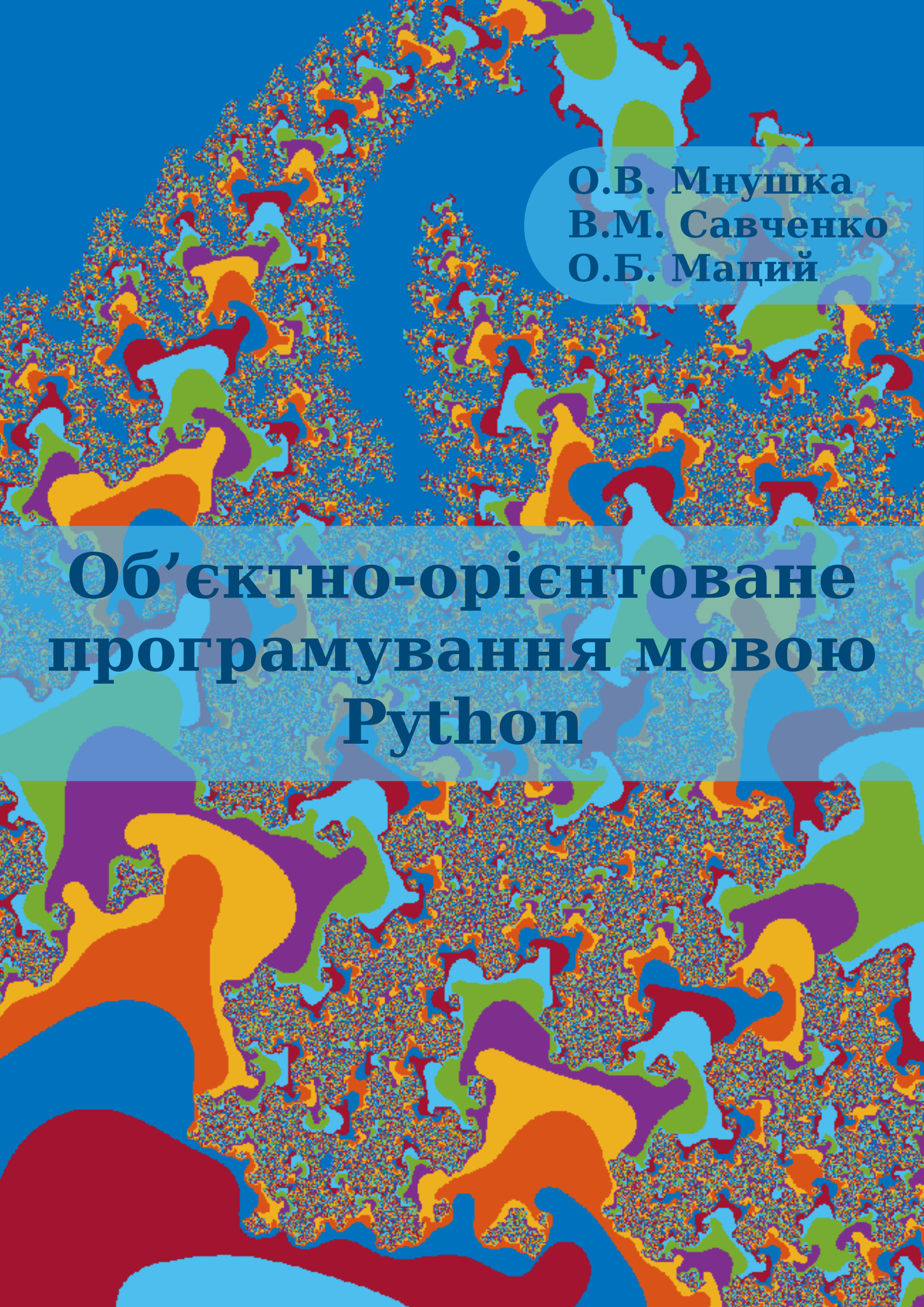

О.В. Мнушка
В.М. Савченко
О.Б. Маций

Об’єктно-орієнтоване програмування мовою Python

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ
АВТОМОБІЛЬНО-ДОРОЖНІЙ УНІВЕРСИТЕТ

Об’єктно-орієнтоване програмування мовою Python

для студентів напрямів підготовки
121 Інженерія програмного забезпечення та
122 Комп’ютерні науки

Харків
ХНАДУ
2021

Рекомендовано до видання рішенням Вченої ради Харківського національного автомобільно-дорожнього університету, протокол № 31/20/5.7 від 18 грудня 2020 р.

Рецензенти:

Нємченко К. Е. – д-р. фіз-мат. наук, професор, завідувач кафедри інформаційних технологій в фізико-енергетичних системах Харківського національного університету ім. В. Н. Каразіна;

Леонов С. Ю. – д. т. н., професор, професор кафедри обчислювальної техніки та програмування Національного технічного університету «ХПІ»;

Пічугіна О. С. – д-р. фіз-мат. наук, доцент, доцент кафедри математичного моделювання та штучного інтелекту Національного аерокосмічного університету ім. М. Є. Жуковського «ХАІ»;

Ніконов О. Я. – Лауреат премії президента України, д. т. н., професор, завідувач кафедри комп'ютерних технологій і мехатроніки Харківського національного автомобільно-дорожнього університету.

Колектив авторів:
Мнушка О.В., ст. викладач,
Савченко В.М., к. т. н.,
Маций О.Б., к. т. н.

Мнушка О.В.
Об’єктно-орієнтоване програмування мовою Python: навчальний посібник для студентів напрямів підготовки 122 Комп’ютерні науки та 121 Інженерія програмного забезпечення / О.В. Мнушка, В.М. Савченко, О.Б. Маций – Х.: ХНАДУ, 2021. – 228 с.

ISBN 978-617-7912-88-9

Розглянуто питання розробки програмного забезпечення у відповідності до принципів об’єктно-орієнтованого дизайну та реалізації програм за технологією об’єктно-орієнтованого програмування. Описані принципи S.O.L.I.D та їх вплив на побудову якісних програмних модулів. Розглянуто поняття та принципи об’єктно-орієнтованого програмування. Проаналізовано проблеми розробки класів об’єктів, побудови їх ієрархій, питання множинного успадковування у сучасних мовах програмування та ін. З точки зору практичного застосування, розглянуті та проаналізовані питання реалізації об’єктно-орієнтованих програм мовою програмування Python 3.

Посібник може бути корисним для студентів, викладачів, науковців та аспірантів вищих навчальних закладів, а також для всіх, хто цікавиться вивченням об’єктно-орієнтованого програмування та мови програмування Python 3.

УДК 004.42

ВСТУП

Об’єктно-орієнтоване програмування (ООП) – популярна методологія розробки програмного забезпечення, в якій програма є сукупністю *об’єктів*, кожний із яких є *екземпляром* деякого *класу*. Об’єкти взаємодіють один із одним для реалізації логіки додатку. У методології ООП розробник мислить категоріями предметної галузі та використовує їх у своїй програмі.

У посібнику для реалізації прикладів використано мову програмування Python[1], тому що це одна із популярних мов програмування, що підтримує усі основні концепції ООП. Python має простий синтаксис і використовується практично на будь-яких обчислювальних платформах та у різних операційних системах.

Для користуванням цим посібником потрібно знати та розуміти основи процедурного програмування будь-якою мовою програмування. Python відноситься до мов із динамічною типізацією, тому в програмах немає оголошення типів змінних, а їх тип визначається із контексту виразу[2].

Особливістю Python є вимоги дотримання прийнятого форматування коду за допомогою відступів, які використовуються для виділення тіла функцій та операторів. Але це не є проблемою при використанні сучасних середовищ розробки, які самі слідкують за дотриманням стилю.

Далі проведемо короткий огляд основ програмування мовою Python 3.

---

[1] Якщо не указане інше, мається на увазі Python 3

[2] А втім, у Python можна вказувати тип змінних у вигляді анотації (https://www.python.org/dev/peps/pep-0484/)

Тема 1. Основи програмування мовою Python

Метою є ознайомлення із основами програмування мовою Python у обсязі, достатньому для подальшої роботи над посібником.

Структура програми

Python-програма є текстовим файлом (скриптом) із послідовністю інструкцій *інтерпретатору мови Python* та може містити наступні елементи (рис. 1.1):

- посилання на файл інтерпретатора у першому рядку програми у вигляді `#!/usr/bin/python`, т. зв. shebang[3];
- коментарі, що починаються із символу «#»;
- секцій _ у вигляді
  - `import module` – імпорт всіх змінних, функцій та класів модулю;
  - `from module import something` – імпорт окремої змінної, функції або класу;
- оголошення та реалізацію класів;

```
1  #!/usr/bin/python3
2  # Приклад програми
3  import typing
4  from logging import Logger
5  MODULE_LEVEL_CONSTANT = 'Constanta'
6
7
8  def fibo(n):
9      '''
10     Рекурсивна функція обчислення чиел Фібоначчі
11     '''
12     if n == 0: return 0
13     if n == 1 or n == 2: return 1
14     return fibo(n-1) + fibo(n-2)
15
16
17 if __name__ == '__main__':
18     n = int(input('Введіть число більше 1: '))
19     print(fibo(n))
```

Рис. 1.1. Програма на Python 3

Програму (рис. 1.1) можна виконати, запустивши інтерпретатор

[3] Shebang (Unix) (https://en.wikipedia.org/wiki/Shebang_%28Unix%29)

Python 3[4] (рис. 1.2) або ipython[5].

Рекомендуємо користуватися текстовими редакторами на кшталт Visual Studio Code, Sublime Text 3, також можна використовувати інтегровані середовища розробника Eclipse або Microsoft Visual Studio Community Edition 2019 («Додаток А. Програмування мовою Python у Microsoft Visual Studio»).

Існує велика кількість онлайнових інтерпретаторів, на таких як: *python.org*, *pythonanywhere.com*, *repl.it*, *live.sympy.org*, *codepad.org* та інших, в яких можна реалізувати більшість прикладів із посібника.

```
Python 3.8.6 (default, Sep 25 2020, 09:36:53)
[GCC 10.2.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> import typing
>>> from logging import Logger
>>> MODULE_LEVEL_CONSTANT = 'Constanta'
>>> def fibo(n):
...     '''
...     Рекурсивна функція обчислення чиел Фібоначчі
...     '''
...     if n == 0: return 0
...     if n == 1 or n == 2: return 1
...     return fibo(n-1) + fibo(n-2)
...
>>> n = int(input('Введіть число більше 1: '))
Введіть число більше 1: 5
>>> print(fibo(n))
5
>>>
```

Рис. 1.2. Робота із інтерпретатором Python 3

Операційні системи Linux/Unix, як правило, мають установлений інтерпретатор python2/python3 у системі, для ОС Windows потрібно додаткове встановлення відповідних додатків.

Уведення/виведення даних у консольних програмах

Для введення даних використовують функцію `input()`, якій у якості параметра можна передати рядок тексту. Результатом її роботи є рядок символів, зчитаних із стандартного пристрою введення – консолі (рис. 1.3).

---

[4] Існує дві версії мови – Python 2 та Python 3. Розробку та підтримку мови Python 2 було припинено 1 січня 2020 року

[5] Інтерактивний Python (ipython) є альтернативною оболонкою з розширеними можливостями (https://uk.wikipedia.org/wiki/IPython)

```
Python 3.8.6 (default, Sep 25 2020, 09:36:53)
Type 'copyright', 'credits' or 'license' for more information
IPython 7.19.0 -- An enhanced Interactive Python. Type '?' for help.

In [1]: n = input('Enter n: ')
Enter n: 10

In [2]: type(n)
Out[2]: str
```

Рис. 1.3. Уведення даних

Для виведення даних у консоль використовують функцію `print()`, яка приймає рядок символів. Ця функція забезпечує достатньо складне форматування тексту (рис. 1.4) та вміє виводити на екран вміст стандартних контейнерів без використання циклів (рис. 1.5).

```
In [3]: print(n)
10

In [4]: print("n = ", n)
n =  10

In [5]: print("n = {}".format(n))
n = 10

In [6]: print(f"n = {n}")
n = 10

In [7]: print("n = {n}, n^2 = {n*n}")
n = {n}, n^2 = {n*n}

In [8]: print(f"n = {n}, n^2 = {n*n}")
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-8-03faa5f5232e> in <module>
----> 1 print(f"n = {n}, n^2 = {n*n}")

TypeError: can't multiply sequence by non-int of type 'str'
```

Рис. 1.4. Виведення даних

Під час виконання останнього оператора виникла помилка, зумовлена забороною множення двох рядків.

У процесі налагодження програм часто стикаються із подібними повідомленнями про помилки – *виняткові ситуації*, що призводить до розкручування стеку викликів інтерпретатором з метою пошуку хибної інструкції («Тема 9. Виняткові ситуації»).

У Python довідка про будь-які об’єкти формується на основі спеціального коментаря або *docstring* на початку модуля (функції).

Коментар у потрійних лапках `'''docstring'''` містить опис

класу або функції (рис. 1.6). Використовують різні стилі написання docstring[6]: *pep257*, *Google*, *numpy*, *sphinx* й т. п. Коментарі у вигляді docstring використовуються системою формування автоматичних підказок *IntelliSense* та їй подібних.

```
In [1]: _list = ["one", "two", "thre"]

In [2]: _dict = {"1": "one", "2": "two"}

In [3]: _tuple = (1, "shell", ["dog", "cat"])

In [4]: _set = {1, 3, 3, 4, 5, 6, 5}

In [5]: print(_list)
['one', 'two', 'thre']

In [6]: print(_dict)
{'1': 'one', '2': 'two'}

In [7]: print(_tuple)
(1, 'shell', ['dog', 'cat'])

In [8]: print(_set)
{1, 3, 4, 5, 6}
```

Рис. 1.5. Виведення вмісту контейнерів

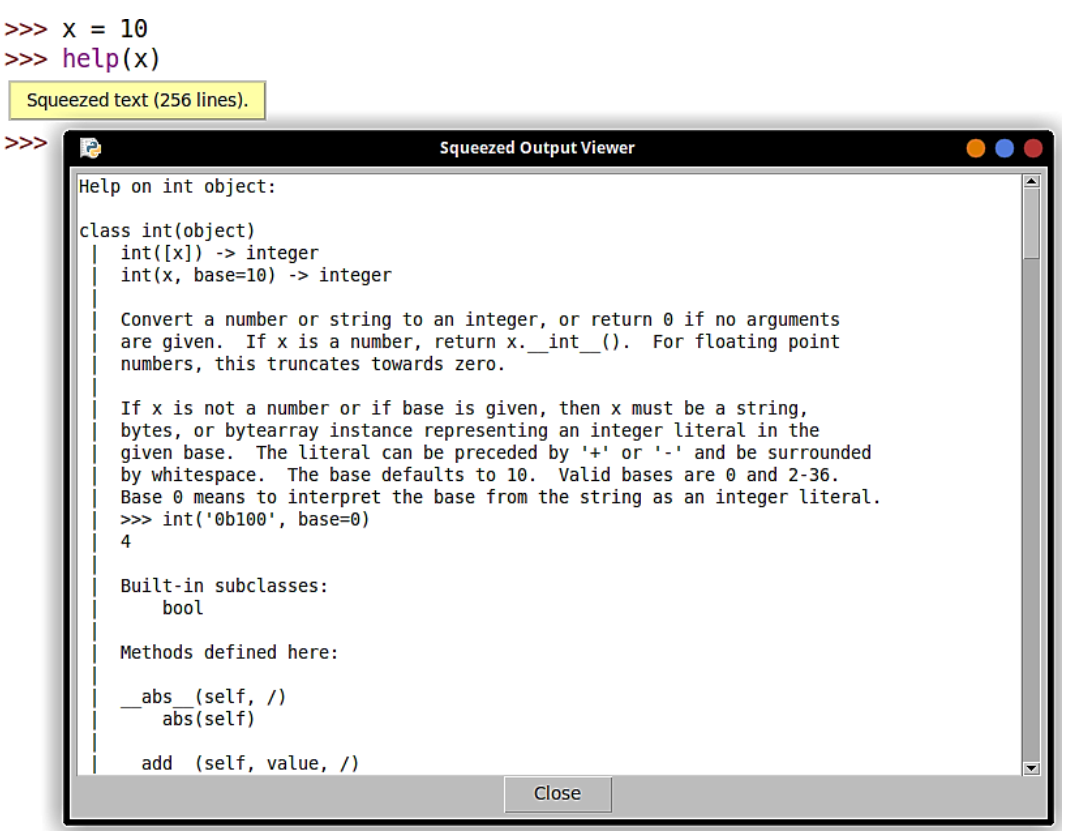

Рис. 1.6. Довідка про об'єкт

[6] PEP 257 – Docstring Conventions (https://www.python.org/dev/peps/pep-0257/)

Типи даних та стандартні контейнери

Python є мовою із динамічною типізацію («Тема 6. Поліморфізм»), а втім, потрібно розуміти тип даних або результату.

Числа й результати обчислень можуть бути:

- `int` (*цілі*) – `1, -300, 123456, 100**100, 25 // 7`;
- `float` (*дійсні*) – `1.23, 25 / 7, sin(x)`[7];
- `complex` (*комплексні*) – `1 + 2j, (3 - 4j) * (2 + 8j)`;
- `bool` (*логічні* підтип `int`) – `True, False, 2 > 3, -5, 0`.

Основні операції над числами: додавання (+), віднімання (-), множення (*), ділення (/), цілочисельне ділення (//), піднесення до степеня (**), ділення за модулем або залишок (%)[8].

Цікавою особливістю цілих чисел є те, що вони можуть бути дуже великими, фактично їх розмір обмежений оперативною пам'яттю комп'ютера[9].

Існує чотири типи *стандартних контейнерів*, що містять у якості елементів певну кількість значень однакових або різних типів:

- `list` (список, `[ ]`) містить елементи *однакових* чи *різних* типів даних – числа, рядки, списки, словники. Доступ до елементу списку здійснюють за допомогою індексації та зрізів, дозволено індексувати список у зворотному порядку за допомогою від'ємних індексів;
- `dict` (словник, `{ }`) містить пари «ключ: значення», доступ до значень здійснюють за допомогою ключів, як ключ використовують будь-які гешовані типи даних;
- `tuple` (кортеж, `( )`) містить елементи різних типів, як і список, але вміст кортежу неможливо змінити після створення.
- `set` (множина, `{ }`) містить невпорядковані дані різних типів в одному екземплярі..

Список є змінюваним (*mutable*) об'єктом, можна додавати чи видаляти елементи, а також змінювати значення елементів (рис. 1.7).

Список використовують як аналог одновимірного масиву у C/C++/C#, або у якості заміни для масиву масивів змінної довжини.

---

[7] Модуль math містить математичні функції та константи, див help(math)

[8] Чисельні методи: numpy (https://numpy.org/) і scipy (https://www.scipy.org/)

[9] Super Long Integers in Python (https://arpitbhayani.me/blogs/super-long-integers)

```
>>> ll = [1,2,3,4,5,6,7] # список
>>> ll[0] # перший елемент
1
>>> ll[-1] # останній елемент
7
>>> ll[0::2] # зріз з кроком 2
[1, 3, 5, 7]
>>> ll[-1::-2] # зріз з кроком 2
[7, 5, 3, 1]
>>> ll[0:6:2] # зріз з кроком 2
[1, 3, 5]
>>> [el * el for el in ll] # генерація нового списку
[1, 4, 9, 16, 25, 36, 49]
>>> ll.extend([7,6,5]) # розширення списку
>>> ll
[1, 2, 3, 4, 5, 6, 7, 7, 6, 5]
>>> ll.append([-1,-2,3]) # додавання елементу
>>> ll
[1, 2, 3, 4, 5, 6, 7, 7, 6, 5, [-1, -2, 3]]
>>> ll.remove(3) # видалення елементу
>>> ll
[1, 2, 4, 5, 6, 7, 7, 6, 5, [-1, -2, 3]]
>>> len(ll) # розмір списку
10
>>> ll.count(7) # кількість входжень
2
```

Рис. 1.7. Робота зі списком

Кортеж є аналогом незмінюваного (*immutable*) списку (рис. 1.8).

```
>>> tt = (1,2,3,4,5,6,7) # кортеж
>>> print(tt)
(1, 2, 3, 4, 5, 6, 7)
>>> tt[0] ; tt[-1] # перший та останній елементи
1
7
>>> tt[0::3] # зріз з кроком 3
(1, 4, 7)
>>> [el * el for el in tt] # генерація списку
[1, 4, 9, 16, 25, 36, 49]
>>> tt == (1,2,3,4,5,6,7)
True
>>> len(tt) # розмір
7
>>> tt.count(7) # кількість входжень
1
```

Рис. 1.8. Робота зі кортежем

Словники у Python є асоціативним масивом на основі геш-таблиць. Словники у якості ключа використовують значення будь-якого незмінюваного типу даних (рис. 1.9). Зверніть увагу, за відсутності ключа у словнику, він може автоматично додатися до словника `dd["1"] = "ONE"` (рис. 1.9) або виникне виняткова ситуація.

Для безпечного доступу до ключів використовують метод `get(key, "value if key not exist")`.

```
>>> dd = {1:"one", 2:"two", 3:"three", 4:"four"} # словник
>>> dd = {1:"one", 2:"two", 3:"three", 4:"four"} ; dd # словник
{1: 'one', 2: 'two', 3: 'three', 4: 'four'}
>>> dd[1]; dd[4] # доступ за ключем
'one'
'four'
>>> dd.keys() # ключі
dict_keys([1, 2, 3, 4])
>>> dd.values() # значення
dict_values(['one', 'two', 'three', 'four'])
>>> dd["1"] = "ONE" ; dd # додати значення
{1: 'one', 2: 'two', 3: 'three', 4: 'four', '1': 'ONE'}
>>> dd[1] = "ONE_ONE" ; dd # змінити значення
{1: 'ONE_ONE', 2: 'two', 3: 'three', 4: 'four', '1': 'ONE'}
>>> dd.get(8, "not avalable") # доступ до елементу
'not avalable'
>>> dd.get(3, "not avalable") # доступ до елементу
'three'
>>> dd.update({8:"eight"}) ; dd # додати елемент
{1: 'ONE_ONE', 2: 'two', 3: 'three', 4: 'four', '1': 'ONE', 8: 'eight'}
>>> dd.update({4:"eight"}) ; dd # замінити елемент
{1: 'ONE_ONE', 2: 'two', 3: 'three', 4: 'eight', '1': 'ONE', 8: 'eight'}
```

Рис. 1.9. Робота зі словником

Множина є змінюваним[10] типом даних, множина містить тільки унікальні елементи *незмінюваних* типів. Допускається у якості елементів одночасно мати різні типи даних Цей тип даних зручно використовувати для видалення дублікатів із інших контейнерів (рис. 1.10).

Основні прийоми роботи із множинами представлені на рис. 1.11, бачимо, що для множин підтримуються всі типи операцій, відомі із курсу «Дискретна математика».

```
>>> ll = [1,2,3,4,4,5,6,7,8,8,9,1,2]
>>> ll = set(ll)
>>> ll
{1, 2, 3, 4, 5, 6, 7, 8, 9}
```

Рис. 1.10. Видалення дублікатів із списку

[10] Є також незмінюваний варіант множини – frozenset (https://docs.python.org/3/library/stdtypes.html?highlight=frozenset#frozenset)

```
>>> st = {1,2,3,4,4,5,4,3,1} ; st # множина
{1, 2, 3, 4, 5}
>>> 7 in st; 1 in st ## ; перевірка на приналежність
False
True
>>> st.add(10); st # додати елемент
{1, 2, 3, 4, 5, 10}
>>> st.discard(10); st # видалити елемент
{1, 2, 3, 4, 5}
>>> st2 = {3,4,5} ; st2 # нова множинина
{3, 4, 5}
>>> st == st2 # множини містять однакові елементи
False
>>> st >= st2 # st надмножина st2
True
>>> st <= st2 # st підмножина st2
False
>>> st | st2  # об'єдання
{1, 2, 3, 4, 5}
>>> st & st2  # перетин
{3, 4, 5}
>>> st - st2 ; st2 - st # різниця
{1, 2}
set()
>>> st ^ st2 ; st2 ^ st # симетрична різниця
{1, 2}
{1, 2}
```

Рис. 1.11. Робота із множиною

Рядки є незмінюваним типом даних. Над рядками допускаються стандартні операції конкатенації, пошуку, заміни (рис. 1.12). У рядку заборонена заміна елемента за індексом. В результаті заміни символів у рядку продукується новий рядок.

У стандартній бібліотеці є модуль `collections`, який містить наступні класи контейнерів:

- `namedtuple` – фабрична функція для створення кортежів (або незмінюваних класів) із іменованими елементами;
- `deque` – двобічна черга;
- `ChainMap` – об’єднує декілька словників в єдине відображення;
- `Counter` – контейнер для підрахунку гешованих об’єктів;
- `OrderedDict` – упорядкований словник, елементи зберігаються у порядку додавання;
- `defaultdict` – словник, який при зверненні до неіснуючого ключа створює його.
- `UserDict`, `UserList`, `UserString` – базові класи для підтримки відповідних класів користувача за допомогою успадковування.

```
>>> str1 = "Hello"; str2 = "World"; str1; str2 # рядки
'Hello'
'World'
>>> txt = str1 + ', ' + str2 + '!' ; txt # конкатенація
'Hello, World!'
>>> txt.replace('World', 'User') # заміна
'Hello, User!'
>>> print(txt) # !!!
Hello, World!
>>> txt[0:] ; txt[-9::-1] # індексація та зрізи
'Hello, World!'
'olleH'
>>> str1 * 3 # повторення
'HelloHelloHello'
>>> str1 += str2 ; str1 # новий рядок, як конкатенація str1 та str2
'HelloWorld'
>>> str1.find("World"); str1.find("User")  # пошук
5
-1
>>> "_".join(list('Hello'))
'H_e_l_l_o'
```

Рис. 1.12. Робота із рядками

Основи роботи із файлами

Умовно всі файли можна розділити на текстові та двійкові. У текстовий файл пишуться рядки символів, а у двійкові – послідовності байтів. Допустимі наступні режими відкриття файлу (табл. 1).

Таблиця 1.1

Режими відкриття

`'r'`	відкриття для читання
`'w'`	відкриття для запису, файл створюється або перезаписується
`'x'`	відкриття на запис тільки нового файлу
`'a'`	відкриття на додавання
`'b'`	відкриття у двійковому режимі
`'t'`	відкриття у текстовому режимі
`'+'`	відкриття на запис і читання

Далі розглянуто основні функції та приклади роботи із текстовим файлом (рис.1.13-1.15). Файл зберігається у каталозі із програмою або у домашньому каталозі користувача. Обробка файлів – джерело потенційних виняткових ситуацій, про що потрібно пам’ятати.

```
# запис у файл
txt = "Hello, world"
with open("myfile.txt", 'w') as fl:
    fl.write(txt + "\n")  # запис у файл
    fl.write(txt[::-1] + "\n")
    fl.writelines(txt + '\n' + txt)

#зчитування із файлу
with open("myfile.txt", 'r') as fl:
   text = fl.readlines()  # зчитування всього файлу у список
   fl.seek(0, 0)  # перейти на початок файлу
   fl.readline()  # пропустити рядок
   line = fl.readline().strip()  # зчитати рядок, видалити переведення рядку
   ch = fl.read(5) #зчитування 5 символів
print("text:>  ", text)
print("line:>  ", line)
print("chars:> ", ch)
```

Рис. 1.13. Робота із текстовим файлом

```
> myfile.txt
Hello, world
dlrow ,olleH
Hello, world
Hello, world
```

Рис. 1.14. Вміст текстового файлу

```
text:>   ['Hello, world\n', 'dlrow ,olleH\n', 'Hello, world\n', 'Hello, world']
line:>   dlrow ,olleH
chars:>  Hello
```

Рис. 15. Результат виконання програми

Робота із двійковим файлом суттєво не відрізняється, послідовно у циклі зчитуються `read()` або записуються `write()` порції даних (байти), а для переміщення файловим потоком використовують функції `seek()` та `tell()`. Інтерпретація зчитаних байтів здійснюється програмістом.

Керуючі конструкції

Існують наступні типи керуючих конструкцій:

- умовний оператор `if … elif … else …`;
- умовний тернарний оператор … `if … else …`;
- цикл `while … else …`;

- цикл `for … in … else…`;

Блоки операторів у керуючих конструкціях виділяються, а після відповідної умови завжди ставиться двокрапка «:». (рис. 1.16).

Для пропуску ітерації використовують continue, а для переривання циклу – `break`. Блок else в операторах циклу виконується тільки у випадку виходу із циклу без допомоги `break`.

```
# умовний оператор
x = 10; y =20
if x == y:
    print("==")
elif x <= y:
    print("<=")
else:
    print(">")

# цикл while
while y:
    print(y, end="")
    y -= 1
print()
```

```
# тернарний оператор
print( "<" if x < y else ">=")

# цикл for
sum = 0
for val in range(-10,10):
    sum += val
print(f"Sum = {sum}")
```

Рис. 1.16. Приклади використання керуючих конструкцій

Функції

Визначення функції починається із ключового слова `def` за яким записують ім’я функції та параметри у круглих дужках. Всі параметри у функцію *передаються за значенням*, тобто – копією параметра. Для змінюваних параметрів використовують `list` або інший змінюваний контейнер. Стандартні параметри `*args` – кортеж неіменованих аргументів, `**kwargs` – словник іменованих аргументів[11].

Для повернення результату або виходу із функції використовують інструкцію `return`. Функція, яка не повертає результат, повертає `None`, що відноситься до `NoneType` та є аналогом `null`, `nil` у інших мовах програмування.

Функції у Python є *об’єктами першого класу*, про особливості їх

[11] *args and **kwargs in Python (https://www.geeksforgeeks.org/args-kwargs-python/)

використання поговоримо, коли будемо розбиратися із роботою декораторів.

```
def customize(a, b, *args, **kwargs):
    """
    Перетворює степінь числа на рядок

    Args:
        a (int): ціле число
        b (int): показник степеню

    Returns:
        str: a ** b із __ між цифрами
    """
    return '__'.join(list(str(a ** b)))
```

Рис. 1.17. Визначення функції

Резюме

Розглянуто основи програмування мовою Python 3, в т. ч. структуру програми, основні типами даних, керуючі конструкції та функції. Для кожного типу даних є певна кількість визначених функцій (методів), ознайомитися з якими можна у документації на сайті Python. Роботу із рядками та винятками буде розглянуто більш докладно у наступних темах.

Представленого матеріалу достатньо для розуміння синтаксису мови та прикладів, представлених у наступних темах.

Стандартна бібліотека Python 3 містить функції майже на всі стандартні випадки, а для специфічних задач є додаткові бібліотеки та фреймворки, тому рекомендуємо ознайомитися із керуванням пакетами за допомогою утиліти pip[12].

Основним джерелом інформації з Python є офіційний сайт https://docs.python.org/3/, додаткову інформацію можна знайти у рекомендованій літератури [1-14] та в Інтернеті [15-18].

[12] pip – PyPy, (https://pypi.org/project/pip/)

Список літератури

1. Объектно-ориентированный анализ и проектирование с примерами приложений, 3-е изд.: Пер. с англ. / Буч Г. и др. – М.: 000 «И. Д. Вильямс», 2008. – 720 с.

2. Лутц М. Изучаем Python, том 1, 5-е изд.: Пер. с англ. – СПб.: ООО «Диалектика», 2019. – 832 с.

3. Лутц М. Изучаем Python, том 2, 5-е изд. : Пер. с англ. – СПб. : ООО «Диалектика», 2020. – 720 с.

4. Хеллман Д. Стандартная библиотека Python 3: справочник с примерами, 2-е изд. : Пер. с англ. – СПб. : ООО «Диалектика», 2019. — 1376 с.

5. Хайнеман Д., Поллис Г., Сеяков С. Алгоритмы. Справочник с примерами на C, C++, Java и Python, 2-е изд.: Пер. с англ. – СПб.: ООО «Альфа-книга», 2017. – 432 с.

6. Гамма Э, Хелм Р, Джонсон Р, Влиссидес Д. Приемы объектно-ориентированного программирования. Паттерны проектирования. – СПб: Питер, 2010. – 368 с.

7. Мнушка О.В., Савченко В.М. Конспект лекцій з дисципліни «Об’єктно-орієнтоване програмування для студентів напрямів підготовки 121 «Інженерія програмного забезпечення» та 122 «Комп’ютерні науки». – Харків, ХНАДУ, 2020.

8. Мнушка О.В., Савченко В.М. Методичні вказівки для проведення практичних робіт з дисципліни «Об’єктно-орієнтоване програмування» для студентів напрямів підготовки 121 «Інженерія програмного забезпечення» та 122 «Комп’ютерні науки». – Харків, ХНАДУ, 2020.

9. Мнушка О.В., Савченко В.М. Методичні вказівки для самостійної роботи з дисципліни «Об’єктно-орієнтоване програмування» для студентів напрямів підготовки 121 «Інженерія програмного забезпечення» та 122 «Комп’ютерні науки». – Харків, ХНАДУ, 2020.

10. Мартинс Р. Чистый код: создание, анализ и рефакторинг. Библиотека программиста. – СПб: Питер, 2019. – 464 с.

11. Кормен Т. Х. Алгоритмы: вводный курс. Пер. с англ. – М. : ООО «И.Д. Вильямс», 2014. – 208 с.

12. Дейтел П., Дейтел Х. Python: Искусственный интеллект, большие данные и облачные вычисления. – СПб.: Питер, 2020. – 864 с.

13. Седер Н. Python. Экспресс-курс. 3-е изд. – СПб.: Питер, 2019. – 480 с.

14. Прохоренок Н. А., Дронов В.А. Python 3 и PyQt 5. Разработка приложений. – 2-е изд., перераб. и доп.. – СПб.: БХВ-Петербург, 2018. – 832 с.

15. Data model. – URL: https://docs.python.org/3.8/reference/datamodel.html

16. PEP 0 – Index of Python Enhancement Proposals (PEPs).– URL: https://www.python.org/dev/peps/

17. Links to Python related information in Ukranian. – URL: https://wiki.python.org/moin/UkranianLanguage

18. Stackoverflow. – URL:https://stackoverflow.com/

Зміст

Навчальне видання

Оксана Василівна МНУШКА
Володимир Миколайович САВЧЕНКО
Ольга Борисівна МАЦИЙ

ОБ’ЄКТНО-ОРІЄНТОВАНЕ ПРОГРАМУВАННЯ

Навчальний посібник

Відповідальний за випуск проф. *О.Я. Ніконов*

В авторський редакції