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 2

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 2 / 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. – 48 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 № 1 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 мовою 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 вказаних спеціальностей 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
Обробка текстової інформації. Робота з файлами в мові Python

Мета: закріпити теоретичні знання і розвинути практичні навички у роботі з файлами та обробкою інформації, що в них міститься.

Теоретичні відомості

Робота з файлами в Python — це база, без якої не обходиться практично жоден серйозний проєкт, особливо у сфері діяльності аналізу даних. Python має вбудовану функцію open(), яка є універсальним ключем до більшості форматів, але для CSV та JSON існують спеціалізовані модулі, що значно полегшують життя.

1. Загальні принципи: Функція "open()"

Перш ніж читати чи писати, файл треба його відкрити. Найкращий спосіб — використання менеджера контексту with, який автоматично закриває файл після завершення роботи.

Основні режими:

- 'r': читання (стандартний).
- 'w': запис (створює новий або перезаписує існуючий).
- 'a': додавання в кінець файлу.

2. Текстові файли (.txt)

Це найпростіший формат. Дані зчитуються як звичайні рядки.

- **Читання:**

Метод	Пояснення
file.read()	весь файл
file.readline()	один рядок або ітерація по об'єкту файлу

- **Запис:** file.write("текст").

3. Файли CSV (Comma Separated Values)

Файли CSV представляють табличні дані. Хоча їх можна читати як *.txt*, краще використовувати вбудований модуль *csv*.

- **csv.reader**: повертає ітератор, де кожен рядок — це список значень.
- **csv.DictReader**: надзвичайно зручний інструмент, який перетворює кожен рядок у словник (dict), де ключами є заголовки стовпців.
- **Запис**: використовуйте csv.writer або csv.DictWriter.

4. Файли JSON (JavaScript Object Notation)

JSON — це стандарт для обміну даними у різноманітних IT-системах. Структура JSON майже ідентична пайтонівським словникам та спискам. Для роботи потрібен модуль json.

Основні методи:

Метод	Пояснення
json.load(file)	Зчитує дані з файлу і перетворює на об'єкт Python
json.loads(string)	Перетворює рядок формату JSON на об'єкт Python
json.dump(obj, file)	Записує об'єкт Python у файл у форматі JSON
json.dumps(obj)	Перетворює об'єкт Python на рядок JSON

Порівняльна таблиця форматів файлів та способів їх використання

Формат	Для чого найкраще підходить	Модуль	Складність структури
TXT	Нотатки, логи, сирий текст	Вбудовано	Низька (рядки)
CSV	Таблиці, бази даних, Excel	csv	Середня (рядки/стовпці)
JSON	Налаштування, API, вкладені дані (в деяких випадках вивантажені дані з бази даних)	json	Висока (ієрархічна)

Приклад 1. Робота з TXT (Текстові файли)

```
# ЗАПИС у файл
lines = ["Перший рядок\n", "Другий рядок\n", "Привіт, Python!"]
with open('example.txt', 'w', encoding='utf-8') as file:
    file.writelines(lines)  # запис списку рядків
    file.write("\nДодатковий текст") # запис одного рядка

# ЧИТАННЯ з файлу
with open('example.txt', 'r', encoding='utf-8') as file:
    content = file.read()  # зчитує весь файл як один рядок
    print("Вміст файлу:\n", content)

# ЧИТАННЯ порядково (економно для пам'яті)
with open('example.txt', 'r', encoding='utf-8') as file:
    for line in file:
        print(f"Обробка: {line.strip()}")
```

Приклад 2. Робота з CSV (Табличні дані)

```
import csv

data = [
    {"name": "Олексій", "age": 25, "city": "Київ"},
    {"name": "Марія", "age": 30, "city": "Львів"}
]
```

```
# ЗАПИС у CSV (як словник)
with open('users.csv', 'w', newline='', encoding='utf-8') as f:
    writer = csv.DictWriter(f, fieldnames=["name", "age", "city"])
    writer.writeheader()  # записуємо заголовки стовпців
    writer.writerows(data)

# ЧИТАННЯ з CSV
with open('users.csv', 'r', encoding='utf-8') as f:
    reader = csv.DictReader(f)
    for row in reader:
        print(f"Користувач {row['name']} живе в місті {row['city']}")
```

Приклад 3. Робота з JSON (Структуровані дані)

```
import json

config = {
    "app_name": "MyStudio",
    "version": 1.5,
    "features": ["auth", "billing", "notifications"],
    "settings": {"theme": "dark", "language": "uk"}
}

# ЗАПИС у JSON
with open('config.json', 'w', encoding='utf-8') as f:
    # ensure_ascii=False дозволяє зберігати кирилицю як текст, а не коди
\u...
    # indent=4 робить файл "красивим" та читабельним для людини
    json.dump(config, f, ensure_ascii=False, indent=4)

# ЧИТАННЯ з JSON
with open('config.json', 'r', encoding='utf-8') as f:
    loaded_data = json.load(f)
    print(f"Версія програми: {loaded_data['version']}")
    print(f"Обрана тема: {loaded_data['settings']['theme']}")
```

Приклад 4. Приклад зчитування JSON із .txt файлу

Припустимо, що існує файл data.txt з таким вмістом: {"user_id": 101, "status": "active", "roles": ["admin", "editor"]}

```
import json

# Шлях до файлу (навіть якщо розширення .txt)
file_path = 'data.txt'
```

```
with open(file_path, 'r', encoding='utf-8') as file:
    # json.load() зчитує текст і автоматично перетворює його на словник
Python
    data = json.load(file)

# Тепер ми можемо працювати з даними як зі словником
print(f"ID користувача: {data['user_id']}")
print(f"Перша роль: {data['roles'][0]}")
```

Важливий нюанс: json.load vs json.loads

Інколи розробники спочатку зчитують файл як звичайний текст, а потім конвертують його. Це виглядає так:

1. **json.load(fp)** — зчитує **безпосередньо з об'єкта файлу** (швидше і зручніше).
2. **json.loads(s)** — зчитує **з рядка** (якщо ви вже зчитали файл методом .read()).

Завдання до лабораторної роботи 1

Варіант 1

Дано файл з курсами валют та цінних металів за період часу за курсом НБУ

Необхідно:

1. Визначити кількість днів, за які наведені курси валют
2. Розрахувати середній курс по кожній з валют за період
3. Визначити мінімальний курс Форинта і день коли він був зафіксований
4. Результат зберегти у вигляді json файлу. Додатково у вихідному файлі записати мінімальну дату періоду та максимальну дату

Варіант 2

Дано файл з показниками погоди в області міста Дарем (Північна Кароліна) за період часу. Тепература вказана у Фаренгейтах

(Довідка $°F = °C * 1.8 + 32$)

Необхідно:

1. Визначити кількість днів, за які наведені показники погоди
2. Визначити яка кількість днів, коли фіксувалась погода, температура була вище 0 за Цельсієм
3. Визначити мінімальну температуру і день її фіксації в межах кожного року
4. Результат зберегти у вигляді json файлу. Додатково у вихідному файлі записати мінімальну дату періоду та максимальну дату

Варіант 3

Дано файл з заявами вступників.

Необхідно:

1. Визначити кількість днів, за які наведені дані

2. Визначити розподіл заяв (кількість) за факультетами

3. Визначити розподіл заяв за спеціальностями.

4. Результат зберегти у вигляді json файлу. Додатково у вихідному файлі записати мінімальну дату періоду та максимальну дату

Варіант 4

Дано файл з заявами вступників до НТУ Дніпровська Політехніка.

Необхідно:

1. Визначити кількість днів, за які наведені дані

2. Визначити розподіл заяв освітнім ступенем (бакалавр\магістр)

3. Визначити розподіл заяв за роком отримання освіти, на базі якої відбувається вступ. Груп виокремити у кількості 5 шт як рівнормірно визначені між максимальною та мінімальною датами отримання документа про освіту попереднього рівня.

4. Результати зберегти у вигляді json файлу. Додатково у вихідному файлі записати мінімальну та максимальну дати періоду

Варіант виконання завдання обирається за номером у списку групи

Варіант 1	**Варіант 2**	**Варіант 3**	**Варіант 4**
1, 5, 9, 13, 17	2, 6, 10, 14, 18	3, 7, 11, 15, 19	4, 8, 12, 16, 20

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	Код професійний, обробляє всі помилки, дані зберігаються без втрат, дотримано всіх стандартів
74–89	Добре	Програма працює правильно, але відсутні деякі перевірки (наприклад, на права доступу до файлу) або кодування
60–73	Задовільно	Основне завдання виконано, але код неоптимальний, відсутня обробка помилок, назви змінних незрозумілі
Менше 60	Незадовільно	Програма не виконує основну функцію (не зчитує або не записує дані)

Контрольні питання:

1. Яка вбудована функція в Python використовується для відкриття файлів?
2. У чому головна перевага використання оператора with (менеджера контексту) порівняно з ручним викликом .close()?
3. Назвіть три основні режими відкриття файлів ('r', 'w', 'a') та поясніть різницю між ними.
4. Який параметр функції open() потрібно обов'язково вказувати для коректної роботи з українським текстом?
5. Чим відрізняється метод .read() від .readlines()?
6. Що станеться з даними у файлі, якщо відкрити його в режимі 'w', записати один рядок і закрити?
7. Чому формат JSON вважається кращим за CSV для збереження налаштувань програми зі складною вкладеною структурою?

Перелік рекомендованих джерел

1. Костюченко А.О. Основи програмування мовою Python: навчальний посібник. Ч.: ФОП Баликіна С.М., 2020. – 180 с.
2. 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.
3. **https://www.python.org/**

Лабораторна робота № 2
Програмування з використанням виключень

Мета: закріпити теоретичні знання і розвинути практичні навички з обробки помилок під час виконання програми, з використання конструкцій try-except-else-finally, з генерації власних виключень, з роботи зі стеком помилок.

Теоретичні відомості

Робота з виключеннями (exceptions) у Python — це механізм, який дозволяє програмі граціозно обробляти помилки під час виконання, не перериваючи роботу всього додатка.

1. Конструкція try-except

Основний блок для обробки помилок. Код, який може викликати виключення, розміщується в try, а логіка обробки — в except.

Приклад 1

```
try:
    результат = 10 / 0
except ZeroDivisionError:
    print("Ділення на нуль неможливе!")
```

Додаткові блоки:

- **else**: Виконується лише тоді, коли в блоці try **не виникло** жодних помилок.
- **finally**: Виконується **завжди**, незалежно від того, була помилка чи ні. Зазвичай використовується для звільнення ресурсів (закриття файлів, з'єднань з БД).

2. Ієрархія виключень

Всі виключення в Python є об'єктами і походять від базового класу BaseException. Більшість стандартних помилок успадковуються від Exception.

Поширені типи виключень:

Тип	Опис
TypeError	Операція застосована до об'єкта невідповідного типу
ValueError	Тип аргументу правильний, але значення недопустиме
IndexError	Індекс списку поза межами діапазону
KeyError	Звернення до неіснуючого ключа в словнику
FileNotFoundError	Спроба відкрити файл, якого не існує

3. Генерація виключень (raise)

Ви можете самостійно викликати помилку за допомогою ключового слова raise. Це корисно для валідації даних.

Приклад 2

```
age = -5
if age < 0:
    raise ValueError("Вік не може бути від'ємним!")
```

4. Створення власних виключень

Для специфічних потреб проєкту можна створювати власні класи помилок, успадковуючись від базового класу Exception. Більш детально ця тема буде висвітлюватись після вивчення базової теорії ООП

Приклад 3

```
class MyCustomError(Exception):
    """Клас для специфічної помилки мого додатка"""
    pass
```

5. Принцип LBYL vs EAFP

У Python домінує підхід **EAFP** (*Easier to Ask for Forgiveness than Permission* — легше попросити вибачення, ніж дозволу).

- **LBYL (Look Before You Leap):** Перевіряти всі умови перед виконанням дії (багато if).
- **EAFP:** Просто виконувати дію і ловити помилку, якщо вона станеться. Це вважається більш "пітонічним" стилем, оскільки він чистіший і часто продуктивніший.

Загальні поради щодо чистого коду:

1. **Не використовуйте "голий" except:** : Завжди вказуйте конкретний тип помилки. Випадкове перехоплення системних сигналів (як-от SystemExit) може завадити зупинці програми.
2. **Мінімізуйте код у блоці try**: Чим менше рядків у try, тим легше зрозуміти, де саме виникла проблема.
3. **Логування**: Замість простого print() у блоці except, використовуйте модуль logging для збереження історії помилок.

Приклад 4. Реалізація логування

```
import logging

# 1. Налаштування логування
# Записуємо у файл 'app.log', рівень DEBUG та вище
logging.basicConfig(
    level=logging.INFO,
    filename='app.log',
    filemode='a',
    format='%(asctime)s - %(levelname)s - %(message)s'
```

```
)

def divide_numbers(a, b):
    try:
        logging.info(f"Спроба ділення {a} на {b}")
        result = a / b
    except ZeroDivisionError:
        # logging.exception автоматично додає Traceback (стек помилки) у
лог
        logging.exception("Виникла помилка: Спроба ділення на нуль!")
        return None
    except TypeError as e:
        logging.error(f"Помилка типів даних: {e}")
        return None
    else:
        logging.info(f"Операція успішна. Результат: {result}")
        return result
    finally:
        logging.info("Завершення блоку обробки ділення.")

# Приклади виклику
divide_numbers(10, 2)
divide_numbers(10, 0)
divide_numbers(10, "abc")
```

Чому це краще за print?

1. **Рівні важливості**: Ви можете фільтрувати повідомлення. Наприклад, у робочому середовищі (production) показувати лише ERROR, а під час розробки — DEBUG.
2. **logging.exception()**: Це "кілер-фіча". Вона записує не просто текст "Сталася помилка", а весь **Traceback** — детальний шлях по коду, який призвів до збою. Це критично важливо для відладки коду.
3. **Автоматизація**: Логи потенційно можуть бути автоматично надісані на зовнішні сервери або до систем моніторингу.

Завдання до лабораторної роботи 2

Варіант 1: Калькулятор індексу маси тіла (BMI)

Напишіть програму, яка запитує вагу (кг) та зріст (м). Обробіть ValueError, якщо введено не число, та виведіть попередження, якщо зріст дорівнює нулю (ZeroDivisionError).

Варіант 2: Конвертор валют

Користувач вводить суму в гривнях та курс долара. Обробіть помилку ділення на нуль, якщо курс введено як 0, та помилку типу даних, якщо введено текст.

Варіант 3: Доступ до елемента словника

Створіть словник із 5 парами "країна: столиця". Користувач вводить назву країни. Обробіть KeyError, якщо такої країни немає у списку.

Варіант 4: Перетворення списку рядків

Дано список ['10', '20', 'apple', '30']. Напишіть цикл, який намагається перетворити кожен елемент на int і додати до загальної суми. Обробіть ValueError для нечислових значень, просто пропускаючи їх.

Варіант 5: Читання конфігураційного файлу

Програма має відкрити файл config.txt. Обробіть FileNotFoundError. Якщо файл знайдено, виведіть повідомлення "Файл зчитано", інакше — "Файл відсутній".

Варіант 6: Обчислення квадратного кореня

Користувач вводить число. Використайте math.sqrt(). Обробіть ValueError, якщо користувач ввів від'ємне число.

Варіант 7: Доступ до списку за індексом

Створіть список фруктів. Користувач вводить порядковий номер. Обробіть IndexError, якщо номер завеликий, та ValueError, якщо введено не ціле число.

Варіант 8: Математична формула

Реалізуйте обчислення $y = 1 / (x - 5)$. Користувач вводить x. Обробіть ситуацію, коли $x = 5$ (ZeroDivisionError).

Варіант 9: Робота з декількома типами помилок

Напишіть програму, яка приймає рядок і намагається:

1) перетворити його на число;

2) поділити 100 на це число.

Обробіть одним блоком (ValueError, ZeroDivisionError).

Варіант 10: Запис у файл з обмеженими правами

Напишіть скрипт, що намагається записати текст у файл, який відкритий лише для читання або знаходиться в захищеній системній папці. Обробіть PermissionError.

Варіант 11: Перевірка довжини пароля

Користувач вводить пароль. Якщо довжина менше 8 символів, примусово викличте виключення за допомогою raise Exception("Пароль занадто короткий") (без створення класу).

Варіант 12: Злиття двох списків

Дано два списки різної довжини. Програма має перемножити елементи з однаковими індексами. Обробіть помилку, коли ітерація виходить за межі коротшого списку.

Варіант 13: Пошук модуля числа

Користувач вводить значення. Спробуйте застосувати функцію abs(). Обробіть TypeError, якщо передано об'єкт, який не підтримує цю операцію (наприклад, список).

Варіант 14: Робота з датами

Користувач вводить день, місяць і рік. Спробуйте створити об'єкт datetime. Обробіть ValueError, якщо введено неіснуючу дату (наприклад, 31 лютого).

Варіант 15: Копіювання вмісту файлу

Програма запитує шлях до двох файлів. З першого читає, у другий пише. Обробіть FileNotFoundError для першого та IOError для другого.

Варіант 16: Сума значень у списку через input

Користувач вводить числа через пробіл. Програма має їх підсумувати. Якщо серед "чисел" є текст, обробіть помилку всередині генератора списку або циклу.

Варіант 17: Видалення елемента із множини (Set)

Створіть множину. Користувач вводить елемент для видалення через set.remove(). Обробіть KeyError, якщо елемента не існує (або поясніть різницю з discard()).

Варіант 18: Конвертація типів у кортежі

Маємо кортеж (1, "2", [3], 4). Програма намагається помножити кожен елемент на 10. Обробіть TypeError для елементів, які не можна множити на число (наприклад, список на число множити можна, але результат буде іншим — розгляньте логічну помилку).

Варіант 19: Робота з системними аргументами

Використайте sys.argv. Програма має вивести перший аргумент командного рядка. Обробіть IndexError, якщо програму запущено без аргументів.

Варіант 20: Recursion Error

Напишіть функцію, яка викликає сама себе без умови виходу. Обробіть RecursionError у блоці try-except і виведіть повідомлення про досягнення ліміту рекурсії.

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	Виключення оброблені професійно, код чистий.
75–89	Добре	Робота виконана повністю, але є незначні зауваження до PEP 8 або оброблені не всі типи помилок.
60–74	Задовільно	Програма працює, але обробка помилок поверхнева (тільки try-except), відсутні блоки else/finally.
< 60	Незадовільно	Критичні помилки в логіці, програма завершується аварійно (traceback) на типових помилках.

Контрольні питання:

1. Різниця між except Exception та except:
2. Навіщо потрібен блок else, якщо код можна просто написати після except?
3. Коли блок finally є незамінним?
4. Що таке "прокидання" виключення (оператор raise)?
5. Чому важливо вказувати конкретний тип помилки (наприклад, ValueError), а не ловити всі одразу?

Перелік рекомендованих джерел

1. Костюченко А.О. Основи програмування мовою Python: навчальний посібник. Ч.: ФОП Баликіна С.М., 2020. – 180 с.
2. Копей В. Б. Мова програмування Python для інженерів і науковців : навч. посіб. / В. Б. Копей. - Івано-Франківськ : ІФНТУНГ, 2019. – 272 с.
3. **https://www.python.org/**

Лабораторна робота № 3
ООП, класи, методи, основні принципи

Мета: закріпити теоретичні знання і розвинути практичні навички з розробки класів та методів класу, використання основних принципів об'єктно-орієнтованого програмування (ООП).

Теоретичні відомості

Об'єктно-орієнтоване програмування (ООП) — це парадигма програмування, де основна увага приділяється не діям (функціям), а об'єктам. У мові Python майже все є об'єктом, від чисел до списків.

1. Основні поняття: Клас та Об'єкт

Щоб зрозуміти різницю, можна використати аналогію з будівництвом:

- **Клас** — це креслення або план будинку. Він описує, які властивості матиме майбутня споруда.
- **Об'єкт (екземпляр)** — це реальний будинок, побудований за цим кресленням. Таких будинків може бути багато, і вони можуть відрізнятися кольором стін чи меблями.
- **Атрибути** — це змінні, що зберігають стан об'єкта (наприклад, колір, марка).
- **Методи** — це функції, визначені всередині класу, що описують поведінку об'єкта (наприклад, "їхати", "гальмувати").

2. Структура класу в Python

Клас створюється за допомогою ключового слова class.

Конструктор __init__

Це спеціальний метод, який автоматично викликається при створенні нового об'єкта. Він ініціалізує початкові значення атрибутів.

Параметр self

Це посилання на поточний екземпляр класу. Завдяки self методи знають, з даними якого саме об'єкта вони працюють.

Приклад 1 Клас машина

```
class Car:
 def __init__(self, brand, color):
     self.brand = brand  # Атрибут екземпляра
     self.color = color

 def drive(self):
     print(f"{self.brand} їде!")
```

3. Чотири "кити" ООП

Коли ми говоримо про класи, ми завжди маємо на увазі ці фундаментальні принципи:

Принцип	Опис
Інкапсуляція	Приховування внутрішньої логіки та захист даних. У Python реалізується через префікси _ (protected) та __ (private).
Успадкування	Можливість створити новий клас на основі існуючого, запозичуючи його функціонал.
Поліморфізм	Здатність різних об'єктів виконувати одну і ту саму дію по-своєму (різні класи мають методи з однаковою назвою).
Абстракція	Виділення головних рис об'єкта, відкидаючи другорядні деталі.

4. Успадкування (Inheritance)

Це механізм, що дозволяє уникнути дублювання коду. Дочірній клас отримує всі методи та атрибути батьківського.

Приклад 2. Успадкування

```
class ElectricCar(Car): # Успадковуємо від Car
    def charge(self):
        print("Батарея заряджається.")
```

5. Методи екземпляра vs Методи класу

1. **Методи екземпляра**: Працюють з конкретним об'єктом через self.
2. **Методи класу (@classmethod)**: Працюють з самим класом, а не об'єктом. Приймають параметр cls.
3. **Статичні методи (@staticmethod)**: Просто функції всередині класу, які не мають доступу ні до self, ні до cls.

Приклад 3. Система "BankSystem"

```
from abc import ABC, abstractmethod

# 1. АБСТРАКЦІЯ: Створюємо шаблон для всіх рахунків
class Account(ABC):
    def __init__(self, owner, balance):
        self.owner = owner
        self._balance = balance  # Інкапсуляція (protected)

    @abstractmethod
    def deposit(self, amount):
        pass

    @abstractmethod
    def withdraw(self, amount):
        pass
```

```python
# 2. УСПАДКУВАННЯ: Створюємо конкретний тип рахунку на основі абстрактного
class SavingsAccount(Account):
    def __init__(self, owner, balance, interest_rate=0.02):
        super().__init__(owner, balance)
        self.__interest_rate = interest_rate  # 3. ІНКАПСУЛЯЦІЯ: Приватний
атрибут (private)

    def deposit(self, amount):
        self._balance += amount
        print(f"Поповнено на {amount}. Новий баланс: {self._balance}")

    def withdraw(self, amount):
        if amount <= self._balance:
            self._balance -= amount
            print(f"Знято {amount}. Залишок: {self._balance}")
        else:
            print("Недостатньо коштів!")

    # Геттер для доступу до приватного поля
    def get_info(self):
        return f"Власник: {self.owner}, Ставка: {self.__interest_rate *
100}%"

# 4. ПОЛІМОРФІЗМ: Інший клас з тими ж методами, але іншою логікою
class BusinessAccount(Account):
    def deposit(self, amount):
        # Бізнес-рахунок отримує бонус 1% при поповненні
        bonus = amount * 0.01
        self._balance += amount + bonus
        print(f"Бізнес-поповнення (+бонус {bonus}): {self._balance}")

    def withdraw(self, amount):
        # Комісія 2% за зняття
        fee = amount * 0.02
        if (amount + fee) <= self._balance:
            self._balance -= (amount + fee)
            print(f"Знято {amount} (комісія {fee}). Залишок:
{self._balance}")

# --- ДЕМОНСТРАЦІЯ ---

accounts = [
    SavingsAccount("Олександр", 1000),
    BusinessAccount("ТОВ 'Світло'", 5000)
]

# Поліморфізм у дії: викликаємо однаковий метод для різних об'єктів
for acc in accounts:
    acc.deposit(500)
```

Коментар до прикладу

Як тут працюють принципи:

1. **Абстракція**: Ми створили клас Account, який не можна створити напряму (ви не можете мати "просто рахунок", він має бути якогось типу). Він лише визначає обов'язкові методи deposit та withdraw.
2. **Інкапсуляція**:
   - _balance (з одним підкресленням) — це сигнал іншим розробникам: "Не змінюйте це поле напряму, використовуйте методи".
   - __interest_rate (з двома підкресленнями) — Python фізично обмежує прямий доступ до цього поля ззовні класу (name mangling).
3. **Успадкування**: SavingsAccount та BusinessAccount успадковують конструктор та атрибут owner від Account, що економить нам написання коду.
4. **Поліморфізм**: Ми пройшлися циклом по списку accounts. Хоча там різні класи, ми викликали один і той самий метод .deposit(), і кожен об'єкт виконав його за своєю логікою (один просто додав гроші, інший нарахував бонус).

Завдання до лабораторної роботи 3

1. Створити клас комплексні числа. Визначити метод для поділу двох комплексних чисел.

2. Створити клас прямокутний паралелепіпед з методом для обчислення площі поверхні і об'єму.

3. Створити клас вектор. Визначити метод знаходження кута між векторами.

4. Створити клас дріб. Визначити метод для поділу двох дробів.

5. Створити клас конус з методом для обчислення площі поверхні і об'єму.

6. Створити клас дріб. Визначити метод для множення двох дробів.

7. Створити клас куля з методом для обчислення об’єму і площі поверхні.

8. Створити клас дріб. Визначити метод для складання та віднімання двох дробів.

9. Створити клас трикутник, задавши в ньому всі довжини сторін. Визначити метод для обчислення радіуса вписаного і описаного кіл.

10. Створити клас дріб. Визначити метод для скорочення дробу.

11. Створити клас трикутник, задавши в ньому всі довжини сторін. Визначити метод для обчислення його площі.

12.Створити клас прямокутник з методом для обчислення площі, периметра і центру тяжіння.

13. Створити клас комплексні числа. Визначити метод для множення двох комплексних чисел.

14. Створити клас трапеція з методом для обчислення площі та визначення можливості вписати коло в дану трапецію.

15. Створити клас дріб. Визначити метод скорочення дробу згідно алгоритму Евкліда.

16. Створити клас вектор. Визначити метод складання векторів.

17. Створити клас прямокутний паралелепіпед з методом для обчислення його діагоналі.

18. Створити клас комплексні числа. Визначити метод для поділу двох комплексних чисел.

19. Створити клас конус з методом для обчислення об'єму і площі поверхні.

20. Створити клас вектор. Визначити метод визначення скалярного добутку

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 правильно. Використано всі принципи ООП. Код чистий, оптимізований, документований, логіка позбавлена надмірності.
74–89	Добре	Основні принципи реалізовані, але є дрібні помилки в інкапсуляції або відсутні магічні методи.
60–73	Задовільно	Класи створені, успадкування працює, але логіка занадто проста або порушено принципи іменування (PEP 8).
Нижче 60	Незадовільно	Відсутнє розуміння self, методи не працюють, або код не запускається через синтаксичні помилки.

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. Креневич А.П. Python у прикладах і задачах. Частина 2. Об'єктно-орієнтоване програмування. Навчальний посібник – К.: ВПЦ "Київський Університет", 2020. – 152 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/**

Лабораторна робота № 4
Unit-тестування програмних кодів

Мета: закріпити теоретичні знання і розвинути практичні навички з розробки тестів з використанням фреймворку модульного тестування Unittest.

Теоретичні відомості

Тестування класів за допомогою модуля unittest є стандартом у Python-розробці. На відміну від тестування простих функцій, перевірка класів потребує особливої уваги до стану об'єктів, взаємодії методів та життєвого циклу тестового випадку.

1. Основні поняття Unittest

unittest — це вбудований фреймворк для модульного тестування (unit testing), побудований на принципах об'єктно-орієнтованого програмування.

- **TestCase (Тестовий випадок):** Найменша одиниця тестування. Вона перевіряє конкретний аспект роботи коду.
- **Test Suite (Тестовий набір):** Сукупність кількох тестових випадків або інших наборів тестів.
- **Test Runner (Виконавець тестів):** Компонент, який керує виконанням тестів і надає результати користувачеві.

2. Структура тестового класу

Для створення тестів необхідно створити клас, що успадковується від unittest.TestCase. Методи всередині цього класу, які починаються з префікса test_, будуть автоматично розпізнані як тести.

3. Методи перевірки (Assertions)

Для верифікації результатів використовуються спеціальні методи класу TestCase. Найпопулярніші з них:

Метод	Перевірка
assertEqual(a, b)	$a == b$
assertNotEqual(a, b)	$a \neq b$
assertTrue(x)	$bool(x)\ is\ True$
assertIsInstance(a, b)	чи є об'єкт екземпляром певного класу
assertRaises(Error)	чи виникає виключення (Exception)
assertIn(item, list)	включення до переліку

4. Життєвий цикл тесту (Fixtures)

Під час тестування класів часто потрібно створювати екземпляр об'єкта перед кожним тестом. Щоб не дублювати код, використовуються методи налаштування:

- **setUp()**: Виконується **перед кожним** тестовим методом. Тут зазвичай створюють об'єкти класу, які будуть тестуватися.
- **tearDown()**: Виконується **після кожного** тестового методу (наприклад, для закриття файлів або з'єднань з БД).
- **setUpClass()**: Виконується один раз на початку роботи всього класу тестування (потребує декоратора @classmethod).

5. Специфіка тестування об'єктів

Загальна практика тестування передбачає важливість перевірки:

1. **Стану об'єкта:** Чи правильно ініціалізуються атрибути в __init__.
2. **Поведінку методів:** Чи повертають методи очікувані значення.
3. **Зміни стану:** Як виклик одного методу впливає на значення атрибутів об'єкта.
4. **Інкапсуляцію:** Як клас реагує на некоректні вхідні дані (викидання помилок).

6. Запуск тестів

Найпростіший спосіб запустити тести всередині скрипта:

```
if __name__ == '__main__':
    unittest.main()
```

Приклад 1. Тестування логіки зі складними типами (Списки/Словники)

```
import unittest
class ShoppingCart:
    def __init__(self):
        self.items = []

    def add_item(self, item):
        self.items.append(item)

class TestShoppingCart(unittest.TestCase):
    def setUp(self):
        self.cart = ShoppingCart()

    def test_add_item(self):
        self.cart.add_item("Apple")
        self.assertIn("Apple", self.cart.items)
        self.assertEqual(len(self.cart.items), 1)
```

```
    def test_cart_is_empty_initially(self):
        self.assertEqual(len(self.cart.items), 0)

if __name__ == '__main__':
    unittest.main()
```

Приклад 2. Тестування обробки виключень (Exceptions)

```
import unittest
class Calculator:
    def divide(self, a, b):
        if b == 0:
            raise ValueError("Ділення на нуль неможливе")
        return a / b

class TestCalculator(unittest.TestCase):
    def test_divide_by_zero(self):
        calc = Calculator()
        # Перевіряємо, чи виникає ValueError при діленні на 0
        with self.assertRaises(ValueError):
            calc.divide(10, 0)

if __name__ == '__main__':
    unittest.main()
```

Приклад 3. Тест класу дріб

```
import unittest
from lab_OOP3 import Fraction

class TestFraction(unittest.TestCase):

    def setUp(self):
        self.x = Fraction(a=1, b=5)
        self.y = Fraction(a=1, b=2)
        self.z = Fraction(a=5, b=15)

    def test1(self):
        z_reduction = Fraction(a=1, b=3)
        self.assertEqual(self.z.reduction(), z_reduction)

     def test2(self):
        '''add()'''
        x_plus_y=Fraction(a=7,b=10)
        self.assertEqual(self.x+self.y,x_plus_y)

    def test3(self):
        '''pow'''
```

```
            x_pow = Fraction(a=1, b=25)
            self.assertEqual(self.x.pow(2), x_pow)

        def test4(self):
            '''lt'''
            self.assertTrue(self.x < self.y)

    if name=='main':
        unittest.main()
```

Завдання до лабораторної роботи 4

1. Ознайомитись з теоретичним матеріалом та вивчити документацію до бібліотеки
2. Ознайомитись з прикладом
3. Написати код тест-скрипту для перевірки можливих випадків для розроблених у лабораторній роботі 3 класів

Варіанти завдань

1. Створити клас комплексні числа. Визначити метод для поділу двох комплексних чисел.

2. Створити клас прямокутний паралелепіпед з методом для обчислення площі поверхні і об'єму.

3. Створити клас вектор. Визначити метод знаходження кута між векторами.

4. Створити клас дріб. Визначити метод для поділу двох дробів.

5. Створити клас конус з методом для обчислення площі поверхні і об'єму.

6. Створити клас дріб. Визначити метод для множення двох дробів.

7. Створити клас куля з методом для обчислення об’єму і площі поверхні.

8. Створити клас дріб. Визначити метод для складання та віднімання двох дробів.

9. Створити клас трикутник, задавши в ньому всі довжини сторін. Визначити метод для обчислення радіуса вписаного і описаного кіл.

10. Створити клас дріб. Визначити метод для скорочення дробу.

11. Створити клас трикутник, задавши в ньому всі довжини сторін. Визначити метод для обчислення його площі.

12.Створити клас прямокутник з методом для обчислення площі, периметра і центру тяжіння.

13. Створити клас комплексні числа. Визначити метод для множення двох комплексних чисел.

14. Створити клас трапеція з методом для обчислення площі та визначення можливості вписати коло в дану трапецію.

15. Створити клас дріб. Визначити метод скорочення дробу згідно алгоритму Евкліда.

16. Створити клас вектор. Визначити метод складання векторів.

17. Створити клас прямокутний паралелепіпед з методом для обчислення його діагоналі.

18. Створити клас комплексні числа. Визначити метод для поділу двох комплексних чисел.

19. Створити клас конус з методом для обчислення об’єму і площі поверхні.

20. Створити клас вектор. Визначити метод визначення скалярного добутку

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	Студент продемонстрував повне розуміння життєвого циклу тесту. Використано setUp, перевірено всі можливі сценарії (включаючи помилки), код чистий.
74–89	Добре	Основна логіка протестована правильно, тести проходять успішно, але не враховано граничні випадки або ігнорується механізм setUp
60–73	Задовільно	Написано мінімальну кількість тестів (лише на успішне виконання), є зауваження до структури коду або іменування методів.
Нижче 60	Незадовільно	Тести не працюють, відсутні перевірки (assert), або код не відповідає заданому класу.

Контрольні питання:

1. Чим клас відрізняється від об'єкта (екземпляра)? (Поясніть на прикладі креслення та готового виробу).
2. Яку роль відіграє метод __init__? Чи можна створити клас без цього методу, і що тоді станеться при створенні об'єкта?
3. Навіщо потрібен параметр self у методах класу? Що станеться, якщо забути вказати його в описі методу, але спробувати викликати цей метод через об'єкт?
4. У чому різниця між атрибутом класу та атрибутом екземпляра? Де зберігається значення, якщо ми змінюємо його через self.name, а де — якщо через ClassName.name?
5. Як реалізувати інкапсуляцію в Python? Яка різниця в поведінці атрибутів _title, __title та title при спробі доступу до них ззовні класу?

6. Що таке super() і в яких випадках без цієї функції не обійтися? Як викликати метод батьківського класу, якщо дочірній клас його перевстановив (overriding)?

7. Як працює поліморфізм у Python? Чому ми можемо викликати метод .draw() у об'єктів різних класів (Квадрат, Коло, Трикутник) в одному циклі?

Перелік рекомендованих джерел

1. https://docs.python.org/3/library/unittest.html
2. Креневич А.П. Python у прикладах і задачах. Частина 2. Об’єктно-орієнтоване програмування. Навчальний посібник – К.: ВПЦ "Київський Університет", 2020. – 152 с.

Лабораторна робота № 5
Графічна бібліотека GUI

Мета: закріпити теоретичні знання і розвинути практичні навички із використання принципів і методів графічної бібіліотеки GUI.

Теоретичні відомості

Tkinter — це стандартна бібліотека для мови Python, призначена для створення графічних інтерфейсів користувача (GUI). Вона є обгорткою над інструментарієм **Tcl/Tk** і входить у стандартний пакет інсталяції Python, що робить її найпопулярнішим вибором для швидкої розробки десктопних програм.

1. Основна концепція: Головний цикл (Mainloop)

Програма на Tkinter працює за принципом **подійно-орієнтованого програмування**. Після запуску програма входить у нескінченний цикл mainloop().

- Він чекає на події (натискання клавіш, рух миші).
- Коли подія стається, викликається відповідний обробник (функція).
- Якщо цикл зупиняється, вікно закривається.

2. Ієрархія віджетів

Все, що ви бачите у вікні (кнопки, текстові поля), називається **віджетами** (widgets).

- **Root (Корінь):** Головне вікно, яке створюється через root = tk.Tk().
- **Parent/Child (Батько/Дитина):** Кожен віджет створюється всередині іншого (контейнера). Якщо видалити батьківський віджет, зникнуть і всі дочірні.

3. Основні класи віджетів

Часто вживаними елемантами форм є наступні:

Віджет	Опис
Label	Відображає статичний текст або зображення.
Button	Виконує функцію (команду) при натисканні.
Entry	Однорядкове поле для введення тексту користувачем.
Text	Багаторядкове поле для редагування великих обсягів тексту.
Checkbutton	Прапорець для вибору формату "так/ні" або декількох варіантів.
Radiobutton	Перемикач для вибору одного варіанту з групи.
Combobox	Випадаючий список (знаходиться у модулі tkinter.ttk).
Message	Аналог Label, але з автоматичним переносом тексту на нові рядки.

4. Менеджери геометрії (Розміщення)

Створення віджета не означає, що він з'явиться на екрані. Його потрібно розмістити одним із трьох методів:

1. **.pack()**: Розміщує віджети один за одним (зверху вниз або зліва направо). Найпростіший, але найменш гнучкий.
2. **.grid()**: Розміщує віджети у вигляді таблиці (сітки) з рядками (row) та колонками (column). Найпотужніший метод для створення форм.
3. **.place()**: Дозволяє вказувати точні координати в пікселях. Використовується рідко, бо не адаптується до зміни розмірів вікна.

5. Змінні Tkinter (Control Variables)

Для зв'язку між віджетами (наприклад, Radiobutton) та даними у Python використовуються спеціальні класи змінних:

- StringVar() — для рядків.
- IntVar() — для цілих чисел.
- BooleanVar() — для логічних значень (True/False).

Вони дозволяють автоматично оновлювати стан віджетів або зчитувати їх за допомогою методів .get() та .set().

6. Модуль ttk (Themed Tkinter)

Модуль tkinter.ttk надає доступ до покращених віджетів, які виглядають більш сучасно та нативно для кожної операційної системи. Наприклад, ttk.Button виглядатиме інакше в Windows та macOS, підлаштовуючись під стиль системи.

Важливо: Для взаємодії з користувачем (діалогові вікна) часто використовується підмодуль messagebox, який потрібно імпортувати окремо: from tkinter import messagebox.

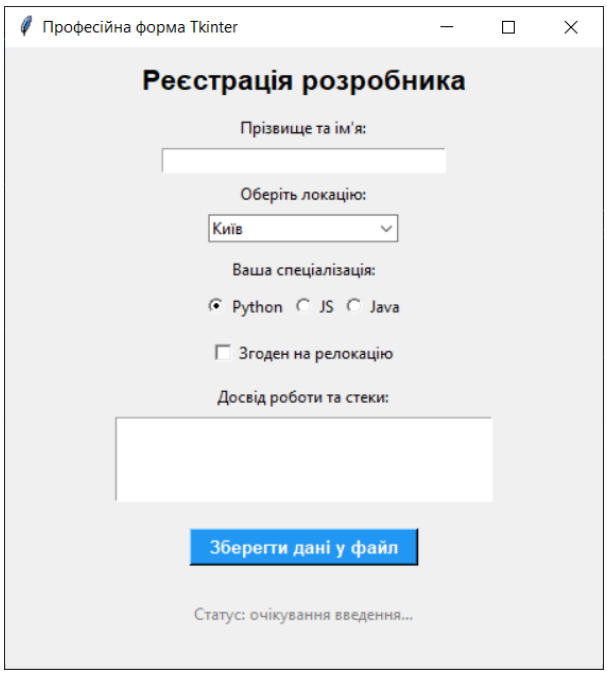
Приклад 1. “Форма реєстрації з валідацією”

```
import tkinter as tk
from tkinter import ttk, messagebox


def save_to_file():
    # 1. Збір даних
    name = name_entry.get().strip()
    city = city_box.get()
    language = lang_var.get()
    experience = "Так" if check_var.get() else "Ні"
    bio = bio_text.get("1.0", tk.END).strip()

    # 2. Валідація (перевірка)
    if not name:
        messagebox.showwarning("Помилка", "Будь ласка, введіть ім'я!")
        return
```

```
    if not bio:
        messagebox.showwarning("Помилка", "Поле 'Про себе' не може бути
порожнім!")
        return

    # 3. Формування рядка для запису
    record = f"Ім'я: {name} | Місто: {city} | Мова: {language} | Досвід:
{experience}\nBio: {bio}\n{'-' * 30}\n"

    # 4. Запис у файл
    try:
        with open("applicants.txt", "a", encoding="utf-8") as file:
            file.write(record)

        # Виведення результату в елемент Message
        summary_msg.config(text=f"Успішно збережено!\nКористувач:
{name}")

        # Очищення полів після збереження
        name_entry.delete(0, tk.END)
        bio_text.delete("1.0", tk.END)
        messagebox.showinfo("Успіх", "Дані збережено у файл
applicants.txt")

    except Exception as e:
        messagebox.showerror("Критична помилка", f"Не вдалося зберегти
файл: {e}")


# Налаштування головного вікна
root = tk.Tk()
root.title("Професійна форма Tkinter")
root.geometry("450x680")

# Заголовок
tk.Label(root, text="Реєстрація розробника", font=("Helvetica", 16,
"bold")).pack(pady=10)

# Поле введення імені
tk.Label(root, text="Прізвище та ім'я:").pack()
name_entry = tk.Entry(root, width=35)
name_entry.pack(pady=5)

# Випадаючий список міст
tk.Label(root, text="Оберіть локацію:").pack()
city_box = ttk.Combobox(root, values=["Київ", "Львів", "Одеса", "Івано-
Франківськ", "Дніпро"], state="readonly")
city_box.current(0)
city_box.pack(pady=5)
```

```
# Вибір мови (Radio)
tk.Label(root, text="Ваша спеціалізація:").pack(pady=5)
lang_var = tk.StringVar(value="Python")
frame_radio = tk.Frame(root)  # Групуємо радіокнопки для охайності
frame_radio.pack()
tk.Radiobutton(frame_radio, text="Python", variable=lang_var,
value="Python").pack(side=tk.LEFT)
tk.Radiobutton(frame_radio, text="JS", variable=lang_var,
value="JavaScript").pack(side=tk.LEFT)
tk.Radiobutton(frame_radio, text="Java", variable=lang_var,
value="Java").pack(side=tk.LEFT)

# Досвід (Check)
check_var = tk.BooleanVar()
tk.Checkbutton(root, text="Згоден на релокацію",
variable=check_var).pack(pady=10)

# Текстова область
tk.Label(root, text="Досвід роботи та стеки:").pack()
bio_text = tk.Text(root, height=4, width=40, font=("Consolas", 10))
bio_text.pack(pady=5)

# Кнопка збереження
save_btn = tk.Button(root, text="Зберегти дані у файл",
command=save_to_file,
                     bg="#2196F3", fg="white", font=("Arial", 10,
"bold"), padx=10)
save_btn.pack(pady=15)

# Поле для статусу (Message)
summary_msg = tk.Message(root, text="Статус: очікування введення...",
width=350, fg="gray")
summary_msg.pack(pady=10)

root.mainloop()
```

Результат виконання:



Коментарі до наведеного прикладу:

Валідація: Програма перевіряє, чи заповнені текстові поля. Якщо ім'я порожнє, з'явиться вікно попередження (messagebox.showwarning), а виконання коду зупиниться завдяки return.

Робота з файлами: Використовується конструкція with open(...), яка автоматично створює файл applicants.txt (якщо його немає) і додає туди новий запис у кінець файлу (mode="a").

Обробка помилок: Блок try...except захищає програму від вильоту, якщо виникнуть проблеми з правами доступу до диску.

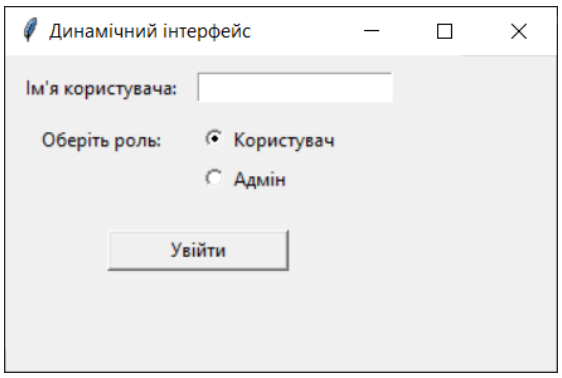
Покращений інтерфейс: Використано tk.Frame для горизонтального розміщення радіокнопок та використано state="readonly" для Combobox, щоб користувач не міг вписувати туди випадкові слова.

Приклад 2. “Динамічний інтерфейс”

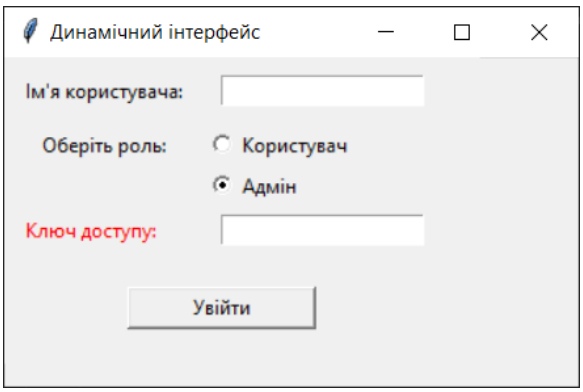
```
import tkinter as tk
from tkinter import ttk, messagebox


def toggle_admin_fields(*args):
    """Функція, що показує або ховає поля залежно від ролі"""
    if role_var.get() == "Admin":
        # Показуємо мітку та поле
        secret_label.grid(row=5, column=0, sticky="w", padx=10)
        secret_entry.grid(row=5, column=1, pady=5, padx=10)
    else:
```

```
        # Ховаємо елементи з сітки
        secret_label.grid_forget()
        secret_entry.grid_forget()


def submit():
    name = name_entry.get()
    role = role_var.get()

    if role == "Admin":
        key = secret_entry.get()
        if key != "1234":
            messagebox.showerror("Помилка", "Невірний ключ
адміністратора!")
            return

    messagebox.showinfo("Успіх", f"Вітаємо, {name}! Вхід виконано як
{role}.")


root = tk.Tk()
root.title("Динамічний інтерфейс")
root.geometry("400x350")

# Змінна для відстеження ролі
role_var = tk.StringVar(value="User")
# trace_add дозволяє викликати функцію автоматично при зміні значення
role_var.trace_add("write", toggle_admin_fields)

# Основні поля
tk.Label(root, text="Ім'я користувача:").grid(row=0, column=0, padx=10,
pady=10)
name_entry = tk.Entry(root)
name_entry.grid(row=0, column=1)

tk.Label(root, text="Оберіть роль:").grid(row=1, column=0, padx=10)
tk.Radiobutton(root, text="Користувач", variable=role_var,
value="User").grid(row=1, column=1, sticky="w")
tk.Radiobutton(root, text="Адмін", variable=role_var,
value="Admin").grid(row=2, column=1, sticky="w")

# Створюємо "секретні" поля заздалегідь, але НЕ пакуємо їх одразу
secret_label = tk.Label(root, text="Ключ доступу:", fg="red")
secret_entry = tk.Entry(root, show="*")  # show="*" ховає символи
(пароль)

# Кнопка підтвердження
tk.Button(root, text="Увійти", command=submit, width=15).grid(row=10,
column=0, columnspan=2, pady=20)

root.mainloop()
```

Результат виконання



Початковий інтерфейс



Динамічно оновлений при виборі «Адмін»

Завдання до лабораторної роботи 5

Використовуючи код і дані з Лабораторної роботи 1 «Обробка текстової інформації. Робота з файлами у мові Python» реалізувати:

- Графічний інтерфейс для перегляду/редагування даних (8-12 полів з загального набору)
- Продемонструвати навички у використанні елементів button, checkbutton, label, message, radoibutton, text, combobox;
- Реалізувати додавання нових записів у файли. З метою збереження структури файлу всі інші поля (що не входять до форми введення даних) повинні записуватись пустими;
- Передбачити контроль 1-2 полів в якості обов'язкових до заповнення;
- Довідкова інформація, що розраховувалась у п.3 повинна відображатись у зручному для користувача вигляді на основній формі, або виводитись в якості допоміжної форми.

3. Додати можливість перегляду статистичної інформації у графічному вигляді (діаграма та графік) по натисканню на окрему кнопку.

Деталізація за варіантами:

- якщо робота ведеться з файлом з курсами валют та цінних металів – динаміка зміни курсу Форинта за будь-який тиждень (на вибір користувача)
- якщо робота ведеться з файлом з показниками погоди в області міста Дарем – динаміка зміни мінімальної температури за роками що містяться у наборі даних

- якщо робота ведеться з файлом про вступ до ЗВО – динаміка зміни кількості заяв на вступ до ЗВО за днями вступної кампанії (для розрахунку використовувати тільки дату створення заяви)

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 бездоганно, має складний інтерфейс та обробку винятків.
75 - 89	Добре	Всі віджети працюють, форма охайна, але є дрібні зауваження до логіки чи дизайну.
60 - 74	Задовільно	Основна мета досягнута, але відсутня валідація або динамічні елементи.
0 - 59	Незадовільно	Програма має критичні помилки, інтерфейс нефункціональний або неповний.

Контрольні питання:

1. Що станеться з графічним інтерфейсом, якщо у коді не викликати метод root.mainloop()? Яка його головна функція?
2. У яких випадках доцільніше використовувати менеджер сітки (grid), а в яких — пакувальник (pack)? Чи можна використовувати обидва методи для віджетів всередині одного контейнера (наприклад, одного Frame)?
3. Який із цих віджетів краще підходить для введення пароля, а який — для написання розлогого відгуку? Чому?
4. Навіщо потрібні класи StringVar, IntVar та BooleanVar? Чим вони кращі за звичайні змінні Python при роботі з віджетами?
5. Як програма розуміє, які саме радіокнопки належать до однієї групи (щоб при виборі однієї інша ставала неактивною)?
6. Як отримати текст, який користувач ввів у Entry, та як отримати весь текст із віджета Text (враховуючи індекси)?
7. Чим віджети з модуля tkinter.ttk відрізняються від стандартних віджетів tkinter? Наведіть приклад віджета, який є лише в ttk.
8. Чому важливо використовувати messagebox при обробці даних із форми? Які типи вікон повідомлень ви знаєте?
9. За допомогою якого методу можна тимчасово прибрати віджет з екрана, не видаляючи його з пам'яті програми?

10. Що таке параметр command у кнопці? Як передати функцію в цей параметр, щоб вона не виконалася миттєво при запуску програми (без натискання)?

Перелік рекомендованих джерел

1. https://docs.python.org/3/library/tkinter.html
2. Alan D. Moore. Python GUI Programming with Tkinter. Published by Packt Publishing Limited, 2023.

Лабораторна робота № 6
Символьні обчислення в Python

Мета: закріпити теоретичні знання і розвинути практичні навички з виконання символьних обчислень використовуючи бібліотеку SymPy.

Теоретичні відомості

SymPy — це бібліотека мови Python для символьної математики. Її мета - стати повноцінною системою комп'ютерної алгебри (CAS), залишаючи код максимально простим та зрозумілим.

1. Основні поняття та ініціалізація

Головна відмінність SymPy від стандартного Python - використання **символьних об'єктів**.

Наприклад, $\sqrt{8}$ у Python поверне 2.828..., а в SymPy — $2\sqrt{2}$.

Для початку роботи необхідно визначити змінні як символи:

Приклад 1

```
import sympy as sp
x, y = sp.symbols('x y')
```

2. Алгебраїчні перетворення

SymPy дозволяє маніпулювати виразами за допомогою вбудованих функцій:

- **expand()**: розкриття дужок та спрощення добутку.
- **simplify()**: універсальна функція для зведення виразу до найпростішого вигляду.
- **factor()**: розкладання багаточлена на множники.
- **subs()**: заміна символу на число або інший вираз.

3. Математичний аналіз

Бібліотека містить потужний інструментарій для розв'язання задач аналізу:

- **Границя функції (limit)**:

$$\lim_{x \to 0} \left(\frac{\sin(x)}{x} \right) = 1$$

- **Диференціювання (diff)**: обчислення похідних будь-якого порядку.
- **Інтегрування (integrate)**: знаходження визначених та невизначених інтегралів.

Невизначений: `sp.integrate(f, x)`

Визначений: `sp.integrate(f, (x, a, b))`

4. Розв’язання рівнянь та систем

Для пошуку коренів використовується функція **solve()** (для алгебраїчних рівнянь) або **dsolve()** (для диференціальних рівнянь). За замовчуванням рівняння прирівнюється до нуля:

- sp.solve(x**2 - 4, x) знайде $x = \pm 2$.

5. Робота з матрицями

Клас sp.Matrix дозволяє виконувати операції лінійної алгебри в символьному вигляді:

- Обчислення детермінанта (det()).
- Пошук оберненої матриці (inv()).
- Знаходження власних чисел та векторів.

Характеристика	**Чисельні (NumPy)**	**Символьні (SymPy)**
Результат	Наближене число (float)	Точний вираз (формула)
Похибка	Можлива похибка округлення	Похибка відсутня
Швидкість	Дуже висока (оптимізовано)	Нижча (через складність структур)
Типові задачі	Data Science, симуляції	Аналітичне доведення, виведення формул

Приклад 2. Алгебра та спрощення виразів

Дано математичний вираз:

$$A = (x + 2)^2 - (x - 2)^2$$

```
import sympy as sp
x = sp.symbols('x')
expr = (x + 2)**2 - (x - 2)**2

expanded = sp.expand(expr)
print(f"Розгорнутий вираз: {expanded}")

value = expanded.subs(x, 10)
print(f"Результат при x=10: {value}")
```

Приклад 3. Математичний аналіз (Похідні та інтеграли)

Для функції $f(x) = x^3\cos(x)$

1. Обчисліть першу похідну $f'(x)$.
2. Обчисліть другу похідну $f''(x)$.
3. Знайдіть неозначений інтеграл $\int f(x)\,dx$
4. Обчисліть означений інтеграл на відрізку $[0, \pi]$.

```
import sympy as sp
x = sp.symbols('x')
f = x**3 * sp.cos(x)
```

```
diff_1 = sp.diff(f, x)
integral_indef = sp.integrate(f, x)
integral_def = sp.integrate(f, (x, 0, sp.pi))

print(f"Похідна: {diff_1}")
print(f"Означений інтеграл: {integral_def}")
```

Приклад 4. Приклад розв’язку завдання у Jupiter notebook

Завдання:

1. Проаналізувати функцію $f(x) = xe^{-2x}$
2. Розв'язати задачу Коші: $y' + y = x, y(0) = 1$

Тема: Символьні обчислення SymPy

Варіант: №0 (Демонстраційний)

Завдання:

1. Проаналізувати функцію $f(x) = x \cdot e^{-x^2}$.
2. Розв'язати задачу Коші: $y' + y = x, \quad y(0) = 1$.

[ ]

```
import sympy as sp
from sympy.plotting import plot

# Налаштування гарного виводу формул (LaTeX)
sp.init_printing(use_latex='mathjax')

# Оголошення символів
x = sp.Symbol('x')
y = sp.Function('y')

print("Середовище готове до роботи.")
```

```
Середовище готове до роботи.
```

2. Аналіз функції $f(x)

Визначення функції:

```
f = x * sp.exp(-x**2)
display(f)
```

$$xe^{-x^2}$$

Обчислення границь:Обчислимо границю при $x \to \infty$:

```
lim_inf = sp.limit(f, x, sp.oo)
print("Границя при x -> +oo:")
display(lim_inf)
```

```
Границя при x -> +oo:
```
$$0$$

Знаходження похідних: Знайдемо першу похідну $f'(x)$ для пошуку критичних точок:

```
f_diff = sp.diff(f, x)
print("Перша похідна:")
display(f_diff)
```

```
Перша похідна:
```
$$-2x^2e^{-x^2} + e^{-x^2}$$

Знаходження інтеграла: Обчислимо невизначений інтеграл $\int f(x)\,dx$:

```
f_int = sp.integrate(f, x)
print("Невизначений інтеграл:")
display(f_int)
```

```
Невизначений інтеграл:
```
$$-\frac{e^{-x^2}}{2}$$

3. Розв'язання задачі Коші Рівняння: $y'(x) + y(x) = x$, умова: $y(0) = 1$.

```
# Запис рівняння
equation = sp.Eq(y(x).diff(x) + y(x), x)
display(equation)

# Розв'язання з початковими умовами (ics)
solution = sp.dsolve(equation, y(x), ics={y(0): 1})

print("Розв'язок задачі Коші:")
display(solution)
```

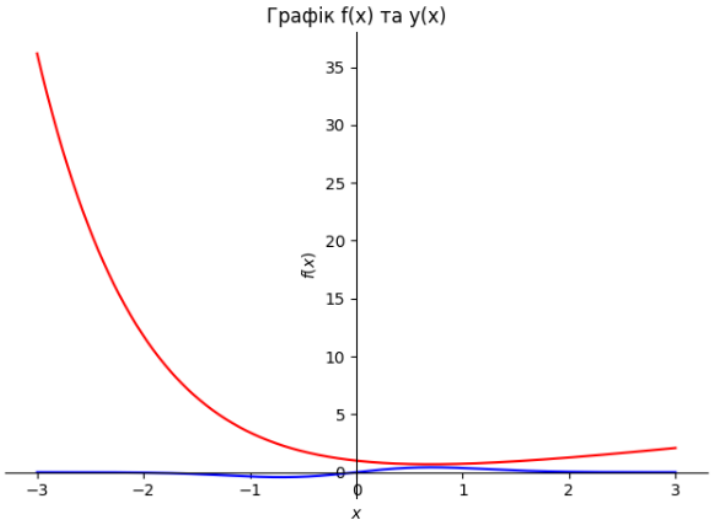
$$y(x) + \frac{d}{dx}y(x) = x$$

```
Розв'язок задачі Коші:
```
$$y(x) = x - 1 + 2e^{-x}$$

4. Візуалізація результатів

Побудуємо графік нашої функції $f(x)$ та знайденого розв'язку $y(x)$.

```
# Графік функції f(x)
p1 = plot(f, (x, -3, 3), show=False, line_color='blue', title="Графік f(x) та y(x)")

# Графік розв'язку y(x) (витягуємо праву частину рівняння через .rhs)
p2 = plot(solution.rhs, (x, -3, 3), show=False, line_color='red')

p1.extend(p2)
p1.show()
```



5. Перевірка розв'язку

Підставимо знайдений розв'язок у початкове диференціальне рівняння, щоб переконатися у правильності.

```
check = equation.subs(y(x), solution.rhs).doit()
print("Перевірка (має бути True):")
display(sp.simplify(check))
```

```
Перевірка (має бути True):
True
```

Завдання до лабораторної роботи 6

Кожне завдання має включати наступні дії:

- Оголосити символьні змінні та функцію.
- Знайти границі функції $f(x)$ у характерних точках $(0, \pm\infty)$
- Обчислити першу та другу похідні $f(x)$.
- Знайти невизначений інтеграл $\int f(x)dx$.
- Розв'язати диференціальне рівняння з початковими умовами (задачу Коші).
- Побудувати графіки функції та знайденого розв'язку ДР на одній площині.
- Виконати перевірку отриманого розв'язку ДР шляхом підстановки.

№	Функція f(x) для аналізу	Диференціальне рівняння	Початкові умови (Задача Коші)
1	$f(x) = \sin(x)/x$	$y' + 2y = e^{3x}$	$y(0) = 1$
2	$f(x) = x^2 e^{-x}$	$y'' - y = 0$	$y(0) = 1, y'(0) = 0$

3	$f(x) = ln(x^2 + 1)$	$y' = x/y + x$	$y(1) = 2$
4	$f(x) = x2 - 4x + 1$	$y'' + 4y = sin(2x)$	$y(0) = 0, y'(0) = 1$
5	$f(x) = cos2(x) - sin(x)$	$xy' + y = x^2$	$y(1) = 0.5$
6	$f(x) = ex \cdot cos(x)$	$y'' - 5y' + 6y = 0$	$y(0) = 2, y'(0) = 3$
7	$f(x) = xx + 1 - 1$	$y' + y \cdot tan(x) = cos(x)$	$y(0) = 1$
8	$f(x) = x \cdot ln(x)$	$y'' + y' = ex$	$y(0) = 0, y'(0) = 1$
9	$f(x) = tan(x)/x$	$y' = 2x(y^2 + 1)$	$y(0) = 0$
10	$f(x) = 1/(1 + e^{-x})$	$y'' - 4y' + 4y = 0$	$y(0) = 1, y'(0) = 2$
11	$f(x) = x^3 - 3x^2 + 2$	$y' + \frac{2}{x}y = x^3$	$y(1) = 0$
12	$f(x) = arcsin(x)$	$y'' + 9y = e^x$	$y(0) = 0, y'(0) = 0$
13	$f(x) = (e^x - e^{-x})/2$	$y' = y \cdot cos(x)$	$y(0) = e$
14	$f(x) = \sqrt{x^2 + 9}$	$y'' + 2y' + y = x$	$y(0) = 1, y'(0) = 0$
15	$f(x) = ln(x)/x$	$x^2y' + y = 1$	$y(1) = 2$
16	$f(x) = sin(x) \cdot ln(sin(x))$	$y'' + y = cot(x)$	$y(\pi/2) = 1, y'(\pi/2) = 0$
17	$f(x) = (x^2 - 1)/(x^2 + 1)$	$y' - y = x \cdot e^x$	$y(0) = 1$
18	$f(x) = 2^x \cdot x^2$	$y'' - 3y' + 2y = cos(x)$	$y(0) = 0, y'(0) = 1$
19	$f(x) = 1/cos(x)$	$y' = (1 + y^2)/(1 + x^2)$	$y(0) = 1$
20	$f(x) = atan(x)$	$y'' + 4y' + 5y = 0$	$y(0) = 1, y'(0) = -2$

Критерії оцінювання лабораторної роботи 6

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	• Студент повністю виконав усі завдання варіанта. • Код структурований, супроводжується поясненнями в Markdown. • Графіки чіткі, з легендою, побудовані у правильних діапазонах. • Зроблено перевірку диференціального рівняння, висновки логічні.
75 - 89	Добре	• Завдання виконані, але є незначні помилки (наприклад, не підписані осі на графіках або пропущено один тип границі). • Код працює, але оформлення мінімальне.

		• Задача Коші розв'язана вірно.
60 - 74	Задовільно	• Виконано основні обчислення (похідні, інтеграли). • Є труднощі з розв'язанням задачі Коші (наприклад, знайдено лише загальний розв'язок без врахування початкових умов). • Відсутня візуалізація або перевірка результатів.
0 - 59	Незадовільно	• Програма містить критичні помилки, які заважають виконанню (SyntaxError). • Виконано менше половини пунктів завдання. • Відсутнє розуміння символьних змінних (спроба рахувати символьні вирази через бібліотеку math).

Контрольні питанння:

1. Яка функція SymPy використовується для оголошення символьних змінних?
2. Чим відрізняється робота функцій math.sin(x) та sympy.sin(x)?
3. Як у SymPy передати початкові умови для диференціального рівняння?
4. Яка різниця між методом .subs() та функцією simplify()?
5. Як отримати праву частину (вираз) з отриманого розв'язку ДР для подальшої побудови графіка?

Перелік рекомендованих джерел

1. Amit Saha. Doing Math with Python: Use Programming to Explore Algebra, Statistics, Calculus, and More! Printed in USA: San Francisco, 2015. – 264 p.

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. Моделювання та реінжиніринг бізнес-процесів: підручн. / С.В. Козир, В.В. Слєсарєв, С.А. Ус, Т.В. Хом’як; М-во освіти і науки України; Нац. техн. Ун-т «Дніпровська політехніка». Дніпро: НТУ «ДП», 2022. 163 с. Режим доступу: https://ir.nmu.org.ua/server/api/core/bitstreams/bdd1cdf1-cce2-429e-8e23-b9949e90b1fb/content
7. 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.
8. Положення про порядок видання в світ інформаційно-методичного забезпечення освітнього процесу в Національному технічному університеті «Дніпровська політехніка» / М-во освіти і науки України, Нац. техн. ун-т «Дніпровська політехніка». Дніпро : НТУ «ДП», 2024, 31 с.
9. 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. https://ir.nmu.org.ua/handle/123456789/173373
10. Руденко В., Жугастров О. Інформатика. Основи алгоритмізації та програмування мовою Python. Харків: Ранок, 2019. – 192 с.
11. Alan D. Moore. Python GUI Programming with Tkinter. Published by Packt Publishing Limited, 2023.
12. Amit Saha. Doing Math with Python: Use Programming to Explore Algebra, Statistics, Calculus, and More! Printed in USA: San Francisco, 2015. – 264 p.

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

У 2 частинах

Частина 2

Видано в авторській редакції.

Електронний ресурс.
Підписано до видання 10.02.2026. Авт. арк. 1,52.

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