

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

МЕТОДИЧНІ ВКАЗІВКИ

до лабораторних робіт з дисципліни «Основи програмування на Python»

для бакалаврів спеціальності 123 Комп'ютерна інженерія
денної форми навчання

Частина III

2024

Методичні вказівки до лабораторних робіт з дисципліни «Основи програмування на Python» для бакалаврів спеціальності 123 Комп'ютерна інженерія, денної форми навчання. Частина III / Укл.: А.В. Тіменко, Н.А. Куликовська – Запоріжжя: НУ «Запорізька політехніка», 2024. – 36 с.

Укладач: А.В. Тіменко, старший викладач
Н.А. Куликовська, старший викладач

Рецензент: М.Ю. Тягунова, к.т.н., доцент

Відповідальний
за випуск: А.В. Тіменко, старший викладач

Затверджено:
На засіданні кафедри
«Комп'ютерні системи та мережі»
Протокол № 7
від 22 березня 2024р.

Рекомендовано до видання
НМК факультету КНТ
Протокол № 8
від 27 березня 2024 р.

ЗМІСТ

1	ЛАБОРАТОРНА РОБОТА №6 ОПАНУВАННЯ СТРУКТУРИ КОДУ	4
1.1	Теоретичні відомості	4
1.2	Хід роботи.....	5
1.3	Зміст звіту	12
1.4	Контрольні питання	12
2	ЛАБОРАТОРНА РОБОТА №7 ОПРАЦЮВАННЯ ФАЙЛІВ	14
2.1	Теоретичні відомості	14
2.2	Хід роботи.....	15
2.3	Зміст звіту	17
2.4	Контрольні запитання.....	18
3	ЛАБОРАТОРНА РОБОТА №8 ВИКОРИСТАННЯ ФУНКЦІЙ ДОСТУПУ ДО ОПЕРАЦІЙНОЇ СИСТЕМИ	19
3.1	Теоретичні відомості	19
3.2	Завдання до виконання	24
3.3	Зміст звіту	24
3.4	Контрольні запитання.....	25
	ЛІТЕРАТУРА.....	26
	ДОДАТОК А Стандартні функції Python.....	27
	ДОДАТОК Б Типові помилки	29
	ДОДАТОК В Пріоритет операторів.....	30
	ДОДАТОК Г Модулі Python.....	31
	ДОДАТОК Е Методи	33

1 ЛАБОРАТОРНА РОБОТА №6 ОПАНУВАННЯ СТРУКТУРИ КОДУ

Мета роботи: отримати навички програмування на мові Python 3 й опанувати структуру коду.

1.1 Теоретичні відомості

Алгоритми в Python – це набір інструкцій, написаних мовою програмування Python, які визначають порядок виконання операцій для розв’язання певного завдання. Алгоритми можуть бути різними за складністю та ефективністю, і вибір правильного алгоритму може суттєво вплинути на продуктивність програми та структуру коду.

Алгоритми, у яких використовується тільки структура «Слідування», називаються лінійними.

Алгоритми, в основі яких лежить структура «Розгалуження», називаються алгоритмами з розгалуженнями.

Алгоритми, в основі яких лежить структура «Повторення», називають циклічними.

На практиці алгоритми розв’язування складних задач містять у собі базові структури всіх трьох типів.

Лінійні алгоритми — це прості алгоритми, які виконуються послідовно, крок за кроком, без використання розгалужень або циклів. Вони розв’язують задачі, де послідовність виконання операцій є прямолінійною і не залежить від зовнішніх умов або змін даних.

Приклади задач, які часто розв’язуються за допомогою лінійних алгоритмів: обчислення простих математичних операцій, пошук максимального/мінімального елемента у списку, знаходження суми елементів списку та інші прості операції, які виконуються послідовно.

Для формування структури коду треба розглянути наступні етапи.

1. Визначити основні операції. Перед вибором структури даних, визначити, які операції будуть виконуватися. Наприклад, якщо потрібно часто виконувати пошук елементів за їхніми ключами, словник (dictionary) може бути відповідним вибором. Якщо необхідно зберігати впорядковані елементи з можливістю повторення, то список (list) або кортеж (tuple) може бути кращим.

2. Враховати швидкість доступу. Деякі структури даних забезпечують швидкий доступ до елементів, наприклад, словники та множини. У той час як інші, такі як списки, можуть мати повільний доступ, особливо під час пошуку елементів.

3. Враховувати унікальність елементів. Якщо потрібно зберігати унікальні елементи, то множина (set) може бути підходящим вибором, оскільки вона автоматично видаляє дублікати. Якщо унікальність не є проблемою, тоді можна використовувати списки або інші структури.

4. Приділити увагу операціям вставки та видалення. Якщо завдання вимагає частих вставок і видалень елементів, то вибрати структуру даних, що забезпечує ефективні операції додавання та видалення елементів. Списки зазвичай забезпечують швидкі операції додавання та видалення в кінець, тоді як множини пропонують ефективні методи додавання та видалення елементів.

5. Розглянути обсяг даних. Якщо завдання включає великий обсяг даних, то звернути увагу на займану пам'ять і швидкість роботи структури даних. Деякі структури можуть споживати більше пам'яті, ніж інші, і забезпечувати повільні операції для великого обсягу даних.

6. Врахувати необхідність сортування. Якщо потрібно підтримувати елементи в упорядкованому вигляді, виберіть структуру даних, яка забезпечує ефективні методи сортування. Списки мають вбудовані методи сортування, тоді як словники та множини зазвичай не гарантують порядок елементів.

7. Врахувати, що вибір структури даних може залежати від конкретного контексту та специфіки завдання.

1.2Хід роботи

Завдання 1. Виконайте в інтерактивному інтерпретаторі тренувальні вправи (завдання не обов'язкові):

1. Які результати обчислення таких виразів?

$(15 > 8)$ and $(2 == 9)$ not $(2 > 1)$ $(6 > 3)$ or $(8 == 7)$ not $((9 > 4)$ or $(3 == 4))$ (True and True) and (True == False) (not False) or (not True)

2. Напишіть код, який виводить різні повідомлення, в залежності від значення, що зберігається у змінній weather_forecast: What a beautiful day!, якщо значення змінної дорівнює sun, Take an umbrella!, якщо значення змінної дорівнює rain і The sun's just gone in! – у іншому випадку.

3. Створіть програму, яка виводить повідомлення про кількість очок, які отримав гравець у комп'ютерній грі. Створіть змінну з ім'ям `points_color` і надайте їй значення 'green', 'yellow' або 'red'. Якщо змінна `points_color` містить значення 'green', виведіть повідомлення про те, що гравець отримав 5 очок, якщо 'yellow' - 10 очок, якщо 'red', виведіть повідомлення про те, що гравець отримав 15 очок.

4. Напишіть ланцюжок **if-elif-else** для визначення віку людини. Надайте значення змінній `age`, а потім виведіть повідомлення. Якщо значення `age` менше 2 - baby, якщо значення `age` більше або дорівнює 2, але менше 4 - kid, якщо значення `age` більше або дорівнює 4, але менше 13 - child, якщо значення `age` більше або дорівнює 13, але менше 20 - teenager. якщо значення `age` більше або дорівнює 20, але менше 65 - grown-up, якщо значення `age` більше або дорівнює 65 - senior.

5. Створіть список трьох своїх улюблених фруктів і назвіть його `favorite_fruits`. Напишіть перевірку на те, чи входить фрукт у список. Якщо фрукт входить у список, виводиться повідомлення, на зразок `You really like peaches!`, у протилежному випадку - повідомлення про відсутність фрукту у списку.

6. Використайте цикл **for**, щоб вивести на екран в окремих рядках значення списку `[5, 4, 3, 2, 1, 0, 'GO!']`.

7. Використайте функцію **zip()**, щоб створити словник `movies`, який об'єднує у пари списки: `seasons = ['summer', 'autumn']` і `months = ['july', 'november']`. Виведіть вміст словника.

8. Використайте відомі вам структури коду для виведення ключів і значень словника `activity = {'business': 'manager', 'it': 'software developer', 'science': 'scientist'}` у вигляді, на зразок «**категорія: професія**».

9. Використайте включення списку, щоб створити список, який містить парні числа у діапазоні `range(12)`.

10. Використайте включення словника, щоб створити словник `squares` з ключами у вигляді цілих чисел з діапазону `range(1, 11)`. Значення словника формуються піднесенням ключів до другого степеня.

11. Використайте цикл **for** для виведення чисел від 1 до 15 включно, в один рядок і пропусками між ними.

12. Створіть список чисел від 1 до 1 000 000. Скористайтеся функціями **min()** і **max()** та переконайтеся у тому, що список дійсно

починається 1 і закінчується 1 000 000. Викличте функцію **sum()** і подивіться, наскільки швидко Python зможе підсумувати мільйон чисел.

13.Скористайтесь третім аргументом функції **range()** для створення списку непарних чисел від 1 до 25 і виведіть усі числа в окремих рядках у циклі **for**.

14.Створіть список перших 10 кубів (тобто кубів усіх цілих чисел від 1 до 10) і виведіть значення усіх кубів у циклі **for** в один рядок з пропусками.

15.Створіть список чисел у діапазоні від 3 до 60 і виведіть усі числа списку у циклі **while** в окремих рядках.

16.Визначте функцію **trees**, яка повертає список ['poplar', 'willow', 'lime']. Викличте функцію.

17.Напишіть функцію **favorite_book()**, яка отримує один параметр **title**. Функція повинна виводити повідомлення, на зразок *One of my favorite books is "The Lord of the Rings"..* Викличте функцію і переконайтеся у тому, що назва книги правильно передається як аргумент при виконанні функції.

18.Напишіть функцію **make_shirt()**, яка отримує розмір футболки і текст, який повинен бути надрукований на ній. Функція повинна виводити повідомлення з розміром і текстом. Викличте функцію з використанням позиційних аргументів. Викличте функцію вдруге з використанням іменованих аргументів. Змініть функцію **make_shirt()**, щоб футболки за замовчуванням мали розмір XL, і на них виводився текст *I love Python!*. Створіть футболку з розміром XL і текстом за замовчуванням.

19.Напишіть функцію **city_country()**, яка отримує назву міста і країну. Функція повинна повертати рядок у форматі *Kyiv, Ukraine*. Викличте функцію як мінімум для трьох пар «місто-країна».

20.Визначте виняток, який називається **MyException**. Згенеруйте його, щоб побачити, що станеться. Потім напишіть код, що дозволяє перехопити цей виняток і вивести рядок *This is my exception..*

21.При введенні числових даних часто зустрічається типова проблема: користувач вводить текст замість чисел. При спробі перетворити дані в **int** генерується виняток **TypeError**. Напишіть функцію, яка приймає два числа, шукає їх суму і виводить результат. Перехопіть виняток **TypeError**, якщо будь-яке із вхідних значень не є числом, і виведіть зручне повідомлення про помилку. Протестуйте

функцію: спочатку введіть два числа, а потім введіть текст замість одного з чисел.

Завдання 2. Напишіть програми у середовищі програмування для розв'язування таких завдань відповідно номеру варіанта:

1. Виведіть вітальне повідомлення для кожного користувача після його входу на сайт. Створіть список з кількох імен користувачів, включаючи ім'я 'Admin'. Перебираючи елементи списку, виведіть повідомлення для кожного користувача. Для користувача з ім'ям 'Admin' виведіть особливе повідомлення - наприклад: Hello Admin, I hope you're well.. У інших випадках виводиться універсальне привітання - наприклад: Hello Alex, thank you for logging in again.. Додайте команду if, яка перевірить, що список користувачів не порожній. Якщо список порожній, виведіть повідомлення: We need to find some users!. Видаліть зі списку всі імена користувачів і переконайтеся у тому, що програма виводить правильне повідомлення.

2. Визначте назву геометричної фігури за введеною кількістю її сторін. Програма повинна підтримувати фігури від 3 до 6 сторін. Якщо введена кількість сторін поза межами цього діапазону, програма повинна відображати відповідне повідомлення.

3. Порядкові числівники у англійській мові закінчуються суфіксом th (окрім 1st, 2nd і 3rd). Створіть список чисел від 1 до 9. Використайте ланцюжок if-elif-else у циклі для виведення правильного закінчення числівника для кожного числа. Програма повинна виводити числівники 1st 2nd 3rd 4th 5th 6th 7th 8th 9th, кожен у новому рядку.

4. Зчитайте ціле число введене користувачем і виведіть повідомлення про те, чи число парне або непарне.

5. Зчитайте назву місяця від користувача як рядок і виведіть кількість днів у вказаному місяці. Врахувати те, що «February» може мати 28 або 29 днів.

6. Потрібно визначити, чи є даний рік високосним. Нагадаємо, що високосними роками вважаються ті роки, порядковий номер яких або кратний 4, але при цьому не кратний 100, або кратний 400 (наприклад, 2000-й рік був високосним, а 2100-й буде невисокосним роком). Програма повинна коректно працювати на числах від 1900 до 3000 років. Виведіть Leap year. у разі, якщо рік є високосним і Ordinary year. у протилежному випадку.

7. Зчитайте зі стандартного вводу цілі числа, по одному числу у рядку, і після першого введенного нуля виведіть суму отриманих на вхід чисел.

8. Зчитайте цілі числа, які вводить користувач, по одному числу у рядку. Для кожного введенного числа виконайте перевірку: якщо число менше 10, то пропускаємо це число, якщо число більше 100, то припиняємо зчитувати числа. У інших випадках вивести це число в окремому рядку.

9. Напишіть простий калькулятор, який зчитує три рядки, які вводить користувач: перше число, друге число і операцію, після чого застосовує операцію до введених числах («перше число» «операція» «друге число») і виводить результат на екран. Підтримувані операції: +, -, /, *, mod, pow, div, де mod - це взяття залишку від ділення, pow - піднесення до степеня, div - цілочисельне ділення. У випадку ділення на 0, необхідно використати обробник винятку, який має вивести повідомлення Division by zero!.

10. Виведіть імена видатних особистостей України, зображених на грошових знаках. Користувач вводить номінал банкноти. Програма виводить значення номіналу і ім'я особи, яка зображена на цій банкноті. Якщо користувач вводить неіснуюче значення номіналу, на екран виводиться відповідне повідомлення.

11. Створіть програму, яка зчитує позицію шахової фігури від користувача і повідомляє про колір квадрату, де знаходиться фігура. Наприклад, якщо користувач вводить a і 1, то програма повинна повідомити, що квадрат має колір black, якщо d і 5 - програма повідомляє, що квадрат має колір white.

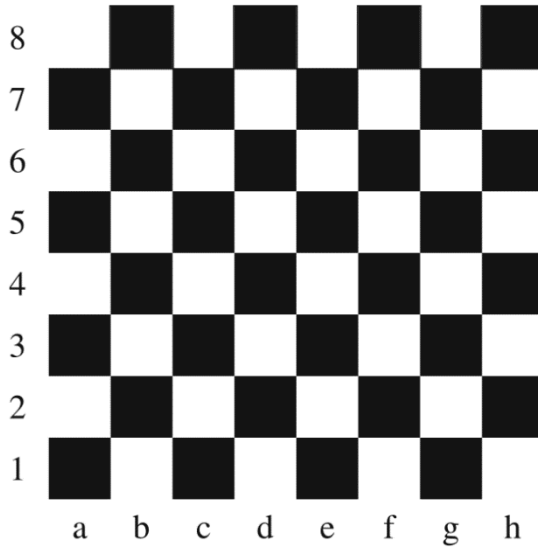


Рисунок 1.1 - Шахова дошка

12. Чи є рядок «паліндромом»? Рядок є паліндромом, якщо він однаково читається зліва направо та справа наліво. Наприклад, слова Level, Noon, Anna є паліндромами, незалежно від регістру літер. Рядки, які треба перевірити, вводяться користувачем. Введення даних можна перервати, якщо ввести слово escape. Програма повинна вивести результат перевірки у вигляді повідомлення is a palindrome або is not a palindrome.

13. Створіть таблицю множення. Розмірність таблиці (кількість рядків і стовпців) вводиться з клавіатури.

14. Як відомо, День програміста припадає на 256 день року, у невисокосний рік це 13 вересня, а у високосний - 12. Дізнайтеся число і номер місяця, що припадають на день, за номером n, який вводиться користувачем, у невисокосному 2017 році.

15. Конвертуйте десяткове число (з основою 10) у бінарне (двійкове, з основою 2). Користувач вводить десяткове число як ціле, а потім використовується алгоритм поділу (псевдокод), показаний нижче, щоб виконати перетворення. Коли алгоритм завершується, результат містить двійкове подання числа. Програма має вивести відповідне повідомлення, на зразок: 2018 in Decimal is 1111100010 in

Binary. Модифікуйте програму таким чином, щоб вона виконувала зворотну операцію: перетворювала двійкове чило у десяткове.

```
Нехай result буде порожнім рядком
Нехай q
представляє ціле число для перетворення
Повторити
Встановити r рівним остачі, коли q ділиться на 2
Перетворити r у рядок та додати його на початку до
result
Розділити q на 2, відкидаючи будь-який
залишок, і зберегти результат назад у q
Поки q не
дорівнює 0
Вивести повідомлення про результат
переведення
```

16. Реалізуйте шифр Цезаря. Один з перших відомих прикладів шифрування був використаний Юлієм Цезарем. Цезар надавав письмові вказівки своїм генералам, але він не бажав, зрозуміло, щоб про них знали вороги. У результаті він створив власне шифрування, що згодом стало відоме як шифр Цезаря. Ідея цього шифру проста (і, як наслідок, вона не забезпечує серйозного захисту). Кожна буква в оригінальному повідомленні зміщується на 3 позиції. В результаті А стає D, В стає Е, С стає F, D стає G і т. д. Останні три букви у алфавіті повертаються до початку: Х стає А, Y стає В, а Z стає С. Нелітерні символи не враховуються шифром. Користувач вводить з клавітури повідомлення та зсув, а потім програма відображає зашифроване повідомлення. Переконайтеся, що програма шифрує як великі та малі літери. Програма також повинна підтримувати значення негативних зсувів, щоб її можна було використовувати як для кодування повідомлень, так і для декодування повідомлень. У нагоді стануть такі функції: `ord()` (перетворює символ у номер позиції цього символу у таблиці ASCII) і `chr()` (перетворює номер позиції символу у таблиці ASCII у відповідний символ).

17. У програмі визначте функцію, яка генерує випадковий пароль. Пароль має бути довільної довжини від 7 до 10 символів. Кожен символ паролю повинен бути випадково обраним з позицій 33 до 126 з таблиці кодів ASCII. Функція повертає випадково сформований пароль як єдиний її результат і він виводиться в основній програмі. Скористайтеся вказівками з попередньої задачі.

18. Напишіть програму, у якій комп'ютер генерує випадкове число, а користувач повинен вгадати число, вводючи його з клавіатури за вказану кількість спроб.

19. Створіть програму за мотивами відомої гри «Камінь, Ножиці, Папір». Застосуйте функціональний підхід при написанні коду програми.

1.3Зміст звіту

Звіт повинен мати такі складові:

- титульний лист;
- мета роботи;
- результат виконання завдань 1, 2;
- відповіді на контрольні питання.

1.4Контрольні питання

1. Які два значення має булевий тип даних?
2. Назвіть три булевих оператори і шість операторів порівняння.
3. Напишіть кілька прикладів із таблиць істинності.
4. Яка різниця між операторами порівняння і присвоювання?
5. Що таке вказівка розгалуження і де вона використовується?
6. Наведіть приклади виразів, що можуть використовуватись як умова.
7. Яке сполучення клавіш використовується для виходу з нескінченного циклу?
8. Чим відрізняються виклики функцій `range(15)`, `range(0, 15)`, `range(0, 15, 1)` у циклі `for`?
9. Навіщо використовувати у програмах функції?
10. Коли виконується код функції: коли вона визначається, чи коли вона викликається?
11. З допомогою якої конструкції створюються функції?
12. Чим відрізняється визначення функції від її виклику?
13. Скільки глобальних областей видимості може мати програма на Python? Скільки локальних?
14. Що таке «значення, яке повертає функція»? Чи може бути це значення у вигляді виразу?

- 15.Що відбувається із змінними, які знаходяться у локальній області видимості, при поверненні їх із функції?
- 16.Яке значення повертає функція, якщо у ній відсутній оператор return?
- 17.Як змусити змінну у функції посилатися на глобальну змінну?
- 18.Що таке тип даних None?
- 19.Як можна запобігти аварійному завершенню програми?
- 20.Які блоки коду записують у блок try? А які у except?

2 ЛАБОРАТОРНА РОБОТА №7 ОПРАЦЮВАННЯ ФАЙЛІВ

Мета роботи: отримати навички програмування на мові Python з використанням файлів.

2.1 Теоретичні відомості

Текстові файли — це файли, що представлені у вигляді рядків типу str. У мові Python для таких рядків відбувається автоматичне кодування/декодування символів Unicode, а також відповідним чином обробляється символ кінця рядка.

Файл може бути **джерелом інформації** (під час введення) і **приймачем інформації** (під час виведення даних). У будь-якому випадку робота з файлом складається з трьох етапів:

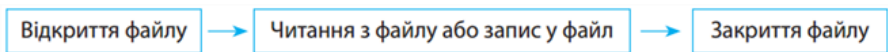


Рисунок 2.1 – Робота із файлом

Для відкриття файлу користуються вбудованою функцією `open()`:

`f = open(<шлях до файлу>, <режим відкриття>)`

де `f` — змінна для збереження файлового об'єкта, який використовується для читання та/або запису у файл.

Ім'я файлу може бути повним або скороченим. Повне ім'я (повний шлях до файлу) в ОС Windows включає:

- назву носія, на якому розміщено файл (диск C:, D: тощо);
- перелік папок від верхнього до нижнього рівня, розділених символом «\». Приклад повного імені: "D:\Моя_папка\myfile.txt".

Якщо файл збережено в одній папці з програмою, достатньо вказати лише ім'я файлу: "myfile.txt".

Файловий об'єкт можна уявити як посередника між програмою і зовнішнім файлом. Через файловий об'єкт відбувається обмін інформацією між програмою і фізичним файлом, який існує на зовнішньому носії.

Якщо шлях до файлу складається лише з назви файлу, Python буде шукати цей файл у папці, у який міститься програма

Параметр режим відкриття визначає, які дії можна буде виконувати з файлом. Основні режими відкриття такі:

- 'r' — тільки для читання;
- 'w' — тільки для запису. Знищує вміст файлу, якщо файл існує, і створює новий файл для запису, якщо не існує;
- 'a' — для дозапису. Інформація додається в кінець наявного файлу; якщо файл не існує, його буде створено.

Читання даних із файлу означає отримання доступу до цих даних.

Функція **readline()** зчитує із файлу один рядок. Після зчитування елемента (символу або рядка) з файлу поточним (доступним) стає наступний елемент.

```
f = open('f1.txt')
s=f.readline()
print(s)
```

Функція **read([n])** зчитує всі дані з файлу, якщо аргумент **n** відсутній, і зчитує **n** символів, якщо задано ціле значення **n**.

2.2 Хід роботи

Завдання 1. Виконайте в інтерактивному інтерпретаторі тренувальні вправи (завдання не обов'язкові):

1. Збережіть рядок 'Test variable to write to file' у змінну test1 і запишіть змінну test1 у файл з ім'ям **tests.txt**.

2. Відкрийте файл **tests.txt** і зчитайте його вміст у рядок test2.

3. Порівняйте рядки test1 і test2?

4. Скопіюйте наступні кілька рядків у файл **painters.csv**.

author, canvas Vincent Willem van Gogh, "Vase with sunflowers"
Rembrandt Harmenszoon van Rijn, "Aristotle" Leonardo da Vinci, "Self-portrait"

5. Використайте модуль **csv** і його функцію **DictReader()**, щоб зчитати вміст файла **painters.csv** у змінну painters. Виведіть значення змінної на екран.

6. Створіть CSV-файл **imdb.csv** з рейтингом фільмів за версією IMDb, використовуючи список словників:

```
imdb = [
    { 'title': 'Lord of the Rings: Two towers',
      'year': 2002,
      'rating': 8.7},
```

```
{
  'title': 'Matrix',
  'year': 1999,
  'rating': 8.7},
{ 'title': 'Interstellar',
  'year': 2014,
  'rating': 8.5},
{ 'title': 'Back to the Future',
  'year': 1985,
  'rating': 8.5},
{ 'title': 'Logan: Wolverine',
  'year': 2017,
  'rating': 8.1}
]
```

Завдання 2. Напишіть програми у середовищі програмування для розв'язування таких завдань відповідно варіанту:

1. Створіть новий файл **numbers.txt** у текстовому редакторі і запишіть у нього 10 чисел, кожне з нового рядка. Напишіть програму, яка зчитує ці числа з файла і обчислює їх суму, виводить цю суму на екран і, водночас, записує цю суму у інший файл з назвою **sum_numbers.txt**.

2. Реалізуйте програму, яка зчитує ціле число, що вводиться з командного рядка, і записує у текстовий файл інформацію, щодо парності або непарності числа.

3. Створіть новий файл у текстовому редакторі і напишіть кілька рядків тексту у ньому про можливості Python. Кожен рядок повинен починатися з фрази: In Python you can Збережіть файл з ім'ям **learning_python.txt**. Напишіть програму, яка зчитує файл і виводить текст з перебором рядків об'єкта файла і зі збереженням рядків у списку з подальшим виведенням списку поза блоком **with**.

4. Функція **replace()** може використовуватися для заміни будь-якого слова у рядку іншим словом. Прочитайте кожен рядок зі створеного у попередньому завданні файла **learning_python.txt** і замініть слово Python назвою іншої мови, наприклад C при виведенні на екран.

5. Створіть порожній файл **guest_book.txt** у текстовому редакторі. Напишіть цикл **while**, який запитує у користувачів імена. При введенні кожного імені виведіть на екран рядок з вітанням для

користувача і запишіть рядок вітання у файл з ім'ям **guest_book.txt**. Простежте за тим, щоб кожне повідомлення розміщувалося в окремому рядку файла. Передбачте вихід з циклу.

6. Зайдіть на сайт [Project Gutenberg's](#) і знайдіть кілька книг для аналізу. Завантажте текстові файли цих творів або скопіюйте текст з браузера у текстовий файл на вашому комп'ютері. Напишіть програму, яка зчитує дані з файла і визначає кількість входжень слова 'the' в кожному тексті незалежно від регістру. Для підрахунку кількості входжень слова або виразу в рядок можна скористатися функцією **count()**.

7. Завантажте текстову версію однієї з книг із сайту [Project Gutenberg's](#). Замініть усі розриви рядків у тексті символом пропуску і запишіть відформатований текст у новий файл **formatted_text.txt**.

8. Завантажте текстову версію книги [The Life and Adventures of Robinson Crusoe By Daniel Defoe](#) із сайту [Project Gutenberg's](#). Витягніть із тексту заголовки усіх розділів, які мають вигляд, на зразок: CHAPTER I—START IN LIFE. Запишіть знайдені назви у новий файл **chapters.txt**.

9. Визначте відсоток малих і великих літер у тексті, що зберігається у файлі. Скористайтесь, як зразком вхідного текстового файла, файл [The Count of Monte Cristo](#) із сайту [Project Gutenberg's](#). Використайте функцію **isalpha()**.

10. Використайте модуль **sqlite3**, щоб створити базу даних SQLite під назвою **imdb.db** і таблицю ratings, що містить наступні поля: id (INTEGER PRIMARY KEY), title (VARCHAR(20)), year (INT), rating (FLOAT). Зчитайте дані з файла **imdb.csv** і додайте їх у таблицю ratings. Зчитайте і виведіть на екран усі значення таблиці ratings у алфавітному порядку за полем title. Зчитайте і виведіть на екран усі записи таблиці ratings з ретингом більшим за 8.70.

2.3 Зміст звіту

Звіт повинен мати такі складові:

- титульний лист;
- мета роботи;
- результат виконання завдань 1, 2;

—відповіді на контрольні питання.

2.4 Контрольні запитання

1. Які аргументи задають режим відкривання файла з допомогою функції `open()`?
2. За допомогою яких функцій можна записувати у файл?
3. Чим відрізняється робота функцій `read()`, `readline()` і `readlines()`?
4. Для чого використовують у роботі з файлами оператор `with`?
5. Як створити і прочитати бінарний файл?
6. Які засоби Python використовуються для відслідковування місцезнаходження конкретних байтів у бінарних файлах?
7. Як використовуються функції `csv.reader()` і `csv.writer`?
8. Яка функція приймає JSON-рядок і повертає структуру даних Python?
9. Яка функція приймає структуру даних Python, а повертає рядок JSON-даних?
10. Які інструменти Python використовують для отримання доступу до баз даних і роботи з ними?
11. Що таке SQL?
12. Наведіть приклад запиту з використанням мови SQL.

3 ЛАБОРАТОРНА РОБОТА №8

ВИКОРИСТАННЯ ФУНКЦІЙ ДОСТУПУ ДО ОПЕРАЦІЙНОЇ СИСТЕМИ

Мета роботи: отримати навички програмування на мові Python з використанням доступу до операційної системи.

3.1 Теоретичні відомості

Робота з файловою системою передбачає створення, перейменування, переміщення або звернення до файлів і каталогів. Усі ці та багато інших завдань можна виконати за допомогою програм, написаних на Python.

Python надає багато системних функцій, що містяться в модулі **os** (скорочення від Operating System - операційна система), який необхідно імпортувати для роботи програм.

Частина назви файла, яка слідує за останньою крапкою, називається **розширенням файла** і вказує на **тип файла**. Розширення файла, зазвичай, надає програма, у якій він був створений.

Наприклад, файл з назвою **projects.docs** має тип файла - документ Word, про що свідчить розширення .docs. Частина назви файла (projects) до останньої крапки - це **ім'я файла**, яке вказує користувач.

Файл має дві ключові ознаки: **назва файла** і **шлях до файла**.

Шлях визначає, де саме на комп'ютері знаходиться файл. Наприклад, файл з назвою **projects.docs** може мати шлях до файла C:\Users\gt\Documents. Це означає, що файл **projects.docs** зберігається у каталозі **Documents**.

Частина C:\ шляху до файла - це **кореневий каталог**, який містить усі інші каталоги (**gt, Users**) і сам файл **projects.docs**.

У системах Windows кореневим каталогом є C:\, а шляхи до файлів записують з використанням **зворотного слешу (\)**, який розділяє імена каталогів. В Linux Ubuntu кореневим каталогом є каталог /, а розділення імен каталогів виконують з використанням **прямого слеша (/)**.

Шлях до файла (або каталогу), який враховує використання обох слешів, можна сформувати з використанням функції **join()**, яка міститься у модулі **path**, який у свою чергу міститься у модулі **os**.

Якщо передати у цю функцію значення назв файлів і каталогів й виконати наступний код

```
import os
print(os.path.join('usr', 'bin', 'media'))
```

функція поверне значення, у якому сформується шлях з коректною підстановкою слеша-розділювача:

```
usr\bin\media
```

Якщо ж виконати даний приклад в інтерактивному режимі інтерпретатора Python, виклик функції **os.path.join()** поверне рядок:

```
'usr\bin\media'
```

Якщо викликати вищезгадану функцію **os.path.join()** в Linux Ubuntu, вона поверне рядок:

```
'usr/bin/media'
```

Отже, для формування шляхів до файлів або каталогів у вигляді рядків використовується функція **os.path.join()**. Наприклад, наступний код

```
import os
myfiles = ['accounts.txt', 'details.csv', 'invite.docs']
for filename in myfiles:
    print(os.path.join('C:\\Users\\gt\\Documents', filename))
```

формує шляхи до файлів, додаючи до імен каталогів назви файлів зі списку:

```
C:\Users\gt\Documents\accounts.txt
C:\Users\gt\Documents\details.csv
C:\Users\gt\Documents\invite.docs
```

Кожна програма, яка виконується на комп'ютері, має свій **поточний робочий каталог** (current working directory - cwd).

Будь-які шляхи до файлів або каталогів, які не починаються з кореневого каталогу, задаються відносно поточного робочого каталогу.

Для отримання значення поточного робочого каталогу у вигляді рядка використовується функція **os.getcwd()**, а для переходу з поточного робочого каталогу у інший каталог - функція **os.chdir()**.

```
import os
print(os.getcwd())
os.chdir('C:\\Windows\\Temp')
```

```
print(os.getcwd())
```

У даному випадку, в якості робочого поточного каталогу встановлюється каталог по шляху C:\Python36, відбувається перехід у інший каталог, який стає поточним:

```
C:\Python36
```

```
C:\Windows\Temp
```

Існує два способи визначення шляху до файлу або каталогу:

- **абсолютний шлях** - завжди починається з імені кореневого каталогу;

- **відносний шлях** - задається відносно поточного робочого каталогу програми.

Існують також каталоги, які позначаються одним (.) або двома (..) символами крапки. Це не реальні каталоги, а спеціальні імена, які використовуються для визначення шляхів.

Для створення нових каталогів використовується функція **os.makedirs()**, яка створює усі необхідні проміжні каталоги:

```
os.makedirs('C:\\Python36\\other\\myprograms')
```

Модуль **os.path** містить функції для визначення абсолютного шляху по заданому відносному, а також для перевірки того, чи шлях є абсолютним.

Виклик **os.path.abspath(path)** - повертає рядок абсолютного шляху до файлу або каталогу.

Виклик **os.path.isabs(path)** - повертає True, якщо шлях є абсолютним, і False у протилежному випадку, коли шлях виявиться відносним.

Виклик **os.path.relpath(path, start)** - повертає рядок відносного шляху від точки start до path. Якщо start не вказаний, то в якості нього використовується поточний робочий каталог.

На наступному прикладі коду:

```
import os
```

```
print(os.path.abspath('.'))
```

```
print(os.path.abspath('.\\Scripts'))
```

```
print(os.path.isabs('.'))
```

```
print(os.path.isabs(os.path.abspath('.')))
```

```
print(os.path.relpath('C:\\Windows', 'C:\\'))
```

```
print(os.path.relpath('C:\\Windows', 'C:\\Python36\\Scripts'))
```

```
print(os.getcwd())
```

робота цих функцій виглядає так:

```

C:\Python36
C:\Python36\Scripts
False
True
Windows
..\..\Windows
C:\Python36

```

Для отримання із повного шляху лише назви файла використовують функцію **os.path.basename()**. Якщо необхідно отримати частину шляху без назви файла використовують функцію **os.path.dirname()**.

Якщо шлях до файла вважати таким рядком 'C:\\Windows\\System32\\calc.exe', то ці дві функції:

```

import os
path = 'C:\\Windows\\System32\\calc.exe'
print(os.path.basename(path))
print(os.path.dirname(path))

```

відпрацюють так:

```

calc.exe
C:\\Windows\\System32

```

Щоб отримати одночасно і назву файла, так і імена каталогів на шляху до нього, можна скористатися функцією **split()** у такому виконанні:

```

import os
path = 'C:\\Windows\\System32\\calc.exe'
print(os.path.split(path))

```

Результатом виведення буде кортеж з двох елементів:

```

('C:\\Windows\\System32', 'calc.exe')

```

Щоб отримати список усіх каталогів на шляху до файла і назву самого файла використовують розділювач **os.path.sep**.

```

import os
path = 'C:\\Windows\\System32\\calc.exe'
print(path.split(os.path.sep))

```

Для системи Windows буде такий результат:

```
['C:', 'Windows', 'System32', 'calc.exe']
```

Для системи Linux Ubuntu

```

import os
print('/usr/bin'.split(os.path.sep))

```

першим елементом такого списку буде порожній рядок:

```
['', 'usr', 'bin']
```

У модулі **os.path** присутні функції, які дозволяють обчислювати розміри файлів у байтах і визначати, які файли та каталоги знаходяться у вказаному каталозі. До таких функцій належать:

- **os.path.getsize(path)** - повертає у байтах розмір файла, шлях до якого вказаний у path;

- **os.listdir(path)** - повертає список рядків з назвами усіх файлів, шлях до яких вказаний у path.

Коли виконати поданий код

```
import os
print(os.path.getsize('C:\\Windows\\System32\\calc.exe'))
print(os.listdir('C:\\Windows\\System32'))
```

отримаємо розмір у байтах вказаного файла і список усіх файлів вказаного каталогу.

Для знаходження сумарного розміру файлів, наприклад, у каталозі по шляху C:\\Python36, можна використати таку програму:

```
total = 0
for filename in os.listdir('C:\\Python36'):
    total = total + os.path.getsize(os.path.join('C:\\Python36',
filename))
print(total)
```

Результатом виконання даної програми буде розмір у байтах усіх файлів каталогу по шляху C:\\Python36:

```
4582106
```

Функції Python для роботи з файлами і каталогами можуть повідомляти про помилки, коли не знаходять шлях до файла або каталогу. Модуль **os.path** містить функції, які перевіряють, чи існує шлях до каталогу або файла:

- **os.path.exists(path)** - повертає значення True, якщо **файл (або каталог)** за вказаним шляхом path існує, і значення False у протилежному випадку;

- **os.path.isfile(path)** - повертає значення True, якщо **файл** за вказаним шляхом path існує, і значення False у протилежному випадку;

- **os.path.isdir(path)** - повертає значення True, якщо **каталог** за вказаним шляхом path існує, і значення False у протилежному випадку.

3.2 Завдання до виконання

Завдання 1. Виконайте в інтерактивному інтерпретаторі тренувальні вправи (завдання не обов'язкові):

1. Виведіть на екран повний шлях і список файлів поточного каталогу.
2. Виведіть на екран розмір у байтах будь-якого файла у каталозі.
3. Здійсніть пошук файлів у поточному каталозі за певним шаблоном і виведіть результати на екран.
4. У поточному каталозі вручну створіть два текстові файли, перший з яких містить довільний текст, другий - порожній. Використовуючи функції для роботи з файлами, виконайте такі дії: виконайте копіювання даних з першого файла у другий, зчитайте з другого файла скопійовані дані і виведіть їх на екран, а потім видаліть перший файл.
5. Виведіть на екран поточну дату й час.
6. Виведіть на екран кількість днів і років, які пройшли з початку «епохи Unix». Вважати 1 рік = 365 днів.

Завдання 2. Напишіть програми у середовищі програмування для розв'язування таких завдань:

1. Запишіть програмно поточні дату й час як рядок у текстовий файл `today.txt`. Зчитайте текстовий файл `today.txt` і розмістіть дані у рядку `today_string`. Розберіть дату з рядка `today_string` на складові, використовуючи функції для роботи з датою й часом, і виведіть їх на екран у вигляді повідомлень з пояснюючим текстом.
2. Створіть об'єкт `harry`, що містить дату вашого народження. Використайте функції для роботи з датою і часом, щоб дізнатися відповіді на такі питання (виведення організуйте в окремий файл у вигляді, на зразок «запитання: відповідь», у окремих рядках):
 - a) "Яка дата вашого народження?"
 - b) "У який день тижня ви народилися?"
 - c) "Коли вам буде (або вже було) 13 330 днів від народження?"

3.3 Зміст звіту

Звіт повинен мати такі складові:

- титульний лист;
- мета роботи;
- результат виконання завдань;
- відповіді на контрольні питання.

3.4 Контрольні запитання

1. Відносно чого задається відносний шлях?
2. З чого починається абсолютний шлях?
3. Яке призначення функцій `os.getcwd()` і `os.chdir()`?
4. Що позначають імена `.` і `..`?
5. Які частини шляху `C:\Users\gt\logging\log12042019.txt` представляють імена каталогів та ім'я файлу?
6. Які функції використовуються для копіювання, перейменування і видалення файлів і каталогів?
7. Що таке «епоха Unix»?
8. Яка функція повертає кількість секунд, які пройшли з моменту початку «епохи Unix»?
9. У чому різниця між об'єктами `datetime` і `timedelta`?
10. Які інструменти локалізації присутні у Python?

JITEPATYPA

1. Barry P. Head First Python: A Brain-Friendly Guide / Paul Barry. – [S. l.] : O'Reilly Media, Incorporated, 2023.
2. Borjigin C. Python Data Science / Chaolemen Borjigin. – [S. l.] : Springer, 2023.
3. Chongchitnan S. Exploring University Mathematics with Python [Electronic resource] / Siri Chongchitnan. – Cham : Springer Nature Switzerland, 2023. – Mode of access: <https://doi.org/10.1007/978-3-031-46270-2>
4. Deep Learning and Its Applications Using Python / Niha K. Basha [et al.]. – [S. l.] : Wiley & Sons, Incorporated, John, 2023.
5. Hazrat R. A Course in Python [Electronic resource] / Roozbeh Hazrat. – Cham : Springer Nature Switzerland, 2023. – Mode of access: <https://doi.org/10.1007/978-3-031-49780-3>
6. Korites B. Python Graphics [Electronic resource] / Bernard Korites. – Berkeley, CA : Apress, 2023. – Mode of access: <https://doi.org/10.1007/978-1-4842-9660-8>
7. Kouh M. Thermal Physics Tutorials with Python Simulations / Minjoon Kouh, Taejoon Kouh. – [S. l.] : Taylor & Francis Group, 2023.
8. Matuszek D. Quick Python 3 / David Matuszek. – [S. l.] : CRC Press LLC, 2023.
9. Miller R. Computer Scripting Techniques with Python / Rick Miller. – [S. l.] : Pulp Free Press, 2023.
10. Nelli F. Python Data Analytics [Electronic resource] / Fabio Nelli. – Berkeley, CA : Apress, 2023. – Mode of access: <https://doi.org/10.1007/978-1-4842-9532->
11. Python Programming Recipes for IoT Applications / Madhusudan Ganuji Lanjewar [et al.]. – [S. l.] : Springer, 2023.

ДОДАТОК А

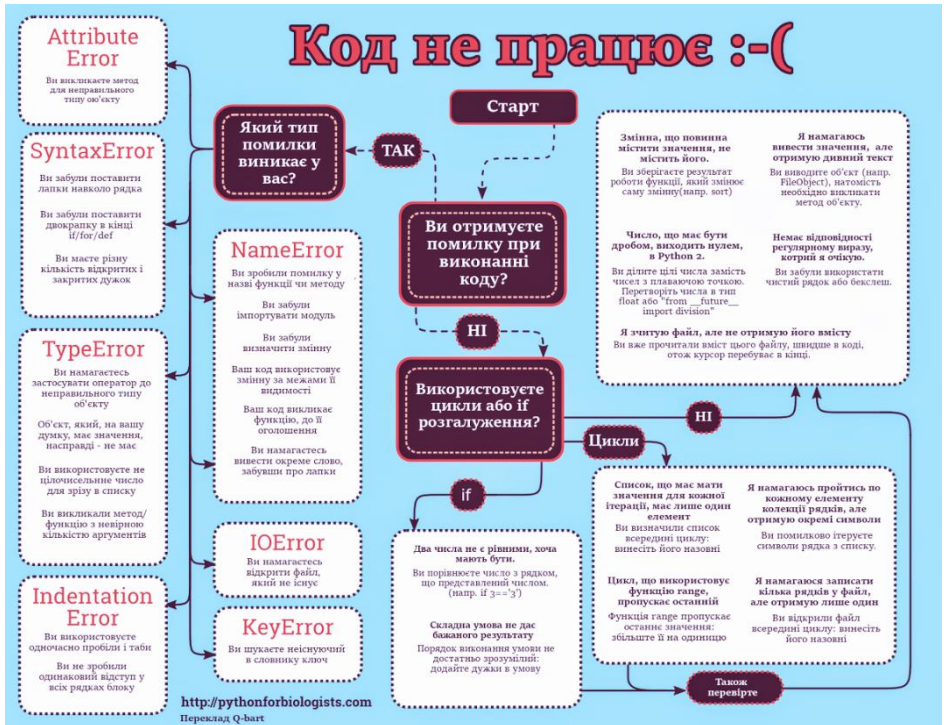
Стандартні функції Python

Виклик функції	Опис
help(x)	Документація об'єкта x
dir(obj)	Список імен/атрибутів об'єкта obj в алфавітному порядку
type(x)	Тип об'єкта x
id(x)	Ідентифікатор (адреса в пам'яті) об'єкта x
print(x)	Виведення x на екран або запис у текстовий файл
open(file, mode)	Відкриття файлу file в одному із режимів mode і повертання відповідного файлового об'єкта
input(x)	Зчитування і повертання рядка x вхідних даних
x.format()	Відформатоване подання рядка x, яке контролюється специфікатором формату
hash(x)	Значення хеш-функції об'єкта x
int(x)	Перетворення об'єкта x в ціле число
float(x)	Перетворення об'єкта x в дійсне (дробове) число
str(x)	Перетворення об'єкта x в рядок
chr(x)	Повертає символ (у вигляді рядка), чия позиція коду для Юнікод дорівнює зазначеному цілому x. Аргумент повинен розташовуватися в межах від 0 до 1,114,111 (0x10FFFF), в іншому випадку порушується виняток ValueError
ord(x)	Повертає для зазначеного Юнікод-символу ціле число, що представляє його позицію коду
round(x)	Найближче ціле число до числа x
abs(x)	Абсолютне значення x
pow(a, b)	Піднесення числа a до степеня b
min(x)	Мінімальний елемент в ітерованому x
max(x)	Максимальний елемент в ітерованому x
sum(x)	Сума елементів в ітерованому x
len(x)	Довжина ітерованого x (кількість елементів в x)
sorted(x)	Новий ітерований об'єкт, що містить елементи x у відсортованому порядку

reversed(x)	Новий ітерований об'єкт, що містить елементи x в зворотному порядку
tuple(x)	Новий кортеж tuple, створений з елементів x
list(x)	Новий список list, створений з елементів x
set(x)	Нова множина set, створена із елементів x
range(i, j)	Новий ітерований об'єкт, який містить числа від i до j-1 (стандартне значення i - 0)

ДОДАТОК Б

Типові помилки



ДОДАТОК В

Пріоритет операторів

(вирази...), [вирази...], {ключ: значення...}, {вирази...}	Вираз в дужках, створення або включення списку/словника/множин и
x[індекс], x[індекс:індекс], x(аргументи ...), x.атрибут	Індекс, зріз, виклик функції, посилання на атрибут
**	Піднесення до степеня
+x, -x, ~x	Знаки «+» і «-», бітове НІ
*, /, //, %	Множення, ділення з плаваючою крапкою, цілочисельне ділення, остача від ділення
+, -	Додавання, віднімання
<<, >>	Бітові зсуви
&	Бітове І
	Бітове АБО
^	Бітове виключення АБО
in, not in, is, is not, <, <=, >, >=, !=, ==	Перевірка на приналежність і рівність
not x	Булеве (логічне) НІ
and	Булеве (логічне) І
or	Булеве (логічне) АБО
if ... else	Умовний вираз
lambda	Лямбда-вираз

ДОДАТОК Г

Модулі Python

Перед використанням функцій модуля необхідно його імпортувати таким чином:

`import назва_модуля .`

Для використання функцій модуля необхідно використовувати такий запис:

`назва_модуля.назва_функції .`

Таблиця Г.1 - Математичні функції і константи

Виклик функції/константи	Опис/значення
<code>math.sin(x)</code>	Синус x (представлений у радіанах)
<code>math.cos(x)</code>	Косинус x (представлений у радіанах)
<code>math.radians(x)</code>	Перетворення x (в градусах) у радіани
<code>math.degrees(x)</code>	Перетворення x (в радіанах) у градуси
<code>math.exp(x)</code>	Експонентна функція x (e^x)
<code>math.sqrt(x)</code>	Квадратний корінь x
<code>math.factorial(x)</code>	Факторіал цілого числа x
<code>math.e</code>	2.718281828459045
<code>math.pi</code>	3.141592653589793

Таблиця Г.2 - Атрибути та їх значення модуля `string`

Атрибут	Значення
<code>ascii_letters</code>	'abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ'
<code>ascii_lowercase</code>	'abcdefghijklmnopqrstuvwxyz'
<code>ascii_uppercase</code>	'ABCDEFGHIJKLMNOPQRSTUVWXYZ'
<code>digits</code>	'0123456789'
<code>hexdigits</code>	'0123456789abcdefABCDEF'
<code>octdigits</code>	'01234567'
<code>punctuation</code>	'!"#\$%&'()*+,-./:;<=>?@[\\]^_`{ }~'
<code>printable</code>	'0123456789abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ!~'"#\$%&'()*+,-./:;<=>?@[\\]^_`{ }~'

	<code>./:;<=>?@[\\]^_`{ }~\t\n\r\x0b\x0c'</code>
whitespace	<code>'\t\n\r\x0b\x0c'</code>

Таблиця Г.3 - Функції для генерування псевдовипадкових чисел

Виклик функції	Опис
<code>random.randrange(start, stop, step)</code>	Випадкове ціле число із діапазону від start до stop
<code>random.randint(A, B)</code>	Випадкове ціле число N ($A \leq N \leq B$)
<code>random.choice(sequence)</code>	Випадковий елемент непорожньої послідовності sequence
<code>random.sample(population, k)</code>	Список довжиною k із послідовності population
<code>random.random()</code>	Випадкове число від 0 до 1

ДОДАТОК Е

Методи

Таблиця Е.1 - Методи рядків

Метод	Призначення
s.capitalize()	Повертає копію рядка s, роблячи першу букву у верхньому регістрі
s.lower()	Повертає копію рядка s із символами у нижньому регістрі
s.swapcase()	Повертає копію рядка s, в якій кожна буква матиме протилежний регістр
s.title()	Повертає копію рядка s, в якій кожне нове слово починається з великої літери
s.upper()	Повертає копію рядка s із символами у верхньому регістрі
s.count(x)	Для рядка s повертає кількість входжень в нього зазначеного підрядка x
s.find(x)	Повертає найменший індекс з рядка s, за яким знаходиться початок зазначеного підрядка x (якщо підрядок x не знайдено, повертає -1)
s.index(x)	Повертає найменший індекс з рядка s, за яким знаходиться початок зазначеного підрядка x (якщо підрядок не знайдено, створюється виняток ValueError)
s.rfind(x)	Повертає найбільший індекс з рядка s, за яким знаходиться початок зазначеного підрядка x (якщо підрядок x не знайдено, повертає -1)
s.rindex(x)	Повертає найбільший індекс з рядка s, за яким знаходиться початок зазначеного підрядка x (якщо підрядок не знайдено, створюється виняток ValueError)
s.replace(a, b)	Повертає копію рядка s, де всі входження підрядка a у рядок s замінюються підрядком b
s.startswith(x)	Повертає True, якщо рядок s починається з вказаного префіксу x (якщо ні, то повертає False)
s.endswith(x)	Повертає True, якщо рядок s закінчується вказаним префіксом x (якщо ні, то повертає False)

s.join(x)	Повертає рядок, складений з елементів ітеративного об'єкта x з розділювачами s
s.lstrip(chars)	повертає копію рядка s, на початку якого видалені символи chars
s.rstrip(chars)	повертає копію рядка s, наприкінці якого видалені символи chars
s.strip(chars)	повертає копію рядка s, на початку і у кінці якого видалені символи chars
s.split(char)	Розділення рядка s по розділювачу char і зберігання у список
s.center(width, fill)	Повертає відцентрований рядок, по краях якого стоїть символ fill (пропуск за замовчуванням)
s.ljust(width, fillchar)	Робить довжину рядка s не меншою width, в разі потреби заповнюючи останні символи символом fillchar
s.rjust(width, fillchar)	Робить довжину рядка s не меншою width, в разі потреби заповнюючи перші символи символом fillchar
s.isalnum()	Чи складається рядок s з букв і цифр?
s.isalpha()	Чи складається рядок s з букв?
s.isdigit()	Чи складається рядок s з цифр?
s.istitle()	Чи починаються слова в рядку s з великої букви?
s.isupper()	Чи складається рядок s із символів у верхньому регістрі?
s.islower()	Чи складається рядок s із символів у нижньому регістрі?

Таблиця E.2 - Методи списків

Метод	Призначення
list.append(x)	Додає елемент x в кінець списку list
list.extend(L)	Розширює список list, додаючи у кінець всі елементи списку L
list.insert(i, x)	Вставляє у списку list на i-ий елемент значення x
list.remove(x)	Видаляє перший елемент у списку list, який має значення x; ValueError, якщо такого елемента не існує

list.pop(i)	Видаляє із списку list i-ий елемент і повертає його; якщо індекс не вказано, видаляється останній елемент
list.index(x, start, end)	Повертає положення у списку list першого елемента зі значенням x (при цьому пошук ведеться від start до end)
list.count(x)	Повертає кількість елементів із списку list зі значенням x
list.sort(reverse=False)	Сортує елементи списку list; якщо необов'язковий параметр reverse має значення True, список сортується в зворотному порядку
list.reverse()	Розвертає список list
list.copy()	Поверхнева копія списку list
list.clear()	Очищає список list

Таблиця Е.3 - Методи словників

Метод	Призначення
dict.clear()	Очистити словник dict
dict.copy()	Повертає копію словника dict
dict.fromkeys(seq, value=None)	Створює словник dict з ключами з послідовності seq і значенням value (за замовчуванням None)
dict.get(key, default=None)	Повертає значення ключа із словника dict, але якщо ключ відсутній, не створює виняток, а повертає default (за замовчуванням None)
dict.items()	Повертає пари (ключ, значення) із словника dict у вигляді списку кортежів [(key1, value1), (key2, value2), ...]
dict.keys()	Повертає ключі словника dict
dict.pop(key, default)	Видаляє ключ і повертає значення словника dict; якщо ключ відсутній, повертає default (якщо default відсутній - створює виняток KeyError)
dict.popitem()	Видаляє і повертає пару (ключ, значення) словника dict; якщо словник порожній, створює виняток KeyError (пам'ятайте, що словники

	невпорядковані)
dict.setdefault(key, default=None)	Повертає значення ключа словника dict, але якщо він відсутній, не створює виняток, а створює ключ із значенням default (за замовчуванням None)
dict.update(other)	Оновлює словник dict, додаючи пари (ключ, значення) із other; існуючі ключі перезаписуються (повертає None)
dict.values()	Повертає значення із словника dict у вигляді списку [value1, value2, ...]