Міністерство освіти і науки України
Чернівецький національний університет
імені Юрія Федьковича

ПРОГРАМУВАННЯ

Методичні рекомендації та завдання для практичних занять

Частина 2

Чернівці
Чернівецький національний університет
2025

УДК 044.43(076.5)
П-784

Рекомендовано до друку Вченою радою факультету математики та інформатики
Чернівецького національного університету імені Юрія Федьковича
(протокол № 9 від « 19 » березня 2025 року)

Укладачі:
Фратавчан Тоня Михайлівна, кандидат фізико-математичних наук, доцент кафедри математичного моделювання;
Дорошенко Ірина Вікторівна, кандидат фізико-математичних наук, доцент кафедри математичного моделювання;
Івасюк Роман Вікторович, асистент кафедри математичного моделювання;
Фратавчан Валерій Григорович, кандидат фізико-математичних наук, доцент кафедри МПУіК.

П-784 Програмування: методичні рекомендації та завдання для практичних занять. Частина 2. Укл.: Т.М.Фратавчан, І.В.Дорошенко, Р.В.Івасюк, В.Г.Фратавчан – Чернівці: Чернівецький національний університет, 2025. – 62 с.

Методичні рекомендації містять завдання для лабораторних робіт та теоретичні відомості, необхідні для їх виконання, з дисципліни «Програмування».
Для студентів, що здобувають освіту в галузі знань „Середня освіта (інформатика)”, „Середня освіта (математика)”, „Математика” та інших спеціальностей з подібною програмою вивчення дисципліни.

УДК 044.43(076.5)

Зміст

Вступ

Дисципліна «Програмування» призначена для вивчення сучасних методів опрацювання інформації, здобуття навиків алгоритмізації та програмування алгоритмічною мовою високого рівня Python та їх застосування у процесі навчання і майбутній професійній діяльності.

Мова програмування Python швидко розвивається. Цьому сприяє не тільки досить вдала концепція мови, а й згуртоване співтовариство, сформоване з розробників і популяризаторів мови. Необхідне програмне забезпечення, включаючи середовища розробки, в основному безкоштовне. Усе це дає підстави розглядати Python як одну з найперспективніших мов програмування.

На сьогодні Python використовується при реалізації найрізноманітніших проектів, серед яких:

- розробка сценаріїв для роботи з Web та Internet-програмами;
- мережеве програмування;
- засоби підтримки технологій HTML і XML;
- програми для роботи з електронною поштою й підтримки Internet-протоколів;
- програми для обслуговування найрізноманітніших баз даних;
- програми для наукових розрахунків;
- програми з графічним інтерфейсом;
- створення ігор і комп'ютерної графіки та багато іншого.

Методичні вказівки містять завдання до практичних робіт з програмування мовою Python, а також теоретичні відомості, необхідні для їх виконання, які дають змогу студентам самостійно засвоїти здобуті знання.

Матеріали методичних вказівок можуть бути корисними студентам різних спеціальностей всіх форм навчання як для розв'язання типових задач, так і для самостійного вивчення теоретичного матеріалу, а також як допоміжний засіб при організації дистанційного навчання.

Мета та програмні результати навчання

Мета навчальної дисципліни: надання студентам базових знань з теорії програмування, формування у студентів здатностей:

− застосовувати професійні знання й уміння на практиці ;

− використовувати інформаційні і комунікаційні технології;

− адаптуватися до різних професійних ситуацій, проявляти творчий підхід, ініціативу;

− аналізувати проблеми, ставити постановку цілей і завдань, виконувати вибір способу й методів дослідження, а також оцінку його якості;

− вирішувати проблеми в професійній діяльності на основі аналізу й синтезу;

− застосовувати сучасні парадигми програмування під час програмної реалізації професійних задач ;

− працювати автономно;

− розробляти і досліджувати математичні моделі явищ, процесів та систем;

− застосовувати чисельні методи для дослідження математичних моделей ;

− використовувати програмні засоби загального та спеціального призначення для розв’язання прикладних задач з математики та інформатики;

− використовувати обчислювальні інструменти для чисельних і символьних розрахунків .

Після вивчення даної дисципліни студент повинен

знати:

− основні поняття та твердження з програмного матеріалу даного курсу;

− основні поняття інформатики – виконавець, алгоритм, програма;

− синтаксис, семантику та властивості структур керування (ланцюг, розгалуження, цикли);

− синтаксичні конструкції мови програмування Python; формальні методи побудови

алгоритмів та програм за допомогою рекурентних співвідношень;

− основні властивості програм; будову простих та складених типів даних;

− опис та використання підпрограм;

− поняття про виключні ситуації;

− роботу з файлами;

− модульне програмування;

− спеціалізовані програмні засоби комп'ютерної та прикладної математики;

вміти:

− будувати лінійні, розгалужені та циклічні алгоритми з використанням підпрограм, модулів у мові програмування Python.

− аналізувати задачі шкільних курсів математики та інформатики різних рівнів

складності, демонструвати здатність їх розв’язувати ;

− застосовувати знання і розуміння для ідентифікації, формулювання і вирішення технічних задач спеціальності використовуючи відомі методи;

− застосовувати сучасні інформаційно-комунікаційні та цифрові технології, спеціалізовані програмні засоби комп’ютерної математики та інтернет-ресурси у професійній діяльності та пошуку наукової інформації для самоосвіти;

− застосовувати базові загальні математичні моделі для специфічних ситуацій, мати навички управління інформацією і застосування комп’ютерних засобів статистичного аналізу даних;

− описувати сучасні тенденції в математиці та інформатиці ;

− системно мислити та застосовувати творчі здібності до формування принципово нових ідей;

− здійснювати пошук інформації в різних джерелах для розв’язання задач спеціальності;

− визначати та застосовувати методи розроблення та дослідження алгоритмів розв’язування задач з інформатики, описувати і застосовувати методи оцінювання ефективності алгоритмів ;

− ефективно працювати як індивідуально, так і у складі команди;

− поєднувати теорію і практику.

Теоретичний зміст програми навчальної дисципліни у другому семестрі

Тема 1. Множини в Python.

Тема 2. Обробка виключень в Python.

Тема 3. Функції в Python.

Тема 4. Робота з файлами.

Тема 5. Бібліотека numpy.

Тема 6. Бібліотека matplotlib.

Тема 7. Бібліотека tkinter.

Тема 8. Елементи об’єктно-орієнтованого програмування в Python.

Практичне заняття №1. Множини та обробка виключень у Python

Мета: Ознайомитися з основними поняттями та методами роботи з множинами, навчитися використовувати множини при написанні програм; навчитися здійснювати обробку виключень (помилок) в Python.

Короткі теоретичні відомості

Множини

Мова програмування Python має засоби для роботи з множинами. У Python **множина** – це невпорядкована колекція елементів, яка має наступні особливості:

- усі елементи множини є унікальні. Іншими словами, у множині не може бути двох однакових елементів;
- елементи множини є незмінюваними об'єктами;
- елементи множини можуть бути різних типів.

У Python множина представлена окремим типом даних. Множини підтримують операції, що існують в теорії множин з курсу математики. Множина може бути пустою (не містити елементів).

Щоб створити об'єкт типу "множина" можна використати один з наступних способів:

- з допомогою присвоєння цьому об'єкту множини елементів фігурних дужках **{ }**;
- з допомогою присвоєння об'єкту деякого результату, що був повернутий функцією set(). Функція set() дозволяє перетворити в множину об'єкти інших типів;
- з допомогою присвоєння об'єкту результату функції frozenset(). Функція frozenset() є конструктором класу frozenset. Ця функція дозволяє отримати незмінну (immutable) множину;
- з допомогою присвоєння об'єкту результату деякої операції над множинами (наприклад, операції & перетину множин);
- з допомогою присвоєння об'єкту результату функції, яка повертає деяку множину.

Мова Python підтримує наступні операції над множинами:

in – перевірка елементу на входження в множину;

– (мінус) – різниця множин;

| – об'єднання множин;

& – перетин множин;

^ – симетрична різниця.

Методи для роботи з множинами в Python

Функції	**Опис**
add()	Додає елемент до множини.
all()	Повертає True, якщо всі елементи множини дорівнюють True (або якщо множина порожня).
any()	Повертає True, якщо хоч один з елементів множини дорівнює True. Якщо множина порожня, повертається False.
clear()	Видаляє всі елементи з множини.
copy()	Повертає копію множини.
difference()	Повертає різницю двох або більше множин у вигляді нової множини.
difference_update()	Видаляє всі елементи іншої множини з поточної множини.
discard()	Видаляє елемент із множини, якщо він є її частиною. (Нічого не робить, якщо елемент не знаходиться у множині)
enumerate()	Повертає об’єкт переліку, який містить індекс і значення для всіх елементів множини у вигляді пари.
intersection()	Повертає перетин двох множин у вигляді нової множини.
intersection_update()	Оновлює множину перетином себе та іншої множини.
isdisjoint()	Повертає True, якщо дві множини не мають перетину.
issubset()	Повертає True, якщо інша множина містить поточну множину.
issuperset()	Повертає True, якщо поточна множина містить іншу множину.
len()	Повертає довжину (кількість елементів) у множині.

max()	Повертає найбільший елемент у множині.
min()	Повертає найменший елемент у множині.
pop()	Видаляє та повертає довільний елемент множини. Викликає помилку KeyError, якщо множина порожня.
remove()	Видаляє елемент із множини. Якщо елемент не є членом множини, виникає помилка KeyError.
sorted()	Повертає новий відсортований список з елементів множини (але не виконує сортування).
sum()	Повертає суму всіх елементів множини.
symmetric_difference()	Повертає симетричну різницю двох множин у вигляді нової множини.
symmetric_difference_update()	Оновлює множину з симетричною різницею себе та іншої.
union()	Повертає об’єднання множин у новій множині.
update()	Оновлює множину з об’єднанням себе та інших.

Обробка помилок в Python

Існує три типи помилок, що виникають під час виконання програми:

- синтаксичні – це помилки, що породжуються неправильним використанням синтаксичних конструкцій мови програмування;
- помилки виконання (runtime error) – це помилки, що виникають у синтаксично правильних програмах під час їхнього виконання, коли інтерпретатор не може коректно розв’язати проблему, що виникла або подальше виконання програми є неможливим;
- логічні помилки – це помилки пов’язані з логікою програми. Інтерпретатор не зможе виявити і ідентифікувати такі помилки, оскільки з точки зору синтаксису програма написана правильно.

Якщо у програмі виникає помилка першого або другого типу, то будемо казати, що відбулася виключна ситуація. Будь-яка виключна ситуація генерує (також використовується термін породжує) виключення.

Виключення (eng: exception) – спеціальний тип даних, що використовується для ідентифікації та детального опису помилки, що виникла у програмі.

Слід зауважити, що виключення породжуються не лише у разі виникнення виключної ситуації, а і як повідомлення про те, що відбулася певна подія. Наприклад, метод списку pop() генерує виключення IndexError, якщо список порожній.

Для обробки виключень використовується оператор try. Його загальний синтаксис такий

try:
processing_code_with_potential_exception
except exception_1
handling_exception_1
...
except exception_N
handling_exception_N
else:
state_else
finally:
obligatory_code

Щоб перевірити стан програми (наприклад для того, щоб переконатися, що подальше її виконання має сенс) використовують інструкцію assert. Вона має такий синтаксис

assert condition, message

де condition – умова (логічний вираз), message – необов’язковий параметр, що є даними, що передаються на обробку при виникненні виключної ситуації. Інструкція **assert** генерує виключення типу AssertionError, якщо логічний вираз condition є хибним. Наприклад,

x = 0
try:

```
assert  x != 0
except AssertionError:
print("Ділити на 0 не можна")
else:
print(1 / x)
```

Завдання для роботи на практичному занятті

І. Використовуючи множини, скласти програму на мові Python

1. У бібліотеці є три відділи: художня література, наукова література та дитяча література. У художньому відділі: «Граф Монте-Крісто», «Віднесені вітром», «Три мушкетери», «Тарас Бульба». У науковому відділі: «Коротка історія часу», «Еволюція видів», «Історія математики», «Фізика майбутнього», «Тарас Бульба», «Чорна рада». У дитячому відділі: «Пригоди Тома Сойєра», «Матильда», «Три мушкетери», «Чарівник країни Оз». Знайти:

1) Книги, які є у всіх відділах.
2) Книги, які є тільки в художньому відділі.
3) Повний список книг у бібліотеці.

2. У Оленки є троє друзів-студентів: Андрій – вивчає математику, фізику, програмування, історію. Ольга – хімію, біологію, філологію, математику. Ігор – програмування, біологію, фізику, історію. Знайти:

1) Курси, які відвідують усі студенти.
2) Курси, які відвідує лише Ольга.
3) Курси, які вивчає Ігор, але не Андрій.

3. Є три сім'ї, які мають тварин: сім'я Петренків має кота, собаку, хом'яка, капібару, сім'я Іваненків – папугу, кота, рибок, єнота, сім'я Василькових – собаку, папугу, єнота, черепаху. Знайти:

1) Тварини, які є у всіх сім'ях.
2) Тварини, які є тільки у Петренків і нема у Василькових та Іваненків.

4. Користувач вводить текст. Необхідно:

1) Визначити кількість унікальних слів у тексті.
2) Вивести всі унікальні слова в алфавітному порядку.
3) Визначити, які слова зустрічаються більше одного разу.

5. Користувач вводить два числа. Потрібно:

1) Визначити, які цифри є спільними для обох чисел.
2) Визначити, які цифри є тільки у першому числі.
3) Визначити, які цифри є тільки у другому числі.

6. Є база паролів зареєстрованих користувачів.
Користувач вводить свій пароль. Потрібно перевірити: Чи є цей пароль у списку заборонених паролів (`password123`, `qwerty`, `123456`, `letmein`, `admin`)? Якщо пароль унікальний – додати його до списку.

7. Користувач вводить два слова. Потрібно:
1) Знайти спільні літери у словах.
2) Літери, які є тільки у першому слові.
3) Літери, які є тільки у другому слові.
4) Усі літери з обох слів.

8. Користувач вводить список чисел. Потрібно:
1) Вивести всі унікальні числа.
2) Визначити, чи є серед них парні.
3) Визначити, які числа відсутні у діапазоні від 1 до максимального числа у списку.

9. Є список можливих слів {«яблуко», «груша», «банан», «персик», «слива»}. Користувач вводить слово. Якщо слово є у списку – вивести "Вірно!". Якщо ні – додати його до списку.

10. Програма випадково генерує множину з 5 чисел (від 1 до 20). Користувач вводить свої 5 чисел, а програма каже, скільки з них є в загаданій множині.
Наприклад:
Загадані числа: {3, 7, 10, 15, 18}
Ваші числа: 2 3 10 19 20
Вгадані числа: {3, 10}

II. Скласти програму на мові Python для розв'язання наступних завдань, використовуючи обробку виключень

1. Користувач вводить два числа. Програма повинна виконати ділення першого числа на друге. Якщо друге число **0**, вивести повідомлення `"Помилка: ділення на нуль!"`. Інакше вивести результат. (*Підказка:* Використовуйте `try-except ZeroDivisionError`.)

2. Користувач вводить число. Якщо введене значення не є числом, вивести `"Помилка: введено не число!"`. Програма повторює запит, поки користувач не введе правильне число. (*Підказка:* Використовуйте `try-except ValueError`.)

3. Користувач вводить два числа та обирає операцію (+, −, *, /). Якщо користувач вводить некоректне число, вивести `"Помилка: введено не число!"`. Якщо ділить на 0, вивести `"Помилка: ділення на нуль!"`. Якщо введено неправильну операцію, вивести `"Помилка: невідома операція!"`. (*Підказка:* Використовуйте `try-except` для `ValueError`, `ZeroDivisionError`, `Exception`.)

4. Є список: `numbers = [10, 20, 30, 40, 50]`. Користувач вводить індекс елемента, який хоче отримати. Якщо індекс виходить за межі списку, вивести `"Помилка: індекс виходить за межі списку!"`. Інакше вивести значення за цим індексом. (*Підказка:* Використовуйте `try-except IndexError`.)

5. Є словник студентів: `students = {"Анна": 19, "Богдан": 20, "Катя": 18}`. Користувач вводить ім’я студента. Якщо ім'я є в словнику, вивести вік. Якщо немає – `"Помилка: студент не знайдений!"`. (*Підказка*: Використовуйте `try-except KeyError`.)

6. Користувач вводить список чисел через пробіл, наприклад: 10 20 30 56 abc 50. Програма має перетворити введені значення у `int` та порахувати їхню суму. Якщо знайдено нечислове значення, ігноруйте його та продовжуйте роботу. (*Підказка:* Використовуйте `try-except ValueError` у циклі.)

Запитання для повторення:

1. *Що таке множина у Python?*
2. *Які способи створення множини?*
3. *Які операції над множинами можливі у Python?*
4. *Чим відрізняються множини від інших колекцій?*

5. *Які дії над множинами дозволені?*
6. *Що таке виключення?*
7. *Які типи помилок виникають під час виконання програми?*
8. *Який оператор використовується для обробки виключень*? *Його загальний синтаксис*?
9. *Для чого і як використовують інструкцію assert?*