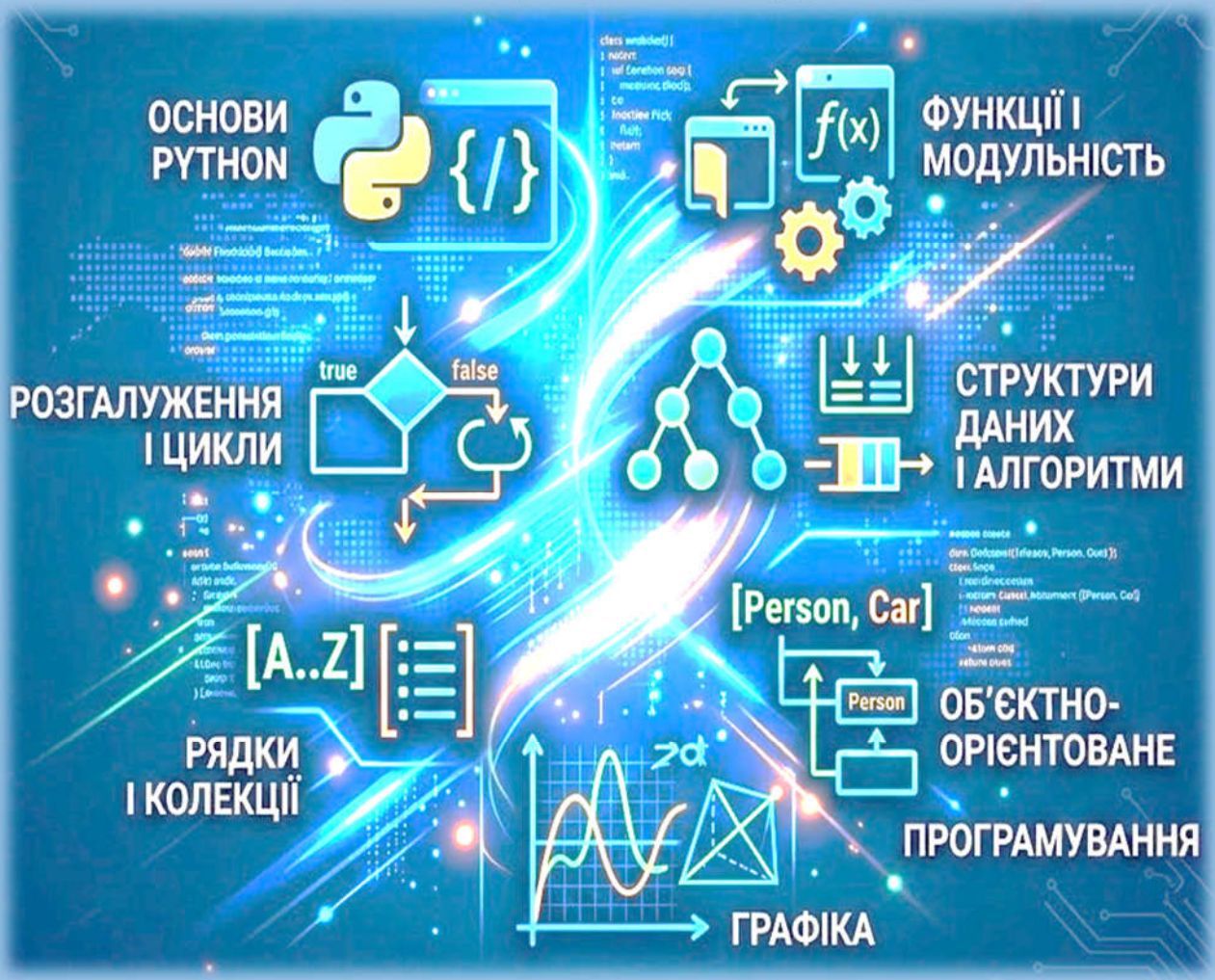
МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
МИКОЛАЇВСЬКИЙ НАЦІОНАЛЬНИЙ АГРАРНИЙ УНІВЕРСИТЕТ
ФАКУЛЬТЕТ МЕНЕДЖМЕНТУ

Кафедра економічної кібернетики,
комп'ютерних наук та інформаційних технологій

АЛГОРИТМІЗАЦІЯ ТА ПРОГРАМУВАННЯ

методичні рекомендації для виконання лабораторних та індивідуальних завдань для здобувачів першого (бакалаврського) рівня вищої освіти ОПП «Комп'ютерні науки» за спеціальністю F3(122) «Комп'ютерні науки» денної форми здобуття вищої освіти



Миколаїв – 2026

УДК 004.056
А45

Друкується за рішенням науково-методичної комісії факультету менеджменту Миколаївського національного аграрного університету (протокол №8 від 23 квітня 2026 року)

Укладач:
О. Ю. Пархоменко – к.ф.-м.н., доцент кафедри економічної кібернетики, комп’ютерних наук та інформаційних технологій Миколаївського національного аграрного університету;

Рецензенти:
В.В.Базаренко – заступник начальника Миколаївської обласної військовоїадміністрації з питань цифрового розвитку, цифрових трансформацій іцифровізації (CDTO);
Д.Л.Кошкін – к.т.н., дцент, доцент кафедри кафедри електроенергетики,електротехніки та електромеханіки Миколаївського національного аграрногоуніверситету.

ЗМІСТ

ПЕРЕДМОВА

Дисципліна «Алгоритмізація та програмування» є фундаментальною у підготовці сучасного ІТ-фахівця. Вона не лише навчає синтаксису конкретної мови, а й формує алгоритмічне мислення – здатність розбивати складні проблеми на прості кроки та знаходити найбільш ефективні шляхи їх розв'язання. Ці методичні рекомендації, що складаються з 19 лабораторних робіт, пропонують структурований шлях від перших рядків коду до розробки комплексних графічних додатків на мові Python.

Python обрано не випадково. Ця мова поєднує в собі лаконічність, потужність та величезну кількість бібліотек, що робить її ідеальним інструментом як для навчання, так і для професійної розробки в галузях від веб-технологій до аналізу даних.

Структура посібника та етапи навчання

Методичні рекомендації побудовано за принципом «від простого до складного». На першому етапі студенти знайомляться з базовим синтаксисом: типами даних, змінними та арифметичними операторами. Особлива увага приділяється культурі написання коду, зокрема використанню зрозумілих назв змінних та сучасним способам форматування виводу через f-рядки.

Наступний логічний блок охоплює керуючі конструкції: розгалуження та цикли. Студенти вчаться будувати складні логічні умови та реалізовувати ітераційні процеси, що є основою будь-якого алгоритму. Лабораторна робота №5 вводить поняття вкладених структур, що відкриває шлях до обробки матриць та складних графічних шаблонів.

Робота з даними та функціональний підхід

Значна частина курсу присвячена структурам даних. Студенти опановують роботу з рядками як незмінними послідовностями символів. Вивчення списків, кортежів, словників та множин дозволяє зрозуміти, як ефективно організовувати інформацію та виконувати операції з пошуку та фільтрації з мінімальною часовою складністю.

Перехід до функцій ознаменовує новий рівень майстерності – декомпозицію. Студенти вчаться розбивати великі завдання на ізольовані модулі, що підлягають повторному використанню. Розширені можливості функцій, такі як рекурсія та функції вищого порядку (map, filter, reduce), дозволяють писати код у функціональному стилі. Модульний підхід, вивчений у 11-й роботі, вчить створювати власні бібліотеки та працювати з файловими системами.

Алгоритми, структури даних та ООП

Розуміння алгоритмічної складності (Big O notation) та реалізація

класичних структур, таких як стеки та черги, є критичними для підготовки інженера. Студенти порівнюють різні алгоритми сортування та пошуку, оцінюючи їхню ефективність на практиці.

Об’єктно-орієнтоване програмування (ООП) представлено як потужна парадигма для моделювання реальних сутностей. Через концепції інкапсуляції, наслідування та поліморфізму студенти вчаться створювати гнучкий та масштабований код. Використання спеціальних («магічних») методів та властивостей (properties) дозволяє створювати «пітонічні» класи, що органічно вписуються в екосистему мови.

Візуалізація, інтерфейси та фінальний проект

Завершальні роботи спрямовані на практичне застосування знань. Візуалізація даних за допомогою бібліотек Turtle та Matplotlib допомагає перетворювати цифри на зрозумілі графічні образи. Створення графічного інтерфейсу користувача (GUI) за допомогою Tkinter та CustomTkinter надає програмам завершеного, професійного вигляду.

Кульмінацією курсу є **Фінальний проект**. Це підсумкова робота, де студент має самостійно пройти всі етапи розробки програмного продукту: від проектування інтерфейсу до обробки винятків та документації.

Методичні особливості

Кожна лабораторна робота містить:

- чітко сформульовану мету та завдання;
- короткі теоретичні відомості з прикладами коду;
- завдання трьох рівнів складності: легкі (для засвоєння синтаксису), середні (для логічного мислення) та складні (для творчого підходу);
- аналіз типових помилок та поради щодо налагодження;
- питання для самоперевірки, що стимулюють глибше розуміння матеріалу.

Окремо заохочується використання інструментів штучного інтелекту не для копіювання, а для аналізу, оптимізації та кращого розуміння власного коду.

Ці методичні рекомендації стануть надійним путівником для кожного, хто прагне не просто вивчити Python, а стати справжнім майстром алгоритмізації.

Бажаємо успіхів у навчанні та натхнення у створенні ваших власних програмних шедеврів!

ЛАБОРАТОРНА РОБОТА №1
Вступ до Python:
перша програма, змінні та типи даних

1. Мета

Засвоєння базових принципів написання програм на Python. Студенти навчаться створювати прості програми, що виводять інформацію на екран, правильно оголошувати та ініціалізувати змінні, працювати з основними типами даних (цілі та дробові числа, рядки, логічні значення), а також виконувати перевірку та перетворення типів за допомогою вбудованих функцій.

2. Завдання

1. Оволодіти написанням першої програми та використанням коментарів.
2. Навчитися оголошувати змінні, присвоювати їм значення різних типів.
3. Зрозуміти відмінності між типами даних int, float, str, bool, NoneType.
4. Опанувати роботу з вбудованими функціями для перевірки (type()) та перетворення (int(), float(), str(), bool()) типів даних.
5. Розробити програми, які комбінують різні типи даних для Рішення простих завдань.

3. Завдання для виконання

Частина 1. Легкі завдання

Завдання 1.1. Форматування привітання

Напишіть програму, яка виводить привітання студенту у форматі: "Вітаю, [Ім’я]! Ласкаво просимо до вивчення Python.". Використайте один виклик print() з конкатенацією рядків.

Вхід: "Анна" →

Вихід: Вітаю, Анна! Ласкаво просимо до вивчення Python.

Вхід: "Олег" →

Вихід: Вітаю, Олег! Ласкаво просимо до вивчення Python.

Вхід: "Марія" →

Вихід: Вітаю, Марія! Ласкаво просимо до вивчення Python.

Завдання 1.2. Таблиця множення (фрагмент)

Виведіть один рядок таблиці множення числа 7 у форматі "7 × 3 = 21". Використайте параметри sep та end функції print() для отримання результату без пробілів навколо знаків.

Вихід: 7×3=21

Завдання 1.3. Каталог змінних

Створіть чотири змінні: product_name (назва товару), price (ціна), in_stock (наявність на складі), weight (вага). Присвойте їм значення довільних, але відповідних типів. Виведіть тип кожної змінної.

Вихід: <class ‘str’>, <class ‘float’>, <class ‘bool’>, <class ‘int’>

Завдання 1.4. Аналіз значень None

Створіть змінні: middle_name зі значенням None, age з числовим значенням. Виведіть їхні типи за допомогою type(). Поясніть різницю у виводі.

Вхід: None, 25 → Вихід: <class ‘NoneType’>, <class ‘int’>

Завдання 1.5. Обчислення вартості

Обчисліть загальну вартість 5 одиниць товару, ціна якого становить 27.50 грн. Результат округліть до двох знаків після коми за допомогою форматування f-рядком.

Вихід: Загальна вартість: 137.50 грн.

Частина 2. Середні завдання

Завдання 2.1. Конвертер валюти (фіксований курс)

Напишіть програму, яка запитує у користувача суму в гривнях (дробове число) та виводить еквівалент в євро за фіксованим курсом (наприклад, 40.5 грн за 1 євро). Виведіть суму в євро з двома знаками після коми.

Вхід: 4050 → Вихід: 100.00 €

Вхід: 202.5 → Вихід: 5.00 €

Вхід: 81 → Вихід: 2.00 €

Завдання 2.2. Валідація номера

Користувач вводить "номер квитка" - послідовність з 6 цифр у вигляді рядка. Програма повинна перетворити цей рядок на ціле число, додати до нього число 1000 та вивести новий "номер" вже у вигляді рядка з 7 символів, доповнивши його нулями зліва за потреби.

Вхід: "001234" → Вихід: "002234"

Вхід: "999999" → Вихід: "1000999"

Вхід: "000001" → Вихід: "001001"

Частина 3. Складні завдання

Завдання 3.1. Калькулятор середнього балу з валідацією

Програма запитує три оцінки студента (від 0 до 100). Оцінки можуть бути введені як цілі або дробові числа. Завдання: перевірити, чи є кожна оцінка коректним числом (не викликає помилку при перетворенні в float). Якщо всі три

коректні - обчислити середній бал, інакше - вивести повідомлення "Помилка введення". Для перевірки використайте try-except. Вивести тип кожної введеної оцінки після перетворення.

Вхід: 85, 92.5, 78 → Вихід: Середній бал: 85.17 | Типи: float, float, float

Вхід: 100, "95", 80 → Вихід: Помилка введення (лапки позначають, що введено рядок)

Вхід: 75.5, 80, abc → Вихід: Помилка введення

Завдання 3.2. Генератор пароля за шаблоном

Користувач вводить три числа: рік народження (4 цифри), місяць (2 цифри), день (2 цифри). Програма має створити "пароль" за шаблоном: "KM-[останні_2_цифри_року][місяць][день*2]". Усі складові мають бути рядками. Якщо день менше 10, на початку має бути 0. Приклад: для 2002-03-07 → "KM-020314". Для перетворення використайте str() та зрізи.

Вхід: 2001, 12, 25 → Вихід: KM-011250

Вхід: 1999, 05, 09 → Вихід: KM-990518

Вхід: 2010, 11, 03 → Вихід: KM-101106

Завдання 3.3. Конвертер фізичних одиниць (комплексний)

Напишіть програму для переведення кілометрів на годину в метри за секунду та навпаки. Користувач вводить значення та одиницю вимірювання ("km/h" або "m/s"). Програма має вивести результат у обох одиницях з точністю до 3 знаків. Формули: m/s = km/h * 1000 / 3600, km/h = m/s * 3600 / 1000. Реалізуйте без умовних операторів, використавши словник (можна як наведений приклад конвертації) або математичну тотожність.

Вхід: 90 "km/h" → Вихід: 90.000 km/h = 25.000 m/s

Вхід: 10 "m/s" → Вихід: 10.000 m/s = 36.000 km/h

Вхід: 120 "km/h" → Вихід: 120.000 km/h = 33.333 m/s

4. Питання для самоперевірки

1. Яка функція в Python використовується для виведення інформації на екран? Які параметри цієї функції ви знаєте?
2. Що таке змінна в програмуванні? Яким оператором здійснюється присвоєння значення змінній в Python?
3. Наведіть приклади значень для кожного з п’яти основних типів даних: int, float, str, bool, NoneType.

ЛАБОРАТОРНА РОБОТА №2
Оператори та введення/виведення даних

1. Мета

Оволодіти практичними навичками роботи з основними операторами мови Python та функціями введення-виведення даних. Закріпити розуміння пріоритетів операцій, навчитися формувати складні логічні умови, коректно оперувати різними типами даних при отриманні їх від користувача та ефективно форматувати результат виведення для його читабельності.

2. Завдання

1. Закріпити знання з синтаксису та семантики основних операторів мови Python.
2. Навчитися отримувати дані від користувача за допомогою функції input() та конвертувати їх у потрібний тип.
3. Опанувати методи форматування рядків виведення з використанням f-рядків (f-strings) та методу format().
4. Розвинути вміння складати арифметичні та логічні вирази для розв’язання практичних задач.
5. Набути досвіду написання простих, але завершених програм, що взаємодіють з користувачем.

3. Завдання для виконання

Частина 1. Легкі завдання

Завдання 1.1. Вітання з користувачем

Напишіть програму, яка запитує ім’я користувача, а потім виводить привітання у форматі: "Вітаю, [ім’я]! Сподіваюся, у вас чудовий день!".

Вхід: Анна → Вихід: Вітаю, Анна! Сподіваюся, у вас чудовий день!

Вхід: Олег → Вихід: Вітаю, Олег! Сподіваюся, у вас чудовий день!

Вхід: Mary → Вихід: Вітаю, Mary! Сподіваюся, у вас чудовий день!

Завдання 1.2. Сума та різниця

Напишіть програму, яка отримує два цілих числа та виводить їх суму і різницю (перше мінус друге) в одному рядку через кому.

Вхід: 7, 3 → Вихід: 10, 4

Вхід: -5, 8 → Вихід: 3, -13

Вхід: 0, 0 → Вихід: 0, 0

Завдання 1.3. Середнє арифметичне трьох чисел

Напишіть програму, яка отримує три числа (можуть бути дробовими) та обчислює їхнє середнє арифметичне. Результат вивести без форматування.

Вхід: 4.5, 5.5, 3.0 → Вихід: 4.333333333333333

Вхід: 10, 20, 30 → Вихід: 20.0

Вхід: 1, 1, 1 → Вихід: 1.0

Завдання 1.4. Перевірка подільності

Напишіть програму, яка запитує ціле число і перевіряє, чи ділиться воно на 5 без остачі. Виведіть True, якщо ділиться, і False – якщо ні.

Вхід: 25 → Вихід: True

Вхід: 17 → Вихід: False

Вхід: 0 → Вихід: True

Частина 2. Середні завдання

Завдання 2.1. Обчислення залишку від ділення на 7

Напишіть програму, яка отримує тризначне ціле число (переконуватися в тризначності не потрібно) і обчислює суму його цифр, а потім знаходить остачу від ділення цієї суми на 7. Використайте лише арифметичні оператори (без рядків).

Вхід: 123 → Вихід: 6 (1+2+3=6, 6%7=6)

Вхід: 999 → Вихід: 6 (9+9+9=27, 27%7=6)

Вхід: 100 → Вихід: 1 (1+0+0=1, 1%7=1)

Завдання 2.2. Форматування вартості товару

Напишіть програму, яка запитує назву товару, його ціну (дробове число) та кількість одиниць. Програма повинна обчислити загальну вартість і вивести результат у вигляді: "Товар: [назва]. Загальна вартість: [сума] грн.", де сума має бути виведена з двома знаками після коми.

Вхід: Яблука, 15.50, 3 → Вихід: Товар: Яблука. Загальна вартість: 46.50 грн.

Вхід: Хліб, 22.0, 1 → Вихід: Товар: Хліб. Загальна вартість: 22.00 грн.

Вхід: Вода, 12.8, 5 → Вихід: Товар: Вода. Загальна вартість: 64.00 грн.

Завдання 2.3. Конвертація метрів

Напишіть програму, яка отримує відстань у метрах (дробове число) та виводить її еквівалент у кілометрах і сантиметрах в одному рядку, розділену комами. Виведіть кілометри з двома знаками після коми, сантиметри – як ціле число.

Вхід: 1250.75 → Вихід: 1.25 км, 125075 см

Вхід: 500 → Вихід: 0.50 км, 50000 см

Вхід: 0.01 → Вихід: 0.00 км, 1 см

Частина 3. Складні завдання

Завдання 3.1. Калькулятор калорій для бігу

Напишіть програму для розрахунку витрачених калорій під час бігу. Відомо, що в середньому витрачається 0.75 калорій на кілограм ваги за кожний кілометр бігу. Програма має:

1. Запитати вагу користувача (кг), відстань пробігу (км) та тривалість пробігу (у форматі "хвилини:секунди", наприклад "30:15").
2. Обчислити загальні витрачені калорії.
3. Обчислити середню швидкість бігу в км/год (округлити до 2 знаків).
4. Вивести детальний звіт у форматі з використанням f-рядків:

=== Звіт про пробіг ===

Вага: [вага] кг

Відстань: [відстань] км

Тривалість: [хвилини] хв [секунди] сек

Витрачено калорій: [калорії] ккал

Середня швидкість: [швидкість] км/год

Вхід: 70, 5.2, 30:0 →

Вихід:

=== Звіт про пробіг ===

Вага: 70 кг

Відстань: 5.2 км

Тривалість: 30 хв 0 сек

Витрачено калорій: 273.0 ккал (70 * 5.2 * 0.75)

Середня швидкість: 10.4 км/год (5.2 / (30/60))

Вхід: 60, 3.0, 22:30 → Вихід: ... Витрачено калорій: 135.0 ккал, Середня швидкість: 8.0 км/год

Завдання 3.2. Калькулятор кінцевої вартості проекту

Студент-фрілансер розраховує вартість проекту. Він працює за ставкою $X за годину. Напишіть програму, яка:

1. Запитує годинну ставку ($), очікувану кількість годин роботи та відсоток податку, який потрібно утримати (напр., 19.5%).
2. Обчислює вартість роботи до оподаткування (ставка × години).
3. Обчислює суму податку та вартість після оподаткування.
4. Виводить три варіанти форматування кінцевої суми в одному рядку через крапку з комою:

a) Без форматування (як є).

b) З двома знаками після коми та знаком долара на початку.

c) З цілою частиною, розділеною пробілами на тисячі та знаком долара в кінці.

Вхід: 25, 160, 19.5 →

Вихід: 3220.0; $3220.00; 3 220$

Розрахунок: 25*160=4000; податок=4000*0.195=780; 4000-780=3220

Вхід: 50, 80, 5 → Вихід: 3800.0; $3800.00; 3 800$ (4000-200)

4. Питання для самоперевірки

1. Який тип даних завжди повертає функція input()? Як отримати з неї ціле число?

2. Назвіть усі арифметичні оператори Python та поясніть різницю між /, // та %.

3. Що повертають оператори порівняння (>, <, ==, тощо`)?

4. У чому різниця між оператором присвоєння = та оператором порівняння ==?

5. Що таке f-рядок (f-string) і для чого він використовується? Наведіть приклад.

ЛАБОРАТОРНА РОБОТА №3
Умовні конструкції if-elif-else

1. Мета

Оволодіти практичними навичками використання умовних конструкцій if, elif, else у мові Python для розгалуження логіки виконання програм. Розвинути вміння аналізувати умови, правильно формулювати логічні вирази та будувати ефективні гілкові алгоритми для розв'язання типових задач програмування.

2. Завдання

1. Закріпити знання про синтаксис та семантику умовних операторів if, elif, else.

2. Навчитися складати складні логічні вирази з використанням операторів порівняння (==, !=, <, >, <=, >=) та логічних операторів (and, or, not).

3. Опанувати практику вкладених умовних конструкцій.

4. Розв'язати набір задач різного рівня складності, спрямованих на застосування умовних операторів для прийняття рішень у програмі.

5. Навчитися аналізувати коректність роботи умовних блоків та уникати типові помилки (наприклад, неповне покриття умов, неправильний порядок перевірок).

3. Завдання для виконання

Частина 1. Легкі завдання

Завдання 1.1. Перевірка додатного числа

Напишіть програму, яка перевіряє, чи є введене число додатним. Виведіть "Додатне", якщо так, або "Не додатне" в іншому випадку (включаючи нуль та від'ємні числа).

Вхід: 5 → Вихід: Додатне

Вхід: -3 → Вихід: Не додатне

Вхід: 0 → Вихід: Не додатне

Завдання 1.2. Подільність на 3

Користувач вводить ціле число. Перевірте, чи ділиться воно на 3 без остачі. Виведіть "Ділиться", якщо так, або "Не ділиться", якщо ні.

Вхід: 9 → Вихід: Ділиться

Вхід: 10 → Вихід: Не ділиться

Вхід: -6 → Вихід: Ділиться

Завдання 1.3. Однакова парність

Дано два цілих числа. Перевірте, чи є вони обидва парними або обидва непарними. Виведіть "Однакова парність", якщо так, або "Різна парність", якщо

ні.

Вхід: 4, 10 → Вихід: Однакова парність

Вхід: 3, 7 → Вихід: Однакова парність

Вхід: 2, 5 → Вихід: Різна парність

Завдання 1.4. Максимум двох чисел

Знайдіть більше з двох введених чисел. Якщо числа рівні, виведіть будь-яке з них.

Вхід: 14, 25 → Вихід: 25

Вхід: -5, -10 → Вихід: -5

Вхід: 7, 7 → Вихід: 7

Частина 2. Середні завдання

Завдання 2.1. Визначення типу року

Напишіть програму, яка визначає, чи є рік високосним. Рік високосний, якщо він ділиться на 4, але не ділиться на 100, або ділиться на 400. Виведіть "Високосний" або "Не високосний".

Вхід: 2024 → Вихід: Високосний

Вхід: 1900 → Вихід: Не високосний

Вхід: 2000 → Вихід: Високосний

Завдання 2.2. Розв’язання квадратного рівняння

Дано коефіцієнти a, b, c квадратного рівняння $ax^2 + bx + c = 0$. Визначте кількість розв’язків: 0 (дискримінант < 0), 1 (дискримінант = 0), 2 (дискримінант > 0). Обробіть випадок, коли a = 0 (рівняння не квадратне).

Вхід: 1, -3, 2 → Вихід: 2 розв’язки

Вхід: 1, 2, 1 → Вихід: 1 розв’язок

Вхід: 0, 5, 10 → Вихід: Рівняння не квадратне

Завдання 2.3. Перевірка трикутника з подальшим визначенням типу

За трьома сторонами спочатку перевірте, чи може трикутник існувати (кожна сторона менша за суму двох інших). Якщо так, визначте його тип: гострокутний (квадрат найбільшої сторони < суми квадратів двох інших), прямокутний (дорівнює), тупокутний (більше).

Вхід: 3, 4, 5 → Вихід: Прямокутний трикутник

Вхід: 5, 5, 8 → Вихід: Тупокутний трикутник

Вхід: 1, 2, 10 → Вихід: Трикутник не існує

Частина 3. Складні завдання

Завдання 3.1. Система знижок для інтернет-магазину

Розрахуйте підсумкову суму покупки з урахуванням знижок. Вхідні дані: сума покупки, статус клієнта ("новий", "постійний", "VIP"), наявність промокоду ("SALE10", "SALE20", нічого). Знижки накладаються послідовно:

1. Статус: новий - 5%, постійний - 10%, VIP - 15%
2. Промокод: SALE10 - 10%, SALE20 - 20%
3. При сумі покупки > 1000 грн - додатково 5%
4. Максимальна знижка не може перевищувати 40%. Виведіть кінцеву суму.

Вхід: 1200, "постійний", "SALE20" → Вихід: 1200 → 1080 → 864 → 820.8 (остаточна сума)

Вхід: 500, "новий", "" → Вихід: 500 → 475 → 475 (без змін)

Вхід: 2000, "VIP", "SALE10" → Вихід: 2000 → 1700 → 1530 → 1453.5 → 1453.5 (обмеження 40%)

Завдання 3.2. Калькулятор податків для фрілансера

Фрілансер вводить свій річний дохід. Розрахуйте суму податку за прогресивною шкалою:

- До 20 000 грн: 5%
- 20 001 - 50 000 грн: 10% з суми понад 20 000
- 50 001 - 100 000 грн: 15% з суми понад 50 000
- Понад 100 000 грн: 20% з суми понад 100 000
- Додатково: якщо дохід > 30 000 грн і є 1 дитина - знижка 2% від загального податку, 2+ дітей - знижка 5%. Мінімальний податок - 100 грн. Виведіть суму податку.

Вхід: 15000, 0 → Вихід: 750.0

Вхід: 75000, 2 → Вихід: 20000*0.05 + 30000*0.1 + 25000*0.15 = 6250 → 6250*0.95 = 5937.5

Вхід: 120000, 1 → Вихід: 20000*0.05 + 30000*0.1 + 50000*0.15 + 20000*0.2 = 15500 → 15500*0.98 = 15190.0

Завдання 3.3. Система прийому замовлень в ресторані

Програма приймає замовлення з наступними параметрами:

1. Час доби: "ранок" (8-11), "обід" (12-16), "вечір" (17-23), "ніч" (0-7) - інші значення недійсні
2. Тип замовлення: "сніданок", "обід", "вечеря", "доставка"
3. Кількість персон (1-10)
4. Правила:

- Сніданок доступний лише вранці, обід - в обідній час, вечеря - ввечері, доставка - будь-коли
- Для доставки: +20% до суми, мінімальна сума замовлення 200 грн

- Кожна персона: базово 100 грн за обід/вечерю, 80 грн за сніданок
- Знижка 10% для замовлень > 500 грн
- Для нічного часу (0-7) +30% націнка
- Виведіть підсумкову суму або повідомлення про неможливість замовлення.

Вхід: "ранок", "сніданок", 3 → Вихід: 3 * 80 = 240 грн

Вхід: "ніч", "доставка", 2 → Вихід: 2 * 100 = 200 → +20% = 240 → +30% = 312 грн

Вхід: "обід", "сніданок", 4 → Вихід: Неможливе замовлення: сніданок недоступний в обідній час

4. Питання для самоперевірки

1. Який синтаксис базової умовної конструкції if в Python?
2. Для чого використовується ключове слово elif і чим воно відрізняється від послідовності незалежних операторів if?
3. Яка роль ключового слова else в умовних конструкціях?
4. Які значення в Python інтерпретуються як False в умовних виразах (хибні значення)?
5. Чому після умови в операторах if, elif, else ставиться двокрапка?
6. Як Python визначає, який код належить до блоку if? Що таке відступи (indentation) і які є вимоги до них?
7. Чи можна мати кілька блоків else в одній умовній конструкції? А кілька блоків elif?
8. Що буде, якщо після if написати умову, що завжди істинна? А завжди хибна?
9. Перерахуйте всі оператори порівняння в Python. Який результат їх роботи?
10. Як працюють логічні оператори and, or, not? Який їх пріоритет виконання?

ЛАБОРАТОРНА РОБОТА №4
Цикли for та while

1. Мета

Освоєння базових концепцій такуального програмування на Python через практичне використання операторів циклів for та while; формування вміння аналізувати умову задачі та вибирати оптимальний тип циклу для її розв'язання; розвиток навичок написання, тестування та налагодження програм з циклами.

2. Завдання

1. Закріпити теоретичні знання щодо синтаксису та семантики циклів for та while.

2. Навчитися визначати необхідність застосування того чи іншого типу циклу залежно від умови задачі.

3. Опанувати методи керування виконанням циклів за допомогою операторів break, continue та else.

4. Набути практичного досвіду створення програм для обробки послідовностей чисел, рядків та розв'язання обчислювальних задач.

5. Розвинути вміння аналізувати та усувати типові помилки, пов'язані з нескінченними циклами та логікою ітерацій.

3. Завдання для виконання
Частина 1. Легкі завдання

Завдання 1.1. Сума чисел у заданому діапазоні

Напишіть програму, яка обчислює суму всіх цілих чисел у діапазоні від N до M включно. Значення N та M вводяться користувачем ($N \leq M$).

Вхід: N=1, M=5 → Вихід: 15

Вхід: N=10, M=10 → Вихід: 10

Вхід: N=-3, M=2 → Вихід: -3 (-3 + -2 + -1 + 0 + 1 + 2)

Завдання 1.2. Добуток парних чисел

Обчисліть добуток всіх парних чисел від 2 до заданого числа K включно.

Вхід: K=6 → Вихід: 48 (2*4*6)

Вхід: K=1 → Вихід: 1

Вхід: K=10 → Вихід: 3840 (2*4*6*8*10)

Завдання 1.3. Виведення квадратів чисел

Виведіть квадрати всіх чисел від a до b у зворотному порядку.

Вхід: a=2, b=4 → Вихід: 16 9 4

Вхід: a=-1, b=1 → Вихід: 1 0 1

Вхід: a=5, b=5 → Вихід: 25

Завдання 1.4. Підрахунок чисел, кратних 7

Підрахуйте кількість чисел у діапазоні від start до end, які діляться націло на 7.

Вхід: start=1, end=20 → Вихід: 2 (7, 14)

Вхід: start=7, end=7 → Вихід: 1

Вхід: start=50, end=60 → Вихід: 0

Завдання 1.5. Форматування виведення таблиці

Виведіть таблицю множення на задане число n (від 1 до 10) у форматі: n x i = результат.

Вхід: n=5 → Вихід:

5 x 1 = 5

5 x 2 = 10

...

5 x 10 = 50

Частина 2. Середні завдання

Завдання 2.1. Симулятор кидків кубика з перемогою

Симулюйте кидки шестигранного кубика (генеруйте випадкові числа від 1 до 6). Кидки продовжуються, доки не випаде 6 або не буде зроблено 10 кидків. Виведіть кількість кидків та суму всіх випалих чисел.

Приклад виконання 1: Випали числа: 2, 4, 1, 6 → Вихід: Кількість кидків: 4, Сума: 13

Приклад виконання 2: 10 кидків без шістки: 1,2,3,4,5,1,2,3,4,5 → Вихід: Кількість кидків: 10, Сума: 30

Приклад виконання 3: Перший кидок 6 → Вихід: Кількість кидків: 1, Сума: 6

Завдання 2.2. Фільтрація введених чисел

Програма зчитує числа від користувача до введення 0. Виведіть: 1) кількість додатних чисел; 2) максимальне від'ємне число (якщо такі є); 3) суму чисел, які закінчуються на 3.

Вхід: 4, -2, 13, -5, 3, 0 → Вихід: Додатних: 3, Макс від'ємне: -2, Сума чисел що закінчуються на 3: 16

Вхід: 0 → Вихід: Додатних: 0, Макс від'ємне: не введено, Сума чисел що закінчуються на 3: 0

Вхід: -10, -3, 23, -1, 0 → Вихід: Додатних: 1, Макс від'ємне: -1, Сума: 20

Завдання 2.3. Генератор "ступінчастої" послідовності

Згенеруйте послідовність, де кожне наступне число утворюється додаванням до попереднього його останньої цифри. Починаємо з start. Зупиняємося, коли число перевищує limit.

Вхід: start=3, limit=20 → Вихід: 3 6 12 14 18

Вхід: start=1, limit=100 → Вихід: 1 2 4 8 16 22 24 28 36 42 44 48 56 62 64 68 76 82 84 88 96

Вхід: start=9, limit=10 → Вихід: 9

Частина 3. Складні завдання

Завдання 3.1. Шифр "подвійного зсуву"

Реалізуйте шифрування рядка. Кожна літера зміщується на позицію її індексу в алфавіті (a=0, b=1... z=25), а потім результат зсувається ще раз на довжину всього слова. Неалфавітні символи залишаються без змін. Використовуйте цикл для перебору символів та вкладений цикл для пошуку в алфавіті.

Вхід: "abc" → Вихід: "def" (a→(0+3)=d, b→(1+3)=e, c→(2+3)=f)

Вхід: "test" → Вихід: "xiwx"

Вхід: "hello world!" → Вихід: "mjqqt btwqi!"

4. Питання для самоперевірки

1. Поясніть принципову різницю між циклами for та while. У яких ситуаціях краще використовувати кожен з них?

2. Що станеться, якщо в циклі while ніколи не зміниться умова, на основі якої він виконується? Наведіть приклад такого коду.

3. Для чого призначена функція range()? Опишіть її основні форми виклику (range(stop), range(start, stop), range(start, stop, step)) та наведіть приклади послідовностей, які вона генерує.

4. Яку роль відіграють оператори break та continue всередині циклу? Чим вони відрізняються? Наведіть короткий приклад для кожного.

5. Коли виконується блок else, що доданий до циклу (for або while)? Чи завжди він виконується після завершення циклу?

6. Що таке "нескінченний цикл" і як його можна навмисно створити? Як безпечно зупинити програму, яка потрапила в нескінченний цикл?

7. Як можна пройтися циклом for по елементах рядка? Наведіть приклад коду, який підраховує кількість певної літери у введеному тексті.

ЛАБОРАТОРНА РОБОТА №5
Вкладені цикли та складні алгоритми

1. Мета

Сформувати практичні навички проектування та реалізації алгоритмів з використанням вкладених циклів у мові програмування Python; закріпити розуміння принципу вкладеності керуючих структур та навчитися аналізувати складність алгоритмів, що базуються на множинних ітераціях; розвинути логічне та алгоритмічне мислення для розв'язання задач обробки матриць (двовимірних масивів), побудови складних графічних шаблонів та роботи з вкладеними структурами даних.

2. Завдання

1. Освоїти синтаксис та принцип роботи вкладених циклів у Python.
2. Навчитися використовувати вкладені цикли для виведення на екран регулярних графічних шаблонів (трикутники, піраміди, таблиці множення).
3. Опанувати техніку обробки двовимірних масивів (матриць) за допомогою вкладених циклів (введення, виведення, пошук елементів, обчислення сум, добутків).
4. Розвинути вміння аналізувати порядок виконання інструкцій у вкладених циклах та оцінювати складність алгоритмів.
5. Набути досвіду розв'язання практичних задач, які вимагають множинної ітерації, таких як знаходження простих чисел в діапазоні, генерація комбінацій, робота з послідовностями.

3. Завдання для виконання
Частина 1. Легкі завдання

Завдання 1.1. Прямокутник з рамкою

Напишіть програму, яка отримує від користувача ширину w та висоту h прямокутника, а потім виводить його на екран. Прямокутник має бути "порожнім" всередині: край складається з символів *, а внутрішня частина – з пробілів.

Вхід: w=5, h=4 → Вихід:

```
*****
*   *
*   *
*****
```

Вхід: w=3, h=3 → Вихід:

```
***
* *
***
```

Вхід: w=2, h=2 → Вихід:

**
**

Завдання 1.2. Таблиця підсумів

Напишіть програму, яка генерує таблицю розміром 5x5, де кожен елемент є сумою свого номера рядка та номера стовпця (починаючи з 1). Виведіть таблицю у відформатованому вигляді.

```
 2  3  4  5  6
 3  4  5  6  7
 4  5  6  7  8
 5  6  7  8  9
 6  7  8  9 10
```

Завдання 1.3. Шахівниця

Напишіть програму, яка виводить на екран шахівницю розміром 8x8. Використовуйте символи # для чорних клітинок та пробіл для білих. Ліва верхня клітинка має бути чорною (#).

```
# # # #
 # # # #
# # # #
 # # # #
# # # #
 # # # #
# # # #
 # # # #
```

..

Частина 2. Середні завдання

Завдання 2.1. Ялинка

Напишіть програму, яка отримує число n – кількість ярусів ялинки – та виводить ялинку з символів *. Кожен ярус є трикутником, ширини ярусів зростають на 2. Ялинка має бути центрованою.

Вхід: n=3 → Вихід:

```
  *
 ***
*****
  *
  *
```

(Останні два рядки - це стовбур ялинки заввишки 2)

Завдання 2.2. Ромб

Напишіть програму, яка отримує непарне число n та виводить на екран

ромб з символів *, висота якого дорівнює n.

Вхід: n=5 → Вихід:

```
  *
 ***
*****
 ***
  *
```

Завдання 2.3. Числова піраміда

Напишіть програму, яка отримує число n та виводить центровану числову піраміду, де кожен рядок містить непарну кількість чисел, що збільшуються від 1 до номера рядка та назад до 1.

Вхід: n=4 → Вихід:

```
   1
  121
 12321
1234321
```

Частина 3. Складні завдання

Завдання 3.1. Генератор усіх можливих паролів

Напишіть програму, яка отримує від користувача максимальну довжину пароля L ($1 \le L \le 4$) та список символів (наприклад, [‘a’, ‘b’, ‘c’]). Програма має згенерувати та вивести ВСІ можливі варіанти паролів довжиною від 1 до L, що складаються з заданих символів (символи можуть повторюватися). Виведення має бути в лексикографічному порядку.

Вхід: L=2, chars=[‘x’, ‘y’] → Вихід:

```
x
y
xx
xy
yx
yy
```

Завдання 3.2. Оптимізація розміщення замовлень

У вас є двомірний масив available_time, де available_time[i][j] – це кількість вільних годин у i-го виконавця в j-ий день (індекси з 0). Також є список orders – кортежів (hours, day), де hours – тривалість замовлення в годинах, day – бажаний день виконання. Напишіть програму, яка розподілить замовлення по виконавцях так, щоб мінімізувати кількість задіяних виконавців (при пріоритеті) або

максимізувати завантаження. Виведіть матрицю розподілу.

Вхід:

available_time = [[4, 2, 5], [3, 1, 0]] # 2 виконавці, 3 дні

orders = [(2, 0), (3, 1), (1, 2), (4, 0)]

Можливий вихід:

Виконавець 0: Замовлення (2,0) та (1,2). Залишок: [1, 2, 4]

Виконавець 1: Замовлення (3,1). Залишок: [3, -2, 0]

Нерозподілене: (4,0)

4. Питання для самоперевірки

1. Що таке вкладені цикли та у яких типових задачах їх необхідно застосовувати?

2. Поясніть порядок виконання інструкцій у конструкції, де цикл for j in range(3) вкладений у цикл for i in range(2). Скільки разів виконається тіло внутрішнього циклу?

3. Яка роль змінної-лічильника зовнішнього циклу при формуванні залежних шаблонів (наприклад, трикутників)? Наведіть приклад діапазону внутрішнього циклу, що залежить від i.

4. У чому різниця між виведенням рядка символів з використанням print(‘*’, end=‘‘) та звичайним print(‘*’) у контексті вкладених циклів?

5. Як створити "порожній" прямокутник (рамку) за допомогою вкладених циклів? За якої умови всередині циклів потрібно виводити пробіл, а не символ?

6. Що таке головна діагональ квадратної матриці? Як умова i == j у вкладених циклах допомагає з нею працювати?

7. Як вивести числовий трикутник, де кожен рядок містить однакові числа (1, 22, 333...), а не зростаючу послідовність?

8. Поясніть призначення оператора break у внутрішньому циклі. Наведіть приклад задачі (наприклад, пошук простих чисел), де його використання є доцільним.

9. Як змінити программу побудови прямокутного трикутника, вирівняного по лівому краю, щоб отримати трикутник, вирівняний по правому краю?

10. Що таке "прапорець" (flag) і навіщо він використовується в алгоритмах із вкладеними циклами? Наведіть приклад.

ЛАБОРАТОРНА РОБОТА №6
Робота з рядками в Python

1. Мета

Оволодіти практичними навичками роботи зі рядками в Python, зокрема освоїти методи їх створення, індексації, обробки (зміна регістру, очищення, розбиття, пошук, заміна), форматування та базового аналізу. Закріпити знання про властивості рядків як незмінних послідовностей символів. Навчитися застосовувати методи рядків для Рішення типових задач, таких як перевірка формату даних, обробка тексту та визначення особливих властивостей (паліндроми, анаграми).

2. Завдання

1. Зрозуміти принципи індексації та зрізів (slicing) рядків.
2. Навчитися модифікувати регістр символів у рядках.
3. Опанувати методи очищення рядків від зайвих символів.
4. Розібратися з пошуком та підрахунком підрядків.
5. Навчитися заміняти частини рядка або розбивати його на складові.
6. Закріпити вміння перевіряти тип вмісту рядка (цифри, букви тощо).
7. Застосувати різні способи форматування рядків для об’єднання даних.
8. Використати отримані знання для Рішення практичних задач (аналіз текстів, робота з паліндромами та анаграмами).

3. Завдання для виконання

Частина 1. Легкі завдання

Завдання 1.1. Вилучення символа за індексом

Напишіть програму, яка отримує рядок та ціле число N (індекс) від користувача. Виведіть символ, який знаходиться на позиції N. Якщо індекс виходить за межі рядка, виведіть повідомлення "Індекс поза межами рядка".

Вхід: "Програмування", 4 → Вихід: "а"

Вхід: "Python", 0 → Вихід: "P"

Вхід: "Hello", 10 → Вихід: "Індекс поза межами рядка"

Завдання 1.2. Форматування номера телефона

Користувач вводить номер телефону у форматі 10 цифр (наприклад, 0951234567). Програма має переформатувати його до вигляду: "+38(0XX)XXX-XX-XX", де XX - відповідні цифри.

Вхід: "0951234567" → Вихід: "+38(095)123-45-67"

Вхід: "0639876543" → Вихід: "+38(063)987-65-43"

Вхід: "0501112233" → Вихід: "+38(050)111-22-33"

Завдання 1.3. Перша та остання літера

Напишіть програму, яка приймає рядок та виводить першу і останню літеру цього рядка разом з їх індексами. Якщо рядок порожній, виведіть "Рядок порожній".

Вхід: "Алгоритм" → Вихід: "Перша: ‘А’(0), остання: ‘м’(7)"

Вхід: "Python" → Вихід: "Перша: ‘P’(0), остання: ‘n’(5)"

Вхід: "О" → Вихід: "Перша: ‘О’(0), остання: ‘О’(0)"

Завдання 1.4. Кожен другий символ

Користувач вводить слово. Виведіть це слово, а під ним - тільки кожен другий символ (починаючи з першого). Використовуйте зрізи (slicing).

Вхід: "Програмування" → Вихід:

Програмування

Пормуаня

Вхід: "Комп’ютер" → Вихід:

Комп’ютер

Кмюе

Вхід: "1234567890" → Вихід:

1234567890

13579

Завдання 1.5. Нормалізація імені користувача

Користувач вводить своє ім’я у будь-якому регістрі (наприклад, "оЛЕНА", "іВАН"). Програма має вивести це ім’я у правильному форматі: перша літера велика, решта - маленькі.

Вхід: "оЛЕНА" → Вихід: "Олена"

Вхід: "іВАН" → Вихід: "Іван"

Вхід: "пЕТРО" → Вихід: "Петро"

Частина 2. Середні завдання

Завдання 2.1. Форматування CSV-даних

Користувач вводить рядок у форматі "Ім’я;Прізвище;Вік;Місто" (наприклад, "Іван;Петренко;20;Київ"). Програма має розбити цей рядок, переформатувати та вивести у вигляді таблиці з заголовками та вирівнюванням.

Вхід: "Іван;Петренко;20;Київ"

Вихід:

Ім’я	Прізвище	Вік	Місто
Іван	Петренко	20	Київ

Вхід: "Марія;Іваненко;19;Львів"
Вихід: (аналогічна таблиця з новими даними)

Завдання 2.2. Генератор паролів (базовий)

Напишіть програму, яка генерує пароль за такими правилами:

1. Користувач вводить слово-основу
2. Програма замінює певні символи: ‘a’ → ‘@’, ‘i’ → ‘!’, ‘o’ → ‘0’, ‘s’ → ‘$’
3. Додає до кінця дві останні цифри поточного року
4. Переводить першу літеру у верхній регістр

Вхід: "password" → Вихід: "P@$$w0rd24" (припустимо, рік 2024)
Вхід: "security" → Вихід: "Security24"
Вхід: "algorithm" → Вихід: "@lgorithm24"

Завдання 2.3. Аналіз логу доступу

Користувач вводить рядок логу у форматі "IP-адреса - дата - запит - код відповіді" (наприклад, "192.168.1.1 - 15/Mar/2024 - GET /index.html - 200"). Програма має виокремити та вивести окремо кожну частину, а також визначити тип відповіді: 2xx - успішно, 3xx - перенаправлення, 4xx - помилка клієнта, 5xx - помилка сервера.

Вхід: "192.168.1.1 - 15/Mar/2024 - GET /index.html - 200"
Вихід:
IP: 192.168.1.1
Дата: 15/Mar/2024
Запит: GET /index.html
Код: 200 (успішно)

Вхід: "10.0.0.5 - 16/Mar/2024 - POST /login.php - 404"
Вихід:
IP: 10.0.0.5
Дата: 16/Mar/2024
Запит: POST /login.php

Код: 404 (помилка клієнта)

Частина 3. Складні завдання

Завдання 3.1. Розширений аналізатор паліндромів

Напишіть програму, яка аналізує текст на наявність паліндромів. Програма повинна:

1. Знайти всі слова, що є паліндромами (довжиною 3+ символів)
2. Для кожного паліндрому вивести його довжину та тип (слово-паліндром чи фраза-паліндром, якщо містить пробіли)
3. Визначити найдовший паліндром у тексті
4. Перевірити, чи можна з усіх знайдених паліндромів скласти новий паліндром (використовуючи перші літери)

Вхід: "Козак з казок. А роза упала на лапу Азора. Тут і там."

Вихід:

Знайдені паліндроми:

1. Козак (5) - слово
2. А роза упала на лапу Азора (24) - фраза

Найдовший: "А роза упала на лапу Азора"

З літер паліндромів: "КА" → не паліндром

Вхід: "Level, radar, civic, rotor"

Вихід:

Знайдені паліндроми:

1. Level (5) - слово
2. radar (5) - слово
3. civic (5) - слово
4. rotor (5) - слово

Найдовший: (всі однакової довжини)

З літер паліндромів: "LRCR" → не паліндром

4. Питання для самоперевірки

1. Яка основна властивість рядків у Python стосується їх змінюваності? Наведіть приклад, що демонструє цю властивість.
2. Поясніть різницю між індексами s[2] та s[-2] для рядка s = "Python". Що повернуть ці вирази?
3. Що поверне вираз "Hello World"[6:11]? Поясніть, чому останній індекс не включається до результату.
4. У чому різниця між методами find() та index()? Коли краще використовувати кожен з них?

ЛАБОРАТОРНА РОБОТА №7
Списки та кортежі

1. Мета

Засвоїти теоретичні знання та набути практичних навичок роботи з основними структурами даних послідовного типу в Python: списками та кортежами. Студент навчиться створювати, модифікувати, обробляти та аналізувати списки й кортежі, використовуючи вбудовані методи та операції, а також розрізняти області застосування цих колекцій.

2. Завдання

1. Оволодіти синтаксисом створення списків та кортежів.
2. Навчитися здійснювати базові операції модифікації списків: додавання, видалення, вставку елементів.
3. Застосовувати методи сортування та пошуку в списках.
4. Освоїти роботу зі зрізами (slice) для отримання підпослідовностей.
5. Набути досвіду роботи з вкладеними списками.
6. Зрозуміти основні властивості та операції з кортежами.
7. Навчитися конвертувати списки в кортежі та навпаки.
8. Застосувати техніку розпакування колекцій.

3. Завдання для виконання

Частина 1. Легкі завдання

Завдання 1.1. Створення списку з користувацьким введенням

Створіть програму, яка запитує у користувача 5 цілих чисел (по одному за раз) та додає їх до списку. Після введення всіх чисел виведіть отриманий список, його довжину та суму всіх елементів.

Вхід: 4, 7, -2, 0, 11 → Вихід: Список: [4, 7, -2, 0, 11], Довжина: 5, Сума: 20

Вхід: 1, 2, 3, 4, 5 → Вихід: Список: [1, 2, 3, 4, 5], Довжина: 5, Сума: 15

Вхід: 0, 0, 0, 0, 0 → Вихід: Список: [0, 0, 0, 0, 0], Довжина: 5, Сума: 0

Завдання 1.2. Додавання елементів різними методами

Створіть порожній список. Додайте до нього число 10 за допомогою append(). Потім додайте список [20, 30] за допомогою extend(). Нарешті, вставте число 5 на початок списку за допомогою insert(). Виведіть кожен проміжний результат.

Вихід після кожного кроку:

1) [10]
2) [10, 20, 30]
3) [5, 10, 20, 30]

Завдання 1.3. Формування списку парних чисел

Задано список numbers = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]. Створіть новий список, який містить лише парні числа з оригінального списку. Виведіть обидва списки.

Вхід: заданий список → Вихід: Оригінал: [1, 2, 3, 4, 5, 6, 7, 8, 9, 10], Парні: [2, 4, 6, 8, 10]

Якщо змінити оригінал на [11, 12, 13, 14] → Вихід: [12, 14]

Якщо оригінал [1, 3, 5] → Вихід: []

Частина 2. Середні завдання

Завдання 2.1. Сортування за різними критеріями

Дано список кортежів, де кожен кортеж містить назву міста та його населення (у тисячах):

cities = [("Київ", 2800), ("Львів", 720), ("Одеса", 990), ("Харків", 1440), ("Дніпро", 980)].

Відсортуйте список: 1) за назвою міста (за алфавітом); 2) за населенням (за спаданням). Виведіть обидва відсортовані варіанти.

Вихід 1: [(‘Дніпро’, 980), (‘Київ’, 2800), (‘Львів’, 720), (‘Одеса’, 990), (‘Харків’, 1440)]

Вихід 2: [(‘Київ’, 2800), (‘Харків’, 1440), (‘Одеса’, 990), (‘Дніпро’, 980), (‘Львів’, 720)]

Якщо додати ("Бердичів", 75) → в алфавітному порядку буде першим.

Завдання 2.2. Пошук елементів за умовою

Користувач вводить список цілих чисел через пробіл. Знайдіть індекси всіх елементів, які більші за середнє арифметичне всього списку. Якщо таких елементів немає, виведіть відповідне повідомлення.

Вхід: 10 20 30 40 50 → Вихід: Середнє: 30.0, Індекси елементів > середнього: [3, 4] (елементи 40, 50)

Вхід: 5 5 5 5 → Вихід: Середнє: 5.0, Немає елементів більших за середнє

Вхід: 1 2 3 4 10 → Вихід: Середнє: 4.0, Індекси: [4] (тільки 10)

Частина 3. Складні завдання

Завдання 3.1. Система обліку товарів на складі

Реалізуйте просту систему обліку товарів. Кожен товар представлений кортежем (код_товару, назва, кількість, ціна). Дано список товарів:

```
inventory = [
   (101, "Ноутбук", 5, 25000),
   (102, "Монітор", 12, 5000),
   (103, "Клавіатура", 25, 800),
   (104, "Мишка", 30, 400)
]
```

Програма повинна:

1. Знайти товар з найбільшою кількістю одиниць на складі
2. Обчислити загальну вартість кожного товару (кількість × ціна)
3. Вивести список товарів, вартість яких перевищує 10000 грн
4. Знайти середню ціну товару на складі
5. Виведіть результати у форматі таблиці.

Вихід:

Товар з найбільшою кількістю: Мишка (30 од.)

Загальна вартість кожного товару:

Ноутбук: 125000 грн

Монітор: 60000 грн

Клавіатура: 20000 грн

Мишка: 12000 грн

Товари з вартістю > 10000 грн: Ноутбук, Монітор, Клавіатура

Середня ціна товару: 7800.0 грн

Завдання 3.2. Аналіз результатів іспиту

Дано список кортежів, де кожен кортеж містить ім’я студента, його вік та бал за іспит:

students = [("Анна", 20, 85), ("Богдан", 22, 92), ("Катерина", 19, 78), ("Дмитро", 21, 95), ("Олена", 20, 88), ("Федір", 22, 62)].

Програма повинна:

1. Розділити студентів на дві групи: молодші за 21 рік та старші/рівні 21
2. Для кожної групи обчислити середній бал
3. Знайти студента з найвищим балом у кожній групі
4. Вивести список студентів, чий бал вище за загальний середній бал усіх студентів
5. Посортувати студентів за віком (зростання), а при однаковому віці - за балом (спадання)
6. Виведіть всі результати структуровано.

Вихід (приблизний):

Група <21 років (середній бал: 81.67):

Найкращий: Катерина - 78

Група >=21 років (середній бал: 83.0):

Найкращий: Дмитро - 95

Студенти вище загального середнього (83.33): Богдан, Дмитро, Олена

Сортування за віком/балом: [("Катерина", 19, 78), ("Анна", 20, 85), ("Олена", 20, 88), ("Дмитро", 21, 95), ("Богдан", 22, 92), ("Федір", 22, 62)]

4. Питання для самоперевірки

1. Яка основна відмінність між списком (list) і кортежем (tuple) у Python?

Наведіть приклади ситуацій, коли краще використовувати кожен з них.

2. Поясніть різницю між методами append(), extend() та insert(). Наведіть приклади їх використання.

3. Чим відрізняються методи remove(), pop() та del для видалення елементів зі списку? У яких випадках виникають винятки при їх використанні?

4. Яка різниця між методами sort() та sorted()? Чому sort() не працює з кортежами?

5. Опишіть синтаксис зрізів (slice) у Python. Що поверне вираз my_list[2:7:2], якщо my_list = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]?

ЛАБОРАТОРНА РОБОТА №8
Словники та множини

1. Мета

Оволодіти практичними навичками роботи з основними структурами даних Python - словниками (dict) та множинами (set). Закріпити теоретичні знання про принципи організації, створення та маніпулювання цими колекціями. Навчитися застосовувати словники для зберігання пар "ключ-значення" та множини для роботи з унікальними елементами. Опанувати методи доступу, модифікації, ітерації та виконання операцій над цими структурами для розв'язання практичних задач обробки даних.

2. Завдання

1. Зрозуміти теоретичні основи роботи зі словниками та множинами.
2. Навчитися створювати, заповнювати та модифікувати словники різними способами.
3. Опанувати методи доступу до елементів словника ([], .get()).
4. Закріпити навички додавання, оновлення та видалення елементів зі словників.
5. Навчитися здійснювати ітерацію по словниках (за ключами, значеннями та парами).

3. Завдання для виконання
Частина 1. Легкі завдання

Завдання 1.1. Створення та зміна словника з інформацією про книгу

Створіть словник book з такими ключами: "title", "author", "year". Заповніть його довільними значеннями. Потім змініть значення ключа "year" на поточний рік і додайте новий ключ "pages" зі значенням 300.

Вхід: book = {"title": "Python Basics", "author": "John Smith", "year": 2020}

Вихід: {"title": "Python Basics", "author": "John Smith", "year": 2024, "pages": 300}

Вхід: book = {"title": "Algorithms", "author": "Robert Brown", "year": 2019}

Вихід: {"title": "Algorithms", "author": "Robert Brown", "year": 2024, "pages": 300}

Вхід: book = {"title": "Data Science", "author": "Anna White", "year": 2021}

Вихід: {"title": "Data Science", "author": "Anna White", "year": 2024, "pages": 300}

Завдання 1.2. Підрахунок кількості парних чисел у словнику

Дано словник numbers, де ключі - це рядки ("num1", "num2", тощо), а значення - цілі числа. Порахуйте, скільки парних чисел міститься у словнику.

Вхід: {"num1": 10, "num2": 15, "num3": 22, "num4": 37, "num5": 40} → Вихід: 3

Вхід: {"a": 1, "b": 2, "c": 3, "d": 4} → Вихід: 2

Вхід: {"x": 11, "y": 13, "z": 17} → Вихід: 0

Завдання 1.3. Знаходження ключа за максимальним значенням

Дано словник із числовими значеннями. Знайдіть ключ, який відповідає максимальному значенню у словнику.

Вхід: {"apple": 50, "banana": 30, "orange": 75, "grape": 20} → Вихід: "orange"

Вхід: {"Math": 95, "Physics": 88, "History": 92} → Вихід: "Math"

Вхід: {"January": 15, "February": 12, "March": 18} → Вихід: "March"

Завдання 1.4. Об’єднання двох словників з додаванням значень

Дано два словники з числовими значеннями. Об’єднайте їх так, щоб у разі спільного ключа значення додавалися, а не перезаписувалися.

Вхід:
dict1 = {"a": 10, "b": 20, "c": 30}
dict2 = {"b": 5, "c": 15, "d": 25}
Вихід: {"a": 10, "b": 25, "c": 45, "d": 25}
Вхід:
dict1 = {"x": 100, "y": 200}
dict2 = {"y": 50, "z": 300}
Вихід: {"x": 100, "y": 250, "z": 300}
Вхід:
dict1 = {"cat": 3, "dog": 5}
dict2 = {"dog": 2, "bird": 4}
Вихід: {"cat": 3, "dog": 7, "bird": 4}

Частина 2. Середні завдання

Завдання 2.1. Інвертування словника (ключ↔значення)

Напишіть програму, яка інвертує словник: ключі стають значеннями, а значення - ключами. Врахуйте, що значення у вихідному словнику можуть повторюватися. У такому разі зберігайте всі відповідні ключі у списку.

Вхід: {"a": 1, "b": 2, "c": 3}

Вихід: {1: "a", 2: "b", 3: "c"}

Вхід: {"apple": "fruit", "carrot": "vegetable", "banana": "fruit"}
Вихід: {"fruit": ["apple", "banana"], "vegetable": ["carrot"]}
Вхід: {"Math": "Smith", "Physics": "Johnson", "Chemistry": "Smith"}
Вихід: {"Smith": ["Math", "Chemistry"], "Johnson": ["Physics"]}

Завдання 2.2. Аналіз успішності студентів за допомогою вкладених словників

Дано словник студентів, де ключ - ім’я студента, а значення - словник з предметами та оцінками. Знайдіть:

1. Студента з найвищим середнім балом
2. Предмет з найвищою середньою оцінкою серед усіх студентів

Вхід:
students = {
"Anna": {"Math": [85, 90], "Physics": [88, 92]},
"Bohdan": {"Math": [78, 82], "Physics": [85, 88]},
"Kateryna": {"Math": [92, 95], "Physics": [90, 94]}
}
Вихід:
Найкращий студент: Kateryna (середній бал: 92.75)
Найкращий предмет: Physics (середня оцінка: 91.5)
Вхід:
students = {
"Ivan": {"Programming": [95, 98], "Algorithms": [88, 92]},
"Maria": {"Programming": [92, 94], "Algorithms": [85, 90]}
}
Вихід:
Найкращий студент: Ivan (середній бал: 93.25)
Найкращий предмет: Programming (середня оцінка: 94.75)

Частина 3. Складні завдання

Завдання 3.1. Система телефонної книги з розширеним функціоналом

Створіть програму телефонної книги, яка підтримує такі операції:

1. Додавання нового контакту (ім’я, список телефонів, email, група)
2. Пошук контакту за іменем (частковий збіг)
3. Виведення всіх контактів певної групи
4. Видалення телефону з контакту
5. Підрахунок кількості контактів у кожній групі

Структура даних: словник, де ключ - унікальний ID контакту, значення - словник з інформацією про контакт.

Вхід:

Додати: Іван Петренко, тел: ["+380501234567", "+380671234567"], email: "ivan@example.com", група: "Друзі"

Додати: Марія Коваленко, тел: ["+380631234567"], email: "maria@example.com", група: "Робота"

Додати: Петро Сидоренко, тел: ["+380501111111"], email: "petro@example.com", група: "Друзі"

Вихід при пошуку "Іван": Контакт знайдено: Іван Петренко

Вихід для групи "Друзі": 2 контакти

Вихід статистики: {"Друзі": 2, "Робота": 1}

4. Питання для самоперевірки

1. Яка основна відмінність між словником (dict) та множиною (set) у Python? Наведіть приклади використання кожної структури.

2. Які типи даних можуть бути ключами в словнику Python? Чому список не може бути ключем?

3. Який спосіб доступу до елемента словника є безпечнішим: dict[key] чи dict.get(key)? Поясніть різницю.

4. Що станеться, якщо спробувати отримати значення за ключем, якого немає в словнику, використовуючи:

a) dict[key]

b) dict.get(key)

c) dict.get(key, "default_value")

5. Які три методи використовуються для ітерації по словнику? Напишіть приклади кожного з них.

ЛАБОРАТОРНА РОБОТА №9
Створення та використання функцій

1. Мета

Оволодіння концепцією функцій в Python як основного інструменту структурного програмування. Формування практичних навичок у створенні, виклику та використанні функцій різних типів, застосуванні їх для декомпозиції складних завдань на простіші блоки. Закріплення розуміння областей видимості змінних та правил документування коду.

2. Завдання

1. Навчитися оголошувати та викликати функції.
2. Опанувати використання параметрів та оператора return.
3. Розрізняти та застосовувати позиційні, іменовані аргументи та параметри за замовчуванням.
4. Зрозуміти різницю між локальними та глобальними змінними.
5. Набути навичок документування функцій за допомогою docstrings.
6. Застосувати принцип декомпозиції для розбиття пракичної задачі на окремі функції.
7. Реалізувати функції для математичних обчислень та обробки даних.

3. Завдання для виконання
Частина 1. Легкі завдання

Завдання 1.1. Вивід привітання

Створіть функцію say_hello без параметрів, яка виводить у консоль українською мовою: "Вітаю! Ласкаво просимо до програми обчислень."

Функція не приймає вхідних даних

Вивід: "Вітаю! Ласкаво просимо до програми обчислень."

Завдання 1.2. Малювання рамки

Створіть функцію draw_frame без параметрів, яка виводить у консоль рамку з зірочок розміром 5x20 (5 рядків висоти, 20 символів ширини).

Функція не приймає вхідних даних

Вивід:

```
********************
*                  *
*                  *
*                  *
********************
```

Завдання 1.3. Поточна дата та час

Створіть функцію show_datetime без параметрів, яка виводить поточну дату та час у форматі: "Сьогодні: DD.MM.YYYY, час: HH:MM"

Функція не приймає вхідних даних

Вивід: "Сьогодні: 15.05.2024, час: 14:30" (актуальна дата та час)

Завдання 1.4. Таблиця множення на 7

Створіть функцію multiplication_table_7 без параметрів, яка виводить таблицю множення числа 7 від 1 до 10 у форматі "7 × 1 = 7".

Функція не приймає вхідних даних

Вивід:

7 × 1 = 7

7 × 2 = 14

...

7 × 10 = 70

Частина 2. Середні завдання

Завдання 2.1. Калькулятор знижок

Створіть функцію calculate_discount з параметрами price, discount_percent та is_member (останній параметр за замовчуванням False). Якщо is_member=True, то до знижки додатково застосовується 5%. Функція повинна повертати кінцеву ціну.

Вхід: calculate_discount(1000, 10) → Вихід: 900.0

Вхід: calculate_discount(500, 20, True) → Вихід: 380.0 (500 -20% = 400, -5% = 380)

Вхід: calculate_discount(750, 15, False) → Вихід: 637.5

Завдання 2.2. Форматування повної адреси

Створіть функцію format_address з параметрами: city, street, house, apartment=None. Функція повинна повертати адресу у форматі "м. City, вул. Street, буд. House, кв. Apartment". Якщо квартира не вказана, не включати "кв. Apartment".

Вхід: format_address("Київ", "Хрещатик", "1") → Вихід: "м. Київ, вул. Хрещатик, буд. 1"

Вхід: format_address("Львів", "Свободи", "15", "42") → Вихід: "м. Львів, вул. Свободи, буд. 15, кв. 42"

Вхід: format_address("Одеса", "Дерибасівська", "22", apartment="5") → Вихід: "м. Одеса, вул. Дерибасівська, буд. 22, кв. 5"

Частина 3. Складні завдання

Завдання 3.1. Система управління бібліотекою

Створіть програму з **п'ятьма функціями** для управління бібліотекою книг:

1. add_book(library, title, author, year, genre) - додає книгу
2. find_books_by_author(library, author) - повертає список книг автора
3. find_books_by_genre(library, genre) - повертає список книг жанру
4. get_books_published_after(library, year) - повертає книги після вказаного року
5. get_library_statistics(library) - повертає статистику: загальна кількість, кількість за авторами, за жанрами

```
library = []
add_book(library, "Мастер и Маргарита", "Булгаков", 1966, "роман")
add_book(library, "Собачье сердце", "Булгаков", 1925, "повесть")
add_book(library, "1984", "Оруэлл", 1949, "антиутопия")
```

4. Питання для самоперевірки

1. Що таке функція в програмуванні та які основні переваги їх використання?
2. Поясніть різницю між параметром та аргументом функції.
3. Яке призначення ключового слова return? Що повертає функція, якщо в ній відсутній оператор return?
4. Яка різниця між позиційними та іменованими аргументами? Наведіть приклад.
5. Що таке параметр за замовчуванням? Чому не рекомендується використовувати мутабельні типи даних (списки, словники) як значення за замовчуванням?

ЛАБОРАТОРНА РОБОТА №10
Розширені можливості функцій: lambda, рекурсія, функції вищого порядку

1. Мета

Сформувати у студентів практичні навички створення та застосування розширених механізмів роботи з функціями в Python. Робота спрямована на засвоєння концепцій функцій як об'єктів першого класу, зокрема створення анонімних lambda-функцій, використання функцій вищого порядку (map(), filter(), reduce()) для функціонального стилю програмування, роботи з довільною кількістю аргументів (*args та **kwargs), а також на розуміння та реалізацію рекурсивних алгоритмів. Здобуті вміння є основою для написання гнучкого, модульного та ефективного програмного коду.

2. Завдання

1. Оволодіти синтаксисом створення та використання lambda-функцій.
2. Навчитися застосовувати вбудовані функції вищого порядку (map(), filter(), reduce()) для обробки послідовностей.
3. Зрозуміти принципи роботи з довільною кількістю позиційних та іменованих аргументів у функціях.
4. Освоїти базовий підхід до реалізації рекурсії: визначення базового випадку та рекурсивного виклику.
5. Розглянути приклади функцій, що приймають інші функції як аргументи або повертають функції як результат.
6. Виконати практичні завдання різного рівня складності для закріплення теоретичного матеріалу.

3. Завдання для виконання

Частина 1. Легкі завдання

Завдання 1.1. Lambda: Чи є число парним

Напишіть lambda-функцію is_even, яка приймає одне ціле число і повертає True, якщо воно парне, і False – якщо ні. Не використовуйте умовні оператори (if/else) в тілі lambda.

Вхід: 10 → Вихід: True

Вхід: 7 → Вихід: False

Вхід: 0 → Вихід: True

Завдання 1.2. Lambda: Конкатенація з форматуванням

Створіть lambda-функцію greet, яка приймає два аргументи: ім'я (name) та вітання (greeting) за замовчуванням "Hello". Функція повинна повертати рядок у

форматі "<greeting>, <name>!".

Вхід: ("Anna", "Hi") → Вихід: "Hi, Anna!"

Вхід: ("Petro") → Вихід: "Hello, Petro!"

Вхід: ("Maria", "Good morning") → Вихід: "Good morning, Maria!"

Завдання 1.3. Lambda: Обчислення площі прямокутника

Створіть lambda-функцію area, яка приймає довжину та ширину прямокутника (цілі або дробові числа) і повертає його площу. Передбачте, що аргументи можуть бути передані в довільному порядку (спочатку ширина, потім довжина). Використайте вбудовані функції для цього.

Вхід: (5, 3) → Вихід: 15.0

Вхід: (2.5, 4) → Вихід: 10.0

Вхід: (3) → Вихід: Помилка (підказка: функція має приймати два аргументи)

Завдання 1.4. Lambda: Перевірка на паліндром

Напишіть lambda-функцію is_palindrome, яка приймає рядок, перетворює його на нижній регістр, видаляє всі пробіли і перевіряє, чи дорівнює рядок сам собі у зворотному порядку. Повертає булеве значення.

Вхід: "Anna" → Вихід: True

Вхід: "Python" → Вихід: False

Вхід: "A man a plan a canal Panama" → Вихід: True

Частина 2. Середні завдання

Завдання 2.1. args: середнє значення довільної кількості чисел

Напишіть функцію flexible_average(*args), яка обчислює середнє арифметичне від довільної кількості переданих чисел. Якщо аргументи не передані, функція має повернути None. Ігноруйте нечислові типи.

Вхід: flexible_average(10, 20, 30) → Вихід: 20.0

Вхід: flexible_average(5) → Вихід: 5.0

Вхід: flexible_average() → Вихід: None

Вхід: flexible_average(1, ‘a’, 3) → Вихід: 2.0 (ігнорує ‘a’)

Завдання 2.2. kwargs: створення HTML-тегу

Створіть функцію make_html_tag(tag_name, content, **attributes), яка повертає рядок HTML-тегу. Параметр tag_name – назва тегу (напр., "div", "a"), content – вміст тегу. Всі інші іменовані аргументи (**attributes) повинні бути додані як атрибути тегу (напр., class="menu", href="#").

Вхід: make_html_tag(‘p’, ‘Hello World’) → Hello World

Вхід: make_html_tag(‘a’, ‘Click here’, href="https://example.com", target="_blank") → Click here

Вхід: make_html_tag(‘div’, ‘‘, id="container", class_="main") →

(Примітка: class – зарезервоване слово, тому використовуйте class_)

Завдання 2.3. Рекурсія: пошук максимального елемента в списку

Напишіть рекурсивну функцію find_max(lst), яка знаходить максимальний елемент у списку чисел (не порожньому). Не використовуйте вбудовану функцію max() або цикли.

Вхід: [3, 7, 2, 9, 1] → Вихід: 9

Вхід: [-5, -1, -10] → Вихід: -1

Вхід: [42] → Вихід: 42

Частина 3. Складні завдання

Завдання 3.1. Складна рекурсія: генерація всіх перестановок

Напишіть рекурсивну функцію permutations(items), яка приймає список items і повертає список всіх можливих перестановок (permutations) його елементів. Елементи унікальні. Не використовуйте модуль itertools.

Підказка: Рекурсивний підхід: для кожного елемента списку, відокремте його, згенеруйте всі перестановки решти списку і приклейте відокремлений елемент на початок кожної з цих перестановок.

Вхід: [1, 2, 3] → Вихід: [[1,2,3], [1,3,2], [2,1,3], [2,3,1], [3,1,2], [3,2,1]]

Вхід: [‘a’] → Вихід: [[‘a’]]

Вхід: [] → Вихід: [[]]

4. Питання для самоперевірки

1. Поясніть основну відмінність між звичайною функцією (def) та lambda-функцією. Наведіть приклад ситуації, коли використання lambda є більш доцільним.

2. Що повертають функції map() та filter() в Python 3? Як отримати з результатів їх роботи звичайний список?

3. Для чого використовують аргументи *args та **kwargs в оголошенні функції? Який тип даних має args, а який – kwargs всередині функції?

4. Опишіть два обов’язкові елементи кожної коректної рекурсивної функції. Чому відсутність будь-якого з них призведе до помилки?

5. Що таке "базовий випадок" (base case) у рекурсії? Наведіть приклад для рекурсивної функції обчислення факторіалу.

ЛАБОРАТОРНА РОБОТА №11
Модулі та робота з файлами

1. Мета

Опанувати принципи модульності в програмуванні та навчитися базовим операціям читання та запису даних у файли в Python. Студенти повинні набути практичних навичок у створенні та використанні власних модулів, використанні вбудованих модулів Python (math, random, datetime), а також ефективній та безпечній роботі з текстовими файлами, включаючи обробку помилок та структурованих даних.

2. Завдання

1. Навчитися створювати власні модулі та імпортувати їх різними способами.
2. Опанувати використання вбудованих модулів math, random та datetime для Рішення прикладних задач.
3. Оволодіти основними методами читання з текстових файлів (read, readline, readlines).
4. Навчитися записувати дані у файли за допомогою методів write та writelines.
5. Зрозуміти та застосовувати контекстний менеджер with для безпечної роботи з файлами.

3. Завдання для виконання
Частина 1. Легкі завдання

Завдання 1.1. Генерація випадкових паролів

Напишіть програму, яка використовує модуль random для генерації 5 випадкових паролів довжиною 8 символів. Кожен пароль повинен містити принаймні одну велику літеру, одну малу літеру та одну цифру. Виведіть паролі у стовпчик.

Вхід: немає

Вихід:

1. A3bC7dEf
2. xY9zP5qR
3. mN8bV2cX
4. pQ1rS6tU
5. kL4jH0wZ

Завдання 1.2. Обчислення геометричних характеристик

Використовуючи модуль math, напишіть функцію, яка приймає радіус кола

та повертає кортеж (площа, довжина кола). Округліть результати до двох знаків після коми.

Вхід: 5 → Вихід: (78.54, 31.42)

Вхід: 2.5 → Вихід: (19.63, 15.71)

Вхід: 10 → Вихід: (314.16, 62.83)

Завдання 1.3. Робота з датою народження

Використовуючи модуль datetime, напишіть програму, яка запитує рік, місяць та день народження користувача та обчислює:

1. Скільки років користувачеві
2. У який день тижня він народився
3. Скільки днів до наступного дня народження

Вхід: 2000, 5, 15 (сьогодні 13.01.2026)

Вихід:

Вік: 25 років

День народження: понеділок

Днів до наступного дня народження: 122

Вхід: 1995, 12, 31

Вихід:

Вік: 30 років

День народження: неділя

Днів до наступного дня народження: 352

Частина 2. Середні завдання

Завдання 2.1. Модуль для обробки текстів

Створіть модуль text_processor.py з такими функціями:

1. count_vowels(text) - повертає кількість голосних літер
2. reverse_words(text) - повертає текст, де слова записані у зворотньому порядку
3. to_pig_latin(text) - перетворює кожне слово на піг-латин (перша літера переміщається в кінець + "ay")

Створіть головну програму, яка демонструє роботу всіх функцій.

Вхід: "Python це цікаво"

Вихід:

Голосних: 5

Зворотній порядок слів: "цікаво це Python"

Піг-латин: "ythonPay цеay ікавоцay"

Вхід: "Привіт світ"

Вихід:
Голосних: 3
Зворотній порядок слів: "світ Привіт"
Піг-латин: "рівитП світау"

Завдання 2.2. Калькулятор з історією

Створіть модуль calculator.py з класом Calculator, який зберігає історію обчислень. Клас повинен мати методи для основних операцій (+, -, *, /) та методи для збереження історії у файл і завантаження з файлу.

Вхідні операції:
calc.add(5, 3) → 8
calc.multiply(4, 2) → 8
calc.divide(10, 2) → 5.0

Вихід (файл history.txt):
2026-01-13 14:30:15 | 5 + 3 = 8
2026-01-13 14:30:16 | 4 * 2 = 8
2026-01-13 14:30:17 | 10 / 2 = 5.0

4. Питання для самоперевірки

1. Яка різниця між командами import math та from math import sqrt? Наведіть приклади використання кожної з них.

2. Що таке модуль у Python? Як створити власний модуль і як його імпортувати в іншій програмі?

3. Опишіть різницю між методами read(), readline() та readlines() для роботи з файлами. Коли краще використовувати кожен з них?

4. Що таке контекстний менеджер with і навіщо його використовують при роботі з файлами? Наведіть приклад.

5. Які режими відкриття файлів існують у Python? Поясніть різницю між режимами ‘r’, ‘w’, ‘a’ та ‘x’.

ЛАБОРАТОРНА РОБОТА №12
Структури даних та алгоритми: стек, черга, сортування та пошук

1. Мета

Оволодіти практичними навичками реалізації та використання основних лінійних структур даних (стек, черга) в Python, навчитися реалізовувати та аналізувати базові алгоритми сортування (бульбашкою, вибором, вставками) та пошуку (лінійний, бінарний), а також засвоїти принципи аналізу алгоритмічної складності для оцінки ефективності програмних рішень. Розвинути вміння застосовувати інструменти штучного інтелекту для аналізу та покращення власного коду.

2. Завдання

1. Реалізувати стек на основі списку Python та його основні операції (push, pop, peek).
2. Використовуючи власну реалізацію стеку, написати програму для перевірки збалансованості дужок у математичних виразах.
3. Реалізувати чергу на основі списку Python та її основні операції (enqueue, dequeue).
4. Написати програму симуляції простої черги обслуговування (наприклад, в банку або call-центрі).
5. Вивчити та використати стандартну структуру deque з модуля collections.

3. Завдання для виконання

Частина 1. Легкі завдання

Завдання 1.1. Реверс рядка за допомогою стеку

Напишіть функцію reverse_string(text), яка використовує стек для реверсу рядка. Створіть власну реалізацію стеку (не використовуйте вбудовані функції reversed() або зрізи [::-1]).

Вхід: "hello" → Вихід: "olleh"
Вхід: "Python" → Вихід: "nohtyP"
Вхід: "12345" → Вихід: "54321"
Вхід: "" → Вихід: ""

Завдання 1.2. Перевірка паліндрому з використанням стеку

Використовуючи стек, напишіть функцію is_palindrome(text), яка перевіряє, чи є рядок паліндромом (читається однаково зліва направо та справа

наліво). Ігноруйте регістр букв та пробіли.

Вхід: "А роза упала на лапу Азора" → Вихід: True

Вхід: "hello" → Вихід: False

Вхід: "Racecar" → Вихід: True

Вхід: "Python" → Вихід: False

Завдання 1.3. Симуляція простої черги заявок

Створіть програму для обробки заявок. Функція process_requests(requests) приймає список заявок (рядки), додає їх у чергу та обробляє по одній, повертаючи список оброблених заявок у порядку їх надходження.

Вхід: ["заявка1", "заявка2", "заявка3"] → Вихід: ["заявка1", "заявка2", "заявка3"]

Вхід: [] → Вихід: []

Вхід: ["task_A", "task_B"] → Вихід: ["task_A", "task_B"]

Частина 2. Середні завдання

Завдання 2.1. Калькулятор зворотньої польської нотації

Реалізуйте калькулятор, який обчислює вирази у зворотній польській нотації (RPN) з використанням стеку. Підтримуйте операції: +, -, *, /. Функція rpn_calculator(expression) приймає рядок з виразом у RPN (числа та оператори розділені пробілами).

Вхід: "3 4 +" → Вихід: 7

Вхід: "5 1 2 + 4 * + 3 -" → Вихід: 14

Вхід: "10 6 9 3 + -11 * / * 17 + 5 +" → Вихід: 22

Завдання 2.2. Симуляція черги з обмеженим часом очікування

Створіть симуляцію черги, де кожен клієнт має час прибуття та максимальний час очікування. Якщо час очікування перевищує максимальний, клієнт залишає чергу. Функція queue_with_timeout(customers, service_time) приймає список клієнтів у форматі (час_прибуття, макс_очікування) та час обслуговування одного клієнта.

Вхід: customers=[(0, 5), (2, 3), (4, 2)], service_time=2 → Вихід: (обслужено: 2, втрачено: 1)

Вхід: customers=[(0, 10), (1, 10), (2, 10)], service_time=3 → Вихід: (обслужено: 3, втрачено: 0)

Вхід: customers=[(0, 1), (0, 5), (1, 1)], service_time=2 → Вихід: (обслужено: 1, втрачено: 2)

Частина 3. Складні завдання

Завдання 3.1. Система обробки завдань з пріоритетами та залежностями

Розробіть систему обробки завдань, де кожне завдання має: ID, пріоритет (1-10), тривалість виконання та список ID завдань, які повинні бути виконані перед ним. Система має визначати оптимальний порядок виконання завдань, враховуючи пріоритети та залежності. Використовуйте комбінацію черг та стеків.

Формат вхідних даних:

```
tasks = [
  {"id": "A", "priority": 3, "duration": 2, "dependencies": []},
  {"id": "B", "priority": 1, "duration": 1, "dependencies": ["A"]},
  {"id": "C", "priority": 2, "duration": 3, "dependencies": ["A"]},
  {"id": "D", "priority": 4, "duration": 2, "dependencies": ["B", "C"]}
]
```

Приклад виходу:

Порядок виконання: A → B → C → D

Загальний час: 8

Час очікування: 4

4. Питання для самоперевірки

1. Який принцип роботи стеку (LIFO/FIFO) і наведіть два реальних приклади його застосування?

2. Чому операція pop(0) у звичайному списку Python має лінійну складність O(n), а popleft() у deque - константну O(1)?

3. Поясніть, як алгоритм перевірки збалансованості дужок використовує стек. Що станеться, якщо після обробки всього рядка стек не порожній?

4. Чому бінарний пошук вимагає відсортованого масиву? Що станеться, якщо спробувати виконати його на невідсортованих даних?

5. Порівняйте часову складність лінійного та бінарного пошуку у найкращому, середньому та найгіршому випадках.

ЛАБОРАТОРНА РОБОТА №13
Робота з колекціями Python та comprehensions

1. Мета

Оволодіти практичними навичками ефективної роботи з вбудованими колекціями та спеціальними конструкціями мови Python для створення лаконічного, читабельного та продуктивного коду. Закріпити розуміння та навчитися застосовувати comprehensions (генератори списків, словників, множин), ітератори, генератори (за допомогою yield) та корисні типи з модуля collections (namedtuple, defaultdict, Counter) для розв'язання типових задач обробки даних.

2. Завдання

1. Вивчити синтаксис та принципи роботи comprehensions, генераторних виразів, генераторів з yield та обраних типів з модуля collections.
2. Навчитися замінювати традиційні цикли for та конструкції зі збором даних на лаконічні comprehensions там, де це покращує читабельність коду.
3. Опанувати створення та використання ітераторів і генераторів для ефективної роботи з послідовностями даних, особливо великими.
4. Набути досвіду використання namedtuple для створення зручних іменованих кортежів, defaultdict для спрощення угруповання даних та Counter для ефективного підрахунку елементів.
5. Розв'язати набір практичних задач різного рівня складності, спрямованих на консолідацію отриманих знань.

3. Завдання для виконання

Частина 1. Легкі завдання

Завдання 1.1. Фільтрація списку чисел

Створіть за допомогою list comprehension список, що містить лише додатні числа з заданого списку numbers. Від'ємні числа та нуль потрібно ігнорувати.

Вхід: [-5, 10, 0, 3, -1, 7] → Вихід: [10, 3, 7]
Вхід: [1, 2, 3, 4, 5] → Вихід: [1, 2, 3, 4, 5]
Вхід: [-10, -20, -30] → Вихід: []

Завдання 1.2. Список довжин слів

Дано список слів words. За допомогою list comprehension створіть новий список, що містить довжини цих слів.

Вхід: ["Python", "is", "awesome"] → Вихід: [6, 2, 7]
Вхід: ["apple", "banana", "cherry"] → Вихід: [5, 6, 6]
Вхід: [""] → Вихід: [0]

Завдання 1.3. Створення словника степенів

Задано список цілих чисел nums. Створіть dictionary comprehension, де ключем буде число зі списку, а значенням – його квадрат.

Вхід: [1, 2, 3, 4] → Вихід: {1: 1, 2: 4, 3: 9, 4: 16}

Вхід: [5] → Вихід: {5: 25}

Вхід: [] → Вихід: {}

Завдання 1.4. Пошук спільних елементів

Дано два списки list1 та list2. За допомогою list comprehension створіть список, що містить елементи, які є одночасно в обох списках (перетин множин). Результат повинен містити лише унікальні значення.

Вхід: [1, 2, 3, 4], [3, 4, 5, 6] → Вихід: [3, 4]

Вхід: ["a", "b", "c"], ["c", "d", "e"] → Вихід: ["c"]

Вхід: [10, 20], [30, 40] → Вихід: []

Частина 2. Середні завдання

Завдання 2.1. Робота з даними студентів (namedtuple)

Створіть namedtuple з іменем Student з полями: first_name, last_name, group, average_mark. Напишіть функцію get_best_student(students), яка отримує список таких кортежів і повертає прізвище та ім’я студента з найвищим середнім балом (у форматі "Прізвище І."). Якщо таких студентів кілька, поверніть першого знайденого.

Вхід: [Student("Іван", "Петренко", "КН-101", 4.5), Student("Марія", "Сидоренко", "КН-101", 4.8), Student("Петро", "Іваненко", "КН-102", 4.5)] → Вихід: "Сидоренко М."

Вхід: [Student("Анна", "Коваль", "ПМ-201", 3.2)] → Вихід: "Коваль А."

Вхід: [] → Вихід: None

Завдання 2.2. Групування товарів за категорією (defaultdict)

Є список словників products, кожен з яких описує товар і має ключі "name" та "category". Використовуйте defaultdict, щоб створити словник, де ключами будуть назви категорій, а значеннями – списки назв товарів, що належать до цієї категорії.

Вхід: [{"name": "iPhone", "category": "Електроніка"}, {"name": "Банан", "category": "Продукти"}, {"name": "Ноутбук", "category": "Електроніка"}, {"name": "Хліб", "category": "Продукти"}]

Вихід: defaultdict(<class ‘list’>, {‘Електроніка’: [‘iPhone’, ‘Ноутбук’],

‘Продукти’: [‘Банан’, ‘Хліб’]})

Вхід: [{"name": "Книга", "category": "Навчання"}] → Вихід: defaultdict(<class ‘list’>, {‘Навчання’: [‘Книга’]})

Вхід: [] → Вихід: defaultdict(<class ‘list’>, {})

Частина 3. Складні завдання

Завдання 3.1. Оптимізація обробки великого лог-файлу (Counter + генератор)

Уявіть, що у вас є дуже великий текстовий лог-файл сервера (симулюйте його як генератор, що видає рядки). Кожен рядок містить IP-адресу користувача. Напишіть функцію find_top_ips(log_generator, top_n), яка отримує генератор log_generator (повертає рядки логу по одному) і ціле число top_n. Функція має повернути список top_n найактивніших IP-адрес (найбільша кількість входжень у лозі), використовуючи Counter. Метою є обробка потенційно нескінченного потоку даних без завантаження всього логу в пам’ять.

Вхід (генератор видає): ["192.168.1.1", "10.0.0.1", "192.168.1.1", "172.16.0.1", "10.0.0.1", "192.168.1.1"], top_n=2

Вихід: [(‘192.168.1.1’, 3), (‘10.0.0.1’, 2)]

Вхід (генератор видає): ["127.0.0.1"] * 5, top_n=1 → Вихід: [(‘127.0.0.1’, 5)]

Вхід: порожній генератор, top_n=5 → Вихід: []

Складність: Потрібно написати як генератор, що симулює файл, так і функцію аналізу, яка працює з цим генератором.

4. Питання для самоперевірки

1. Які три основні типи comprehensions існують у Python? Наведіть приклад синтаксису кожного з них.

2. Чим відрізняється генераторний вираз (x for x in range(10)) від list comprehension [x for x in range(10)] з точки зору продуктивності та пам’яті?

3. У чому полягає ключова відмінність між функцією, яка повертає список, та функцією-генератором з використанням ключового слова yield?

4. Для чого використовується defaultdict з модуля collections? Наведіть конкретний приклад задачі, де його використання є значно зручнішим, ніж робота зі звичайним dict.

5. Що поверне метод Counter("abracadabra").most_common(2)? Поясніть, що роблять клас Counter та його метод most_common.

ЛАБОРАТОРНА РОБОТА №14
Класи та об’єкти

1. Мета

Засвоїти базові принципи об’єктно-орієнтованого програмування (ООП) на Python через практичне створення класів та об’єктів. Студенти навчаться оголошувати класи, визначати атрибути (екземпляра та класу) та методи (екземпляра, класу, статичні), використовувати конструктор __init__ для ініціалізації об’єктів, а також моделювати взаємодію між декількома об’єктами одного класу.

2. Завдання

1. Оволодіти синтаксисом оголошення класу в Python.
2. Навчитися створювати об’єкти (екземпляри) класу.
3. Розрізняти та застосовувати атрибути екземпляра та атрибути класу.
4. Освоїти роботу з конструктором __init__ для ініціалізації початкового стану об’єкта.
5. Навчитися проектувати та викликати методи екземпляра, класу та статичні методи.
6. Розвинути навички моделювання простих реальних сутностей за допомогою класів.
7. Реалізувати взаємодію між декількома об’єктами одного класу в програмі.

3. Завдання для виконання
Частина 1. Легкі завдання

Завдання 1.1. Клас "Точка"

Створіть клас Point, який представляє точку в двовимірному просторі. Клас має мати конструктор __init__(self, x, y), що приймає координати x та y. Додайте метод __str__(self), який повертає рядок у форматі "(x, y)".

Вхід: p = Point(3, 4) → Вихід при print(p): (3, 4)

Вхід: p = Point(-1, 0) → Вихід при print(p): (-1, 0)

Вхід: p = Point(0, 0) → Вихід при print(p): (0, 0)

Завдання 1.2. Клас "Книга" (базовий варіант)

Створіть клас Book. Конструктор має приймати параметри title (назва) та author (автор). Додайте метод get_info(self), який повертає рядок: "Книга: [назва], Автор: [автор]".

Вхід: b = Book("1984", "Джордж Орвелл") → Вихід при b.get_info(): "Книга: 1984, Автор: Джордж Орвелл"

Вхід: b = Book("Кобзар", "Тарас Шевченко") → Вихід: "Книга: Кобзар,

Автор: Тарас Шевченко"

Вхід: b = Book("Маленький принц", "Антуан де Сент-Екзюпері") → Вихід: "Книга: Маленький принц, Автор: Антуан де Сент-Екзюпері"

Завдання 1.3. Клас "Прямокутник"

Створіть клас Rectangle. Конструктор має приймати width (ширина) та height (висота). Додайте метод area(self), який обчислює та повертає площу прямокутника.

Вхід: r = Rectangle(5, 3) → Вихід при r.area(): 15

Вхід: r = Rectangle(7, 2) → Вихід: 14

Вхід: r = Rectangle(4, 4) → Вихід: 16

Частина 2. Середні завдання

Завдання 2.1. Клас "Користувач" з атрибутом класу

Створіть клас User. Додайте атрибут класу user_count = 0, який відстежує загальну кількість створених користувачів. У конструкторі збільшуйте цей лічильник на 1 при кожному створенні нового об’єкта. Додайте метод класу get_total_users(cls), який повертає поточну кількість користувачів.

Вхід: User.get_total_users() → Вихід: 0 → Після u1 = User("Аліса") → User.get_total_users(): 1 → Після u2 = User("Богдан") → User.get_total_users(): 2

Вхід: u1 = User("Марія") → User.get_total_users(): 1

Вхід: User.get_total_users() → 0 → u1 = User("Іван") → u2 = User("Оксана") → u3 = User("Петро") → Вихід: User.get_total_users(): 3

Завдання 2.2. Клас "Магазин" зі статичним методом

Створіть клас Store. Додайте статичний метод is_valid_product_name(name), який перевіряє, чи назва товару відповідає критеріям: не порожня, довжина не менше 3 символів, не починається з пробілу. Метод повертає True або False. Конструктор класу приймає name (назва магазину). Додайте метод add_product(self, product_name), який додає товар тільки якщо він валідний (використовуючи статичний метод), і повертає відповідне повідомлення.

Вхід: Store.is_valid_product_name("Молоко") → Вихід: True

Вхід: Store.is_valid_product_name(" a") → Вихід: False (пробіл на початку та коротка назва)

Вхід: s = Store("Продукти") → s.add_product("Хліб") → Вихід: "Товар ‘Хліб’ додано" → s.add_product(" ") → Вихід: "Невірна назва товару"

Частина 3. Складні завдання

Завдання 3.1. Система "Бібліотека" з взаємодією об’єктів

Створіть клас LibraryBook (книга в бібліотеці) та клас LibraryMember (читач).

LibraryBook має: title, author, book_id, is_available (доступність, за замовчуванням True), методи borrow() (позначає книгу як взяту, якщо вона доступна) та return_book() (позначає як доступну).

LibraryMember має: name, member_id, borrowed_books (список id взятих книг), методи borrow_book(book) (додає id книги до свого списку, якщо книга доступна, та викликає метод borrow() книги) та return_book(book) (видаляє id зі списку та викликає return_book() книги). Додайте метод show_borrowed(self) для виведення списку взятих книг.

Вхід:

```
book1 = LibraryBook("1984", "Орвелл", "B001")
book2 = LibraryBook("Кобзар", "Шевченко", "B002")
member = LibraryMember("Олена", "M001")

member.borrow_book(book1)
member.borrow_book(book2)
member.show_borrowed()
```

Вихід: ["B001", "B002"] (статус book1 та book2: is_available = False)

Вхід: (продовження) member.return_book(book1) → book1.is_available: True → member.show_borrowed(): ["B002"]

Вхід: member.borrow_book(book1) → member.borrow_book(book1) (спроба вдруге) → Вихід: "Книга B001 вже взята вами або недоступна"

4. Питання для самоперевірки

1. Що таке клас в Python і чим він відрізняється від об’єкта?
2. Для чого використовується метод __init__ у класі? Як він називається?
3. Що таке параметр self у методі класу? Чи можна дати йому інше ім’я?
4. Як створити екземпляр (об’єкт) класу Car? Наведіть приклад.
5. Що таке атрибут екземпляра? Наведіть приклад його ініціалізації в конструкторі та доступу до нього.
6. Що таке метод екземпляра? Наведіть приклад оголошення та виклику.
7. Яка різниця між атрибутом екземпляра та атрибутом класу? Наведіть приклад кожного.
8. Як змінити значення атрибута класу, і як ця зміна вплине на вже створені об’єкти?

ЛАБОРАТОРНА РОБОТА №15
Наслідування та поліморфізм

1. Мета

Засвоєння принципів об’єктно-орієнтованого програмування (ООП) – наслідування та поліморфізму, формуванні практичних навичок створення ієрархій класів, перевизначення та розширення методів, використання множинного наслідування, абстрактних класів та забезпечення поліморфної поведінки об’єктів у мові Python.

2. Завдання

1. Оволодіти синтаксисом оголошення батьківського (базового) та дочірніх класів.
2. Навчитися використовувати функцію super() для доступу до методів батьківського класу.
3. Опанувати техніки перевизначення (overriding) та розширення (extending) методів.
4. Зрозуміти принципи множинного наслідування та MRO (Method Resolution Order).
5. Навчитися застосовувати функції isinstance() та issubclass() для перевірки типів.

3. Завдання для виконання
Частина 1. Легкі завдання

Завдання 1.1. Створення базового класу "Тварина"

Створіть базовий клас Animal з атрибутами name (ім’я) та species (вид). Додайте метод make_sound() та get_info(). Створіть дочірній клас Dog, який наслідує Animal та перевизначає метод make_sound() для повернення "Гав!". Використовуйте super() для виклику батьківського конструктора.

Вхід: dog = Dog("Рекс", "Собака домашній")

Вихід при виклику dog.get_info(): "Рекс - Собака домашній"

Вихід при виклику dog.make_sound(): "Гав!"

Вхід: dog2 = Dog("Бакс", "Лабродор")

Вихід при виклику dog2.get_info(): "Бакс - Лабродор"

Вхід: dog3 = Dog("Шарик", "Вівчарка")

Вихід при виклику dog3.make_sound(): "Гав!"

Завдання 1.2. Розширення класу "Студент"

Створіть клас Student з атрибутами name та student_id. Створіть клас

ExtendedStudent, який наслідує Student та додає атрибут major (спеціальність). Реалізуйте метод display_info(), який у дочірньому класі повинен показувати всю інформацію про студента.

Вхід: s = ExtendedStudent("Олена Петренко", "S12345", "Комп'ютерні науки")

Вихід: s.display_info() → "Студент: Олена Петренко, ID: S12345, Спеціальність: Комп'ютерні науки"

Вхід: s2 = ExtendedStudent("Іван Сидоренко", "S67890", "Математика")

Вихід: s2.display_info() → "Студент: Іван Сидоренко, ID: S67890, Спеціальність: Математика"

Вхід: s3 = ExtendedStudent("Марія Коваленко", "S11111", "Фізика")

Вихід: s3.display_info() → "Студент: Марія Коваленко, ID: S11111, Спеціальність: Фізика"

Завдання 1.3. Наслідування класу "Банківський рахунок"

Створіть базовий клас BankAccount з атрибутами owner (власник) та balance (баланс). Додайте методи deposit(amount) (поповнити) та get_balance(). Створіть клас SavingsAccount, який наслідує BankAccount та додає атрибут interest_rate (відсоткова ставка). Використовуйте super() у конструкторі.

Вхід: acc = SavingsAccount("Петро Іванов", 1000, 0.05)
acc.deposit(500)
Вихід: acc.get_balance() → 1500

Вхід: acc2 = SavingsAccount("Оксана Сидорова", 2000, 0.03)
Вихід: acc2.get_balance() → 2000

Вхід: acc3 = SavingsAccount("Михайло Петренко", 0, 0.07)
acc3.deposit(3000)
Вихід: acc3.get_balance() → 3000

Частина 2. Середні завдання

Завдання 2.1. Множинне наслідування: "Розумний будинок"

Створіть три базові класи: LightDevice (метод turn_on(), turn_off()), Thermostat (метод set_temperature(temp)), SecurityDevice (метод activate_alarm()). Створіть клас SmartHomeController, який наслідує всі три класи. Додайте метод evening_mode(), який включає світло, встановлює температуру 22°C та активує сигналізацію. Продемонструйте MRO.

Вхід: controller = SmartHomeController()
Вихід при виклику controller.evening_mode() →
"Світло увімкнено"
"Температура встановлена на 22°C"
"Сигналізацію активовано"

Вхід: controller.turn_off() → "Світло вимкнено"
Вхід: controller.set_temperature(25) → "Температура встановлена на 25°C"
Вихід: SmartHomeController.__mro__ → показує порядок пошуку методів

Завдання 2.2. Поліморфізм у системі обробки документів

Створіть базовий клас Document з абстрактним методом process(). Створіть три дочірніх класи: TextDocument (повертає "Обробка текстового документу"), SpreadsheetDocument (повертає "Обробка електронної таблиці"), PDFDocument (повертає "Обробка PDF-документу"). Напишіть функцію process_documents(documents), яка обробляє список документів різних типів.

Вхід: docs = [TextDocument(), SpreadsheetDocument(), PDFDocument()]
Вихід process_documents(docs):
"Обробка текстового документу"
"Обробка електронної таблиці"
"Обробка PDF-документу"

Вхід: docs2 = [PDFDocument(), PDFDocument(), TextDocument()]
Вихід:
"Обробка PDF-документу"
"Обробка PDF-документу"
"Обробка текстового документу"

Вхід: docs3 = [SpreadsheetDocument(), TextDocument()]
Вихід:
"Обробка електронної таблиці"
"Обробка текстового документу"

Частина 3. Складні завдання

Завдання 3.1. Система управління персоналом з абстрактними класами

Створіть абстрактний базовий клас Employee з абстрактними методами calculate_salary() та get_role(). Створіть три конкретних класи: HourlyEmployee (атрибути: hours_worked, hourly_rate), SalariedEmployee (атрибути:

monthly_salary, bonus), CommissionEmployee (атрибути: sales_amount, commission_rate). Створіть клас Department для управління співробітниками різних типів. Реалізуйте функціонал додавання співробітників, підрахунку загальних витрат на зарплату та пошуку співробітників за роллю.

Вхід:

dept = Department("IT")

dept.add_employee(HourlyEmployee("Іван", "розробник", 160, 250))

dept.add_employee(SalariedEmployee("Олена", "менеджер", 30000, 5000))

dept.add_employee(CommissionEmployee("Петро", "продажі", 500000, 0.05))

Вихід: dept.total_payroll() → сума всіх зарплат

Вихід: dept.find_by_role("розробник") → список співробітників з цією роллю

Вихід для кожного співробітника: employee.calculate_salary() → конкретна зарплата

Вхід:

dept2 = Department("Sales")

dept2.add_employee(CommissionEmployee("Марія", "продажник", 1000000, 0.07))

Вихід: dept2.total_payroll() → 70000 (7% від 1 000 000)

4. Питання для самоперевірки

1. Що таке наслідування в ООП і які переваги воно дає?
2. Поясніть різницю між батьківським (базовим) та дочірнім (похідним) класом.
3. Для чого використовується функція super() і в яких випадках її обов’язково викликати?
4. Що таке перевизначення методів (method overriding) і чим воно відрізняється від перевантаження?
5. Поясніть поняття поліморфізму в ООП. Наведіть приклад з життя.

ЛАБОРАТОРНА РОБОТА №16
Інкапсуляція та спеціальні методи

1. Мета

Засвоїти принципи інкапсуляції в об’єктно-орієнтованому програмуванні на Python та навчитися використовувати спеціальні (магічні) методи для створення зручних, безпечних та інтуїтивно зрозумілих класів. У ході роботи студенти навчаться контролювати доступ до атрибутів об’єкта, перевизначати поведінку операторів і вбудованих функцій, а також створювати об’єкти, які можуть взаємодіяти з контекстними менеджерами та протоколами ітерації.

2. Завдання

1. Оволодіти концепцією інкапсуляції та її синтаксичним вираженням у Python через приватні (_, __) атрибути.
2. Навчитися керувати доступом до атрибутів класу за допомогою властивостей (@property) та методів-сеттерів і геттерів.
3. Зрозуміти призначення та реалізувати магічні методи для рядкового представлення об’єктів (__str__, __repr__).
4. Освоїти перевантаження основних арифметичних операторів (наприклад, __add__, __sub__) та операторів порівняння (наприклад, __eq__, __lt__).
5. Навчитися створювати класи-контейнери, реалізуючи методи __len__, __getitem__, __setitem__.
6. Розібратися з механізмом роботи контекстних менеджерів через методи __enter__ та __exit__.
7. Зрозуміти принцип роботи ітераторів та реалізувати їх за допомогою методів __iter__ та __next__.

3. Завдання для виконання

Частина 1. Легкі завдання

Завдання 1.1. Клас для зберігання пароля

Створіть клас PasswordManager, який зберігає пароль у приватному атрибуті. Реалізуйте властивість password з геттером та сеттером. Сеттер повинен перевіряти, що пароль містить принаймні 8 символів та хоча б одну цифру. При ініціалізації пароль встановлюється у пустий рядок.

Вхід:

pm = PasswordManager()

pm.password = "qwerty"

Вихід: ValueError("Пароль має містити принаймні 8 символів та хоча б одну цифру")

Вхід:
pm = PasswordManager()
pm.password = "securepass123"
print(pm.password)
Вихід: "securepass123"

Вхід:
pm = PasswordManager()
pm.password = "short1"
Вихід: ValueError("Пароль має містити принаймні 8 символів та хоча б одну цифру")

Завдання 1.2. Клас для обмеження віку

Створіть клас Person, який зберігає ім’я та вік особи. Вік має бути приватним атрибутом. Реалізуйте властивість age з геттером та сеттером. Сеттер не дозволяє встановити вік менше 0 або більше 120 років. При спробі встановити некоректне значення генерується виняток ValueError.

Вхід:
person = Person("Іван", 25)
print(person.age)
Вихід: 25

Вхід:
person = Person("Марія", 30)
person.age = 150
Вихід: ValueError("Вік має бути від 0 до 120 років")

Вхід:
person = Person("Петро", -5)
Вихід: ValueError("Вік має бути від 0 до 120 років")

Частина 2. Середні завдання

Завдання 2.1. Клас для комплексних чисел

Створіть клас ComplexNumber, який реалізує комплексні числа. Перевантажте оператори +, -, * та /. Також реалізуйте метод __abs__, який повертає модуль комплексного числа. Кожна операція має повертати новий об’єкт ComplexNumber.

Вхід:
a = ComplexNumber(1, 2)

```
b = ComplexNumber(3, 4)
print(a + b)
```
Вихід: ComplexNumber(4, 6)

Вхід:
```
a = ComplexNumber(5, 7)
b = ComplexNumber(2, 3)
print(a * b)
```
Вихід: ComplexNumber(-11, 29) # (5+7i)*(2+3i) = 10+15i+14i+21i² = -11+29i

Вхід:
```
a = ComplexNumber(3, 4)
print(abs(a))
```
Вихід: 5.0

Частина 3. Складні завдання

Завдання 3.1. Контекстний менеджер для транзакцій бази даних

Створіть контекстний менеджер Transaction, який імітує роботу з транзакціями бази даних. При вході в контекст створюється "точка збереження", при успішному виході - "фіксація", при виникненні винятку - "відкат". Клас має зберігати список операцій (рядки), які виконуються в транзакції. Реалізуйте методи add_operation(operation) для додавання операції до транзакції.

Вхід:
```
with Transaction() as t:
    t.add_operation("INSERT INTO users VALUES (1, ‘John’)")
    t.add_operation("UPDATE accounts SET balance = 1000")
print("Транзакція успішна")
```
Вихід:

Початок транзакції

Транзакція успішна

Фіксація транзакції: [‘INSERT INTO users VALUES (1, ‘John’)’, ‘UPDATE accounts SET balance = 1000’]

Вхід:
```
try:
    with Transaction() as t:
        t.add_operation("DELETE FROM logs WHERE date < ‘2024-01-01’")
        raise ValueError("Помилка під час виконання")
except ValueError:
```

```
    print("Транзакція скасована")
```
Вихід:
Початок транзакції
Відкат транзакції через помилку
Транзакція скасована

Вхід:
```
with Transaction() as t:
    t.add_operation("CREATE TABLE test (id INT)")
    # Ніяких помилок
print("Створено таблицю")
```
Вихід:
Початок транзакції
Створено таблицю
Фіксація транзакції: [‘CREATE TABLE test (id INT)’]

4. Питання для самоперевірки

1. Що таке інкапсуляція в об’єктно-орієнтованому програмуванні та які механізми її реалізації існують у Python?
2. Яка різниця між захищеними (_attribute) та приватними (__attribute) атрибутами в Python? Навіщо потрібен механізм name mangling?
3. Поясніть призначення декоратора @property. Чим властивості кращі за звичайні методи getter/setter?
4. Які переваги дає використання властивостей (@property) порівняно з прямим доступом до атрибутів класу?
5. Для чого використовуються магічні методи __str__ та __repr__? Яка між ними різниця та коли кожен з них викликається?

ЛАБОРАТОРНА РОБОТА №17

Графічна бібліотека Turtle та візуалізація даних з Matplotlib

1. Мета

Опанувати навички візуалізації в Python з використанням графічної бібліотеки Turtle для створення програмних векторних зображень та бібліотеки Matplotlib для побудови наукових графіків і діаграм. Мета включає розуміння основних принципів графічного програмування (система координат, керування пером, колірні моделі) та навички представлення та аналізу даних у графічній формі.

2. Завдання

1. Навчитися створювати прості та складні геометричні фігури за допомогою Turtle.
2. Застосувати цикли та функції для генерації графічних патернів і візерунків.
3. Оволодіти техніками заповнення фігур кольором.
4. Навчитися будувати основні типи графіків (лінійні, стовпчасті, кругові, діаграми розсіювання) з використанням Matplotlib.
5. Опанувати налаштування зовнішнього вигляду графіків (осі, легенда, заголовки, мітки).
6. Навчитися створювати складні композиції з декількох графіків (підграфіки).
7. Отримати навички читання даних із файлів та їх подальшої візуалізації.

3. Завдання для виконання

Частина 1. Легкі завдання

Завдання 1.1. Кольоровий квадрат із променем

Намалюйте квадрат зі стороною 120 пікселів. Кожна сторона квадрата повинна бути іншого кольору (наприклад, червона, зелена, синя, чорна). Від центру квадрата до кожного з його кутів має йти лінія (промінь) того ж кольору, що й відповідна сторона.

Вхід: Програма не має вхідних даних.

Вихід: Графічне вікно із зображенням: квадрат з чотирьох різнокольорових сторін та чотири лінії від центру до кутів.

Завдання 1.2. Шахівниця Turtle

За допомогою черепашки намалюйте шахівницю розміром 4x4 клітинки. Розмір однієї клітинки – 40 пікселів. Клітинки повинні чергуватися двома кольорами (наприклад, сірий та білий). Фон вікна встановіть у світлий колір.

Вхід: Програма не має вхідних даних.

Вихід: Графічне вікно із зображенням шахівниці 4x4.

Завдання 1.3. Сходинки до неба

Намалюйте сходи, що ведуть зліва направо вгору. Перша сходинка має висоту 10 пікселів і ширину 20 пікселів. Кожна наступна сходинка має висоту на 5 пікселів більшу за попередню, а ширина залишається 20 пікселів. Всього намалюйте 8 сходинок. Кожна сходинка – іншого відтінку сірого (від темного до світлого).

Вхід: Програма не має вхідних даних.

Вихід: Графічне вікно із зображенням сходів із 8 сходинок, що зростають у висоті та змінюють колір.

Частина 2. Середні завдання

Завдання 2.1. Квітка з n пелюстками

Напишіть програму, яка запитує у користувача ціле число n (кількість пелюсток, n >= 3). Програма має намалювати квітку, де кожна пелюстка є колом. Всі кола мають однаковий радіус (50 пікселів) і повинні торкатися один одного, утворюючи квітку навколо центру. Колір пелюсток має змінюватися плавно по колу (використовуйте кольорову модель HSV через colorsys або циклічну зміну RGB).

Вхід 1: n = 6 → Вихід: Квітка з 6 пелюстками-колами.

Вхід 2: n = 4 → Вихід: Квітка з 4 пелюстками-колами.

Вхід 3: n = 8 → Вихід: Квітка з 8 пелюстками-колами.

Завдання 2.2. Фрактал «Крива Коха» (сніжинка)

Реалізуйте малювання фрактала «Сніжинка Коха» рекурсивно. Глибину рекурсії level задає користувач (1 <= level <= 5). Намалюйте сніжинку на основі рівностороннього трикутника (довжина сторони початкового трикутника – 250 пікселів). Колір ліній на кожному рівні рекурсії має ставати світлішим (від темно-синього до світло-блакитного).

Вхід 1: level = 1 → Вихід: Рівносторонній трикутник.

Вхід 2: level = 2 → Вихід: Трикутник із заміненими середніми третинами сторін на «зірочки».

Вхід 3: level = 4 → Вихід: Деталізована сніжинка Коха.

Завдання 2.3. Анімований годинник Turtle

Намалюйте циферблат аналогового годинника (коло з розміткою 12 годин). Додайте стрілки: годинну, хвилинну та секундну. Запрограмуйте їх рух,

використовуючи поточний системний час (модуль datetime). Годинник має оновлюватися (рухатися) у реальному часі. Використовуйте turtle.ontimer() для анімації.

Вхід: Програма не має вхідних даних.

Вихід: Графічне вікно із циферблатом та трьома рухомими стрілками, які показують поточний час.

Частина 3. Складні завдання

Завдання 3.1. Інтерактивний малювальник Turtle

Створіть інтерактивну програму-малювальник. Користувач повинен керувати черепашкою за допомогою клавіш:

- W, A, S, D – рух вперед, вліво, назад, вправо.
- Q, E – поворот вліво/вправо на 15 градусів.
- Пробіл – підняти/опустити перо (перемикач).
- C – очистити екран.
- 1, 2, 3 – змінити колір пера на червоний, зелений, синій.
- R – змінити колір заповнення на випадковий (при заповненні фігури).
- F – увімкнути/вимкнути режим автоматичного заповнення замкнутої фігури.

Програма має виводити коротку інструкцію на самому графічному вікні.

Вхід: Взаємодія користувача з програмою через клавіатуру.

Вихід: Інтерактивне графічне вікно, в якому користувач може малювати та керувати процесом.

4. Питання для самоперевірки

1. Які дві основі функції керування станом пера в Turtle і для чого вони використовуються?
2. Як розрахувати кут повороту черепашки для малювання правильного n-кутника? Наведіть формулу.
3. Що відбудеться, якщо команди t.begin_fill() та t.end_fill() у Turtle розмістити в різних частинах програми або забути одну з них?
4. Яка різниця між об’єктно-орієнтованим (fig, ax = plt.subplots()) та процедурним (plt.plot()) підходами до побудови графіків у Matplotlib? Який підхід є кращим для складних візуалізацій і чому?
5. Як додати легенду до графіка в Matplotlib? Які дві обов’язкові умови для її появи?

ЛАБОРАТОРНА РОБОТА №18
Створення GUI з Tkinter та CustomTkinter

1. Мета

Опанувати базові навички створення графічного інтерфейсу користувача (GUI) у мові програмування Python за допомогою бібліотеки Tkinter та її сучасного розширення CustomTkinter. Студенти навчаться розміщати віджети (елементи управління) у вікні, обробляти події та створювати інтуїтивно зрозумілі програми з візуальним інтерфейсом.

2. Завдання

1. Вивчити базові принципи роботи з бібліотеками Tkinter та CustomTkinter.
2. Навчитися створювати головне вікно програми та налаштовувати його властивості.
3. Опанувати роботу з основними віджетами (кнопки, мітки, поля введення тощо).
4. Навчитися розміщувати віджети у вікні за допомогою геометр-менеджерів Pack, Grid та Place.
5. Реалізувати обробку подій (натискання кнопок, введення тексту).
6. Створити прості програми з GUI, які виконують обчислення або обробляють дані, введені користувачем.

3. Завдання для виконання
Частина 1. Легкі завдання

Завдання 1.1. Вітальна картка

Створіть програму, яка відображає вітальну картку. Вікно розміром 400x300 пікселів має заголовок "Моя вітальна картка". У вікні розмістіть:

- Заголовок "Вітаю!" шрифтом Arial 18 пунктів
- Ваше ім’я під заголовком
- Назву вашої групи
- Кнопку "Закрити" яка завершує роботу програми

Приклад виходу: Вікно з вказаними елементами. При натисканні кнопки "Закрити" програма завершується.

Завдання 1.2. Конвертер температури

Створіть програму для конвертації градусів Цельсія в Фаренгейти. Вікно має містити:

- Поле для введення температури в Цельсіях
- Кнопку "Конвертувати"
- Мітку для відображення результату в Фаренгейтах

Формула конвертації: F = C × 9/5 + 32
Вхід: 0 → Вихід: 32.0
Вхід: 100 → Вихід: 212.0
Вхід: -40 → Вихід: -40.0

Завдання 1.3. Простий лічильник

Створіть програму з лічильником, який має:

- Мітку з поточним значенням лічильника (починається з 0)
- Кнопку "+" для збільшення значення на 1
- Кнопку "-" для зменшення значення на 1
- Кнопку "Скинути" для повернення до 0

Приклад роботи: Початкове значення: 0. При 3 натисканнях "+": 3. Потім 1 натискання "-": 2. Натискання "Скинути": 0.

Частина 2. Середні завдання

Завдання 2.1. Менеджер завдань (To-Do List)

Створіть простий менеджер завдань. Вікно має:

- Поле для введення нового завдання
- Кнопку "Додати" для додавання завдання до списку
- Список (Listbox) з усіма завданнями
- Кнопку "Видалити" для видалення вибраного завдання
- Кнопку "Очистити всі" для видалення всіх завдань
- Лічильник кількості завдань

Приклад роботи: Додати "Купити молоко" → список: "Купити молоко", лічильник: 1. Вибрати та натиснути "Видалити" → список порожній.

Завдання 2.2. Редактор тексту з форматуванням

Створіть простий текстовий редактор з можливістю форматування:

- Багаторядкове текстове поле (Text)
- Кнопки для форматування: "Жирний", "Курсив", "Підкреслений"
- Вибір розміру шрифту (Combobox з варіантами: 10, 12, 14, 16, 18)
- Вибір кольору тексту (3 кнопки: чорний, синій, червоний)
- Кнопки "Копіювати", "Вставити", "Очистити"

Приклад роботи: Введення тексту, вибір "Жирний" та "16" → текст стає жирним зі шрифтом 16 пунктів.

Частина 3. Складні завдання

Завдання 3.1. Текстовий редактор з меню та панеллю інструментів

Створіть повноцінний текстовий редактор з такими елементами:

1. **Головне меню:**

- Файл (Новий, Відкрити, Зберегти, Зберегти як, Вийти)
- Редагування (Вирізати, Копіювати, Вставити, Видалити, Вибрати все)
- Формат (Шрифт, Розмір, Колір, Вирівнювання)
- Довідка (Про програму)

2. **Панель інструментів** з іконками/кнопками для основних дій
3. **Статусний рядок** з інформацією:

- кількість символів
- кількість слів
- поточна позиція курсора

4. **Основна область** - багаторядкове текстове поле з прокруткою
5. **Функціонал:**

- робота з файлами (відкриття, збереження)
- пошук та заміна тексту
- друк документу (симуляція)
- автозбереження кожні 5 хвилин

Приклад роботи: Створення документа, форматування, збереження в .txt файл, статистика в статусному рядку.

4. Питання для самоперевірки

1. Яка основна функція методу mainloop() в Tkinter і чому вона критично важлива?
2. Назвіть три основні геометр-менеджери Tkinter та в чому їхні ключові відмінності?
3. Що таке callback-функція (функція зворотного виклику) і як вона пов’язана з обробкою подій у GUI?
4. Як правильно обробляти помилки введення даних користувачем (наприклад, введення тексту замість числа)? Наведіть приклад коду.
5. Поясніть різницю між tk.Entry та tk.Text. У яких випадках краще використовувати кожен з них?

ЛАБОРАТОРНА РОБОТА №19
Фінальний проект: GUI-додаток з використанням Tkinter/CustomTkinter

1. Мета

Систематизувати та застосувати знання, отримані протягом курсу, для розробки повноцінного, функціонального графічного додатку з інтуїтивно зрозумілим інтерфейсом. Проект має продемонструвати вміння студента інтегрувати основні концепції програмування (змінні, структури даних, функції, ООП, робота з файлами, обробка винятків) у практичному продукті.

Тип роботи

Індивідуальний або командний (2 особи) підсумковий проект. Проект складається з послідовних етапів, кожен з яких має чіткі критерії виконання та оцінювання.

Прикладні теми проектів (на вибір)

Студент може обрати одну з наведених тем або запропонувати свою (з обов'язковим затвердженням викладачем). Приклад власної теми має включати схожий рівень складності.

1. **Персональний менеджер завдань (To-Do List):** додавання, видалення, редагування завдань, відмітка про виконання, сортування за пріоритетом/датою, збереження списку у файл.
2. **Простий текстовий редактор:** відкриття, збереження, створення нових .txt файлів. Опціонально: пошук по тексту, зміна шрифту.
3. **Калькулятор розрахунку витрат (Budget Tracker):** введення статей доходів/витрат, категорії, суми, дати. Візуалізація за допомогою простих кругових або стовпчастих діаграм (бібліотека matplotlib або прості засоби Tkinter Canvas). Збереження даних.
4. **Додаток для квізу (вікторини):** завантаження питань та варіантів відповідей з файлу (JSON/CSV), таймер, підрахунок балів, відображення результату.
5. **Гра "Вгадай число" або "Хрестики-нулики" з історією:** гра проти комп'ютера або другого гравця, ведення таблиці рекордів, яка зберігається між сесіями.

2. Етапи виконання та критерії оцінювання

Загальна максимальна оцінка: 100 балів.

Етап 1: Планування та проектування (10 балів)

Завдання: створити технічне завдання та макет вашого додатку.

1.1. (3 бали) Вибір та обґрунтування теми проекту. Короткий опис (5-7

речень): що робить додаток, яку проблему вирішує?

1.2. (4 бали) Створення ескізу інтерфейсу (wireframe). Можна намалювати від руки, в Paint, або використати спеціальні інструменти (diagrams.net). Ескіз має показати розташування **всіх** основних віджетів (кнопок, полів вводу, міток, списків тощо) на головному та додаткових вікнах.

1.3. (3 бали) Опис функціональних вимог у вигляді нумерованого списку. Наприклад: "1. Програма повинна дозволяти користувачу ввести нову задачу у текстове поле. 2. При натисканні кнопки ‘Додати’ задача має з’явитися у списку. 3. Користувач повинен мати можливість видалити обрану задачу."

Етап 2: Розробка базового інтерфейсу (GUI) (15 балів)

Завдання: реалізувати "макет" програми без активної логіки, згідно з ескізом.

2.1. (5 балів) Створення головного вікна програми з заголовком та базовими налаштуваннями (розмір, можливість зміни розміру).

2.2. (5 балів) Коректне розміщення всіх запланованих віджетів за допомогою менеджерів компонування (pack, grid або place). Обов’язково використати мінімум **5 різних типів** віджетів (наприклад: Label, Entry, Button, Listbox/Treeview, Combobox, Checkbutton, Text, Canvas).

2.3. (5 балів) Базова стилізація: вибір кольорової схеми (фон, кнопки), налаштування шрифтів для заголовків та основного тексту. Інтерфейс має бути акуратним та зручним для сприйняття.

Етап 3: Реалізація бізнес-логіки та функціоналу (40 балів)

Завдання: "оживити" інтерфейс, додавши обробку дій користувача та основний функціонал програми.

3.1. (10 балів) Написання функцій-обробників подій (event handlers) для всіх інтерактивних елементів (кнопки, поля вибору тощо). Код має реагувати на дії користувача (наприклад, виводити повідомлення при натисканні).

3.2. (10 балів) Реалізація валідації введених користувачем даних (перевірка на порожність, правильний тип даних, діапазон значень). Виведення зрозумілих повідомлень про помилку (наприклад, у Label або messagebox).

3.3. (10 балів) Реалізація роботи з файлами для збереження стану програми (наприклад, списку завдань, історії операцій). Дані мають зберігатися при закритті та завантажуватися при запуску. Формат: JSON, CSV або текстовий файл.

3.4. (10 балів) Інтеграція всієї логіки програми. Усі модулі (інтерфейс, обробка даних, робота з файлами) мають працювати як єдине ціле. Додаток виконує свою основну функцію (керує завданнями, веде бюджет і т.д.).

Етап 4: Обробка помилок та тестування (10 балів)

Завдання: забезпечити стабільну роботу програми в нестандартних

ситуаціях.

4.1. (5 балів) Використання конструкцій try-except-finally для всіх критичних операцій: робота з файлами (відсутній файл, відмова в доступі), потенційні помилки перетворення типів (ValueError), ділення на нуль тощо.

4.2. (5 балів) Проведення ручного тестування: перевірити всі сценарії використання (коректні та некоректні). Створити короткий звіт або коментар у коді/README про те, що було протестовано.

Етап 5: Документація та якість коду (20 балів)

5.1. Документація (10 балів):

- README.md у корені проекту з назвою, описом, **скріншотами інтерфейсу**, інструкцією зі встановлення (потрібні бібліотеки) та запуску.
- Чіткі інлайнові коментарі для неочевидних ділянок коду.
- Docstrings для всіх функцій та класів (короткий опис, аргументи, що повертає).

5.2. Якість коду (10 балів):

- Логічне розділення коду на функції/класи/модулі (наприклад, окремий файл для логіки та окремий для GUI).
- Відсутність дублювання коду (DRY principle).
- Читабельність: зрозумілі назви змінних і функцій, дотримання стилю PEP 8 (відступи, довжина рядків).
- Використання ООП підходу (наприклад, створення класу для головного додатку або для об'єктів даних) – **заохочується, але не є обов'язковим для всіх тем**.

Етап 6: Презентація та здача (5 балів)

6.1. (5 балів) Підготовка до захисту проекту. Студент має бути готовий:

- продемонструвати основний функціонал програми;
- пояснити ключові архітектурні рішення;
- розповісти про виниклі складнощі та способи їх подолання;
- відповісти на запитання щодо коду.

Обов'язкові технічні вимоги до проекту

- використання стандартної бібліотеки tkinter або сучасної customtkinter;
- наявність мінімум **5 різних типів** віджетів;
- обробка подій користувача (кліки, введення тексту);
- збереження та завантаження даних у файл (стан програми не має втрачатися після перезапуску);
- наявність обробки помилок (try-except);
- код повинен бути чистим, добре структурованим і закоментованим;
- проект має бути розміщений на **GitHub** (або аналогічному сервісі);
- репозиторій **обов'язково** має містити файл README.md.

3. Форма здачі проекту

1. **Посилання на публічний репозиторій GitHub.** Назва репозиторію: Final-Project-GUI-[Прізвище] (наприклад, Final-Project-GUI-Petrenko).
2. **Репозиторій має містити:**
   - повний вихідний код проекту;
   - файл README.md (за описом вище);
   - файли даних (якщо потрібно) або папку з ресурсами;
   - файл requirements.txt зі списком залежностей (наприклад, customtkinter), якщо вони використовуються.
3. **Усна презентація та захист проекту** на останньому занятті або за розкладом.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Алгоритмізація і програмування: метод. реком. до виконання практичних робіт здобувачами вищої освіти ступеня «молодший бакалавр» факультету менеджменту спеціальності 122 «Комп'ютерні науки» денної форми навчання / уклад. Ю. В. Волосюк. Миколаїв: МНАУ, 2021. 52 с.

URL: https://dspace.mnau.edu.ua/jspui/handle/123456789/10866

2. Алгоритмізація та програмування. Частина 1. Базові концепції програмування. Лабораторний практикум: навчальний посібник / уклад. В. В. Романов, Т. І. Просянкіна-Жарова, О. Ю. Безносик. Київ: КПІ ім. Ігоря Сікорського, 2022. 150 с.

URL: https://ela.kpi.ua/server/api/core/bitstreams/596d62b6-8251-4013-9c48-960e10b6dfb2/content.

3. Бандоріна Л. М., Климкович Т. О., Удачина К. О. Основи алгоритмізації та програмування: навч. посібник. Дніпро: УДУНТ, 2022. 158 с.

URL: https://crust.ust.edu.ua/server/api/core/bitstreams/bfb9f823-172b-49bf-92a2-48bbbef5f113/content.

4. Беррі П. Head First Python. Легкий для сприйняття довідник. Харків: Фабула, 2023. 624 с.

5. Булгакова О. С., Зосімов В. В., Ходякова Г. В. Алгоритмізація і програмування: теорія та практика: навчальний посібник для дистанційного навчання. Миколаїв: СПД Румянцева, 2021. 138 с.

URL: http://dspace.mdu.edu.ua/jspui/handle/123456789/931.

6. Васильєв О. М. Програмування мовою Java. Тернопіль: Богдан НК, 2022. 696 с.

7. Висоцька В. А., Оборська О. В. Python: алгоритмізація та програмування: навчальний посібник. Львів: Новий світ-2000, 2021. 514 с.

8. Григорович В. Г. Алгоритмізація та програмування. Частина 1: навчальний посібник. Львів: Магнолія 2006, 2023. 357 с.

9. Злобін Г. Г. Основи алгоритмізації та програмування мовою Сі: підручник. Київ: Каравела, 2022. 168 с.

10. Кублій Л. І. Алгоритмізація та програмування: Практикум: навч. посіб. для здобувачів ступеня бакалавра за спеціальністю 122 "Комп'ютерні науки" / ; КПІ ім. Ігоря Сікорського. Київ: КПІ ім. Ігоря Сікорського, 2019. 209 с.

URL: https://ela.kpi.ua/server/api/core/bitstreams/a2c179fd-fb1f-4536-aae7-08a8928f8569/content

11. Лосєв М. Ю., Федорченко В. М. Програмування мовою Python: навчальний посібник. Харків; Львів: Новий Світ – 2000, 2024. 178 с.

12. Мартін Р. Чистий кодер. Кодекс поведінки для професійних розробників. Харків: Фабула, 2023. 256 с.

13. Пастернак І. І., Костик А. Т. Інструментальні засоби веб-технологій: навчальний посібник. Львів: Магнолія, 2024. 197 с.

14. Ришковець Ю. В., Висоцька В. А. Алгоритмізація та програмування. Частина 2: навчальний посібник. Львів: Новий Світ – 2000, 2020. 320 с.

15. Рудий Т. В., Паранчук Я. С., Сеник В. В. Алгоритмізація та програмування. Частина 2. Модульне програмування: навчальний посібник. Львів: Львівський державний університет внутрішніх справ, 2024. 176 с.

URL: https://dspace.lvduvs.edu.ua/handle/1234567890/6994

16. Рудий Т. В., Паранчук Я. С., Сеник В. В. Алгоритмізація та програмування. Частина 1. Структурне програмування: навчальний посібник. Львів: Львівський державний університет внутрішніх справ, 2023. 240 с.

URL: https://dspace.lvduvs.edu.ua/handle/1234567890/5515

17. Селіверстов Р., Мельничин А. Основи програмування мовою Python: навчальний посібник. Львів: ЛНУ, 2020. 190 с.

18. Спірінцева О. В., Литвинов О. А., Герасимов В. В. Java-технології та мобільні пристрої. Алгоритми і структури даних: навч. посіб. Дніпро: ДНУ, 2016. 140 с.

19. Трофименко О. Г., Манаков С. Ю., Ларін Д. Г. Основи програмної інженерії: навч.-метод. посібник. Одеса: Фенікс, 2022. 197 с.

URL: https://dspace.onua.edu.ua/items/25698e60-3d50-42c0-a87a-27bd8b22a54d

20. Трофименко О. Г., Прокоп Ю. В., Логінова Н. І., Задерейко О. В. C++. Алгоритмізація та програмування: підручник. Одеса: Фенікс, 2019. 477 с.

URL: https://dspace.onua.edu.ua/items/6c40c92b-c7d4-43ae-93da-e195f3daf3d1

21. Щербаков О. В., Парфьонов Ю. Е., Федорченко В. М. Основи об`єктно-орієнтованого програмування: навчальний посібник. Харків: ХНЕУ, 2019. 237 с. URL: http://surl.li/oecpq.

Навчальне видання

АЛГОРИТМІЗАЦІЯ ТА ПРОГРАМУВАННЯ

Методичні рекомендації до виконання лабораторних робіт
для здобувачів першого (бакалаврського) рівня вищої освіти
ОПП «Комп'ютерні науки» спеціальності F3 «Комп'ютерні науки»
денної форми здобуття вищої освіти

Укладач: **Пархоменко** Олександр Юрійович

Формат 60х84 1/16. Ум. друк. арк. 4,0.
Тираж 20 прим. Зам. № ____

Надруковано у видавничому відділі
Миколаївського національного аграрного університету
54008, м. Миколаїв, вул. Георгія Гонгадзе, 9

Свідоцтво суб'єкта видавничої справи ДК № 4490 від 20.02.2013 р.