ДЕРЖАВНИЙ УНІВЕРСИТЕТ ЕКОНОМІКИ і ТЕХНОЛОГІЙ

STATE UNIVERSITY OF ECONOMICS AND TECHNOLOGY
ДЕРЖАВНИЙ УНІВЕРСИТЕТ ЕКОНОМІКИ І ТЕХНОЛОГІЙ

SYLLABUS / РОБОЧА ПРОГРАМА НАВЧАЛЬНОЇ ДИСЦИПЛІНИ

Course of study (code) / Назва дисципліни (шифр)	# ОБ'ЄКТНО-ОРІЄНТОВАНЕ ПРОГРАМУВАННЯ (IT1212BOOPP)
Academic year / Навчальний рік - Семестр	**2022/2023 –1 семестр**
Course of study / Назва спеціальності	**121 Інженерія програмного забезпечення**
Educational program / Освітня програма **Education - ECTS / Рівень – Кредити** **Status / Статус** **Learning language / Мова навчання**	**«Інженерія програмного забезпечення»** **Перший (бакалаврський) рівень -6 ECTS** **Обов'язкова** **Українська**
Author / Укладач	**Дмитро Медведєв, кандидат технічних наук,** **Державний університет економіки і технологій,** **e-mail: medvediev_dg@kneu.dp.ua, http://orcid.org/0000-0002-3747-1717** **моб. +380688535681**
Консультації	**чт, 14.00-15.00**

A. OBJECTIVE OF THE SUBJECT / МЕТА ТА ЗАВДАННЯ ДИСЦИПЛІНИ

Мета викладання навчальної дисципліни - це забезпечити отримання студентами теоретичних знань і практичних навичок прикладного програмування. За результатами вивчення дисципліни студент буде знати базові алгоритмічні конструкції та їх представлення на мові Python, засоби представлення в програмі даних та базових структур даних, формує вміння та навички складання алгоритмів різної складності, буде вміти оцінювати їх ефективність, виконувати тестування та налагодження. Також за підсумками вивчення дисципліни студент бути мати досвід використовувати отримані знання та вміння на практиці при розв'язуванні прикладних та нестандартних завдань.

B. SUBJECT PROGRAM / ПРОГРАМА ДИСЦИПЛІНИ

Тема 1.Поняття об'єктно-орієнтованого програмування на мові Python.

Об'єктно-орієнтоване програмування – метод програмування, заснований на поданні програми як сукупності взаємодіючих об'єктів, кожен з яких є екземпляром певного класу, а класи є членами певної ієрархії наслідування

Об'єктно-орієнтоване програмування має на увазі повторне використання. Комп'ютерна програма написана в формі об'єктів і класів може бути використана знову в інших проектах без повторення коду;

Використання модульного підходу в об'єктно-орієнтованому програмуванні дозволяє отримати читається і гнучкий код.

В об'єктно-орієнтованому програмуванні кожен клас має певне завдання. Якщо помилка виникне в одній частині коду, то її можна виправити локально, без необхідності втручатися в інші частини коду;

Інкапсуляція даних (яку ми розглянемо далі в статті) вносить додатковий рівень безпеки в розроблювану програму з використанням об'єктно-орієнтованого підходу;

Тема 2. *Об'єкти і класи.*

Отже, в Python об'єкти - це значення, що створюються на основі шаблону - класу. Програміст описує за допомогою спеціального синтаксису вміст класу і потім під час виконання створює об'єкти - екземпляри (instances) цього класу. У класу є свої дані - атрибути класу. До них мають доступ всі екземпляри класу. При цьому екземпляри мають свої атрибути - атрибути примірника. Ці дані доступні тільки об'єкту власнику.

Деякі атрибути можуть бути функціями. У цьому випадку такі атрибути називають методами.

Клас в об'єктно-орієнтованому програмуванні виступає в ролі креслення для об'єкта. Клас можна розглядати як карту будинку. Ви можете зрозуміти, як виглядає будинок, просто глянувши на його карту.

Тема 3. Інкапсуляція та приховування інформації.

Інкапсуляція - це пакування даних і поведінки (процедур, які працюють з даними) в один об'єкт. Інкапсуляція переслідує все ту ж мету - приховування складності за абстракцією. Але просто так складати все підряд в якийсь об'єкт не слід. Даних і поведінки має бути стільки, скільки необхідно і достатньо: об'єкт повинен зберігати всі свої і тільки свої дані самостійно і надавати всі необхідні засоби для маніпуляції цими даними.

- Визначення та використання класів
- Поля і методи класів
- Інкапсуляція та приховування інформації
- Конструктори і деструктори

Тема 4. *Наслідування в Python. Класи і підкласи.*

Python дозволяє вам створити розширений клас з одного або багатьох інших класів. Цей клас називається похідний клас (derived class) або просто підклас.

Підклас успадкує атрибути, методи, і інші члени з батьківського класу. Він так само може перевизначати (override) методи батьківського класу. Якщо підклас не визначає свій конструктор, він успадкує конструктор батьківського класу за замовчуванням. В Python, конструктор використовується для створення об'єкта і прикріплює значення атрибутам (attribute).

Конструктор підкласів завжди викликається конструктором батьківського класу, щоб ініціалізувати значення для атрибутів батьківського класу, потім він прикріплює значення цих атрибутів.

- Конструктор копіювання
- Вкладені класи
- Статичні елементи класу
- Дружні функції і класи

Тема 5. *Об'єкти (екземпляри)* класу. Конструктор __init__.
Об'єкт також називається екземпляром. Проте, процес створення об'єкта класу називається ініціалізація. В Python, щоб створити об'єкт класу, нам просто потрібно вписати назву класу, з подальшими дужками, що відкриваються і закриваються.
Клас, як ми вже побачили, може зберігати дані. Але типовий клас присутній в програмі в єдиному екземплярі. Тому сам по собі клас не надто корисний, адже зберігати визначення можна і в модулях. Весь сенс використання класів полягає в їх інстанціюванії.
Інстанціюванієм (instantiation) називають процес (акт) створення на основі класу примірника (instance) - такого об'єкта, який отримує доступ до всього вмісту класу, але при цьому володіє і здатністю зберігати власні дані. При цьому, маючи об'єкт, завжди можна дізнатися, екземпляром якого класу він є.
Тема 6. *Атрибути класу. Атрибути класу проти атрибутів екземплярів*.
В Python, кожен об'єкт містить певні атрибути за замовчуванням і методи в додаток до певних користувачем атрибутами. Щоб подивитися на всі атрибути і методи об'єкта, використовуйте вбудовану функцію під назвою dir ().
Атрибути можуть бути наочно віднесені до двох типів:
атрибути класу
атрибути екземплярів
Атрибути класу діляться серед всіх об'єктів класу, в той час як атрибути екземплярів є власністю екземпляра.
Екземпляр - це просто альтернативна назва об'єкта.
Атрибути примірника оголошуються всередині будь-якого методу, в той час як атрибути класу оголошуються поза будь-якого методу.
Тема 7. *Методи для реалізації функціоналів об'єкта.*
Метод __init__ запускається, як тільки об'єкт класу реалізується. Цей метод корисний для здійснення різного роду ініціалізації, необхідної для даного об'єкта. Класи / об'єкти можуть мати методи, що представляють собою функції, за винятком додаткової змінної self.
Метод, який може бути викликаний безпосередньо за допомогою імені класа називається статичним методом (@staticmethod).
Тема 8. *Локальні і глобальні змінні. Поліморфізм.*
Є два типи атрибутів Python: атрибути примірника і атрибути класу.
Атрибути класу також називаються змінними. Залежно від області видимості, змінні також можуть ставитися до двох типів: локальні змінні і глобальні змінні.
Локальна змінна в класі - це змінна, доступ до якої можливий тільки всередині блоку коду, в якому вона визначена. Наприклад, якщо ви визначите змінну всередині методу, до нього не вдасться отримати доступ звідки-небудь поза методу. Глобальна змінна визначається поза будь-якого блоку, тобто методу, операторів-if, тощо. Доступ до глобальної змінної може бути отриманий де завгодно в класі.
Поліморфізм в об'єктно-орієнтованому програмуванні - це можливість обробки різних типів даних, т. Е. Що належать до різних класів, за допомогою "одне і тієї ж" функції, або методу. Насправді однаковим є тільки ім'я методу, його вихідний код залежить від класу. Крім того, результати роботи однойменних методів можуть істотно відрізнятися. Тому в даному контексті під поліморфізмом розуміється безліч форм одного і того ж слова - імені метод.

C. LIST OF COMPETENCIES AND STUDIES TARGETED RESULTS / ПЕРЕЛІК КОМПЕТЕНТНОСТЕЙ ТА ПРОГРАМНИХ РЕЗУЛЬТАТІВ НАВЧАННЯ

Загальні компетентності (ЗК)	ЗК-1. Здатність до абстрактного мислення, аналізу та синтезу. ЗК-2. Здатність застосовувати знання у практичних ситуаціях. ЗК-3. Здатність проведення теоретичних та прикладних досліджень на відповідному рівні ЗК-8. Знати класифікацію програмного забезпечення та призначення його складових частин. ЗК-9. Здатність використовувати сучасні інтегровані прикладні програмні системи для обробки числової, текстової та графічної інформації.
Спеціальні (фахові) компетентності (ФК)	ФК 2. Здатність розробляти архітектури, модулі та компоненти програмних систем. ФК 3. Знання і розуміння специфікацій, стандартів, правил і рекомендацій в професійній галузі, уміння оцінювати ступінь обґрунтованості їх застосування, здатність дотримуватися їх при реалізації процесів життєвого циклу. ФК 4. Здатність забезпечувати технічну підтримку і навчання користувачів програмного забезпечення. ФК 5. Уміння готувати та презентувати документацію та методичні матеріали щодо програмного забезпечення. ФК 6. Здатність накопичувати, обробляти та систематизувати професійні знання щодо створення і супроводження програмного забезпечення та визнання важливості навчання протягом всього життя.

	ФК 7 Здатність здійснювати процес інтеграції системи, застосовувати стандарти і процедури управління змінами для підтримки цілісності загальної функціональності і надійності програмного забезпечення. ФК 8. Здатність до алгоритмічного та логічного мислення. ФК 9. Здатність приймати участь у проектуванні програмного забезпечення, включаючи проведення моделювання (формальний опис) його структури, поведінки та процесів функціонування.
Програмні результати навчання (ПРН)	ПРН 2. Розуміти, аналізувати, цілеспрямовано шукати і вибирати необхідні для вирішення професійних завдань інформаційно-довідникові ресурси і знання з урахуванням сучасних досягнень науки і техніки. ПРН-3. Знати, розуміти і застосовувати ефективні підходи щодо проектування програмного забезпечення. ПРН 7. Знати, розуміти, аналізувати, вибирати, кваліфіковано застосовувати засоби забезпечення інформаційної безпеки і цілісності даних відповідно до розв'язуваних прикладних завдань та створюваних програмних систем. ПРН-8. Знати, розуміти і застосовувати відповідні математичні поняття, методи доменного, системного і об'єктно-орієнтованого аналізів та математичного моделювання для розробки програмного забезпечення. ПРН-11. Набувати нові наукові і професійні знання, вдосконалювати навички, прогнозувати розвиток програмних систем та інформаційних технологій. ПРН 12. Знати і уміти застосовувати інформаційні технології обробки, зберігання та передачі даних. ПРН-15. Знати та вміти застосовувати методи верифікації та валідації програмного забезпечення.

D. SEMESTER PLAN / СЕМЕСТРОВИЙ ПЛАН

Тиждень/ Дата	Тема, план/короткі тези	Форма діяльності (заняття), години, формат	Завдання для СРС (література, ресурси в інтернеті, презентація, відеокурси)
1 тиждень	Тема 1.Поняття об'єктно-орієнтованого програмування на мові Python. Переваги об'єктно-орієнтованого програмування: використання модульного підходу; кожен клас має певне завдання,виправити помилку можна локально, без необхідності втручатися в інші частини коду; інкапсуляція даних вносить додатковий рівень безпеки в розроблювану програму	Лекція, (2 год), F2F	Опрацювання літератури: основна 1, 3, 5 додаткова 3
		Лабораторні заняття (2 год), F2F	Теми до вивчення: основне призначення ООП в мові Python; пошук успадкованих атрибутів; різниця між об'єктом класу і об'єктом екземпляра; особливість першого аргументу в методах класів; метод __init__; створення класу; визначення суперкласів для класу Лабораторна робота №1
2-3 тиждень	Тема 2. Об'єкти і класи. Імена класів - Створення класу Метод **__init__** як конструктор Метод **__init__** визначається з трьома параметрами у дужках: **self**, name та age Параметр **self** є обов'язковим у визначенні методу, він повинен бути першим у списку усіх параметрів - Створення екземпляру класу - Доступ до атрибутів Написання тексту програми Запуск програми Виправлення помилок в програмі	Лекція, (2 год), F2F	Опрацювання літератури: основна 1, 3, 5 додаткова 3
		Лабораторні заняття (4 год), F2F	Теми до вивчення: Створення класів і екземплярів класів Інструкція class Операції присвоювання всередині інструкції class Метод **__init__** Параметр **self** Лабораторна робота №2
4-5 тиждень	Тема 3. Інкапсуляція та приховування інформації • Визначення та	Лекція, (2 год), F2F	Опрацювання літератури: основна 1, 2, 3, 5, 8 додаткова 1,2,3

	використання класів • Поля і методи класів • Інкапсуляція та приховування інформації • Конструктори і деструктори Написання тексту програми Запуск програми Виправлення помилок в програмі	Лабораторні заняття (4 год), F2F	Теми до вивчення: створення захищеного атрибута self.__name = name метод @property метод назва_методу_для_отримання_значення_атрибута.setter Лабораторна робота №3
6-7 тиждень	Тема 4. Наслідування в Python. Класи і підкласи. Метод __init__ отримує інформацію, необхідну для створення екземпляру. Функція super () - спеціальна функція, яка допомагає Python зв'язати нащадків з батьком. Написання тексту програми Запуск програми Виправлення помилок в програмі	Лекція, (2 год), F2F	Опрацювання літератури: основна 1, 2, 3, 5, 7, 8 додаткова 1, 2, 3
		Лабораторні заняття (4 год), F2F	Теми до вивчення: • Конструктор копіювання • Вкладені класи • Статичні елементи класу • Дружні функції і класи Лабораторна робота №4
8-9 тиждень	Тема 5. Об'єкти (екземпляри) класу. Конструктор __init__. • Створення класу • Створення екземпляру класу • Доступ до атрибутів • Виклик методів • Створення декількох екземплярів	Лекція, (2 год), F2F	Опрацювання літератури: основна 1, 5, 7, 8 додаткова 1,2,3,4
		Лабораторні заняття (4 год), F2F	Теми до вивчення: Конструктор и деструктор метод __del__() параметр методу __init __ (). Перший його параметр - self - посилання на сам щойно створений об'єкт. Лабораторна робота №5
10-11 тиждень	Тема 6. Атрибути класу. Атрибути класу проти атрибутів екземплярів. Написання тексту програми Запуск програми Виправлення помилок в програмі	Лекція, (2 год), F2F	Опрацювання літератури: основна 1, 2, 3,4, 8 додаткова 1,2,3,4
		Лабораторні заняття (4 год), F2F	Теми до вивчення: атрибути класу атрибути екземплярів Статичні і динамічні атрибути класу Лабораторна робота №6
12-13 тиждень	Тема 7. Методи для реалізації функціоналів об'єкта. статичні методи класу методи екземпляра класу. Статичні методи за допомогою декоратора staticmethod Написання тексту програми Запуск програми Виправлення помилок в програмі	Лекція, (2 год), F2F	Опрацювання літератури: основна 1, 2, 3,4, 8 додаткова 1,2,3,4
		Лабораторні заняття (4 год), F2F	Теми до вивчення: Види методів: статичні, класу і екземпляра класу. Вбудований приклад методу екземпляра - str.upper (),Вбудований приклад методу класу - dict.fromkeys () Статичні методи за допомогою декоратора staticmethod Лабораторна робота №7
14-15 тиждень	Тема 8. Локальні і глобальні змінні. Поліморфізм	Лекція, (2 год), F2F	Опрацювання літератури: основна 1, 2, 3,4, 8 додаткова 1,2,3,4
		Лабораторні заняття (4 год), F2F	Теми до вивчення: • локальні змінні • глобальні змінні • поліморфізм Лабораторна робота №8

16 тиждень	Заняття для захисту індивідуальних робіт.	Лабораторне заняття (2 год), F2F	Виправлення недоліків індивідуальних робіт, виконання програмного коду, розрахунок контрольного прикладу

Вивчення дисципліни передбачає виконання двох аудиторних тестових завдань за допомогою програми Zelis. Перша контрольна модульна робота виконується за темами 1-4, друга – за темами 5-8 у тестовій формі. Під час виконання студенти мають продемонструвати уміння та навички залучати набуті теоретичні та практичні знання з програмуванні на мові Python.

Замість науково-дослідницької роботи студенти можуть отримати 10 балів за вивчення онлайн курсу «**Програмуванні на мові Python**» на платформі COURSERA (https://www.coursera.org/), посилання на курс: https://www.coursera.org/specializations/python

Детальний план проведення лабораторних занять, завдання для індивідуальних робіт містяться в системі MOODLE у відповідних розділах.

E. BASIC LITERATURE (OBLIGATORY TEXTBOOKS) / ОСНОВНА ЛІТЕРАТУРА (ОБОВ'ЯЗКОВІ ПІДРУЧНИКИ)

1. Путівник мовою програмування Python [Електронний ресурс]. – Режим доступу : http://pythonguide.rozh2sch.org.ua.
2. ArcGIS Pro Python [Електронний ресурс]. – Режим доступу : https://pro.arcgis.com/ru/pro-app/arcpy/main/arcgis-pro-arcpy-reference.htm.
3. Python. Обучение программированию [Электронный ресурс]. – Режим доступа : https://younglinux.info/python
4. Python ООП: https://python-scripts.com/object-oriented-programming-in-python#pros-cons-oop
5. Лутц М. Изучаем Python / М. Лутц. – СПб. : Символ-Плюс, 2011. – 1280 с.
6. Лутц М. Программирование на Python : в 2 томах / М. Лутц. – СПб. : Символ-Плюс, 2011. – Т. 1. – 992 с.
7. Дэвид М. Бизли Python. Подробный справочник / Дэвид М. Бизли. – СПб. : Символ-Плюс, 2010. – 864 с.
8. Саммерфилд М. Программирование на Python 3. Подробное руководство / М. Саммерфилд. – СПб. : Символ-Плюс, 2009. – 608 с.
9. Саммерфилд М. Python на практике / М. Саммерфилд – М. : ДМК Пресс, 2014. – 338 с.
10. Сузи Р. А. Язык программирования Python : учеб. пособие / Р. А. Сузи. – М. : ИНТУИТ, БИНОМ. Лаборатория знаний, 2006. – 328 с.
11. Доусон М. Программируем на Python / М. Доусон. – СПб. : Питер, 2012. – 432 с.
12. Хахаев И. А. Практикум по алгоритмизации и программированию на Python: учебник / И. А. Хахаев. – М. : Альт Линукс, 2010. – 126 с.

F. COMPLEMENTARY LITERATURE / ДОДАТКОВА ЛІТЕРАТУРА

1. А. Мюллер, С. Гвидо - Введение в машинное обучение с помощью Python. Руководство для специалистов по работе с данными – М. 2017. — 393 с.
2. Буйначев С.К., Боклаг Н.Ю. - Основы программирования на языке Python – Екатеренбург. - 2014. — 90 с.
3. Бэрри П. - Изучаем программирование на Python (Мировой компьютерный бестселлер). - М.- 2017. — 618 с.
4. Зед Шоу - Легкий способ выучить Python (Мировой компьютерный бестселлер) - 2017. — 353 с.
5. Федоров Д. - Основы программирования на примере языка Python - М.- 2018. – 167 с.
Чан Уэсли Дж. - Python. создание приложений (Библиотека профессионала) –М. – 2015.- 794 с.

G. THE MOST IMPORTANT PUBLICATIONS OF THE AUTHOR(S) CONCERNING PROPOSED CLASSES / ОСНОВНІ ПУБЛІКАЦІЇ АВТОРА, ЩО ПОВЯЗАНІ З ТЕМАТИКОЮ ЗАПЛАНОВАНИХ ЗАНЯТТЬ

1.

H. PREREQUISITE AND POSTREQUISITE / ПРЕРЕКВІЗИТИ ТА ПОСТРЕКВІЗИТИ

Дисципліни, які є необхідними передумовами для вивчення зазначеного курсу: «Вища математика», «Основи програмування».

I. SCOPE AND TYPE / КІЛЬКІСТЬ ВІДВЕДЕНИХ ГОДИН ТА ФОРМА ПРОВЕДЕННЯ ЗАНЯТЬ

	Денна	Заочна
Лекції	32	10
Практичні (лабораторні)	32	10
Самостійна робота студента (СРС)	98	142
Індивідуально-консультативна робота (ІКР)	18	18
Курсова робота	–	–

J. CURRENT AND FINAL EVALUATION / ПОТОЧНЕ ТА ПІДСУМКОВЕ ОЦІНЮВАННЯ

	Денна	Заочна
Поточний контроль, в т.ч.:	50	50
оцінювання під час аудиторних занять	10	10
виконання контрольних (тестових) робіт	20	20

виконання і захист завдань самостійної роботи	60	60
науково-дослідницька робота	10	10
Підсумковий контроль (екзамен)	-	-
Разом	100	100

J. CURRENT AND FINAL EVALUATION / ПОТОЧНЕ ТА ПІДСУМКОВЕ ОЦІНЮВАННЯ

	Денна	Заочна
Шкала балів	Оцінка за 4-бальноюшкалою	Шкала ECTS
90 – 100	Відмінно	A
80 – 89	Добре	B
70 – 79		C
66 – 69	Задовільно	D
60 – 65		E
21 – 59	незадовільно з можливістю повторного складання екзамену (заліку)	FX
0 – 20	незадовільно з можливістю вивчення дисципліни за індивідуальним графіком у формі додаткової індивідуально-консультативної роботи.	F

K. CODE OF CONDUCT OF THE COURSE / КОДЕКС ПОВЕДІНКИ ПІД ЧАС ВИВЧЕННЯ КУРСУ

Для успішного проходження курсу та складання контрольних заходів необхідним є виконання наступних обов'язків:

- не запізнюватися на заняття;
- не пропускати заняття (як лекційні. так і практичні), в разі хвороби мати довідку або її ксерокопію;
- самостійно опрацьовувати весь лекційний матеріал та ресурси для самостійної роботи;
- конструктивно підтримувати зворотній зв'язок з викладачем на всіх етапах проходження курсу (особливо під час виконання індивідуальних проектів/курсового проекту);
- своєчасно і самостійно виконувати всі передбачені програмою лабораторні завдання;
- брати очну участь у контрольних заходах;
- будь-яке відтворення результатів чужої праці (виключаючи практичну роботу над командним проектом), в тому числі використання завантажених з Інтернету матеріалів, як власних результатів, кваліфікується, як порушення норм і правил академічної доброчесності, та передбачає притягнення до відповідальності у порядку, визначеному чинним законодавством.

L. METHODS OF CONDUCTING / МЕТОДИ НАВЧАННЯ

Для формувань компетентностей застосовуються такі методи навчання:
вербальні/словесні (лекція, пояснення, дискусія);
наочні (спостереження, ілюстрація, демонстрація, відео-уроки);
практичні (різні види лабораторних завдань, вирішення задач, проведення експерименту, практики);
пояснювально-ілюстративний, який передбачає пред'явлення готової інформації викладачем та її засвоєння студентами;
метод проблемного викладу;
дослідницький.

M. TOOLS, EQUIPMENT AND SOFTWARE / ІНСТРУМЕНТИ, ОБЛАДНАННЯ ТА ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ

мультимедійний клас з ПК, цифровий проектор, інтегрована середа IDLE Python
Zoom – хмарна платформа для відео і аудіо конференцій та вебінарів
Тестування знань та отримання оцінки з екзамену:
ZELIS - система призначена для тестування знань студентів в двох режимах: автоматизований контроль знань та тестування по бланкам.

N. STUDENT RESOURCES, MOOC PLATFORMS / ЦИФРОВІ РЕСУРСИ ДЛЯ СТУДЕНТІВ ТА ВІДКРИТІ ДИСТАНЦІЙНІ ОНЛАЙН КУРСИ

Студентам пропонується доступ до навчальних матеріалів дисципліни - http://moodle.kneu.dp.ua.
Coursera – безкоштовні онлайн-курси з різних дисциплін, у разі успішного закінчення яких користувач отримує сертифікат про проходження курсу.
EdX – онлайн-курси від закладів вищої освіти.
Prometheus — український громадський проект масових відкритих онлайн-курсів.

O. FEEDBACK/ ЗВОРОТНІЙ ЗВ'ЯЗОК

Електронні листи є найкращим способом зв'язатися з керівником курсу, і, будь ласка, додайте шифр групи в темі листа. Якщо ви надішлете мені електронне повідомлення, надайте мені, принаймні, 48 годин (протягом робочого тижня), щоб відповісти. Якщо ви не отримаєте відповідь, відправте листа повторно.

P. ACADEMIC HONESTY/ АКАДЕМІЧНА ДОБРОЧЕСНІСТЬ

Державний університет економіки і технологій очікує від студентів розуміння та підтримання високих стандартів академічної чесності. Приклади академічної не доброчесності включають такі: плагіат, зловживання інформацією із застарілих джерел мережі. Очікується, що вся робота, виконана відповідно до вимог курсу, є

власною роботою студента. Під час підготовки роботи, яка відповідає вимогам курсу, студенти повинні відрізняти власні ідеї від інформації, отриманої з інших джерел. Без попереднього письмового схвалення викладачем, студенти можуть не подавати один і той же звіт двічі.

APPROVED / ЗАТВЕРДЖЕНО

Рішенням кафедри інформатики і прикладного програмного забезпечення Державного університету економіки і технологій - протокол № 1 від 25.06.2022 року

Укладач: Дмитро МЕДВЕДЄВ

ЗАТВЕРДЖЕНО:

Кафедрою інформатики та прикладного програмного забезпечення
Протокол № ___ від __ _____ 2022 року
Завідувач кафедри Олександр ЗЕЛЕНСЬКИЙ

Науково-методичною радою Державного університету економіки і технологій
Протокол № _1 від 20_вересня 2020 року
Голова науково-методичної ради Валентин ОРЛОВ