ЗМІСТ

ПЕРЕДМОВА

Ідея навчання на задачах не нова. Вона успішно застосовується при навчанні дисциплін, для яких написані збірники задач з розв'язаннями. З програмування, особливо для початківців, такої літератури дуже мало. Виходять, в основному, книги-довідники, які описують синтаксис того чи іншого середовища програмування, його оператори та структури даних. Застосування мовних конструкцій в них ілюструються лише найпростішими прикладами. Такі книги, безумовно, потрібні, але практичний досвід викладання переконує у необхідності та ефективності розв'язування задач.

При навчанні програмування необхідно вивчати і запам'ятовувати, розробляти і записувати (кодувати) алгоритми.

Не менш важливо отримати завершену програму, як і в математиці довести задачу до відповіді. "*Вважаю важливим показавши програму в остаточному вигляді, звертати увагу на деталі, оскільки саме в них криються труднощі програмування. Опис принципів самого алгоритму та його математичний аналіз можуть стимулювати "академічний розум" і кинути йому виклик, але це було б не чесно по відношенню до програміста-практика. Тому я суворо дотримуюся правила давати програми в остаточному вигляді тією мовою, якою вони дійсно можуть бути виконані на обчислювальній машині*" [12].

Викладачі погодяться з цією думкою і з тим, що не повинно бути пріоритету ідеї алгоритму чи налагодженої програми на початковій стадії навчання програмування.

Скрипти в посібнику написані на Python (перевірені у версії Python 3.12).

Популярність мови зростає тривалий час серед професійних програмістів і вона *все ширше використовується у навчальних процесах шкіл і вишів*. Українські школярі за окремими підручниками розпочинають знайомство з нею у 5 класі. Один із найкращих у світі технічних університетів – Массачусетський технологічний інститут (MTI) активно використовує Python як у навчальному процесі так і в наукових дослідженнях.

У сучасному світі технологій вміння програмувати є одним з ключових. І Python, як універсальна мова програмування, відкриває широкий спектр можливостей.

Python відомий своєю простотою, що робить його ідеальним вибором для тих, хто тільки починає свій шлях у програмуванні; мова відома простотою, читабельністю та широкими можливостями; Python – проста і гнучка мова тощо. Подібні твердження містяться в усіх передмовах до книжок і матеріалах мережі про Python.

Лаконічність і читабельність – так! Але варто наголосити, що код, створений засобами мови надвисокого рівня, не *може бути простим завжди* (!). Напевно, правильніше вважати, що *Python володіє чудовим балансом між складністю і синтаксисом*.

У великих проєктах *для спрощення коду* окрім фізичного розділення на частини (функції, класи, модулі, пакети) застосовуються різнорівневі засоби абстракції – типи, спеціалізований синтаксис, менеджери контексту, асинхронне та метапрограмування. Для написання (і навіть розуміння) відповідного об'ємного коду пам'ять програміста повинна зберігати абстракції і велику кількість понять різного змісту і обсягу. Відомо, що пам'ять людини обмежена (пригадується гаманець Міллєра, правило 7 ± 2).

У підсумку, все ж таки, виходить складний код.

З досвіду викладання: власне сприйняття абстракцій мови можна перевірити на найпростішій синтаксичній абстракції – the walrus operator, якому у посібнику приділена достатня увага; власну думку радимо формувати опрацьовуючи гарно структуровану фірмову документацію, посібники, відеоуроки і, звичайно, створюючи скрипти; розуміння мови сформується лише після набуття *значного досвіду* програму-вання, участі в онлайн-форумах, конференціях, відкритих проєктах.

Посібник містить більше 250 завершених програм, написаних у процедурному стилі.

Увага приділяється вбудованим структурам даних, алгоритмічним структурам, рекурсії, алгоритмам і лаконічності їх реалізації. Доступно і чітко викладений матеріал охоплює основні теми для ***правильного вступу в світ алгоритмів і програм***.

Глави 4-6 присвячені алгоритмічним структурам – слідування, розгалуження, цикл; глава 7 – функціям і рекурсії, глава 8 – ітераторам і генераторам. У главі 10 розглянуті масиви і ключові пакети для наукових обчислень та візуалізації даних – numpy і matplotlib; у главі 11 – текстові та двійкові файли; у главі 12 – рядкові методи в задачах.

Велику увагу приділено графічним побудовам (глава 9), що обумовлено динамічним розвитком комп'ютерних інформаційно-графічних технологій – інтерфейси опера-ційних систем, гіпертекстовий формат вебсторінок, нові можливості, які створює штучний інтелект (генеративні моделі, реалістична візуалізація, розпізнавання об'єктів, реконструкція 3D моделей, відеоігри тощо). Навчання комп'ютерної графіки не слід ототожнювати з вивченням і застосуванням графічних редакторів. Значно важливіше моделювати графічні об'єкти, демонструючи і візуалізуючи важливі поняття і прийоми програмування – вибір структур даних, подання і кодування даних, вкладені цикли, параметри функцій.

Окремі задачі глави 13 пропонувались на олімпіадах з програмування різних рівнів, співбесідах. Вони можуть бути використані для підготовки учнів та студентів до олімпіад з програмування і майбутніх співбесід при працевлаштуванні.

Виконуючи тестові вправи, можна оцінити рівень володіння мовою.

Наступні кроки у бік поглибленого програмування – це вичерпне вивчення алгоритмів і структур даних, розділів дискретної математики, лінійного, асинхронного та мета-програмування, конструювання даних з динамічною структурою, знайомство з особливостями операційних систем, опанування *принципів та техніки об'єктно-орієнтованого програмування,* необхідність у використанні яких виникає при розробці складних і об'ємних проєктів.

Хоча посібник є практикумом, у ньому не вдалось уникнути великої кількості детальних пояснень важливих понять та методів, наприклад, структур даних (глава 3), на які "багатий" Python, і, відповідно, прикладів-ілюстрацій, таблиць. Адже це потужне професійне середовище з величезною кількістю відкритих бібліотек і більшість глав можна розгорнути в окремий посібник, описуючи, наприклад, багаті можливості тієї чи іншої бібліотеки, структури даних, оброблення файлів різних форматів тощо.

Заради економії місця на сторінках книги правила гайду по стилях коду (PEP 8) дотримуються не завжди (модулі імпортуються не в окремих рядках, мінімалістичний код не завжди документується, після двокрапки зустрічається код і т.п.).

Книга розрахована на ***старшокласників, просунутих початківців, слухачів курсів, студентів і викладачів.***

Вона з успіхом може використовуватись для самостійного навчання і, сподіваємось, виявиться корисною для викладачів, які проводять практичні заняття.

Сподіваємось також, що посібник стане надійним помічником у вивченні чудової мови і відкриє двері у світ програмування.

Python – особлива, своєрідна і гармонійна мова. Отже, її вивчення – вибір сучасний, раціональний і перспективний.

Бажаємо успіхів у навчанні та натхнення у створенні програм!

ГЛАВА 1. IDLE. ОБ’ЄКТИ. ТИПИ ДАНИХ

IDLE: Shell Window, Edit Window. Інтерактивний режим.
Імпорт модулів. Об’єкти. Величини. Адресація об’єктів у пам’яті.
Категорії операторів. Оператори як функції.
Прості та складені типи даних. Перетворення типів.

Python – мультипарадигменна, інтерпретована, об’єктно-орієнтована, універсальна мова програмування високого рівня з динамічною семантикою. Підтримує модулі та пакети модулів. Парадигми програмування: *імперативне* (процедурне), *декларативне* (логічне, функціональне), *об’єктно-орієнтоване*.

Програма (скрипт) на Python – послідовність операторів (команд, інструкцій), записаних у текстовому файлі з розширенням ру.

За роки існування Python накопичилася величезна кількість відкритих бібліотек, використання яких значно прискорює розроблення програм.

Велика і дружня спільнота розробників сприяє зростанню популярності мови.

◈ IDLE

Integrated **D**evelopment and **L**earning **E**nvironment – інтегроване середовище для написання програм. Встановлюється разом з інтерпретатором Python і є альтернативою командному рядку. Запускає програму на виконання лише після її збереження. Існують інші середовища для написання і налагодження коду.

Інтерактивний режим

Приклади цього розділу записані й виконані в Shell Window (Options/Configure/IDLE/Windows/Open Shell Window або Run/Python Shell). Такий режим називають ***інтерактивним*** або *режимом калькулятора*.

Режим зручний на етапі вивчення синтаксису мови, для отримання довідки, різних обчислень; опрацювання операторів, методів, типів, структур даних тощо. Декілька інструкцій в одному рядку розділяються знаком крапка з комою (**;**).

Команди, записані справа від знаку **>>>**, виконуються після натискання Enter, результати виводяться у наступному рядку.

```
>>> (7+2/3)*3 - 25          >>> 119e-14 * 17e+14     >>> abs(-10-3)
    -2.0                         2023                     13
>>> 13_703_703_579/111      >>> (3+2j)*(3-2j)         >>> 4**300>3**400
    123456789.0                  (13+0j)                  False
```

Обчислені: значення числового виразу; добуток чисел, записаних в експоненційному форматі, модуль різниці; частка двох чисел, добуток спряжених комплексних чисел. У нерівності порівнюються значення степенів. *Довга арифметика* вбудована в мову. Це набір алгоритмів для виконання дій з цілими числами довільної довжини (значення степенів у прикладі мають довжини 181 і 191).

Shell Window **Інтерактивне вікно** (режим калькулятора)	Edit Window **Вікно редагування** (режим програмування)
Виконання команд (>>> – підказка; Ctrl + space – список, Alt + N, Alt + P – наступна, попередня)	Робота з програмним кодом (написання, редагування, зберігання, виконання)
Довідка, списки атрибутів: **>>> import math; dir(math)** **>>> help(math)** **>>> from math import sin, floor** **>>> help(sin); help(floor)** **>>> from winsound import Beep** **>>> help(Beep)** **>>> dir(list)** **['__len__', 'append', 'clear', 'copy', 'count', 'extend', 'index', 'insert', 'pop', …]**	Програма (скрипт): **from math import cos** **def Bisect(a, b, eps, f):** ** while b - a > 2*eps:** ** c = (a + b)/2** ** if f(a)*f(c)<=0:** ** b = c** ** else: a = c** ** return (a + b)/2** **print(Bisect(0, 1, 0.000001, lambda x: x - cos(x)))**
Відображення результатів	File / Save As (Ctrl + Shift + S)
0.7390851974487305	New File: Ctrl + N; Open: Ctrl + O; Close Window: Alt + F4; Exit: Ctrl + Q

```
>>> a, b, c = -3, -12, 7
>>> min(a,b,c), max(a,b,c), sum((a,b,c))        #(-12, 7, -8)
>>> bin(21), oct(21), hex(21), int('0b10101',2)
    ('0b10101', '0o25', '0x15', 21)
>>> chr(65), chr(90), ord('A'), ord('Z')
    ('A', 'Z', 65, 90)
>>> divmod(42,5), divmod(-42,5), divmod(-42.0,-5.0)
    ((8, 2), (-9, 3), (8.0, -2.0))
>>> ~5, 5|13, 5&13, 5^13, 13>>1, 13<<1
    (-6, 13, 5, 8, 6, 26)
>>> 145*23 + 13*2023          >>>'Допоміжна '
    29634                        'Допоміжна '
>>> _ - 9634                  >>> _ + 'змінна'
    20000                        'Допоміжна змінна'
```

Використані окремі ***вбудовані функції.*** Коли інструкції дозволяється розділяти комою, повертаються *кортежі-результати*.

У прикладах: знаходиться найбільше і найменше з трьох чисел; число 21 перетворюється у системи числення з основами 2, 8, 16 та отримується за двійковим кодом; взаємно обернені функції виводять символи за кодами і навпаки; функція **`divmod`** повертає неповну частку і остачу при цілочисельному діленні; виконані логічні операції (детальніше про них далі); в останньому прикладі змінна _ (одне підкреслення – underscore) *зберігає останній результат* і фактично є вбудованою допоміжною змінною.

Складніші інструкції, наприклад цикл, в Shell Window також виконуються:

```
>>> codes = 176,178,179,449,945,8707,8709,8776
>>> for x in codes: print(chr(x), end = '  ')
... ° ² ³ ǁ α ∃ ∅ ≈
```

Для написання програм використовуватимемо основний режим Edit Window, який дає змогу редагувати, зберігати і завантажувати файли.

◈ Імпорт модулів

Практично всі програми на Python імпортують вбудовані модулі. Користувацькі модулі можуть створюватись додатково і об’єднуватись в пакети (глава 11).

```
>>> from winsound import Beep
    Beep(777, 77)                  #частота, тривілість

>>> from math import e, pi, sin
>>> e**pi > pi**e, 3*sin(pi/2) == 9**0.5
   (True, True)                      #булівський кортеж

>>> from math import hypot
>>> hypot(3,4), hypot(5,12), hypot(8,15),hypot(7,24)
   (5.0, 13.0, 17.0, 25.0)     #базові Піфагорові трійки

>>> from fractions import Fraction as F
>>> F(1,3)-F(4,7), F(1,3)/F(4,7)    #арифметичні дії над
   (Fraction(-5,21), Fraction(7,12)) #звичайними дробами

>>> from array import array
>>> from random import randrange
>>> #масив з 7 випадкових двозначних чисел
>>> array('i',(randrange(11,99) for k in range(7)))
>>> array('i', [86, 22, 51, 20, 68, 22, 38])

>>> import timeit
>>> def y(): L = [i**2 for i in range(1_000_000)]
...
>>> print(timeit.timeit("y()", setup = \
... "from __main__ import y", number = 10))
    1.9967969999997877
```

У прикладах: імпортована функція **Beep** генерує звуковий сигнал заданої частоти і тривалості; **pi**, **e**, **sin** використовуються у виразах, булівські значення яких зберігаються в кортежі; гіпотенузи базових трійок Піфагора обчислюються після імпорту **hypot**; арифметичні дії над звичайними дробами можливі після імпорту **Fraction** (перейменування **as F** для скорочення записів); модулі **array** і **randrange** імпортовані для створення масивів; **timeit** – для визначення тривалості виконання коду (функції **y**).

```
>>> import math; dir(math)
[ …'acos', 'asin', 'atan', 'ceil', 'copysign', 'cos', 'dist',
'degrees', 'e', 'exp', 'fabs', 'factorial', 'floor', 'gcd',
'hypot', 'log', 'log10', 'log2', 'nan', 'perm', 'pi', 'pow', … ]
```

Команди **import math; dir(math)** ***виводять список об'єктів модуля*** (список – скорочений).

```
>>> from random import sample; help(sample)
Help on method sample in module random:
sample(population, k, *, counts=None) method of random. Random
instance Chooses k unique random elements from a sequence.
```

Команда **help(sample)** після імпорту з модуля **random** ***виводить довідку*** (скорочена) для функції **sample.**

◈ Об'єкти, величини

У Python змінні, функції, класи – **об'єкти** – абстракції даних. Для доступу до них у програмах використовують **імена** (*ідентифікатори*). *Оператор* (команда, інструкція) *присвоювання* **a = 7** пов'язує довільно іменований об'єкт, розміщений в пам'яті, з посиланням на нього, адресою (уявляємо **a** як мітку, бирку).

```
>>> a = 7; id(a); id(7)   #140708184445928  140708184445928
```

Програмуючи, вважатимемо, що змінній **a** при *ініціалізації* надається значення **7** – числова (скалярна) ***величина***, тип якої визначається автоматично,

Типізація в Python динамічна, неявна і строга за сумісністю типів.

Величина – одне з фундаментальних понять математики. Величини класифікують як: а) скалярні/векторні; б) співрозмірні/неспіврозмірні; в) порядкові/непорядкові; г) пов'язані рівняннями зв'язку/непов'язані; д) одного роду/різного роду; е) сталі/змінні тощо.

У програмуванні найважливіші дві останні класифікації.

Сталі величини (константи) не змінюють присвоєне їм значення; *змінні величини* можуть змінювати його у процесі виконання коду. Іменовані константи у середовищі окремо не виділяються і тому позначаються аналогічно до змінних.

Унікальні імена змінних не повинні збігатися із зарезервованими словами (keywords), з ідентифікаторами (identifiers) вбудованих класів, функцій, модулів тощо. Великі та малі літери в іменах розрізняються (!).

Отже, основні характеристики змінних величин: ***ім'я, тип, адреса, значення***.