати сучасні технології програмування та тестування програмного забезпечення;*
  - *здатність зрозуміти постановку завдання, сформульовану мовою певної предметної галузі, здійснювати пошук та збір необхідних вихідних даних;*
  - *здатність до організації роботи колективу виконавців, приймання доцільних та економічно обґрунтованих організаційних та управлінських рішень;*
- *знання:*
  - *принципів ООП;*
  - *принципів SOLID;*
  - *понять про об'єкти та класи, їх властивості, структуру та взаємозв'язки;*
  - *підходів до ефективної роботи у команді;*
  - *базових конструкцій UML;*
  - *підходів до об'єктної декомпозиції предметної області;*
  - *мотивації та структури патернів проектування;*
  - *виражальних засобів мови Python для реалізації ООП;*
  - *підходів до побудови «чистого» коду;*
- *уміння:*
  - *виконувати об'єктно-орієнтовану декомпозицію предметної області;*
  - *будувати UML-діаграми класів;*
  - *аналізувати діаграми класів для визначення проблем чи порушення принципів SOLID;*
  - *писати об'єктно-орієнтований код мовою Python;*
  - *ідентифікувати та використовувати патерни проектування;*
- *навички:*
  - *роботи із засобами графічного представлення діаграм класів;*
  - *написання об'єктно-орієнтованого коду мовою Python;*
  - *декомпозиції предметної області на об'єкти;*
  - *використання систем контролю версій для колективної розробки;*
  - *написання закінченого програмного продукту;*
- *досвід:*
  - *роботи у команді;*
  - *роботи із UML-діаграмами класів;*
  - *об'єктно-орієнтованого програмування мовою Python;*
  - *реалізації завершеного проекту від ідеї до програми.*

2. Пререквізити та постреквізити дисципліни (місце в структурно-логічній схемі навчання за відповідною освітньою програмою)

Дисципліна вивчається у осінньому семестрі 2 курсу та базується на результатах навчання з дисциплін:

- *Алгоритми і структури даних*

- *Програмування*
- *Програмування мовою Python*

Успішне проходження курсу дозволить студентам продовжити вивчення наступних дисциплін з навчального плану підготовки бакалаврів:

- *Автоматизоване тестування програмного забезпечення*
- *Бази даних та інформаційні системи*
- *Функційне програмування*
- *Структура та інтерпретація комп'ютерних програм*
- *Front-end розробка*
- *Побудова REST-сервісів*

та дисциплін з навчального плану підготовки магістрів:

- *Інтелектуальний аналіз даних*
- *Машинне навчання*

3. Зміст навчальної дисципліни

Розділ 1. Об'єктно-орієнтований дизайн.

Тема 1.1. Об'єкти та їх реалізація.

Тема 1.2. Характеристики об'єктів. Інкапсуляція.

Тема 1.3. Зв'язки між об'єктами. Поліморфізм.

Тема 1.4. Наслідування та інтерфейси.

Тема 1.5. Принципи SOLID. UML.

Розділ 2. Об'єктно-орієнтоване програмування у Python.

Тема 2.1. Особливості ООП у Python.

Тема 2.2. Представлення об'єктів у пам'яті.

Тема 2.3. Мета-класи і рефлексія.

Розділ 3. Спеціальні теми ООП.

Тема 3.1. Генерики і шаблони. Типізація у ООП.

Тема 3.2. Зв'язок ООП та функціонального програмування. Функції як об'єкти.

Тема 3.3. Управління пам'яттю у мовах з garbage collection.

Розділ 4. Патерни проектування.

Тема 4.1. Абстрактні структури та архітектура програм.

Тема 4.2. Генеративні патерни.

Тема 4.3. Поведінкові патерни.

Тема 4.4. Структурні патерни.

4. Навчальні матеріали та ресурси

Підручники

1. *Чертов О.Р., Объектно-ориентированное программирование, 2012.*
   Частина 1 — програма мінімум для того, аби опанувати дисципліну.
   Електронний примірник доступний на кафедральних ресурсах.
2. *Dusty Phillips, Python 3 Object-Oriented Programming, 3 ed., 2018.*
   Розділи 1-3, 5, 8-10. Вибрані частини розділу 7.
3. *Будай А., Дизайн-патерни — просто як двері, 2016.*
   Це наше основне джерело для вивчення патернів проектування.

Посилання на онлайн-версію: https://sites.google.com/site/designpatternseasy/download

Додаткові матеріали

1. *Документація Python*
   Посилання: https://docs.python.org/
2. *Zed Shaw, Learn Python 3 the Hard Way, 2017.*
   Розділ 40 і далі.
3. *Erich Gamma et al., Design Patterns: Elements of Reusable Object-Oriented Software, 1994.*
   Це класична книга по патернам проектування, в якій вони вперше були описані. Вона доволі тяжко читається як першоджерело, краще користуватися нею як довідником.

Рекомендовані онлайн-курси

1. *Coursera. Python Classes and Inheritance*
   https://www.coursera.org/learn/python-classes-inheritance
2. *Coursera. ООП и паттерны проектирования в Python*
   https://www.coursera.org/learn/oop-patterns-python

Навчальний контент

5. Методика опанування навчальної дисципліни (освітнього компонента)

Лекції

Розділ 1. Об'єктно-орієнтований дизайн

Лекція 1. Парадигми програмування. Вступ до об'єктно-орієнтованого програмування. Об'єкти та їх реалізації.

Лекція 2. Принципи побудови програми в ООП. Характеристики об'єкту. Класи. Інкапсуляція та рівні доступу до даних і методів.

Лекція 3. Типи зв'язків між об'єктами: асоціація, агрегація, наслідування. Метакласи і шаблони, генерики. Поліморфізм.

Лекція 4. Наслідування. Множинне наслідування і проблема ромба. Інтерфейси.

Лекція 5. Принципи SOLID. UML-діаграми класів.

Розділ 2. Об'єктно-орієнтоване програмування у Python

Лекція 6. ООП у Python. Виконання програм у віртуальній машині. Особливості реалізації інкапсуляції та поліморфізму. Приклад простого класу.

Лекція 7. Особливості реалізації наслідування. Представлення класів у пам'яті. Множинне наслідування у Python. Інтерфейси та абстрактні класи.

Лекція 8. Метакласи у Python. ООП засноване на прототипах. Рефлексія. Динамічне конструювання типу.

Лекція 9. Метакласи у Python. Реалізація власного метакласу. Порядок пошуку методів. Проблеми наслідування у метакласованих структурах. Випадки застосування метакласів. Метаклас ABCMeta.

Розділ 3. Спеціальні теми ООП

Лекція 10. Генерики та динамічна типізація. Відмінність реалізації генериків і шаблонів. Проблеми наслідування у генериків. Вбудовані генерики.

Лекція 11. Елементи функціонального програмування. Функції як об'єкти. Лямбда-функції у Python.

Лекція 12. Управління пам'яттю у мовах з віртуальними машинами. Garbage collector. Детерміноване звільнення ресурсів. Try-with-resources та контекстні менеджери.

Розділ 4. Патерни проектування

Лекція 13. Патерни проектування. Чому раціонально використовувати патерни? Абстрактні структури та архітектура програм. Типи патернів. Генеративні патерни. Синглтон. Фабричний метод. Абстрактна фабрика.

Лекція 14. Поведінкові патерни. Обсервер. Команда. Стратегія. Структурні патерни. Декоратор. Адаптер. Фасад.

Практики

Розділ 1. Об'єктно-орієнтований дизайн

Практика 1. Основи об'єктно-орієнтованого дизайну. Розбиття предметної області на об'єкти. Виділення об'єктів та їх властивостей. Встановлення зв'язків між об'єктами. Зв'язки як характеристики.

Практичне завдання (колективне): вибрати предметну область, виділити в ній головні об'єкти, визначити зв'язки між ними.

Практика 2. Виділення класів на основі об'єктного розбиття. Графічне зображення структури класів, UML-діаграми. Прояви інкапсуляції. Представлення характеристик об'єкта як полів та зв'язків. Рівні доступу до даних. Типи зв'язків.

Практичне завдання (колективне): на основі результатів попередньої практичної роботи побудувати UML-діаграму класів.

Практика 3. Типові помилки в об'єктно-орієнтованому дизайні. Гранулярність і загальність класів. Мінімальні вимоги до класів. God-класи. Прояви наслідування. Абстрактні класи і узагальнюючі об'єкти. Множинне наслідування і проблема ромба.

Практичне завдання (колективне): виявити проблеми в дизайні, розробленому на попередній практичній роботі. Запропонувати вирішення проблеми ромба.

Практика 4. Вирішення проблеми ромба. Інтерфейси. Заміна наслідування на агрегацію. Проблеми агрегації. Дизайн за контрактом. Застосування принципів SOLID.

Практичне завдання (колективне): вирішити проблеми ромба у дизайні з попередньої практики заміною наслідування на агрегацію. Побудувати інтерфейси для системи класів. Виявити порушення принципів SOLID.

Лабораторна 0. Обговорення та затвердження тем навчальних проектів.

Завдання: вибрати тему проекту та побудувати UML-діаграму класів.

Розділ 2. Об'єктно-орієнтоване програмування у Python.

Лабораторна 1. Структура класів та екранні форми.

Завдання: реалізувати принаймні кілька класів, що демонструють розмежування доступу до даних, геттери і сеттери, функціональне призначення відповідних об'єктів. Підготувати ескізи чи макети користувацького інтерфейсу.

Лабораторна 2. Наслідування.

Завдання: Реалізувати схему наслідування між класами, абстрактні класи. Реалізувати множинне наслідування. Реалізувати зв'язок між класами на основі контракту (інтерфейсів).

Розділ 3. Спеціальні теми ООП.

Лабораторна 3. Виключні ситуації.

Завдання: визначити негативні шляхи руху користувача та можливі точки збою у рамках проекту. Реалізувати обробку виключних ситуацій.

Лабораторна 4. Інтерфейс користувача та обробка вводу/виводу.

Завдання: реалізувати інтерактивну роботу програми з користувачем — по суті, графічний інтерфейс. Організувати роботу з файловою системою, базою даних, мережею в рамках проекту.

Розділ 4. Патерни проектування

Лабораторна 5. Патерни проектування.

Завдання: ідентифікувати та реалізувати принаймні один з патернів проектування.

Захист проектів.

Завдання: Підготувати коротку доповідь та демонстрацію роботи проекту.

6. Самостійна робота студента

До самостійного опрацювання виносяться:

- *підготовка до аудиторних занять — до 2 годин на тиждень;*
- *підготовка до контрольних робіт та заліку — до 20 годин за семестр;*
- *виконання лабораторних робіт — до 50 годин;*
- *вивчення наступних тем:*
  - *Теоретичні засади UML.*
  - *Принципи колективної роботи з системою контролю версій (git flow).*
  - *Перевантаження операторів у Python.*
  - *Патерни з GoF, що не увійшли у лекції.*
  - *Архітектурні патерни (MVP, MVC, MVVM).*

Політика та контроль

7. Політика навчальної дисципліни

Відвідування

Відвідування лекцій необов'язкове, але ми заохочуємо студентів не пропускати лекційні заняття через можливість ставити уточнюючі питання та брати участь у живому обговоренні. Записи лекцій (та оглядової частини практичних занять) поточного чи минулих років будуть доступні онлайн.

Відвідування практичних необов'язкове, однак їх пропуск буде сильно ускладнювати продуктивну роботу на наступних практичних заняттях.

Виконання і захист лабораторних робіт обов'язкове. Активність на лабораторних заняттях становить 50% семестрового рейтингу.

Пропущені контрольні роботи можна перездати за погодженням із викладачем.

Оцінювання

Оцінювання контрольних та лабораторних робіт відбувається з точністю до десятих, округлення за звичними правилами.

Дедлайни

Контрольні роботи мають бути здані у рамках часу, відведеного на їх проведення.

Лабораторні роботи мають окремі визначені терміни (дедлайни). Роботи, здані після цих термінів, будуть оцінюватися з модифікатором:

- *здані після* **софт дедлайну** *(окремі для кожної роботи, але не раніше ніж за 2 тижні після отримання завдання) — 0,5, тобто отримають половину балів;*

- *здані після* **хард дедлайну** *(тиждень до заліку) — не отримають балів.*

Фінальну презентацію проекту потрібно провести до залікового заняття.

Додаткові бали

Активність на лекціях та лабораторних заняттях — відповіді на запитання викладача, знаходження помилок у лекційних чи лабораторних матеріалах; питання, що свідчать про вдумливу роботу з навчальним матеріалом, надання оригінальних рішень у лабораторних завданнях тощо — заохочується додатковими балами на розсуд викладача.

Крім того, заохочується додатковими балами підтверджене сертифікатами проходження курсів (онлайн чи офлайн), що стосуються тем дисципліни.

Протягом курсу можна отримати не більше 10 додаткових балів.

Академічна доброчесність

Ми підтримуємо принципи академічної доброчесності і рівності всіх студентів. У випадку виявлення випадків списування (у контрольних чи лабораторних роботах) чи плагіату — бали за відповідні роботи будуть анульовані. Повторні порушення принципів академічної доброчесності можуть призвести до недопуску до складання заліку.

Викладачі можуть перевіряти роботи, виконані у рамках курсу, за допомогою систем виявлення плагіату Unicheck та MOSS.

Якщо не зазначено іншого, усі контрольні заходи проводяться у форматі «відкритої книги». Це означає, що ви маєте право користуватися будь-якими ресурсами, окрім допомоги сторонніх осіб. Ми довіряємо нашим студентам і покладаємо надію на те, що вони не порушать цю довіру.

8. Види контролю та рейтингова система оцінювання результатів навчання (РСО)

Поточний контроль:

- ***модульні контрольні роботи*** *(40%):*
  10 балів х 4 роботи = ***40*** *балів*
  Модульні контрольні роботи представляють собою тестові завдання та завдання на написання/редагування коду за темою модуля. На виконання кожної модульної контрольної роботи виділяється 2 години. Ці роботи проводяться замість лекційних занять. Вас буде заздалегідь попереджено про проведення контрольної роботи.
- ***захист лабораторних робіт*** *(50%):*
  10 балів х 5 робіт = ***50*** *балів*
  Захист лабораторних робіт займає 10-15 хвилин та складається із демонстрації коду та працюючої програми, а також співбесіди. Через це, на кожну лабораторну роботу виділяється кілька занять.
  Лабораторні роботи мають дедлайни як описано вище.
- ***презентація семестрового проекту*** *(10%):*
  10 балів х 1 заняття = ***10*** *балів*
  Семестровий проект є сумарним результатом виконання усіх лабораторних робіт та представляє собою програмний продукт із завершеним функціоналом, визначеним студентами та викладачем на першому лабораторному занятті. Презентація відбувається у вигляді короткої (до 5 хв) доповіді та демонстрації роботи програми.

Календарний контроль: проводиться двічі на семестр як моніторинг поточного стану виконання вимог силабусу. Календарний контроль проводиться за результатами лабораторних робіт, проведених на момент початку контролю.

Семестровий контроль: залік.

Залікова оцінка виставляється на основі семестрового рейтингу, або — за бажанням студента чи при семестровому рейтингу менше 60 балів — за результатами написання залікової контрольної роботи.

Залікова контрольна робота складається із 2 теоретичних та 2 практичних питань, що оцінюються у 25 балів за кожне.

Умови допуску до семестрового контролю: мінімальний рейтинг не нижче 25 балів та виконання усіх лабораторних робіт.

Таблиця відповідності рейтингових балів оцінкам за університетською шкалою:

Кількість балів	*Оцінка*
100-95	Відмінно
94-85	Дуже добре
84-75	Добре
74-65	Задовільно
64-60	Достатньо
Менше 60	Незадовільно
Не виконані умови допуску	Не допущено

9. Додаткова інформація з дисципліни (освітнього компонента)

- *додатки до силабусу — приклади модульних та залікової контрольних робіт, положення про РСО;*
- *зазвичай залік проходить на останньому занятті з дисципліни, відповідно, хард дедлайн буде за тиждень до цього;*
- *у випадку проведення курсу дистанційно, результати контролю (контрольні та залікові роботи) мають бути виконані в цифровому вигляді: як текстові файли, чи файли з кодом, або за неможливості — у вигляді розбірливих фото. Усі такі матеріали мають бути завантажені на указаний викладачем ресурс у терміни відведені під відповідний тип контролю. Зазвичай ми надаватимемо буферні 5 хвилин на випадок форс-мажорних подій. У окремих випадках, залікова контрольна може бути проведена у формі співбесіди.*

Робочу програму навчальної дисципліни (силабус):

Складено асистентом Борисенком Павлом Борисовичем

Ухвалено кафедрою ПМА (протокол № 12 від 02.06.2021)

Погоджено Методичною комісією факультету прикладної математики (протокол № 7 від 24.06.2021)