МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ
«ДНІПРОВСЬКА ПОЛІТЕХНІКА»

ФАКУЛЬТЕТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
Кафедра системного аналізу та управління

Т.В. Хом’як, Ю.О. Шевченко, Д.М. Гаранжа

ПРОГРАМУВАННЯ ТА АЛГОРИТМІЧНІ МОВИ

Методичні рекомендації до виконання лабораторних робіт
для здобувачів ступеня бакалавра
освітньо-професійної програми «Системний аналіз»
зі спеціальності 124 Системний аналіз

У 2 частинах

Частина 1

Дніпро
НТУ «ДП»
2026

Програмування та алгоритмічні мови [Електронний ресурс] : методичні рекомендації до виконання лабораторних робіт для здобувачів ступеня бакалавра освітньо-професійної програми «Системний аналіз» зі спеціальності 124 Системний аналіз. У 2 ч. Ч 1 / уклад.: Т.В. Хом'як, Ю.О. Шевченко, Д.М. Гаранжа ; М-во освіти і науки України, Нац. техн. ун-т «Дніпровська політехніка». – Дніпро : НТУ «ДП», 2026. – 63 с.

Укладачі:

Т.В. Хом'як, канд. фіз.-мат. наук, доц.;
Ю.О. Шевченко, асистент;
Д.М. Гаранжа, ст. викл.

Затверджено науково-методичною комісією зі спеціальності F4 Системний аналіз та наука про дані (протокол № 2 від 05.02.2026) за поданням кафедри системного аналізу та управління (протокол № 2 від 05.02.2026).

Наведено теоретичні відомості за темами, приклади вирішення задач, завдання до лабораторних робіт з критеріями їх оцінювання, контрольні питання, список рекомендованих джерел.

Орієнтовано на активізацію навчальної діяльності здобувачів ступеня бакалавра спеціальностей 124 Системний аналіз і закріплення практичних навичок у засвоєнні дисципліни «Програмування та алгоритмічні мови».

Відповідальний за випуск завідувач кафедри системного аналізу та управління Т.А. Желдак, канд. техн. наук, доц.

ЗМІСТ

ВСТУП

Методичні рекомендації орієнтовані на використанням будь-якого середовища, де є можливість розробляти та тестувати програмний код написаний на Python (наприклад PyCharm або Google Colab). Варіанти лабораторних робот представлені після теоретичних відомостей до кожної теми. Предметні області для лабораторних робіт обрані такими, що зрозумілі здобувачам і не потребуватимуть спеціальних знань.

Здобувач обирає варіант лабораторної роботи відповідно до номера свого прізвища у списку групи. До лабораторного заняття здобувач зобов'язаний прочитати методичні рекомендації до лабораторної роботи, прослухати лекцію і виконати самостійно завдання. Під час лабораторного заняття здобувач показує викладачеві результати роботи. Коли робота виконана, здобувач повинен її захистити. Захист полягає у виконанні практичного завдання або відповіді на питання за темою лабораторної роботи.

За результатами виконання кожної роботи здобувач повинен оформити звіт. Звіти оформляються за допомогою текстового редактора на папері формату А4 відповідно до вимог стандартів на оформлення технічної документації.

Мета дисципліни – формування у здобувачів вищої освіти компетентностей щодо для розробки алгоритмів та написання програмного коду за допомогою сучасних мов програмування. Дисципліна орієнтована на вивчення мови програмування Python з використанням середовища PyCharm та Google Colab.

Під час вивчення дисципліни здобувачі набувають таких дисциплінарних результатів навчання:

- будувати ефективні розгалужені та циклічні алгоритми
- створювати функції для вирішення задач
- обробляти одновимірні та двовимірні масиви даних (списки)
- розробляти програмний код мовою Python із застосуванням різних структур даних для розв’язання задач
- обробляти текстову інформацію найпоширеніших форматів представлення із записуванням (зчитуванням) в файл
- створювати синтаксично правильні програми методами структурного програмування з використанням винятків та виключень
- розробляти класи, методи класу, застосовувати механізм успадкування та множинного успадкування
- отримувати результати тестування програмних кодів
- розробляти графічний інтерфейс користувача для візуалізації результатів аналізу
- виконувати символьні обчислення для вирішення задач математичного аналізу, лінійної алгебри, дискретної математики

Це видання має допомогти здобувачам у виконанні лабораторних робіт з дисципліни. Розроблені завдання мають допомогти здобувачам вказаної спеціальності у застосуванні на практиці навичок програмування та візуалізації алгоритмів.

Практичні навички, отримані при вивченні даної дисципліни, можуть бути використані при розробці, реалізації, тестуванні, впровадженні, супроводженні, експлуатації програмних засобів та подальшій роботі з даними.

Лабораторна робота № 1
Розробка лінійних, розгалуджених та циклічних алгоритмів

Мета: закріпити теоретичні знання і розвинути практичні навички з розробки та візуалізації алгоритмів.

Теоретичні відомості

Алгоритм – це опис процесу вирішення того чи іншого завдання. Алгоритмом називається кінцевий набір правил, розташованих у певному логічному порядку, що дозволяє виконавцю вирішувати будь-яку конкретну задачу з деякого класу однотипних задач.

Алгоритми мають ряд важливих властивостей:

Скінченність. Алгоритм має завжди завершуватись після виконання скінченної кількості кроків. Процедуру, яка має решту характеристик алгоритму, без, можливо, скінченності, називають методом обчислень.

Дискретність. Процес, що визначається алгоритмом, можна розчленувати (розділити) на окремі елементарні етапи (кроки), кожен з яких називається кроком алгоритмічного процесу чи алгоритму.

Визначеність. Кожен крок алгоритму має бути точно визначений. Дії, які необхідно здійснити, повинні бути чітко та недвозначно визначені для кожного можливого випадку.

Вхідні дані. Алгоритм має деяку кількість (можливо, нульову) вхідних даних, тобто, величин, заданих до початку його роботи або значення яких визначають під час роботи алгоритму.

Вихідні дані. Алгоритм має одне або декілька вихідних даних, тобто, величин, що мають досить визначений зв’язок із вхідними даними.

Ефективність. Алгоритм вважають ефективним, якщо всі його оператори досить прості для того, аби їх можна було точно виконати за скінченний проміжок часу за допомогою олівця та аркушу паперу.

Найбільш часто вживані способи представлення алгоритмів:

Спосіб	**Опис**
Словесний спосіб (Природна мова)	Це опис алгоритму звичайною людською мовою. Він не має суворих стандартів і часто використовується для пояснення ідеї «на пальцях». • **Переваги:** Зрозуміло будь-кому, не потребує спеціальних знань. • **Недоліки:** Неточність, багатослівність та можливість неоднозначного трактування.
Графічний спосіб (Блок-схеми)	Це представлення алгоритму за допомогою геометричних фігур (блоків), з'єднаних лініями переходу. Кожна фігура має своє стандартне значення (овал — початок/кінець, ромб — умова тощо).

	• **Переваги:** Наочність, легке відстеження логічних зв'язків та розгалужень. • **Недоліки:** Громіздкість при описі великих і складних програм.
Псевдокод	Це напівформалізована мова, яка за структурою нагадує мову програмування (використовує конструкції “якщо-то-інакше”, “поки”, “для”), але не має жорсткого синтаксису. Псевдокод орієнтований на людину, а не на комп'ютер. **Приклад:** ЯКЩО температура > 30 ТО ДРУКУВАТИ "Спекотно" ІНАКШЕ ДРУКУВАТИ "Нормально"
Програмний спосіб (Текст програми)	Це запис алгоритму конкретною мовою програмування (наприклад, Python, C++, Java). Це єдина форма, яку комп'ютер може виконати безпосередньо (після компіляції чи інтерпретації).

Хоча існує багато методів, найбільш часто вживаним і універсальним способом формалізації алгоритмів є **блок-схема (графічний спосіб)**.

Саме цей метод став стандартом у навчанні, системному аналізі та технічній документації завдяки своїй наочності. Проте варто розуміти, що в професійному середовищі розробники часто використовують **псевдокод** як більш швидку альтернативу.

Ось чому блок-схема тримає першість у більшості випадків:

1. Візуальний стандарт (ISO та ДСТУ)

Блок-схема — це не просто малюнок, а міжнародно визнана мова. Існують конкретні стандарти (наприклад, **ISO 5807:1985**), які чітко визначають, яка фігура за що відповідає. Це робить схему зрозумілою для будь-якого фахівця у світі, незалежно від того, якою мовою він програмує.

2. Етапність використання

Блок-схема є "золотою серединою" між людською мовою та програмним кодом:

- **Проектування:** Спочатку малюємо схему, щоб зрозуміти логіку.
- **Реалізація:** Перетворюємо схему на код.
- **Документація:** Додаємо схему до звіту (як у вашій лабораторній), щоб пояснити, як працює програма.

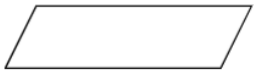
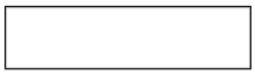
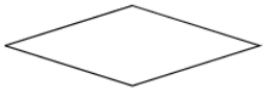
3. Чому не код чи слова?

- **Словесний опис** занадто неточний для складних систем.

- **Програмний код** занадто деталізований і "засмічений" синтаксичними особливостями конкретної мови (дужками, крапками з комою), що заважає бачити загальну структуру.
- **Псевдокод** — чудовий інструмент для досвідчених програмістів, але він не дає такої наочності для відстеження розгалужень та циклів, як графіка.

2. Побудова алгоритмів з використанням блок-схем

Основні графічні елементи використовуються при побудові блок-схем:

Елемент	Назва	Функція
Овал (або скруглений прямокутник) 	Термінатор	Початок або кінець алгоритму.
Паралелограм 	Введення/ Виведення	Отримання даних від користувача або друк результату.
Прямокутник 	Дія	Виконання математичних операцій або присвоєння.
Ромб 	Розгалудження	Перевірка умови (логічний вузол).
Шестикутник 	Цикл	Заголовок циклу

Типові структури алгоритмів:

1. Лінійні алгоритми

Це найпростіший вид структури, де дії виконуються **послідовно**, одна за одною, у порядку їх запису. У таких алгоритмах немає умов або повторень.

- **Ключова особливість:** Кожен блок виконується рівно один раз.
- **Приклад:** Обчислення площі прямокутника за двома сторонами ($S = a\ b$).

2. Розгалужені алгоритми (Алгоритми вибору)

Ця структура використовується, коли шлях виконання програми залежить від певної **умови** (логічного виразу). Залежно від того, чи є умова істинною (True) чи хибною (False), програма обирає один із напрямків.

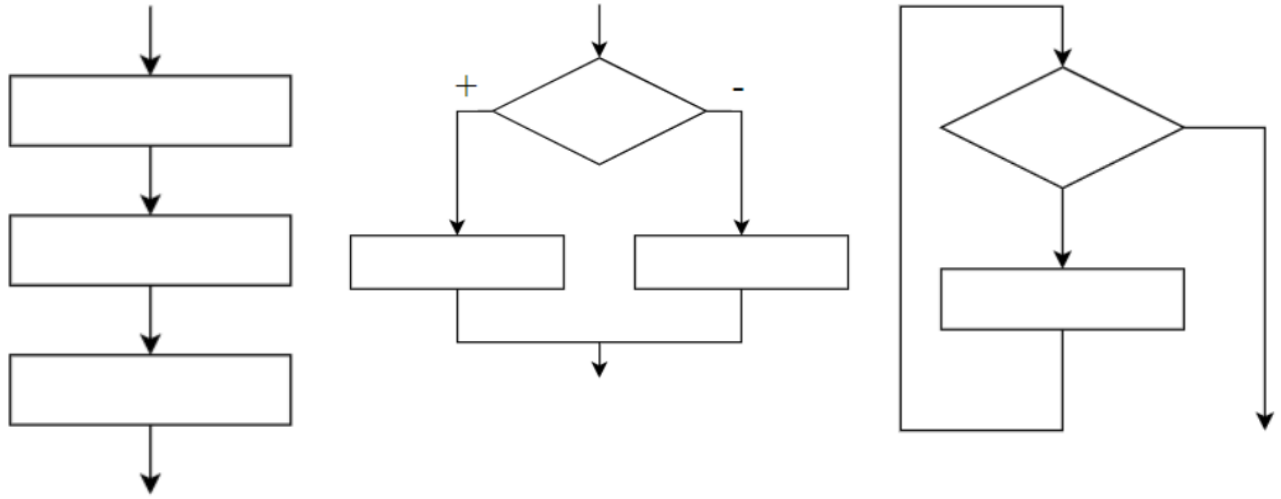
- **Типи розгалужень:**
  - **Повне:** Має дві гілки («ТАК» та «НІ»).
  - **Неповне:** Дія виконується лише у разі виконання умови, інакше програма просто йде далі.
  - **Множинний вибір:** Вибір одного варіанта з багатьох (наприклад, оператор switch або case).

- **Приклад:** Визначення, чи є число парним.

3. Циклічні алгоритми

Алгоритми циклічної структури передбачають **багаторазове повторення** певної послідовності дій (тіла циклу), поки виконується (або не виконується) певна умова.

- **Види циклів:**
  - **Цикл з передумовою (while):** Умова перевіряється *перед* виконанням тіла циклу. Якщо умова відразу хибна, цикл не виконається жодного разу. (актуальний для мови програмування Python)
  - **Цикл з післяумовою (do-while):** Умова перевіряється *після* виконання тіла. Цикл гарантовано виконається хоча б один раз. (фактично відсутній тип для мови програмування Python)
  - **Цикл з параметром (for):** Використовується, коли кількість повторень відома заздалегідь. (актуальний для мови програмування Python)



Лінійний алгоритм Розгалужений алгоритм Циклічний алгоритм

Застосунки для складання блок-схем

Для створення блок-схем існує безліч інструментів: від простих онлайн-редакторів до потужних професійних середовищ проектування. Вибір залежить від того, чи потрібна вам швидка схема для звіту, чи складна архітектурна діаграма.

Ось найпопулярніші застосунки, розподілені за категоріями:

1. Спеціалізовані онлайн-сервіси (найпопулярніші)

Ці інструменти працюють у браузері, не потребують встановлення та мають величезні бібліотеки стандартних символів.

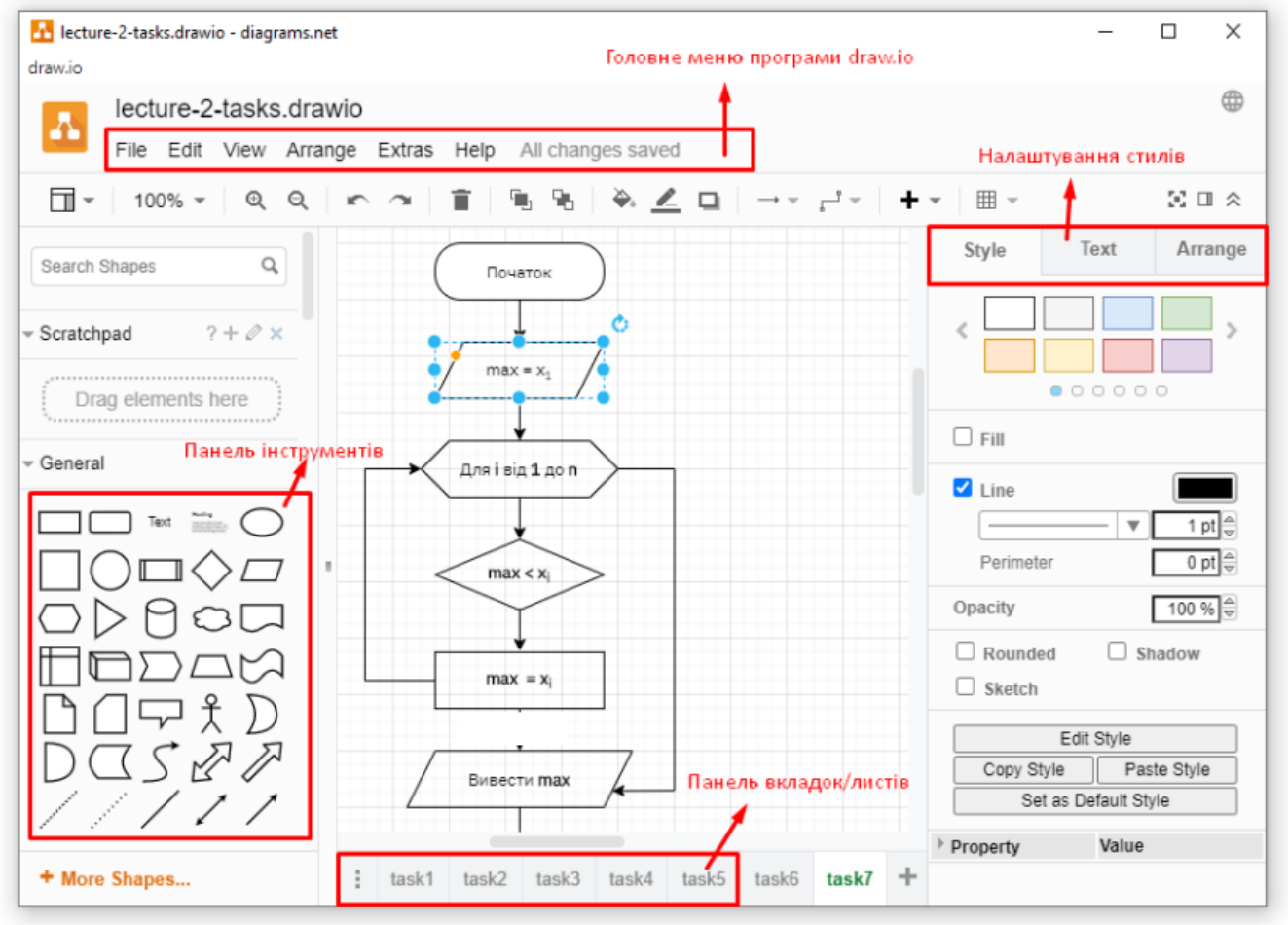
- **Draw.io (diagrams.net):**

Переваги: Повністю безкоштовний, має відкритий код.

Особливості: Інтегрується з Google Drive, OneDrive та GitHub. Має всі необхідні блоки за стандартом ISO.

Досить зручною є функція збереження діаграм для публікації у веб або на паперових носіях формати файлів (png, jpeg, svg, pdf, html, xml), а також вбудовування їх у різні типи документів

Графічний інтерфейс *draw.io* є зручним у користуванні і дозволяє доповнити перелік доступних зображень, блоків, логотипів продуктів для створення схем будь-якого рівня складності.



2. Професійні десктопні програми

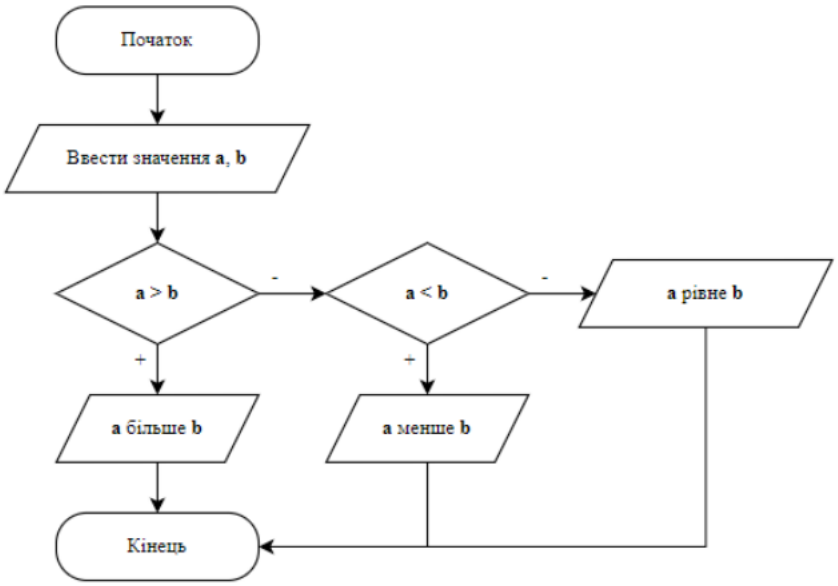
Використовуються для складних технічних проектів та великих схем.

- **Microsoft Visio:**
  - **Статус:** Промисловий стандарт.
  - **Особливості:** Величезна кількість інструментів для автоматизації, зв'язок схем із даними Excel, але програма платна і досить важка в освоєнні (в залежності від складності поставленного завдання).

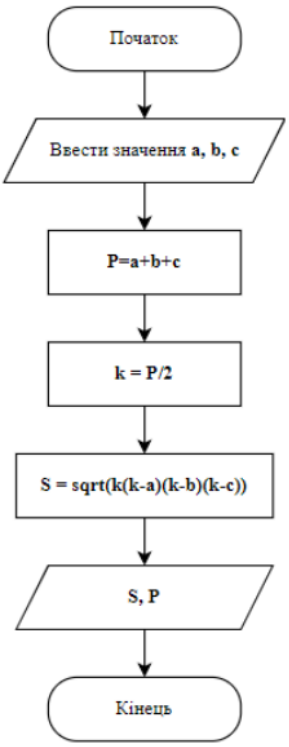
Кожен студент маяючи обліковий запис Microsoft365 може скористатись безкоштовною десктом або хмарною версією застосунку. Для потреб складання блок-схем в рамках поточного курсу можливостей цих програм абсолютно достатньо.

Приклади блок-схем алгоритмів

Приклад 1. Побудувати блок-схему алгоритму порівняння двох чисел.

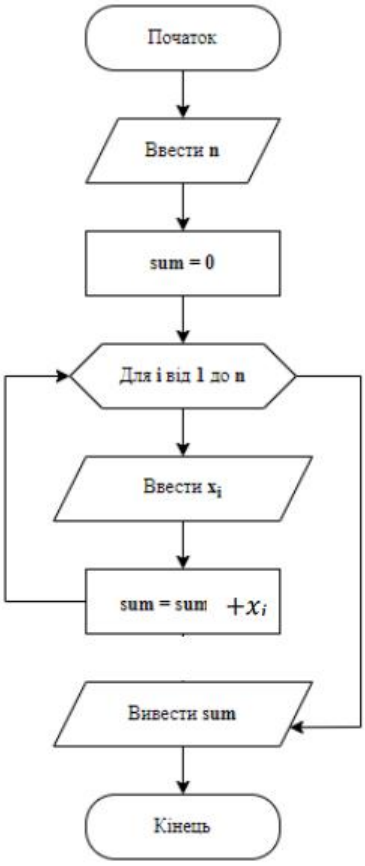


Приклад 2. Побудувати блок-схему алгоритму зходження периметра та площі трикутника за формулою Герона.



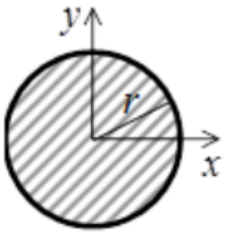
Приклад 3. Побудувати блок-схему алгоритму знаходження суми елементів у масиву.

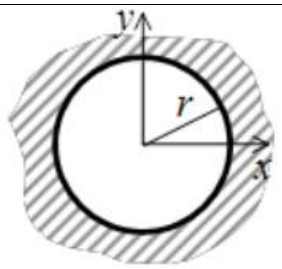
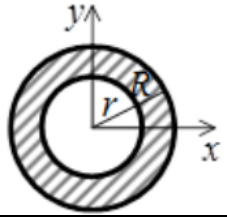
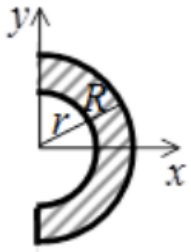
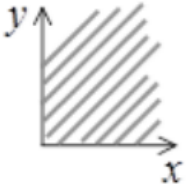
Введемо позначення: n – кількість елементів масиву, sum – змінна, що визначає суму, x_i – i-й елемент масиву.

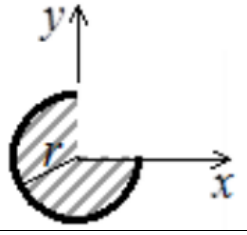
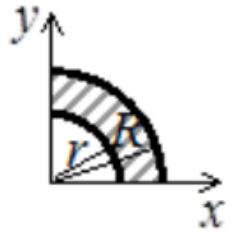
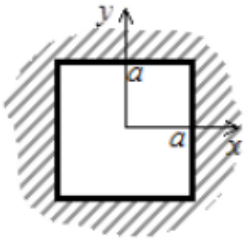


Завдання до лабораторної роботи 1

Розробити блок-схему для вирішення поставленої задачі відповідно варіанту.

Варіант	Завдання
1	Визначити, чи належить задана на площині точка заштрихованій області. 
2	Обчислити значення виразу $y = \begin{cases} \cos^2 x, 0 < x < 2 \\ 1 - \sin x^2, x \le 0, x \ge 2 \end{cases}$
3	Визначити, чи належить задана на площині точка заштрихованій області.

	
4	Обчислити значення виразу $$z=\begin{cases}\max(x,\ y),\ x<0;\\ \min(x,\ y),\ x\geq 0;\end{cases}$$
5	Визначити, чи належить задана на площині точка заштрихованій області. 
6	Обчислити значення виразу $$z=\begin{cases}\dfrac{ax+b}{x^2+y^2}+1,\ x^2+y^2\leq 1;\\ \dfrac{abxy}{x^2+y^2}-x^3,\ x^2+y^2>1;\end{cases}$$
7	Визначити, чи належить задана на площині точка заштрихованій області. 
8	Обчислити значення виразу $$y=\begin{cases}\dfrac{ax+b}{ax}+ax,\ x<-1;\\ ax+b^2,\ -1\leq x\leq 3.5;\\ \dfrac{ax+b}{ax}-a^2x,\ x>3.5;\end{cases}$$
9	Визначити, чи належить задана на площині точка заштрихованій області. 

10	Визначити, чи є серед цифр заданого натурального трьохзначного числа однакові.
11	Визначити, чи належить задана на площині точка заштрихованій області. 
12	Відомі координати трьох точок $A(x_1, y_1), B(x_2, y_2), C(x_3, y_3)$. З'ясувати, чи лежать ці точки на одній прямій.
13	Визначити, чи належить задана на площині точка заштрихованій області. 
14	Визначити, чи дорівнює сума двох перших цифр заданого натурального чотирьохзначного числа сумі двох останніх чисел.
15	Визначити, чи належить задана на площині точка заштрихованій області. 
16	На площині задані пряма $y = kx + b$ і точка $A(x, y)$ З'ясувати, чи належить точка А цій прямій.
17	Визначити, чи належить задана на площині точка заштрихованій області. 
18	На площині задані прямі $y = k_1x_1 - b_1$ і $y = k_2x_2 - b_2$. Визначити взаємне їх розташування на площині. Указівки: умова паралельності двох прямих $k_1 = k_2$; умова перпендикулярності двох прямих $1 + k_1k_2 = 0$
19	Відомі координати вершин трикутника $A(x_1, y_1), B(x_2, y_2), C(x_3, y_3)$.

	Визначити, чи трикутник рівнобедрений. При обчисленні застосувати формулу відстані між двома точками.
20	Сторони трикутника задані рівняннями прямих $y = k_1x_1 - b_1$, $y = k_2x_2 - b_2$ і $y = k_3x_3 - b_3$. Визначити, чи трикутник прямокутний. При обчисленні застосувати умову перпендикулярності прямих $1 + k_1k_2 = 0$, де k_1 і k_2 - коефіцієнти прямих, заданих рівняннями $y = k_1x_1 - b_1$, $y = k_2x_2 - b_2$
21	Обчислити значення виразу $$y = \begin{cases} 0, & \text{при } x < 0; \\ 1/(x^2+1), & \text{при } 0 \le x \le 1; \\ x^3, & \text{при } 1 < x \le 4; \\ 62 + \log_8 x, & \text{при } x > 4. \end{cases}$$
22	Відомі координати вершин чотирикутника $A(x_1, y_1), B(x_2, y_2), C(x_3, y_3), D(x_4, y_4)$. Визначити, чи цей чотирикутник – квадрат.
23	Обчислити значення виразу $$v = \begin{cases} xy, & \text{при } x < 0,\ y \le 0; \\ \operatorname{ctg}\dfrac{x}{y}, & \text{при } 0 \le x < 10,\ 0 < y \le 9; \\ -100, & \text{в інших випадках.} \end{cases}$$
24	Відомі координати вершин чотирикутника $A(x_1, y_1), B(x_2, y_2), C(x_3, y_3), D(x_4, y_4)$. Визначити, чи цей чотирикутник – ромб
25	Обчислити значення виразу $$z = \begin{cases} \dfrac{1}{x^2}, & \textit{при } x < -5 \\ \dfrac{1}{y^2}, & \textit{при } y > 5 \end{cases}$$
26	Обчислити значення виразу $$y = \begin{cases} 10, & \text{при } x < 0; \\ x(x^2+1), & \text{при } 0 \le x \le 1; \\ x\ , & \text{при } 1 < x \le 5; \end{cases}$$
27	Коло задано рівнянням $x^2 + y^2 = 25$. Визначити, чи належить задана точка колу.

Критерії оцінювання лабораторної роботи 1

За умови своєчасного та в повному обсязі виконаного завдання здобувач може отримати 100 балів (матеріал якісно викладено в логічній послідовності, без

помилок, зроблено висновки до виконаного завдання, виявлено підвищений рівень володіння матеріалом).

Бали	Оцінка	Характеристика
90–100	Відмінно	Алгоритм ідеально виконує задачу, дотримано всіх стандартів оформлення, помилки відсутні.
74–89	Добре	Задача розв'язана вірно, але є незначні зауваження до оформлення (відсутні стрілки, не скрізь підписані умови).
60–73	Задовільно	Логіка в цілому вірна, але допущені помилки в графічних символах (наприклад, переплутано блок процесу та введення) або схема надмірно заплутана.
Менше 60	Незадовільно	Алгоритм не розв'язує задачу, містить грубі логічні помилки (нескінченні цикли) або оформлений без врахування стандартів.

Контрольні питання

1. Дати визначення поняття «алгоритм».
2. Перелічіть властивості алгоритмів.
3. Якими способами можна представити алгоритм?
4. Які типові блоки для формування блок-схем алгоритмів ви знаєте? Як вони зображуються графічно?
5. Основні алгоритмічні структури?

Перелік рекомендованих джерел

1. Кормен, Томас Г. Вступ до алгоритмів: Переклад з англійської третього видання [укр.] Introduction to Algorithms: Third Edition: [пер. з англ.] / Томас Г. Кормен, Чарлз Е. Лейзерсон, Роналд Л. Рівест, Кліфорд Стайн, –К.: К. І. С., 2019. – 1288 с.
2. Руденко В., Жугастров О. Інформатика. Основи алгоритмізації та програмування мовою Python. Харків: Ранок, 2019. – 192 с.
3. DiagramDesigner https://logicnet.dk/DiagramDesigner/
4. Draw.IO https://www.diagrams.net/

Лабораторна робота № 2
Написання найпростіших програм в інтегрованому середовищі розробки IDLE, використання середовища PyCharm, Google Colab

Мета: навчитися писати найпростіші програми та закріпити основи синтаксису мови Python.

Теоретичні відомості

Одним з можливих середовищ розробки програм може бути використане середовище Google Colaboratory (https://colab.research.google.com/).

З точки зору розробника на Python, **Google Colaboratory** — це не просто «блокнот у хмарі», а повноцінна інтерактивна оболонка (REPL), яка базується на середовищі **Jupyter**.

Ось детальний технічний розбір Colab як інструменту програмування:

1. Архітектура середовища (Backend)

Коли ви відкриваєте ноутбук, Google виділяє вам віртуальну машину (VM) на базі **Ubuntu**.

- Runtime (Середовище виконання): Ви працюєте з дистрибутивом Python (зазвичай актуальна стабільна версія 3.x).
- Системний доступ: Ви можете виконувати bash-команди прямо з комірок, додаючи префікс !. Це дозволяє керувати системою як терміналом:
  - !ls -la — перегляд файлів.
  - !pip install — встановлення специфічних бібліотек.
  - !apt-get install — встановлення системних утиліт (наприклад, ffmpeg).

2. Попередньо встановлений стек (Batteries Included)

Colab усуває проблему "dependency hell" (пекла залежностей). У середовище вже інтегровані сотні бібліотек:

- Аналіз даних: Pandas, NumPy, SciPy.
- Візуалізація: Matplotlib, Seaborn, Plotly.
- Machine Learning: Scikit-learn, XGBoost.
- Deep Learning: TensorFlow, Keras, PyTorch (з повною підтримкою CUDA для GPU).

3. Робота з даними та файловою системою

Оскільки це хмарна машина, робота з файлами має свої особливості:

- Тимчасове сховище: У вас є папка /content, куди можна завантажувати файли, але після завершення сесії вони видаляються.
- Монтування Google Drive: За допомогою бібліотеки google.colab можна підключити свій диск як локальну директорію.
- Робота з API: Colab дозволяє легко створювати форми для введення даних (Sliders, Input fields), що зручно для швидкого тестування параметрів функцій.

4. Магічні команди (Magics)

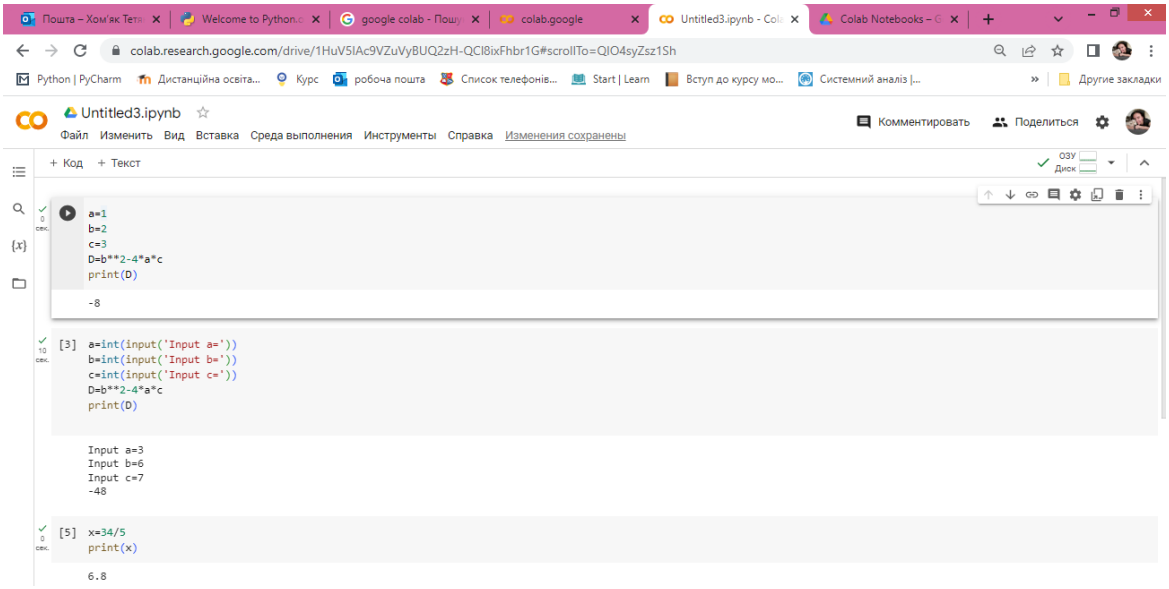
Colab підтримує "Magic commands" від IPython, які розширюють можливості Python:

- %timeit — вимірювання часу виконання коду.
- %%writefile script.py — запис вмісту комірки у файл.
- %load_ext — завантаження розширень (наприклад, для візуалізації профілювання пам'яті).

5. Інтерактивність та Output

Кожна комірка може виводити не лише текст, а й:

- Rich HTML: Таблиці Pandas з можливістю фільтрації.
- Інтерактивні графіки: Які можна масштабувати прямо в браузері.
- Media: Відео, аудіо та зображення, згенеровані кодом.



Зовнішній вигляд середовища Google Colab

Перед написанням коду вкрай бажано ознайомитись з **PEP 8** (https://peps.python.org/pep-0008/).

PEP 8 — це офіційний посібник зі стилю написання коду на Python. Його головна мета — зробити код максимально читабельним та одноманітним, щоб різні розробники могли легко розуміти проєкти один одного.

Ось основні правила, розбиті за категоріями:

1. Зовнішній вигляд коду (Layout)

- **Відступи:** Використовуйте рівно **4 пробіли** для кожного рівня відступу. Не використовуйте табуляцію (Tab).
- **Довжина рядка:** Максимальна довжина рядка — **79 символів**. Для довгих блоків тексту (коментарів) — 72 символи.
- **Порожні рядки:**
  - Відокремлюйте функції верхнього рівня та описи класів **двома** порожніми рядками.
  - Методи всередині класу відокремлюйте **одним** порожнім рядком.
- **Кодування:** Завжди використовуйте **UTF-8**.

2. Імпорти (Imports)

- Імпорти мають бути на самому початку файлу, після коментарів до модуля та рядків документації (docstrings).

- **Групування:** Імпорти слід розділяти на три групи (в такому порядку):

1. Стандартні бібліотеки Python.
2. Сторонні бібліотеки (third-party).
3. Локальні модулі поточного проєкту.

- Кожна група має бути відокремлена порожнім рядком.
- **Стиль:** Краще писати import os, а не import os, sys. Проте конструкція from subprocess import Popen, PIPE дозволена.

3. Пробіли у виразах

- Уникайте зайвих пробілів:
  - Одразу всередині дужок: spam(ham[1]) — Ні. spam(ham[1]) — Так.
  - Перед комою, двокрапкою чи крапкою з комою.
  - Перед дужкою, що починає виклик функції: dict ['key'] — Ні.
- Навколо операторів: Завжди оточуйте бінарні оператори (=, +, -, ==, <, >, !=, in, is, and, or) одним пробілом з обох боків.
- Пріоритет: Якщо використовуєте оператори з різним пріоритетом, можна прибрати пробіли навколо тих, що мають вищий пріоритет:

$y = x^2 + 5$ (у коді: y = x**2 + 5)

4. Коментарі

- Актуальність: Коментарі, які суперечать коду, гірші, ніж їх відсутність. Завжди оновлюйте їх!
- Коментарі в рядку (Inline): Використовуйте їх рідко. Вони мають бути відокремлені від коду принаймні двома пробілами та починатися з # і одного пробілу.
- Docstrings: Пишіть документацію для всіх публічних модулів, функцій, класів та методів.

5. Правила іменування (Naming Conventions)

Це одна з найважливіших частин PEP 8:

Об'єкт	Стиль іменування	Приклад
Змінні / Функції	snake_case (малі літери з підкресленням)	my_variable, calculate_sum()
Класи	PascalCase (CapWords)	MySuperClass, UserProfile
Константи	UPPER_CASE (великі літери)	MAX_TIMEOUT, PI
Модулі / Пакети	короткі малі літери, бажано без підкреслень	requests, numpy

Порада: Ніколи не використовуйте символи 'l' (мала L), 'O' (велика o) або 'I' (велика i) як однолітерні змінні, бо їх легко сплутати з 1 та 0.

6. Програмування (Рекомендації)

- Порівняння з None робіть через is або is not, а не через оператори рівності.
- Використовуйте метод .startswith() та .endswith() замість зрізів рядків для перевірки префіксів/суфіксів.
- Перевірка на порожнечу: пишіть if not my_list:, а не if len(my_list) == 0:.

Базовий синтаксис

Синтаксис мови Python, як і сама мова, достатньо простий. Можна виділити наступні твердження.

• Кінець рядка є кінцем інструкції (прикінцеві символи непотрібні).

• Для надання значень змінним оператором присвоювання є знак дорівнює «=». Формат оператора: ім’я_змінної=вираз.

• Основна інструкція та вкладені інструкції (вкладений блок інструкцій) записуються відповідно до одного шаблону: основна інструкція завершується двокрапкою, наступними рядками розташовуються вкладені інструкції, з однаковим відступом на початку рядків по відношенню до основної інструкції. **Наприклад**:

Основна інструкція:
    Вкладений блок інструкцій

Тобто вкладені інструкції об'єднуються в блоки за величиною відступів. Відступ може бути будь-яким, головне, щоб в межах одного вкладеного блоку відступ був однаковий. Не варто забувати про читабельність коду, так, відступ в 1 пропуск є малочитабельним. **Рекомендується використовувати 4 пропуски**.

Можна бачити, що синтаксис оформлення основної інструкції та вкладеного блоку інструкцій істотно відрізняється від синтаксису більшості мов, в яких використовуються операторні дужки для виділення вкладеного блоку інструкції (наприклад, *begin … end* в Паскалі або *{ … }* в Сі).

• Розмір літер має значення, тобто великі і маленькі літери вважаються різними. Більшість службових слів (окрім: False, None, True) та вбудованих функцій пишуться маленькими літерами. Проте існує декілька спеціальних випадків:

• Можна записати кілька інструкцій в одному рядку, розділяючи їх крапкою з комою:

```
a = 1; b = 2; print(a, b)
```

• Можна записувати одну інструкцію в декілька рядків. Для цього необхідно розмістити її в парі круглих, квадратних або фігурних дужок:

```
if (a < 1 and b < 2
    and c < 3 and d < 4):
  print('spam' * 3)
```

• Якщо тіло вкладеної інструкції містить єдиний оператор, то він може розташовуватися в тому ж рядку, що і основна інструкція:

```
if x > y: print(x)
```

Крім конструкцій мови, програма може містити коментарі.

Базові математичні операції

Символ операції	Призначення	Використання	Приклад
+	Додавання	x + y	2 + 3 результат 5 При a=3 та b=2 a + b результат 5
-	Віднімання	x – y	3 - 2 результат 1
*	Множення	x * y	2 * 3 результат 6
/	Ділення	x / y	4 / 2 результат 2.0 5 / 2 результат 2.5
//	Знаходження цілої частини від цілочисельного ділення	x // y	7 // 3 результат 2 7 // -3 результат -3 -7 // 3 результат -3 -7 // -3 результат 2
%	Знаходження залишку від цілочисельного ділення	x % y	7 % 3 результат 1 7 % -3 результат -2 -7 %3 результат 2 -7 % -3 результат -1 *Пояснення:* *c = a // b* *z = a % b, z має той же знак що і b* *таким чином, щоб* *a = c * b + z*
**	Піднесення до степеня	x**y	2**3 результат 8 4**0.5 результат 2.0

Порядок обчислення операцій

Якщо маємо вираз виду 2 + 3 * 4, то з шкільного курсу математики відомо, що спочатку виконується операція множення, а вже потім операція додавання, оскільки операція множення має вищий пріоритет, ніж операція додавання.

Так само і в операціях мови Python, спершу обчислюються оператори і вирази з вищим пріоритетом, а потім поступово за спаданням пріоритету.

В таблиці наведено пріоритет операторів мови Python, починаючи з самого низького (зверху таблиці) і до найвищого (внизу таблиці).

Оператор	Опис
lambda	Лямбда-вираз (Лямбда-функція)
or	Логічне «АБО»
and	Логічне «І»
not x	Логічне «НІ»
in, not in	Перевірка приналежності
is, is not	Перевірка тотожності
<, <=, >, >=, !=, ==	Оператори порівняння

|	Бітове «АБО»
^	Бітове «ВИКЛЮЧАЮЧЕ АБО»
&	Бітове «І»
<<, >>	Зсуви
+, -	Додавання та віднімання
*, /, //, %	Множення, ділення, цілочисельне ділення та залишок від ділення
+x, -x	Додатнє, від'ємне
~x	Бітове «НІ»
**	Піднесення до степеня
x.attribute	Посилання на атрибут
x[індекс]	Звернення за індексом
x[індекс1:індекс2]	Зріз
f(аргументи …)	Виклик функції
(вираз, …)	Кортеж (tuple)
[вираз, …]	Список (list)
{ключ:дані, …}	Словник (dict)

В таблиці оператори з рівним пріоритетом розміщені в одному рядку (наприклад, + та – мають рівний пріоритет)

Використання вбудованих математичних функцій

Модуль **math** (https://docs.python.org/3/library/math.html) є частиною стандартної бібліотеки Python і надає доступ до математичних функцій, визначених стандартом C. Він ідеально підходить для роботи з дійсними числами (float).

Ось список часто вживаних на практиці функцій, розбитих за категоріями:

1. Теоретико-числові функції

Ці функції допомагають округлювати числа та працювати з цілою/дробовою частинами.

- math.ceil(x) — округлення вгору до найближчого цілого.
- math.floor(x) — округлення вниз до найближчого цілого.
- math.factorial(n) — обчислення факторіала числа n!.
- math.gcd(a, b) — знаходження найбільшого спільного дільника (НСД).
- math.fsum(iterable) — точне обчислення суми чисел з плаваючою крапкою (мінімізує похибки).

2. Степеневі та логарифмічні функції

Використовуються для роботи з коренями, експонентами та логарифмами.

- math.sqrt(x) — квадратний корінь із x.
- math.pow(x, y) — піднесення x до степеня y (завжди повертає float).
- math.exp(x) — обчислення експоненти e^x.
- math.log(x, [base]) — натуральний логарифм (або за вказаною основою).
- math.log10(x) — десятковий логарифм.

3. Тригонометричні функції

Нагадування: усі кути передаються та повертаються в **радіанах**.

- math.sin(x), math.cos(x), math.tan(x) — основні тригонометричні функції.
- math.asin(x), math.acos(x), math.atan(x) — обернені тригонометричні функції (арксинус і т.д.).
- math.hypot(x, y) — обчислення гіпотенузи трикутника за двома катетами

4. Конвертація кутів

Оскільки людина зазвичай думає в градусах, а комп'ютер — у радіанах, ці функції незамінні:

- math.radians(d) — конвертує градуси в радіани.
- math.degrees(r) — конвертує радіани в градуси.

5. Математичні константи

Модуль також містить готові значення констант з високою точністю:

- math.pi — число Пі (π ≈3.14159).
- math.e — число Ейлера (≈2.71828).
- math.tau — число Тау (≈6.28318).
- math.inf — позитивна нескінченність.

Розгалудження

Конструкція **if-else** у Python — це основний інструмент керування логікою програми. Вона дозволяє комп'ютеру "приймати рішення": виконувати певний блок коду лише тоді, коли справджується конкретна умова.

1. Базова структура (Синтаксис)

У Python логічні блоки виділяються **відступами** (зазвичай 4 пробіли). Це критично важливо, оскільки без відступів код видасть помилку.

```
if умова:
    # Код, який виконається, якщо умова істинна (True)
else:
    # Код, який виконається, якщо умова хибна (False)
```

Приклад

```
age = 18
if age >= 18:
    print("Ви повнолітні")
else:
    print("Ви ще неповнолітні")
```

2. Кілька умов: оператор elif

Якщо у вас більше ніж два варіанти розвитку подій, використовується *elif* (скорочено від *else if*). Ви можете додавати скільки завгодно *elif* між *if* та *else*.

Приклад

```
score = 85

if score >= 90:
    print("Відмінно")
```

```
elif score >= 75:
    print("Добре")
elif score >= 60:
    print("Задовільно")
else:
    print("Незадовільно")
```

3. Логічні оператори в умовах

Для створення складних перевірок використовуються оператори порівняння та логічні зв’язки:

- **Порівняння:** == (рівно), != (не дорівнює), >, <, >=, <=.
- **Логічні зв'язки:**
  - and: істинно, якщо **обидві** умови виконуються.
  - or: істинно, якщо виконується **хоча б одна** умова.
  - not: інверсія (перетворює True на False і навпаки).

4. Тернарний оператор (Скорочений запис)

Якщо вам потрібно просто присвоїти значення змінній залежно від умови в один рядок, можна використати таку конструкцію:

Приклад.

```
status = "Доступ дозволено" if age >= 18 else "Доступ заборонено"
```

Важливі правила для початківця:

1. **Двокрапка:** Завжди ставте : після if, elif та else.
2. **Тип Boolean:** Умова в if завжди приводиться до логічного типу (True або False).
3. **Порядок перевірки:** Python перевіряє умови зверху вниз. Як тільки він знаходить першу істинну умову, він виконує її блок і **пропускає всі інші** гілки (elif та else).

Приклад

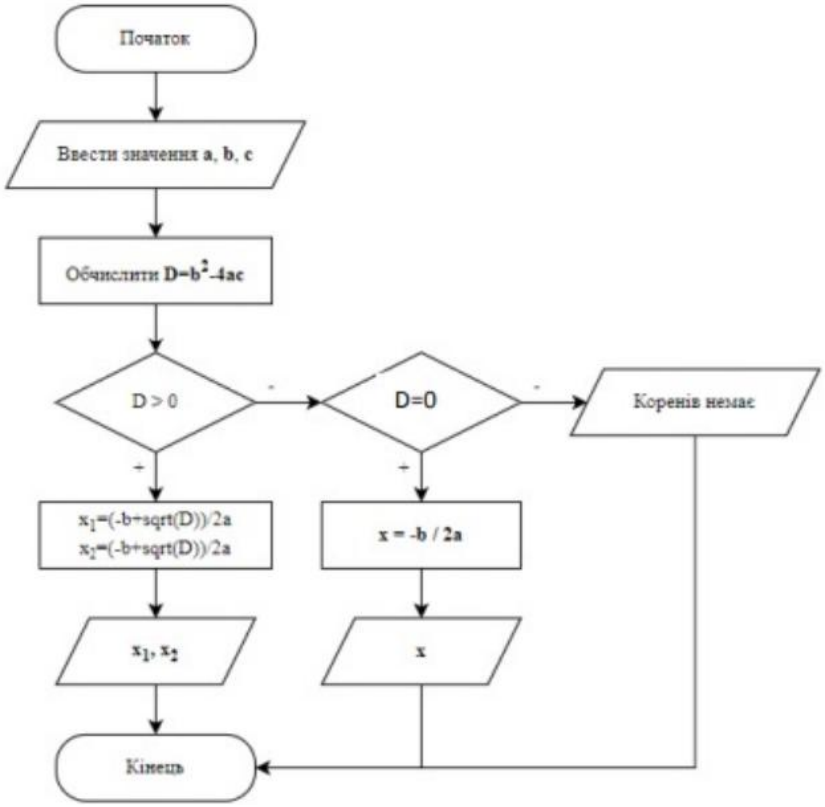
```
import math

a=int(input("Enter a="))
b=int(input("Enter b="))
c=int(input("Enter c="))
D=b**2-4*a*c

if D>0:
    x1=(-b+math.sqrt(D))/(2*a)
    x2=(-b-math.sqrt(D))/(2*a)
    print('x1=',x1,'x2=',x2)
else:
    if D==0:
        x=-b/(2*a)
        print('x=',x)
```

```
else:
    print('not root')
```

Блок-схема алгоритму



Завдання до лабораторної роботи 2

Скласти блок-схему алгоритму, що розв’язує завдання відповідно до варіанту завдання;

Написати программу, у відповідності до складеної блок-схеми;

Провести тестування складеної програми за всіма можливими гілками;

Оформити звіт з лабораторної роботи, в якому мають бути вказані:

- індивідуальне завдання у відповідності до варіанту
- блок-схему алгоритму
- код програми, яка виконує поставлене завдання
- результати тестування
- висновки

Варіант	Вираз для обчислення
1	$C = \begin{cases} a^2 - (b+2)\sin(a-1), & \text{коли } a>b; \\ a^2 + (b-2)\sin(a-1), & \text{коли } a=b; \\ a^2 - (b+2)\cos(a+1), & \text{коли } a<b; \end{cases}$
2	$C = \begin{cases} b^2 - (b+tg(a))\cos(a-b), & \text{коли } a>b; \\ b^2 - (b+\lg(a))\cos(a-b), & \text{коли } a=b; \\ b^2 - (b+a)\sin^2(a-b), & \text{коли } a<b; \end{cases}$
3	$C = \begin{cases} \ln a^2 - (b+5)tg^2(a-1), & \text{коли } a>\sin(b); \\ \lg a^2 - (b+5)ctg(a-1), & \text{коли } a=\sin(b); \\ \ln a^2 - (b^4+5)\cos^2(a-1), & \text{коли } a<\sin(b); \end{cases}$
4	$C = \begin{cases} e^{2+a} - (\cos(b)+2)\cos(a-1), & \text{коли } a>0 \text{ інакше:} \\ e^{2+a} - (\cos(b+a)+2)\sin(a-1), & \text{коли } a>=\ln(b); \\ e^{2+a} - (\cos(b)+2)\sin^3(a-1), & \text{коли } a<\ln(b); \end{cases}$
5	$C = \begin{cases} a^{1/2} - (b+2)\sin(e^a-1), & \text{коли } a>e \text{ інакше;} \\ a^{1/4} - (b+a^2)\sin(e^a-1), & \text{коли } a<=tg(b); \\ a^{1/2} - (b+2)tg(e^a-1), & \text{коли } a>tg(b); \end{cases}$
6	$C = \begin{cases} a^2 - \ln(b+a^2)\sin^2(a-1), & \text{коли } a>b \text{ інакше;} \\ a^2 - \ln(b+\sin^2(a))\sin^2(a-1), & \text{коли } \lg(a)<=tg(b) \\ a^3 - \ln(b+2)tg^2(a-1), & \text{коли } \lg(a)tg(b); \end{cases}$
7	$C = \begin{cases} \ln^2(a^3) - \sin(b+2)\sin(a+b), & \text{коли } a=1 \text{ інакше;} \\ \ln^2(a^2) - \cos(b+2)\sin(\lg(a)+1), & \text{коли } \lg(a)<b; \\ \ln^2(a^2) - \cos(b+a)\cos(a+5), & \text{коли } \lg(a)>=b; \end{cases}$
8	$C = \begin{cases} a^{2\sin(a)} - e^{(b+2)}\ln(\lg(a)-1), & \text{коли } a>b \text{ інакше;} \\ a^{2b} - e^{(b+2)}\lg(a-1) \end{cases}$
9	$C = \begin{cases} \cos(a^2+b^2) - e^{(b+a)}+1, & \text{коли } \ln(a)<b; \\ \sin(a^2+b^2) - e^{(a+2)}+1, & \text{коли } \ln(a)=b; \\ \lg(a^{2a}+b) - e^{(b+1)}+1, & \text{колиln}(a)>b; \end{cases}$
10	$C = \begin{cases} b^2/a^2 - (b+2a)\sin(e^a), & \text{коли } \cos(a)<b; \\ b^2/a^2 - (\sin(b)+a)\sin(e^a), & \text{коли } \cos(a)=b; \\ b^2/a^2 - (b+a)\lg(e^a), & \text{коли } \cos(a)>b; \end{cases}$
11	$C = \begin{cases} ab^2 - (b+2)\sin^2(a), & \text{коли } a=0; \\ ab^2 - (b+3)\sin^3(a), & \text{коли } a=1 \text{ або } 2 \text{ або } 3; \\ ab^4 - (b+4)\sin^4(a) & \text{в інших випадках} \end{cases}$
12	$C = \begin{cases} b/a^3 - (b+2)/\cos(a-1), & \text{коли } a=b; \\ b/a^2 - (b+2)/tg(a-b), & \text{коли } a=1,\ b>0; \\ \cos(b)/a^2 - (b+a^{1/2})/\sin(a-1) & \text{в інших випадках} \end{cases}$
13	$C = \begin{cases} a^2/((b+2)\sin(a-1)), & \text{коли } a>0 \text{ або } b>0; \\ a^2/((b+2)\cos(a-1)), & \text{коли } a=0,\ b>0 \text{ та } b<2; \\ a^2 & \text{в інших випадках.} \end{cases}$

14	$C = \begin{cases} e^x/\sin^4(y-\lg(x-1)), & \text{коли } x>0 \text{ або } y>x; \\ e^x/\sin^2(y-\cos^3(x-1)), & \text{коли } x>y \text{ та } y>0; \\ e^x/\cos^2(y-\cos(x-y)) & \text{в інших випадках} \end{cases}$
15	$C = \begin{cases} \lg(\sin^2(2x))+y-\cos(1/x^{1/2}), & \text{коли } x=1,\ x=2,\ x=4\ \ y=0; \\ e^x\sin^2 y-\cos(3x^{1/2}), & \text{коли } y>0 \text{ або } x=3; \\ e^x\sin(y+1) & \text{в інших випадках.} \end{cases}$
16	$C = \begin{cases} \ln^x(y)/\sin^2 y-tg(x-y), & \text{коли } x>y \text{ або } \ y=0; \\ \lg^x(x+y)/\sin^2 y-\sin(x-5), & \text{коли } x=y \text{ та } y=0; \\ e^x/\cos^2 y-\cos(x-1), & \text{коли } y<0; \end{cases}$
17	$C = \begin{cases} x^x/\sin^2 y+\cos(1/y-1), & \text{коли } y>0 \text{ та } \sin(y)>0; \\ y^x/\sin^3 y+\cos(1/x-1), & \text{коли } y=0 \text{ та } \sin(x)>0; \\ e^x/tg^2 y+\cos(y/x-x) & \text{в інших випадках} \end{cases}$
18	$C = \begin{cases} ye^{xy}\sin^2 y+e^{\cos(y-1)}, & \text{коли } (x+y)>5\ \ y>0; \\ xe^y\sin^2 y+e^{\cos(y-x)}, & \text{коли } y<0; \\ xe^{2x}\sin^2 y+e^{\sin(x-1)} & \text{в інших випадках;} \end{cases}$
19	$C = \begin{cases} ctn(\sin(1-\cos^2(x-1))), & \text{коли } x=0 \text{ та } y=1; \\ \ln(\sin(y-\cos^3(x))), & \text{коли } x<0 \text{ або } \ y<0; \\ \lg(\sin(x-\cos^4(x-y))) & \text{в інших випадках} \end{cases}$
20	$C = \begin{cases} e^{(\sin(x)+2y)}/\ln(xy-1), & \text{коли } xy>0; \\ e^{(x+2y)}/\lg(x^2y+1), & \text{коли } x=y=1; \\ e^{(2x+y)}/ctn(1+xy) & \text{в інших випадках} \end{cases}$
21	$C = \begin{cases} y^x/ctn^2(y+\cos(x-1)), & \text{коли } x-y=1 \text{ та } \ y>0; \\ 2^x/\sin^2(y-\cos(x-1)), & \text{коли } y=0; \\ e^x/\cos^2(x-\sin(x-y)) & \text{в інших випадках} \end{cases}$
22	$C = \begin{cases} (e^x-e^y)^2/\sin(y)-\cos(x), & \text{коли } \cos(x)>0\ \ y>0; \\ (2^x-2^y)^{2x}/\sin(y^2)-\cos(y^2)-\cos(y), & \text{коли } \sin(y)<>0\ \ x>0; \\ (e^x-e^y)^2/\sin(x^2+1)-ctg(y) & \text{в інших випадках;} \end{cases}$
23	$C = \begin{cases} ((x^2+1)^2+1)^2-y-1, & \text{коли } (x+y)=1, \text{ або } (x+y)=2, \text{ або } (x+y)=4; \\ ((y^2+1)^2+1)^2-x-1, & \text{коли } (x+y)=3, \text{ або } (x+y)=5 \\ x+y & \text{в інших випадках;} \end{cases}$
24	$C = \begin{cases} e^x\sin(x^2-\cos(x^2-1)), & \text{коли } (x^2>1)\ \ (y^3>1); \\ e^x\sin(y^2-\sin(x^2-1)), & \text{коли } x=0\ \ y=0; \\ e^x\sin(x+y^2-\cos(y^2-1)) & \text{в інших випадках} \end{cases}$
25	$C = \begin{cases} \sin(e^x)\sin^2(y-x-1), & \text{коли } (x>1) \text{ та } (y>1); \\ \cos(e^x)\sin^3(y-x-1), & \text{коли } (x=1) \text{ та } (y<1); \\ tg(e^x)\sin^{1/3}(y-x-1) & \text{в інших випадках.} \end{cases}$

Критерії оцінювання лабораторної роботи 2

За умови своєчасного та в повному обсязі виконаного завдання здобувач може отримати 100 балів (матеріал якісно викладено в логічній послідовності, без помилок, зроблено висновки до виконаного завдання, виявлено підвищений рівень володіння матеріалом).

Бали	Рівень	Коментар
90–100	Відмінно	Коректно складена блок-схема, оптимальний код, продемонстровано повний спектр тестів.
75–89	Добре	Програма виконує завдання, блок-схема вірна, але є дрібні недоліки в стилі коду або оформленні
60–74	Задовільно	Програма працює лише на частині даних, у блок-схемі є помилки в позначеннях, студент плутається в поясненні пріоритетів логічних операторів.
< 60	Незадовільно	Критичні помилки в логіці, програма завершується аварійно (traceback) на типових помилках.

Контрольні питання

1. **Пріоритет операторів:** У якому порядку обчислюється вираз if a > 5 or b < 10 and not c == 0? Як змінити цей порядок?
2. **Вкладені умови:** Що таке "пекло відступів" (indentation hell) і як можна переписати вкладені if, щоб код став чистішим?
3. **Специфіка elif:** Чи обов'язково завершувати конструкцію if - elif блоком else? У яких ситуаціях його відсутність може призвести до логічної помилки?
4. **Константи модуля math:** Як отримати значення числа π та числа Ейлера (e) з високою точністю? Чому краще використовувати math.pi, аніж вписувати 3.14 вручну?
5. **Округлення:** Чим відрізняються функції math.ceil(), math.floor() та math.trunc()? Яку з них ви оберете, щоб знайти кількість цілих пачок товару, які потрібно купити?
6. **Робота з коренями та степенями:** У чому різниця між використанням оператора ** 0.5 та функції math.sqrt()?
7. **Тригонометрія:** У яких одиницях вимірювання (градуси чи радіани) приймають аргументи функції math.sin() та math.cos()? Як швидко конвертувати одне в інше за допомогою цього ж модуля?
8. **Перевірка значень:** Які функції модуля math дозволяють перевірити, чи є число "не числом" (NaN) або нескінченністю (inf)? Коли це може знадобитися в обчисленнях?
9. **Факторіал та НСД:** Які готові методи модуля math існують для знаходження факторіала числа та найбільшого спільного дільника (НСД) двох чисел?

Перелік рекомендованих джерел

1. Основи програмування. Python. Частина 1 [Електронний ресурс]: підручник для студ. спеціальності 122 "Комп'ютерні науки" /А.В. Яковенко; КПІ ім. Ігоря Сікорського. – Київ: КПІ ім. Ігоря Сікорського, 2018. – 195 с.

2. Трінтіна Н.А., Негоденко О.В., Гаманюк І.М., Шевченко С.М. Програмування мовою Python. Навчальний посібник підготовлено до друку для самостійної роботи студентів вищих навчальних закладів. – К.: ННІІТ ДУТ, 2022. – 113 с.

3. https://www.python.org/

4. https://www.w3schools.com/python/default.asp

Лабораторна робота № 3
Розробка програм з використанням логічних операторів та операторів циклів

Мета: набути практичних навичок по використанню логічних операторів та операторів циклу.

Теоретичні відомості

Синтаксис операторів циклів

1. **Цикл while** (Цикл із передумовою)

Цикл while виконує блок коду доти, доки певна логічна умова залишається істинною (True).

Синтаксис:

```
while умова:
    # блок коду, що виконується
```

2. Цикл for (Цикл перерахування)

Цикл for в Python є ітераційним. Він призначений для перебору елементів будь-якої послідовності (списку, рядка, кортежу) або об'єктів, що ітеруються.

Синтаксис:

```
for елемент in послідовність:
    # блок коду
```

Функція range()

Найчастіше цикл for використовують разом із функцією range(), яка генерує послідовність чисел:

- range(5) — від 0 до 4.
- range(1, 10, 2) — від 1 до 9 з кроком 2.

3. Керування циклами: break, continue та else

Для більш гнучкого керування логікою всередині циклів використовують спеціальні інструкції:

- **break**: негайно перериває виконання циклу та виходить із нього.
- **continue**: пропускає залишок поточної ітерації та переходить до наступної перевірки умови.
- **Блок else**: унікальна особливість Python. Код у блоці else виконається лише в тому випадку, якщо цикл завершився **природним шляхом** (умова стала хибною або ітерації закінчилися), а не був перерваний командою break.

Використання блоку else у циклах — це цікава особливість Python, яка часто плутає новачків. Найкраще розуміти це так: **блок else виконується лише тоді, коли цикл завершився сам по собі**, а не був перерваний командою break.

Приклад: Пошук "забороненого" елемента

Уявімо, що ми перевіряємо список продуктів на наявність алергену (наприклад, горіхів).

```
products = ["яблуко", "банан", "молоко", "мед"]
search_item = "горіхи"

for item in products:
    if item == search_item:
        print(f"Обережно! Знайдено {search_item}. Зупиняємо перевірку.")
        break  # Перериваємо цикл, бо ціль знайдена
else:
    # Цей код виконається, тільки якщо цикл пройшов по ВСЬОМУ списку
    # і жодного разу не зустрів break
    print(f"Перевірка завершена. {search_item} не знайдено, все безпечно.")
```

Як це працює (логіка)

1. **Якщо ми знайдемо "горіхи":** Спрацює break. Програма "вистрибне" з циклу, і блок else буде **проігнорований**.
2. **Якщо горіхів немає:** Цикл перебере всі елементи, дійде до кінця і "спокійно" завершиться. Після цього Python виконає код у блоці else.

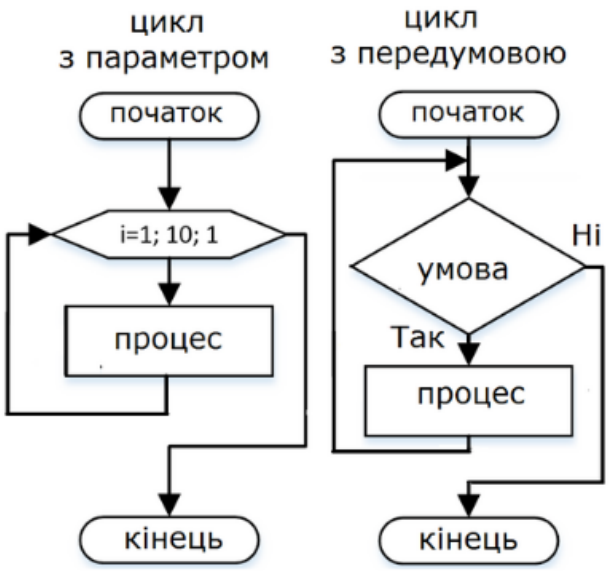
Чому це зручно?

Без else вам довелося б створювати додаткову змінну-прапорець (наприклад, found = False), змінювати її всередині циклу, а потім після циклу писати окрему перевірку if not found:. Блок else робить код коротшим та чистішим.

Коротке правило для запам'ятовування:

else у циклах працює як "якщо не було break".

Представлення циклів у блок-схемах



Завдання до лабораторної роботи 3

1. Скласти 2 окремі програми, які знаходять значення виразу у відповідності до поставленого завдання використовуючи *for* та *while*;
2. Виконати тестування кожної з програм;
3. Передбачити виведення вхідних об'єктів (матриць або векторів) «до» та «після» перетворень, у випадку якщо це доцільно за умовою завдання.
4. Скласти блок-схему алгоритму для кожної з програм.
5. Оформити звіт з виконаної роботи.

Варіант	**Формула**	**Умови**
1.	$p=N!/2^N$	$N=15$
2.	$p=(1-1/2^2)(1-1/3^2)...(1-1/n^2)$	$n=10$
3.	$p=cos(1)+cos(2)+cos(3)+...+cos(n-1)+cos(n)$	$n=15$
4.	$p=1/(x+10)+1/(2x+9)+1/(3x+8)+...+1/(10x+1)$	$x=10$
5.	$p=1/cos1+2/cos2+...+N/cosN!$	$N=8$
6.	$p=1/sin1+1/(sin1+sin2)+...+1/(sin1+sin2+...+sinN)$	$N=10$
7.	$p=(a+1)(a+2)(a+3)...(a+n)$	$n=10$ $a=0,5$
8.	$p=1/a+/(a(a+1))+...1/(a(a+1)...(a+n))$	$n=10$, $a=1$
9.	$p=1/a^2+2/a^4+...+N/a^{2n}$	$n=6$
10.	$p=a(a-n)(a-2n)...(a-n^2)$	$n=10$, $a=4$
11.	$p=sin(x+1)+sin(2x+2)+...+sin(nx+n)$	$n=10$, $x=0,5$
12.	$p=sin(1/x)+sin^2(1/x^2)+...+sin^n(1/x^n)$	$n=10$
13.	$p=cos(x^2)+2cos(2x^2)+...+Ncos(Nx^2)$	$N=10$, $x=0,5$
14.	$p=x^{2n}/3^n$	$n=6$, $x=1,2$
15.	$p=1/2+2/3+3/4+...+(n/n+1)$	$n=10$
16.	$p=(1+2)/(2+3)+(2+3)/(3+4)+...+(n+n+1)/(n+1+n+2)$	$n=10$
17.	$p=(1+2)/x+(2+3)/2x+...+(n+n+1)/nx$	$n=10$
18.	$p=(1-1/2^2)(1-1/3^2)...(1-1/n^2)$	$n=10$
19.	$p=1^2/(1^2+2)+2^2/(2^2+4)+...+n^2/(n^2+2n)$	$n=10$
20.	$p=((1+1)/(1+2))((2+1)/(2+2))...((n+1)/(n+2))$	$n=10$
21.	$p=(2+1/1!)(2+1/2!)...(2+1/n!)$	$n=10$
22.	$p=(1+1/1!)(1-1/2!)...(1-1/n!)$	$n=10$
23.	$p=1/(2+1)^2+2/(4+1)^2+3/(6+1)^2+...+n/(2n+1)^2$	$n=10$
24.	$p=1!/sin(1)^2+2!/sin(2)^2+...+n!/sin(n)^2$	$n=10$
25.	$p=1/2+3/4+5/6+...+(2n-1)/2n$	$n=10$

Критерії оцінювання лабораторної роботи 3

За умови своєчасного та в повному обсязі виконаного завдання здобувач може отримати 100 балів (матеріал якісно викладено в логічній послідовності, без помилок, зроблено висновки до виконаного завдання, виявлено підвищений рівень володіння матеріалом).

Бали	Рівень	Коментар
90–100	Відмінно	Програма працює ідеально. Код чистий, документований, логіка позбавлена надмірності. Повна відповідність PEP 8.
74–89	Добре	Програма виконує всі завдання, але алгоритм може бути не найбільш оптимальним (наприклад, зайві вкладені умови, які можна було спростити через elif). Загалом код чистий, але є поодинокі порушення PEP 8 (зайві пробіли, задовгі рядки)
60–73	Задовільно	Програма працює, але реалізація "важка". Використовуються лише найпростіші конструкції. Можливі помилки в граничних значеннях (наприклад, цикл спрацьовує на один раз менше, ніж потрібно). Код недбалий, відступи можуть бути хаотичними.
Нижче 60	Незадовільно	Програма не запускається через синтаксичні помилки або логіка повністю суперечить завданню. Наявні нескінченні цикли, що призводять до зависання системи.

Перелік рекомендованих джерел

1. Основи програмування. Python. Частина 1 [Електронний ресурс]: підручник для студ. спеціальності 122 "Комп'ютерні науки" /А.В. Яковенко; КПІ ім. Ігоря Сікорського. – Київ: КПІ ім. Ігоря Сікорського, 2018. – 195 с.
2. Трінтіна Н.А., Негоденко О.В., Гаманюк І.М., Шевченко С.М. Програмування мовою Python. Навчальний посібник підготовлено до друку для самостійної роботи студентів вищих навчальних закладів. – К.: ННІІТ ДУТ, 2022. – 113 с.
3. https://www.python.org/
4. https://www.w3schools.com/python/default.asp

Лабораторна робота № 4
Програмування функцій, рекурсивних функцій

Мета: набути практичних навичок у розробці функцій та їх використанні.

Теоретичні відомості

У Python функції — це основний інструмент для структурування коду. Вони дозволяють уникати дублювання, розбиваючи програму на логічні блоки, які можна викликати багаторазово.

1. Користувацькі функції (def)

Функція оголошується за допомогою ключового слова def, після якого йде назва, круглі дужки з параметрами та двокрапка.

Структура функції:

- **Назва**: має відображати дію (зазвичай дієслово).
- **Параметри**: змінні, які функція отримує на вхід.
- **Тіло**: блок коду з відступом.
- **Return**: інструкція, яка повертає результат (якщо її немає, функція поверне None).

Приклад

```
def calculate_grade(score):
    """Визначає результат тесту на основі балів."""
    if score >= 60:
        return "Пройдено"
    else:
        return "Не здано"

# Виклик функції
result = calculate_grade(85)
print(result)  # Виведе: Пройдено
```

2. Параметри та аргументи

Python пропонує гнучкі способи передачі даних у функції:

1. **Позиційні аргументи**: передаються в тому ж порядку, що й параметри.
2. **Іменовані аргументи**: передаються із зазначенням імені (name="Олена").
3. **Значення за замовчуванням**: параметри, які мають значення, якщо їх не вказали при виклику.

Приклад

```
def greet(name, message="Вітаємо"):
    print(f"{message}, {name}!")

greet("Олексій")             # Використає "Вітаємо"
greet("Марія", "Добридень") # Перекриє стандартне значення
```

3. Лямбда-функції (Анонімні функції)

Лямбда-функція — це невелика анонімна функція, яка записується в один рядок. Вона не має імені (якщо ви не присвоїте її змінній) і ключового слова def.

Синтаксис: lambda параметри: вираз

Особливості:

- Може приймати будь-яку кількість аргументів, але містить лише **один вираз**.
- Автоматично повертає результат обчислення виразу.
- Використовується там, де функція потрібна на короткий час (наприклад, всередині інших функцій).

Приклад

```
# Звичайна функція
def double(x):
    return x * 2

# Лямбда-функція
double_lambda = lambda x: x * 2

print(double_lambda(5))  # Виведе: 10
```

Представлення функцій у блок-схемах

1. Виклик функції (Зумовлений процес)

Для позначення виклику користувацької функції, яка описана десь в іншому місці, використовується символ **"Зумовлений процес"** (Predefined Process).

- **Вигляд:** Прямокутник із двома вертикальними лініями всередині по боках.
- **Зміст:** Всередині пишеться назва функції та аргументи, які в неї передаються.

2. Опис самої функції (Окремий алгоритм)

Якщо ви малюєте блок-схему не всієї програми, а конкретної функції, вона має свою власну структуру:

- **Початок:** Замість слова "Початок" у термінаторі (овалі) пишеться назва функції та список параметрів. Наприклад: Функція get_sum(a, b).
- **Тіло функції:** Використовуються стандартні блоки: прямокутники для дій, ромби для умов, паралелограми для введення/виведення.
- **Кінець (Повернення значення):** Замість слова "Кінець" часто використовується термінатор із написом return та назвою змінної, яку функція віддає назад.

3. Рівні деталізації

Зазвичай при проектуванні складних систем використовують два типи блок-схем:

1. **Основна схема (Main flow):** Показує загальну логіку програми. Функції там виглядають як прості блоки "Зумовленого процесу". Це дозволяє не захаращувати схему деталями.

2. **Детальна схема (Function flow):** Окрема блок-схема для кожної функції. Вона пояснює, як саме обчислюються дані всередині цього блоку.

Приклад

Завдання до лабораторної роботи 4

1. Скласти програми, які реалізують поставлені задачі.
2. Виконать тестування складеного коду.
3. Представити алгоритми складених програм у вигляді блок-схем.
4. Оформити результати у вигляді звіту.

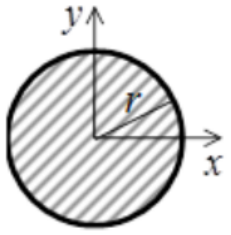
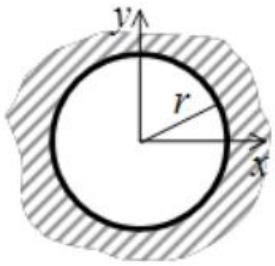
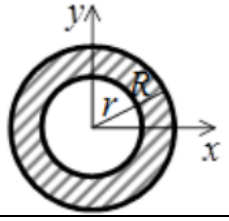
Задача 1.

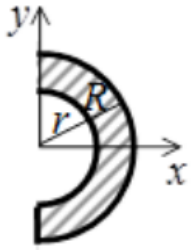
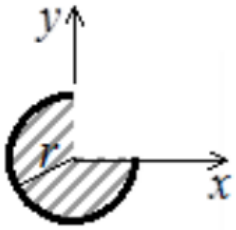
Варіант	Умова
1	Скласти функцію для розрахунку кількості поєднань з m елементів по n елементів: $C_n^m = \frac{n!}{m!(n-m)!}$. Для перевірки роботи функції написати основну програму, яка обчислює C_2^4, C_3^6, C_2^6.
2	Написати функцію для обчислення суми n членів арифметичної прогресії за такою формулою: $S = \frac{(a_1 + a_n) \cdot n}{2}$. Обчислити суми арифметичних прогресій: 1, 2, 3,..., 100 1, 4, 6,..., 100 1, 3, 5, ..., 101

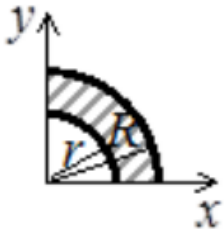
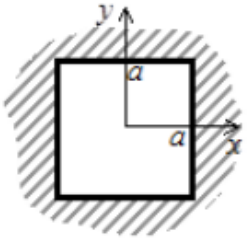
<table>
<tr><td>3</td><td>Написати функцію для обчислення третьої сторони трикутника за відомими двома сторонами та кутом між ними. Обчислити сторони трикутників за табличними даними. Для обчислень користуватися формулою: $a^2 = b^2 + c^2 - 2 \cdot a \cdot c \cdot \cos\alpha$.
<table>
<tr><td>Сторона a, см</td><td>Сторона b, см</td><td>Кут α, рад</td></tr>
<tr><td>3</td><td>4</td><td>π/2</td></tr>
<tr><td>2</td><td>3</td><td>π/4</td></tr>
<tr><td>5.5</td><td>3.1</td><td>2π/3</td></tr>
<tr><td>1.75</td><td>4.9</td><td>3π/4</td></tr>
</table></td></tr>
<tr><td>4</td><td>Написати функцію для зведення числа x до ступеня m (m -натуральне число). Обчислити x^3, x^4, x^5.</td></tr>
<tr><td>5</td><td>Написати функцію обчислення арксинуса. Обчислити $arcsin(0.9)$, $arcsin(0.1)$, $arcsin(-0.9)$, $arcsin(0.99)$. Для обчислень користуватися формулою: $\arcsin(x) = arctg\left(\frac{x}{\sqrt{1-x^2}}\right)$.</td></tr>
<tr><td>6</td><td>Написати функцію для обчислення арккосинусу. Обчислити $arccos(0.5)$, $arccos(0.6)$, $arccos(-0.5)$, $arccos(0.99)$. Для обчислень користуватися формулою: $\arccos(x) = arctg\left(\frac{\sqrt{1-x^2}}{x}\right)$.</td></tr>
<tr><td>7</td><td>Написати функцію для обчислення площі за формулою Герона $S = \sqrt{p(p-a)(p-b)(p-c)}$, де a, b, c – сторони трикутника, а p – напівпериметр. Обчислити площу трикутників за такими вихідними даними:
<table>
<tr><td>Сторона a, см</td><td>Сторона b, см</td><td>Сторона c, см</td></tr>
<tr><td>3</td><td>4</td><td>5</td></tr>
<tr><td>3.1</td><td>4.5</td><td>1.2</td></tr>
<tr><td>4.0</td><td>6.7</td><td>2.9</td></tr>
</table></td></tr>
<tr><td>8</td><td>Написати функцію обчислення об’єма кульового сектора $V = \frac{2}{3}\pi R^2 H$, де R - радіус сфери, а H - висота сектора. Обчислити об’єми кульових секторів за такими вихідними даними:
<table>
<tr><td>R, см</td><td>H, см</td></tr>
<tr><td>6</td><td>4</td></tr>
<tr><td>7.5</td><td>4</td></tr>
<tr><td>12</td><td>3.2</td></tr>
</table></td></tr>
<tr><td>9</td><td>Написати функцію для обчислення знака числа за такою формулою: $sign(x) = \begin{cases} 1, x > 0 \\ 0, x = 0 \\ -1, x < 0 \end{cases}$. Обчислити такі вирази: sign(a b), sign(a/b), sign(-b), sign(-a).</td></tr>
</table>

10	Написати функцію обчислення суми цифр тризначного натурального числа. Обчислити суму цифр для чисел від 100 до 120

Задача 2

Варіант	**Завдання**
1	Написати функцію, яка визначає, чи належить задана на площині точка заштрихованій області. 
2	Написати функцію, яка виконує обчислення виразу, та протабулювати значення на інтервалі (-5;5) з кроком 0,25 $y = \begin{cases} \cos^2 x, 0 < x < 2 \\ 1 - \sin x^2, x \le 0, x \ge 2 \end{cases}$
3	Написати функцію, яка визначає, чи належить задана на площині точка заштрихованій області. 
4	Написати функцію, яка виконує обчислення виразу, та протабулювати значення на інтервалі (-5;5) з кроком 0,25 $z = \begin{cases} \max(x, y), & x < 0; \\ \min(x, y), & x \ge 0; \end{cases}$
5	Написати функцію, яка визначає, чи належить задана на площині точка заштрихованій області. 
6	Написати функцію, яка виконує обчислення виразу, та протабулювати значення на інтервалі (-5;5) з кроком 0,25 $z = \begin{cases} \dfrac{ax+b}{x^2+y^2} + 1, & x^2 + y^2 \le 1; \\ \dfrac{abxy}{x^2+y^2} - x^3, & x^2 + y^2 > 1; \end{cases}$

7	Написати функцію, яка визначає, чи належить задана на площині точка заштрихованій області. 
8	Написати функцію, яка виконує обчислення виразу, та протабулювати значення на інтервалі (-5;5) з кроком 0,25 $y = \begin{cases} \frac{ax+b}{ax} + ax, & x < -1; \\ ax + b^2, & -1 \le x \le 3.5; \\ \frac{ax+b}{ax} - a^2 x, & x > 3.5; \end{cases}$
9	Написати функцію, яка визначає, чи належить задана на площині точка заштрихованій області. 
10	Написати функцію, яка визначає, чи є серед цифр заданого натурального трьохзначного числа однакові.
11	Написати функцію, яка визначає, чи належить задана на площині точка заштрихованій області. 
12	Написати функцію, яка за координатами трьох точок $A(x_1, y_1), B(x_2, y_2), C(x_3, y_3)$ з'ясувує, чи лежать ці точки на одній прямій.
13	Написати функцію яка визначає, чи належить задана на площині точка заштрихованій області. 

14	Написати функцію, яка визначає, чи дорівнює сума двох перших цифр заданого натурального чотирьохзначного числа сумі двох останніх чисел.
15	Написати функцію, яка визначає, чи належить задана на площині точка заштрихованій області. 
16	Написати функцію, яка за коефіцієнтами прямої k і b ($y = kx + b$) та координатами точки $A(x, y)$ дає відповідь, чи належить точка А цій прямій.
17	Написати функцію, яка визначає, чи належить задана на площині точка заштрихованій області. 
18	На площині задані прямі $y = k_1x_1 - b_1$ і $y = k_2x_2 - b_2$. Визначити взаємне їх розташування на площині. Указівки: умова паралельності двох прямих $k_1 = k_2$; умова перпендикулярності двох прямих $1 + k_1k_2 = 0$
19	Написати функцію, яка за координатами вершин трикутника $A(x_1, y_1), B(x_2, y_2), C(x_3, y_3)$ визначає, чи трикутник рівнобедрений. При обчисленні застосувати формулу відстані між двома точками.
20	Написати функцію, яка виконує обчислення виразу, та протабулювати значення на інтервалі (-5;5) з кроком 0,25 $y = \begin{cases} 0, & \text{при } x < 0; \\ 1/(x^2+1), & \text{при } 0 \le x \le 1; \\ x^3, & \text{при } 1 < x \le 4; \\ 62 + \log_8 x, & \text{при } x > 4. \end{cases}$
21	Написати функцію, яка за координатами вершин чотирикутника $A(x_1, y_1), B(x_2, y_2), C(x_3, y_3), D(x_4, y_4)$ визначає, чи цей чотирикутник – квадрат.
22	Написати функцію яка виконує обчислення виразу, та протабулювати значення на інтервалі (-5;5) з кроком 0,25

	$$v=\begin{cases} xy, & \text{при } x<0,\ y\le 0; \\ \operatorname{ctg}\dfrac{x}{y}, & \text{при } 0\le x<10,\ 0<y\le 9; \\ -100, & \text{в інших випадках.} \end{cases}$$
23	Написати функцію, яка за координатами вершин чотирикутника $A(x_1,y_1), B(x_2,y_2), C(x_3,y_3), D(x_4,y_4)$ визначає, чи цей чотирикутник – ромб
24	Написати функцію, яка виконує обчислення виразу, та протабулювати значення на інтервалі (-5;5) з кроком 0,25 $$z=\begin{cases} \dfrac{1}{x^2}, & \textit{при } x<-5 \\ \dfrac{1}{y^2}, & \textit{при } y>5 \end{cases}$$
25	Написати функцію, яка виконує обчислення виразу, та протабулювати значення на інтервалі (-5;5) з кроком 0,25 $$y=\begin{cases} 10, & \text{при } x<0; \\ x\left(x^2+1\right), & \text{при } 0\le x\le 1; \\ x\ , & \text{при } 1<x\le 5; \end{cases}$$

Задача 3

Скласти лямбда-функцію, яка виконує розрахунок заданого виразу, та протабулювати значення функції на інтервалі (-5;5) з довільним кроком який вказується користувачем

№	$f(x)$	№	$f(x)$	№	$f(x)$
1	$y=3^x$	6	$y=e^{x\cos\frac{\pi}{4}}\cos\left(x\cdot\sin\dfrac{\pi}{4}\right)$	11	$y=\left(1+2x^2\right)\cdot e^{x^2}$
2	$y=-\ln\left\|2\sin\dfrac{x}{2}\right\|$	7	$y=\cos x$	12	$y=-\dfrac{1}{2}\ln\left(1-2x\cos\dfrac{\pi}{3}+x^2\right)$
3	$y=\sin x$	8	$y=\dfrac{x\cdot\sin\dfrac{\pi}{4}}{1-2x\cdot\cos\dfrac{\pi}{4}+x^2}$	13	$y=\dfrac{1}{2}\ln x$
4	$y=\dfrac{x}{2}$	9	$y=\dfrac{1}{4}\ln\dfrac{x+1}{1-x}+\dfrac{1}{2}arctg(x)$	14	$y=\dfrac{1}{4}\left(x^2-\dfrac{\pi^2}{3}\right)$
5	$y=e^x$	10	$y=e^{\cos(x)}\cos(\sin(x))$	15	$y=\dfrac{1+x^2}{2}arctg(x)-\dfrac{x}{2}$

Критерії оцінювання лабораторної роботи 4

За умови своєчасного та в повному обсязі виконаного завдання здобувач може отримати 100 балів (матеріал якісно викладено в логічній послідовності, без помилок, зроблено висновки до виконаного завдання, виявлено підвищений рівень володіння матеріалом).

Бали	Оцінка	Характеристика
90–100	Відмінно	Програма працює ідеально. Код чистий, документований, логіка позбавлена надмірності. Повна відповідність PEP 8.
74–89	Добре	Програма виконує всі завдання, але алгоритм може бути не найбільш оптимальним (наприклад, зайві вкладені умови, які можна було спростити через elif). Загалом код чистий, але є поодинокі порушення PEP 8 (зайві пробіли, задовгі рядки)
60–73	Задовільно	Програма працює, але реалізація "важка" і може містити незначні помилки. Використовуються лише найпростіші конструкції. Код недбалий, відступи можуть бути хаотичними.
Нижче 60	Незадовільно	Програма не запускається через синтаксичні помилки або логіка повністю суперечить завданню.

Контрольні питання

1. Як створити функцію, де один із параметрів є необов'язковим? Що станеться, якщо спробувати поставити параметр із дефолтним значенням *перед* обов'язковим параметром?
2. Поясніть різницю між викликом calculate(10, 5) та calculate(b=5, a=10). Чи можна передати іменований аргумент перед позиційним в одному виклику?
3. Назвіть хоча б дві вбудовані функції Python, у які найчастіше передають лямбда-вирази як аргументи.
4. Як визначити параметр функції, який буде приймати конкретне значення, якщо користувач не передав його при виклику?

Перелік рекомендованих джерел

1. Копей В. Б. Мова програмування Python для інженерів і науковців : навч. посіб. / В. Б. Копей. - Івано-Франківськ : ІФНТУНГ, 2019. – 272 с.
2. Костюченко А.О. Основи програмування мовою Python: навчальний посібник. Ч.: ФОП Баликіна С.М., 2020. – 180 с.
3. Основи програмування. Python. Частина 1 [Електронний ресурс]: підручник для студ. спеціальності 122 "Комп’ютерні науки"

/А.В. Яковенко; КПІ ім. Ігоря Сікорського. – Київ: КПІ ім. Ігоря Сікорського, 2018. – 195 с.
4. https://www.python.org/
5. https://www.w3schools.com/python/default.asp

Лабораторна робота № 5
Обробка списків, вкладених списків

Мета: закріпити теоретичні знання і розвинути практичні навички роботи із списками та вкладеними списками.

Теоретичні відомості

Робота зі списками (list) — це важлива тема у Python, оскільки це основний тип даних для зберігання впорядкованих колекцій об'єктів.

1. Створення списків

Списки є **змінними** (mutable) і можуть містити елементи різних типів.

- Порожній список: my_list = [] або my_list = list()
- Зі значеннями: numbers = [1, 2, 3, 4, 5]
- Різнотипний список**:** mixed = [1, "Apple", 3.14, True]

2. Доступ до елементів (Індексація та Зрізи)

Python використовує нульову індексацію.

- Пряма індексація: my_list[0] — перший елемент.
- Зворотна індексація: my_list[-1] — останній елемент.
- Зрізи (Slices): my_list[start:stop:step]
  - my_list[1:4] — елементи з 1-го по 3-й.
  - my_list[::-1] — розгортає список у зворотному порядку.

3. Додавання та розширення елементів

Існує три основні методи для додавання даних:

1. append(obj): Додає один об'єкт у кінець списку.
2. extend(iterable): Розширює список, додаючи елементи з іншої колекції (списку, кортежу).
3. insert(index, obj): Вставляє об'єкт у вказану позицію, зсуваючи інші елементи.

4. Видалення елементів

1. remove(value): Видаляє перший знайдений елемент із вказаним значенням (якщо значення немає — викликає помилку).
2. pop(index): Видаляє елемент за індексом і повертає його. Якщо індекс не вказано, видаляє останній.
3. clear(): Повністю очищує список.
4. del my_list[i]: Інструкція для видалення елемента або зрізу за індексом.

5. Пошук та інформація

- len(list): Кількість елементів.
- value in list: Перевірка наявності елемента (повертає True або False).
- index(value): Повертає індекс першого входження значення.

- count(value): Рахує, скільки разів значення зустрічається в списку.

6. Сортування та зміна порядку

- sort(): Сортує поточний список (змінює оригінал).
- sorted(list): Функція, що повертає новий відсортований список, не змінюючи старий.
- reverse(): Розгортає список "задом наперед".

7. Вкладені списки (Матриці)

Список може містити інші списки. Це дозволяє створювати структури типу таблиць.

```
matrix = [
    [1, 2, 3],
    [4, 5, 6],
    [7, 8, 9]
]
# Доступ до числа 5:
element = matrix[1][1] # Другий рядок, другий елемент
```

8. Генератори списків (List Comprehension)

Це компактний спосіб створення списків на основі існуючих послідовностей.
Синтаксис: [вираз for елемент in послідовність if умова]

Приклад *(квадрати парних чисел):*

```
squares = [x**2 for x in range(10) if x % 2 == 0]
```

9. Вкладені цикли

Python, як і будь-яка інша мова програмування дозволяє розміщувати один цикл усередині іншого. Це часто використовується для роботи з двовимірними структурами даних (матрицями, таблицями).

```
for i in range(3):
    for j in range(3):
        print(f"({i}, {j})", end=" ")
```

Приклад. Генерація таблиці множення від 1 до 10:

```
# 1. Формування таблиці за допомогою вкладених циклів
size = 10
multiplication_table = []

for i in range(1, size + 1):
    row = []
    for j in range(1, size + 1):
        row.append(i * j)
```

```
        multiplication_table.append(row)

    # 2. Красиве виведення таблиці на екран
    print("Таблиця множення:")
    for row in multiplication_table:
        # Використовуємо форматування рядків для вирівнювання колонок
        print(" ".join(f"{item:4}" for item in row))
```

Розбір логіки роботи:

1. **Зовнішній цикл for i**: Відповідає за множник першого числа (рядки). На кожній ітерації створюється порожній список row.
2. **Внутрішній цикл for j**: Відповідає за множник другого числа (колонки). Він наповнює поточний рядок результатами множення i * j.
3. **multiplication_table.append(row)**: Після того, як внутрішній цикл завершився, готовий рядок додається до головного списку.
4. **Форматування {item:4}**: Це вказує Python виділити під кожне число рівно 4 символи. Це потрібно для того, щоб колонки не "з'їжджали", коли числа стають двозначними.

"Пайтонічний" спосіб (List Comprehension)

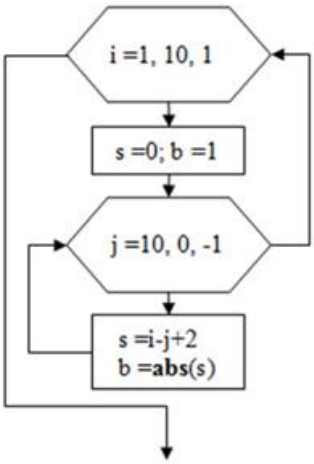
У Python цей же результат можна отримати набагато коротше, використовуючи вкладені генератори списків:

```
size = 10
# Створення таблиці в один рядок
table = [[i * j for j in range(1, size + 1)] for i in range(1, size + 1)]

# Вивід (перші 3 рядки для прикладу)
for row in table[:3]:
    print(row)
```

Такий підхід працює ідентично до попереднього, але виглядає професійніше та працює трохи швидше. Ви можете звернутися до будь-якого елемента таблиці за індексами, наприклад, table[4][4] видасть результат множення 5×5 (враховуючи, що індексація починається з 0).

Приклад зображення подвійних вкладених циклів у блок-схемах



Завдання до лабораторної роботи 5

Варіант	**Завдання**
1.	В матриці $n \times m$ замінити елементи третього стовпчика на їх синуси.
2.	В матриці $n \times n$ замінити елементи головної діагоналі на їх квадрати.
3.	В матриці $n \times m$ підрахувати суму елементів другого порядку.
4.	В матриці $n \times m$ замінити всі нулі на одиниці.
5.	З матриці $n \times m$ вивести на дисплей другий стовпчик.
6.	Знайти мінімальний елемент матриці $n \times m$.
7.	Вивести на екран вектор побудований з сум рядків матриці $n \times m$
8.	Побудувати матрицю $n \times m$, елементи якої дорівнюють сумі своїх індексів.
9.	Із вектора побудувати матрицю $n \times m$ першим та останнім рядком якого буде заданий вектор, другою та третьою - квадрати та куби його відповідних елементів.
10.	На базі матриці $n \times m$ побудувати іншу матрицю, елементи якої будуть дорівнювати квадратам елементів заданої матриці.
11.	В матриці $n \times m$ обчислити суму максимального та мінімального елементів.
12.	Із матриці $n \times n$ поміняти місцями першу строку та перший стовпчик.
13.	Обчислити суму елементів матриці $n \times m$.
14.	Транспонувати матрицю $n \times m$ (не використовуючи функції бібліотек Python). (Поміняти місцями її рядки та стовпчики).
15.	Знайти мінімальний елемент в першому стовпчику матриці $n \times m$
16.	З матриці $n \times n$ винести на екран елементи розташовані вище головної діагоналі.
17.	В матриці $n \times n$ знайти середнє арифметичне елементів її головної діагоналі.
18.	В матриці $n \times m$ замінити максимальний елемент на нуль.
19.	В матриці $n \times m$ замінити всі елементи менші 4 на нулі.

20.	Підрахувати в матриці $n \times m$ суму всіх елементів більших ніж 2.
21.	Підрахувати скільки елементів матриці $n \times m$ більші за 2.
22.	Обчислити середнє арифметичне першого рядка матриці $n \times m$
23.	Побудувати матрицю $n \times m$, елементи якої дорівнюють квадрату різниці їх індексів.
24.	Замінити в матриці $n \times n$ всі елементи розташовані на головній діагоналі на нулі.
25.	Замінити на нулі елементи матриці $n \times n$ розташовані нижче головної діагоналі.

Критерії оцінювання лабораторної роботи 4

За умови своєчасного та в повному обсязі виконаного завдання здобувач може отримати 100 балів (матеріал якісно викладено в логічній послідовності, без помилок, зроблено висновки до виконаного завдання, виявлено підвищений рівень володіння матеріалом).

Бали	Оцінка	Характеристика
90 - 100	Відмінно	Код чистий, документований, логіка позбавлена надмірності. Повна відповідність PEP 8.
75 - 89	Добре	Впевнене володіння списками, дрібні помилки.
60 - 74	Задовільно	Матеріал засвоєний, але є серйозні прогалини в логіці або стилі
0 - 59	Незадовільно	Програма має критичні помилки, потребує суттєвого доопрацювання.

Контрольні питання

1. Яким символом (дужками) позначаються списки в Python і чи можуть вони містити дані різних типів одночасно?
2. У чому різниця між методами .append() та .extend()? Що станеться, якщо передати список у метод .append()?
3. Як отримати останній елемент списку, не знаючи його точної довжини?
4. Яка функція використовується для визначення кількості елементів у списку?
5. Як працює інструкція del порівняно з методом .remove()?
6. Як перевірити, чи існує певне значення у списку, не використовуючи цикл for?
7. Який метод дозволяє вставити елемент у конкретну позицію списку, а не в кінець?

Перелік рекомендованих джерел

1. Копей В. Б. Мова програмування Python для інженерів і науковців : навч. посіб. / В. Б. Копей. - Івано-Франківськ : ІФНТУНГ, 2019. – 272 с.
2. Костюченко А.О. Основи програмування мовою Python: навчальний посібник. Ч.: ФОП Баликіна С.М., 2020. – 180 с.
3. Основи програмування. Python. Частина 1 [Електронний ресурс]: підручник для студ. спеціальності 122 "Комп'ютерні науки" /А.В. Яковенко; КПІ ім. Ігоря Сікорського. – Київ: КПІ ім. Ігоря Сікорського, 2018. – 195 с.
4. https://www.python.org/
5. https://www.w3schools.com/python/default.asp

Лабораторна робота № 6
Робота із структурами даних: кортежі, словники, множини

Мета: закріпити теоретичні знання і розвинути практичні навички по роботі зі структурами даних у Python.

Теоретичні відомості

Рядки - це впорядковані послідовності символів, що використовують для зберігання і подання текстової інформації (символів і слів, наприклад, ваше ім'я, змісту текстових файлів, завантажених в пам'ять, адрес в Інтернеті, програми на Python тощо). Рядки володіють потужним набором засобів для їх обробки.

Рядок (str) — це впорядкована послідовність символів Юнікоду (Unicode). Це означає, що Python однаково добре працює як з латиницею, так і з кирилицею чи емодзі.

Ключова особливість: Незмінність (Immutability)

Рядки в Python є **незмінними**. Ви не можете змінити символ всередині існуючого рядка. Будь-яка операція, яка нібито "змінює" рядок (наприклад, заміна букви), насправді створює **новий** об'єкт у пам'яті.

1. Створення та представлення

Рядки можна створювати кількома способами:

- **Одинарні або подвійні лапки:** 'Hello' або "Hello". Вони ідентичні.
- **Потрійні лапки:** '''Текст''' або """Текст""". Використовуються для багаторядкових текстів або документації (docstrings).
- **Спецсимволи:** \n (новий рядок), \t (табуляція), \\ (зворотний слеш).

2. Індексація та зрізи (Slicing)

Оскільки рядок — це послідовність, кожен символ має свій номер (індекс).

- **Пряма індексація:** Починається з 0 (ліворуч).
- **Зворотна індексація:** Починається з -1 (праворуч).

Зрізи

Формула: string[start:stop:step]

- start: Індекс початку (включно).
- stop: Індекс кінця (не включаючи).
- step: Крок (наприклад, 2 для кожного другого символу).

3. Основні методи та операції

Python має багату стандартну бібліотеку для маніпуляцій з текстом:

Математичні оператори

- **Конкатенація (+):** Об'єднання двох рядків.
- **Дублювання (*):** Повторення рядка n разів.

Метод	Опис
S.upper() / .lower()	Переведення у верхній/нижній регістр.
S.strip()	Видалення пробілів по боках.
S.replace(old, new)	Заміна підрядка.
S.split(separator)	Розбиття рядка в список за роздільником.
S.join(iterable)	Об'єднання списку в рядок.
S.find(sub)	Пошук індексу першого входження підрядка.
S.isdigit()	Перевірка вмісту
S .endswith(‘ spam ‘)	Перевірка закінчення рядка
for x in S: print(x)	Обхід в циклі

4. Форматування рядків

Існує три основні покоління форматування, але зараз стандартом є **f-рядки**:

1. **f-рядки (Python 3.6+):** f"Привіт, {name}!" — найшвидший і найчитабельніший спосіб.
2. **Метод .format():** "{} {}".format(a, b).
3. **Оператор %:** Застарілий спосіб у стилі мови C

Кортеж (tuple) - це незмінна структура даних, яка дуже схожа на список.

Кортежі можна використовувати замість списків, але вони мають менше можливостей, оскільки кортеж не може бутизмінений після створення.

1. Способи створення кортежів:

- **Використання дужок ():** найпоширеніший спосіб.

```
my_tuple = (1, 2, 3, "apple", True)
```

- **«Пакування» кортежу (без дужок):** елементи, розділені комами, автоматично стають кортежем.

```
another_tuple = 1, 2, 3  # (1, 2, 3)
```

- **Функція tuple():** деретворює ітеровані об'єкти (списки, рядки) на кортеж.

```
list_to_tuple = tuple([1, 2, 3]) # (1, 2, 3)
string_to_tuple = tuple("hello") # ('h', 'e', 'l', 'l', 'o')
```

- **Кортеж з одного елемента:** обов'язково потрібно ставити кому після елемента.

```
single_tuple = (5,)
# single_tuple = (5) -> Це буде просто число (int), а не кортеж
```

- **Порожній кортеж:**

```
empty_tuple = ()
```

2. Індексація та зрізи: доступ до елементів за індексом, як у списках, наприклад, t[0] або t[1:3].

3. Основні вбудовані методи кортежів:

- **count(x)** - повертає кількість разів, коли елемент x з'являється в кортежі.

```
t = (1, 2, 3, 2, 4)
print(t.count(2))  # Виведе: 2
```

- **index(x)** - повертає індекс першого входження елемента x. Якщо елемент не знайдено, викликається помилка ValueError.

```
t = ('a', 'b', 'c')
print(t.index('b'))  # Виведе: 1
```

- **max(), min(), sum():** Пошук найбільшого, найменшого елемента або суми значень (для числових кортежів).
- **sorted(t):** Створює новий відсортований список на основі елементів кортежу (сам кортеж залишається незмінним)

4. Операції, що часто використовуються з кортежами:

- **Конкатенація (+):** Об'єднання двох кортежів у новий.
- **Повторення (*):** Створення кортежу, що повторюється.
- **Індексація та зрізи ([]):** Доступ до елементів, наприклад, t[0] або t[1:3].

Оскільки кортежі незмінні, такі методи як **append(), remove(), pop(),** доступні для списків, у кортежах **відсутні**.

5. Переваги кортежів:

– захист від дурня, оскільки кортеж захищений від змін, як навмисних, так і випадкових,

– кортежі займають менше місця в пам’яті, працюють швидше,

– можна використовувати як ключі словника.

Словник — це впорядкована, змінна структура даних, що зберігає пари у вигляді «ключ: значення». Ключі унікальні та незмінні (рядки, числа, кортежі), а значення можуть бути будь-якого типу. Словники ідеальні для швидкого пошуку, додавання та оновлення даних за допомогою ключа, а не індексу.

1. Основні операції зі словниками:

- **Створення:** Використовуються фігурні дужки {} або конструктор dict().

```
user = {"name": "Tanya", "age": 25}
# Або
user = dict(name="Tanya", age=25)
```

- **Доступ до елементів:**

```
print(user["name"])  # Виведе: Tanya
print(user.get("age")) # Безпечний доступ (поверне None, якщо ключа немає)
```

- **Додавання/Зміна:**

```
user["city"] = "Kyiv"  # Додавання нового ключа
user["age"] = 26       # Зміна існуючого значення
```

- **Видалення:**

```
del user["city"]       # Видалення за ключем
user.pop("age")        # Видалення з поверненням значення
```

2. Методи словників:

- **.keys()** — повертає список ключів.
- **.values()** — повертає список значень.
- **.items()** — повертає пари (ключ, значення) для циклів.
- **.update()** — оновлює словник іншим словником або парою ключ-значення.
- **len(dict)** — кількість елементів у словнику.

3. Особливості роботи із словниками:

- ключі не можуть дублюватися,
- словники можна вкладати один в одного (словник у словнику) для зберігання складних даних.

Множина — це набір унікальних даних. Елементи множини не можуть дублюватися. Множина може містити будь-яку кількість елементів, і вони можуть бути різних типів (int, float, кортеж, рядки тощо). Але множина не може

мати ***змінювані*** елементи, такі як списки, словники або інші множини. Для створення множини всі елементи поміщають усередині фігурних дужок {}, розділених комами.

1. Створення множин

```
# Створення через фігурні дужки
my_set = {1, 2, 3, 3, 4}  # Результат: {1, 2, 3, 4}
# Створення порожньої множини (важливо: {} створить словник)
empty_set = set()
```

2. Основні операції та методи

- **додавання:**

```
my_set.add(5)
```

- **видалення:**

```
my_set.remove(3) (помилка, якщо елемента немає)
my_set.discard(3) (без помилки)
```

- **об'єднання (Union):**

```
set1 | set2
set1.union(set2)
```

- **перетин (Intersection):**

```
set1 & set2
set1.intersection(set2)
```

- **різниця (Difference):**

```
set1 - set2
set1.difference(set2)
```

- **симетрична різниця:**

```
set1 ^ set2
set1.symmetric_difference(set2)
```

Frozenset (Незмінна множина) — це версія множини, яку не можна змінювати після створення (immutable), що дозволяє використовувати її як ключ у словниках.

```
fs = frozenset([1, 2, 3, 4])
```

Множини ідеально підходять для швидкої перевірки членства (чи є елемент у наборі) та очищення даних від дублікатів.

Приклад. Є дані про студентів, їхні курси та оцінки. Потрібно:

1. Зберігати список студентів (ім'я + курс).
2. Зберігати оцінки студентів з різних предметів.
3. Знайти унікальні предмети, з яких складали іспити.
4. Обчислити середній бал для кожного студента.

```
# 1. Tuple (Кортеж) - для незмінних даних (студент: ім'я, курс)
student1 = ("Олексій", 2)
student2 = ("Марія", 3)
student3 = ("Іван", 2)

# 2. List (Список) - для зберігання колекції студентів
students = [student1, student2, student3]

# 3. Dict (Словник) - для зв'язку студента з його оцінками
# Ключ - ім'я студента, значення - словник предметів та оцінок
grades = {
    "Олексій": {"Математика": 90, "Програмування": 95},
    "Марія": {"Математика": 85, "Програмування": 98, "Фізика": 90},
    "Іван": {"Математика": 70, "Фізика": 75}
}

# 4. Set (Множина) - для пошуку унікальних предметів
unique_subjects = set()
for student_grades in grades.values():
    for subject in student_grades.keys():
        unique_subjects.add(subject)

print(f"Унікальні предмети: {unique_subjects}")

# Обчислення середнього бала
for student in students:
    name = student[0]
    if name in grades:
        scores = grades[name].values()
        average = sum(scores) / len(scores)
       print(f"Студент {name} (курс {student[1]}) - сер. бал: {average:.2f}")
```

Результат:

Унікальні предмети: {'Математика', 'Програмування', 'Фізика'}
Студент Олексій (курс 2) - сер. бал: 92.50
Студент Марія (курс 3) - сер. бал: 91.00
Студент Іван (курс 2) - сер. бал: 72.50
Чому використані саме ці структури:

- **Tuple ():** Використано для ("Олексій", 2), оскільки ім'я та курс студента — це цілісний, незмінний запис.

- **List []:** Використано для students, бо нам потрібно впорядковано зберігати всіх студентів і, можливо, додавати нових.
- **Dict {}:** Найкращий вибір для grades, оскільки дозволяє швидко знайти оцінки за іменем студента (ключем).
- **Set {}:** Використано для unique_subjects, оскільки множина автоматично видаляє дублікати (наприклад, "Математика" зустрічається у всіх, але в set потрапить лише один раз).

Цей приклад показує, як різні структури даних працюють разом для ефективного вирішення задачі.

Завдання до лабораторної роботи 6

Завдання 1.

Варіанти:

1. Задано речення. Скласти програму, яка визначає і виводить на екран найбільшу кількість прогалин, розташованих підряд.
2. Задано текст, в якому є дві і більше однакові літери. Скласти програму, яка визначає і виводить на екран найбільшу кількість однакових символів, розташованих підряд.
3. Задано слово. Скласти програму, яка визначає і виводить на екран кількість різних символів в ньому.
4. Задано слово, в якому є дві і більше однакові літери. Скласти програму, яка їх визначає і виводить на екран.
5. Задано три слова. Скласти програму, яка визначає і виводить на екран літери, які не повторюються в них.
6. Задано два слова. Скласти програму, яка визначає і виводить на екран ті літери слів, які є тільки в одному з них (в тому числі повторювані). Наприклад, якщо задано слова «процесор» та «інформація», то відповідь має вигляд: п е с і н ф м а і я.
7. Задано два слова. Скласти програму, _ка визначає, чи можна з літер першого з них здобути друге. Розглянути такі варіанти: 1) повторювані літери другого слова можуть в першому слові не повторюватися; 2) кожна літера другого слова повинна входити у перше слово стільки раз, скільки вона входить у друге.
8. Задано три слова. Скласти програму, яка визначає і виводить на екран ті літери слів, які є лише в одному зі слів. Розглянути такі варіанти: 1) повторювані літери кожного слова розглядаються; 2) повторювані літери кожного слова не розглядаються.
9. Задано три слова. Скласти програму, яка визначає і виводить на екран їх загальні літери. Повторювані літери кожного слова не розглядати.
10. Задано речення з десяти слів. Скласти програму, яка визначає і виводить

на екран заповнений ними список. Розділових знаків в реченні немає.

11. Задано речення. Скласти програму, яка визначає і виводить на екран речення, в якому слова розташовані в зворотному порядку (наприклад, речення «мама мила раму» буде змінено на «раму мила мама»).

12. Задано речення. Скласти програму, яка визначає і виводить на екран речення, в якому слова змінено місцями (наприклад, замість першого слова розташовано останнє, а замість останнього - перше).

13. Задано речення. Скласти програму, яка визначає і виводить на екран всі його слова, відмінні від слова «привіт».

14. Задано речення. Скласти програму, яка визначає і виводить на екран: а) кількість слів, які розпочинаються з літери «н»; б) кількість слів, які закінчуються на літеру «р».

15. Задано речення. Скласти програму, яка визначає і виводить на екран: слова а) які розпочинаються і закінчуються на одну і ту ж літеру; б) які містять три літери «е»; в) які містять хоча б одну літеру «о».

16. Задано речення. Скласти програму, яка визначає і виводить на екран будь-яке його слово, що розпочинається на літеру «к».

17. Задано речення. Скласти програму, яка визначає і виводить на екран довжину його самого короткого слова.

18. Задано речення. Скласти програму, яка визначає і виводить на екран його найдовше слово (прийняти, що таке слово є одним).

19. Задано речення. Скласти програму, яка визначає і виводить на екран, чи правдивим є твердження, що його найдовше слово має більше 10 символів.

20. Задано речення. Скласти програму, яка визначає і виводить на екран всі слова в порядку неспадання їх довжин.

21. Задано речення. Скласти програму, яка визначає і виводить на екран всі слова, які зустрічаються в реченні один раз.

22. Задано речення. Скласти програму, яка визначає і виводить на екран всі його різні слова.

23. Задано речення, в якому є тільки два однакових слова. Скласти програму, яка визначає їх і виводить на екран.

24. Задано речення. Скласти програму, яка визначає і виводить на екран всі його слова, попередньо перетворивши кожне слово за таким правилом: а) замінити першу зустрінуту літеру «а» на «о»; б) видалити зі слова всі входження останньої літери (крім неї самої), в) залишити в слові тільки перші входження кожної літери (інші видалити); г) в самому довгому слові видалити середню (середні) літери (прийняти, що таке слово є одним 0.

25. Задано текст з цифр і літер латинського алфавіту. Скласти програму, яка визначає, яких літер - голосних *{a, e, i, o, u, y}* або ‘приголосних більше в цьому тексті.

26. Задано текст з латинських літер. Скласти програму, яка визначає і виводить на екран в алфавітному порядку по одному разу всі голосні літери латинського алфавіту (множина *{a, e, i, o, u, y}),* які входять в цей текст. Текст та елементи множини задано в одному реєстрі (нижньому або верхньому).

Завдання 2.

Варіанти:

1. Напишіть програму, яка приймає рядок тексту (речення) від користувача (input). Розділіть рядок на слова (список). Використайте словник, щоб підрахувати кількість входжень кожного слова (частотний словник). Виведіть слова, що повторюються більше 2х разів.

2. Дано: students = [("Іван", [85, 90, 78]), ("Марія", [92, 88, 95]), ("Олег", [70, 75, 80])]. Створити словник {ім'я: середній бал}, вивести студентів із середнім балом > 85.

3. Бібліотека книг books = [("1984", "Орвелл", 1949), ("Кобзар", "Шевченко", 1840), ("Майстер і Маргарита", "Булгаков", 1967)]. Створити словник {автор: список книг}, знайти найстарішу книгу, створити множину книг.

4. Курси студентів courses = {"Іван": ("математика", "фізика"), "Марія": ("біологія", "хімія"), "Олег": ("математика", "інформатика")}. Створити множину всіх курсів, знайти студентів, які вивчають біологію або інформатику, створити словник {курс: список студентів}.

5. Є текст: text = "python code code data structures python data". Розбити на слова (список), створити множину унікальних слів, побудувати словник {слово: кількість}. Вивести топ-3 найчастіших слів

6. Магазин товарів у вигдяді (назва, кількість, ціна): products = [("яблуко", 10, 2.5), ("банан", 5, 3.0), ("апельсин", 8, 4.0)]. Створити словник {назва: загальна вартість}, знайти найдорожчий товар (за ціною) , створити множину товарів, яких більше ніж 7.

7. Соціальна мережа users = {"Іван": {"Марія", "Олег"}, "Марія": {"Іван", "Анна"}, "Олег": {"Іван"}}. Знайти спільних друзів Івана і Марії, додати нового користувача, рекомендувати друзів (друзі друзів як множина) .

8. Аналіз покупок orders = [("Іван", "яблуко"), ("Марія", "банан"), ("Іван", "банан"), ("Олег", "яблуко")]. Створити словник {користувач: список товарів}, знайти унікальні товари (множина) , визначити найпопулярніший товар.

9. Фільтрація чисел numbers = [1, 2, 3, 4, 5, 6, 6, 3, 2]. Створити множину унікальних чисел, розділити на парні і непарні (два списки), створити словник {число: квадрат} .

10. Курси студентів courses = {"Іван": ("математика", "фізика"), "Марія": ("біологія", "хімія"), "Олег": ("математика", "інформатика")}. Створити множину всіх курсів, знайти студентів, які вивчають математику, створити словник {курс: список студентів}.

11. Дано: students = [("Іван", [85, 90, 78]), ("Марія", [92, 88, 95]), ("Олег", [70, 75, 80])]. Створити словник {ім'я: середній бал}, вивести студентів із середнім балом в діапазоні між 74 та 82.

12. Ігрова статистика players = [("Іван", 120), ("Марія", 150), ("Олег", 120)]. Створити словник {гравець: очки}, знайти топ-гравця, згрупувати гравців за однаковими очками {очки: [гравці]}.

13. Email-фільтр emails = ["test@gmail.com", "user@yahoo.com", "admin@gmail.com", "hello@ukr.net"]. Отримати домени (множина), створити словник {домен: список email}, порахувати кількість email на кожному домені

14. Бібліотека книг books = [("1984", "Орвелл", 1949), ("Кобзар", "Шевченко", 1840), ("Майстер і Маргарита", "Булгаков", 1967)]. Створити словник {автор: список книг}, знайти найстарішу книгу, створити множину авторів.

15. Фільтрація чисел numbers = [1, 2, 3, 4, 5, 6, 6, 13, 12, 8]. Створити множину унікальних чисел, розділити на прості і складені (два списки), створити словник {число: квадрат} .

16. Магазин товарів у вигдяді (назва, кількість, ціна): products = [("яблуко", 10, 2.5), ("банан", 5, 3.0), ("апельсин", 8, 4.0)]. Створити словник {назва: загальна вартість}, знайти найдешевший товар (за ціною) , створити множину товарів, ціна яких більше ніж середня.

17. Розклад занять schedule = [("Понеділок", "математика"), ("Вівторок", "фізика"), ("Понеділок", "інформатика"), ("Середа", "математика")]. Створити словник {день: список предметів}. Знайти всі унікальні предмети (множина). Вивести дні, коли є математика, порахувати кількість занять на кожен день.

18. Аналіз логів входу logs = [("Іван", "success"), ("Марія", "fail"), ("Іван", "fail"), ("Олег", "success")]. Створити словник {користувач: список статусів}. Порахувати кількість успішних входів для кожного. Знайти користувачів, у яких були помилки (множина), визначити користувача з найбільшою кількістю невдач.

19. Фільми та жанри movies = [("Inception", ("sci-fi", "thriller")), ("Titanic", ("romance", "drama")), ("Matrix", ("sci-fi", "action"))]. Створити словник {жанр: список фільмів, знайти всі унікальні жанри, вивести фільми жанру "sci-fi". Знайти жанр із найбільшою кількістю фільмів.

20. Результати тестування results = [("Іван", {"math": 80, "eng": 75}), ("Марія", {"math": 90, "eng": 85}), ("Олег", {"math": 70, "eng": 60})]. Створити словник {ім'я: середній бал}. Знайти студента з найкращим результатом з математики. Створити множину всіх предметів. Вивести студентів, у яких середній бал > 80

21. Транспорт і маршрути routes = [("bus_1", ["A", "B", "C"]), ("bus_2", ["B", "C", "D"]), ("bus_3", ["A", "D"])]. Створити словник {маршрут: список зупинок}. Знайти всі унікальні зупинки. Визначити зупинки, які обслуговуються більше ніж одним маршрутом. Знайти маршрути, які проходять через зупинку "C".

Критерії оцінювання лабораторної роботи 5

За умови своєчасного та в повному обсязі виконаного завдання здобувач може отримати 100 балів (матеріал якісно викладено в логічній послідовності, без помилок, зроблено висновки до виконаного завдання, виявлено підвищений рівень володіння матеріалом).

Бали	Оцінка	Характеристика
90 - 100	Відмінно	Код чистий, документований, логіка позбавлена надмірності. Використано методи до відповідних структур даних.
75 - 89	Добре	Впевнене володіння матеріалом, дрібні помилки.
60 - 74	Задовільно	Матеріал засвоєний, але є серйозні прогалини в логіці використання структур даних.
0 - 59	Незадовільно	Програма має критичні помилки, потребує суттєвого доопрацювання.

Контрольні питанння

1. Чим кортеж (tuple) відрізняється від списку (list) у Python? Наведіть щонайменше дві відмінності.
2. Як створити кортеж з одного елемента? Чому простий запис у дужках може не працювати?
3. Що таке ключ і значення у словнику (dict)? Які типи даних можуть бути ключами словника?
4. Як додати нову пару «ключ–значення» у словник та як безпечно отримати значення за ключем, якщо такого ключа може не існувати?
5. Чим множина (set) відрізняється від списку та кортежа? Яку основну властивість мають елементи множини?
6. У яких випадках доцільніше використовувати: кортеж, словник, множину? Наведіть по одному прикладу для кожної структури.

Перелік рекомендованих джерел

1. Копей В. Б. Мова програмування Python для інженерів і науковців : навч. посіб. / В. Б. Копей. - Івано-Франківськ : ІФНТУНГ, 2019. – 272 с.
2. Костюченко А.О. Основи програмування мовою Python: навчальний посібник. Ч.: ФОП Баликіна С.М., 2020. – 180 с.
3. Основи програмування. Python. Частина 1 [Електронний ресурс]: підручник для студ. спеціальності 122 "Комп’ютерні науки" /А.В. Яковенко; КПІ ім. Ігоря Сікорського. – Київ: КПІ ім. Ігоря Сікорського, 2018. – 195 с.
4. https://www.python.org/
5. https://www.w3schools.com/python/default.asp

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Кормен, Томас Г. Вступ до алгоритмів : Переклад з англійської третього видання : [укр.] Introduction to Algorithms: Third Edition : [пер. з англ.] / Томас Г. Кормен, Чарлз Е. Лейзерсон, Роналд Л. Рівест, Кліфорд Стайн, –К.: К. І. С., 2019. – 1288 с.
2. Трінтіна Н.А., Негоденко О.В., Гаманюк І.М., Шевченко С.М. Програмування мовою Python. Навчальний посібник підготовлено до друку для самостійної роботи студентів вищих навчальних закладів. – К.: ННІІТ ДУТ, 2022. – 113 с.
3. Копей В. Б. Мова програмування Python для інженерів і науковців : навч. посіб. / В. Б. Копей. - Івано-Франківськ : ІФНТУНГ, 2019. – 272 с.
4. Костюченко А.О. Основи програмування мовою Python: навчальний посібник. Ч.: ФОП Баликіна С.М., 2020. – 180 с.
5. Креневич А.П. Python у прикладах і задачах. Частина 2. Об’єктно-орієнтоване програмування. Навчальний посібник – К.: ВПЦ "Київський Університет", 2020. – 152 с.
6. Основи програмування. Python. Частина 1 [Електронний ресурс]: підручник для студ. спеціальності 122 "Комп’ютерні науки" /А.В. Яковенко; КПІ ім. Ігоря Сікорського. – Київ: КПІ ім. Ігоря Сікорського, 2018. – 195 с.
7. Руденко В., Жугастров О. Інформатика. Основи алгоритмізації та програмування мовою Python. Харків: Ранок, 2019. – 192 с.
8. Хом'як Т.В., Коханчик Н.С., Малієнко А.В. Вирішення задачі маршрутизації транспорту на підприємстві // Збірник наукових праць НГУ, № 63, 2020. – С. 145-155. (https://doi.org/10.33271/crpnmu/63.145)
9. Хом’як Т.В., Прус О. Системний аналіз виявлення проблем системи освіти та шляхи їх вирішення // Information Technology: Computer Science, Software Engineering and Cyber Security, Вип. 3, 2024. – С. 180-188. (https://doi.org/10.32782/IT/2024-3-1)
10. Хом’як Т,, Малієнко А., Безугла О., Гаранжа Д. Аналіз і розрахунок оптимальної кількості операторів контактного центру компанії // Information Technology: Computer Science, Software Engineering and Cyber Security, Вип. 2, 2025. – С. 166-174. (https://doi.org/10.32782/IT/2025-2-17)
11. Хом’як Т.В. Аналіз та моделювання діяльності додаткових курсів на кафедрі: дипл. проєкт магістра. Запоріжжя : Запорізька політехніка, 2023. 87 с. Режим доступу: https://eir.zp.edu.ua/items/99cbe81d-579b-411e-8842-2023ee03929c

ДОДАТОК А. ПРИКЛАД ОФОРМЛЕННЯ ТИТУЛЬНОЇ СТОРІНКИ

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ
«ДНІПРОВСЬКА ПОЛІТЕХНІКА»

ФАКУЛЬТЕТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
Кафедра Системного аналізу та управління

ЛАБОРАТОРНА РОБОТА №___
з дисципліни «Програмування та алгоритмічні мови»
на тему: _____

Виконав:
здобувач групи 124-24-1
ПІБ

Прийняв:
ПІБ викладача (ів)

Дніпро
2026

Навчальне видання

Хом'як Тетяна Валеріївна
Шевченко Юлія Олександрівна
Гаранжа Дмитро Миколайович

ПРОГРАМУВАННЯ ТА АЛГОРИТМІЧНІ МОВИ

Методичні рекомендації до виконання лабораторних робіт
для здобувачів ступеня бакалавра
освітньо-професійної програми «Системний аналіз»
зі спеціальності 124 Системний аналіз

Видано в авторській редакції.

Електронний ресурс.
Підписано до видання 10.02.2026. Авт. арк. 2,1.

Національний технічний університет «Дніпровська політехніка».
49005, м. Дніпро, просп. Дмитра Яворницького, 19.