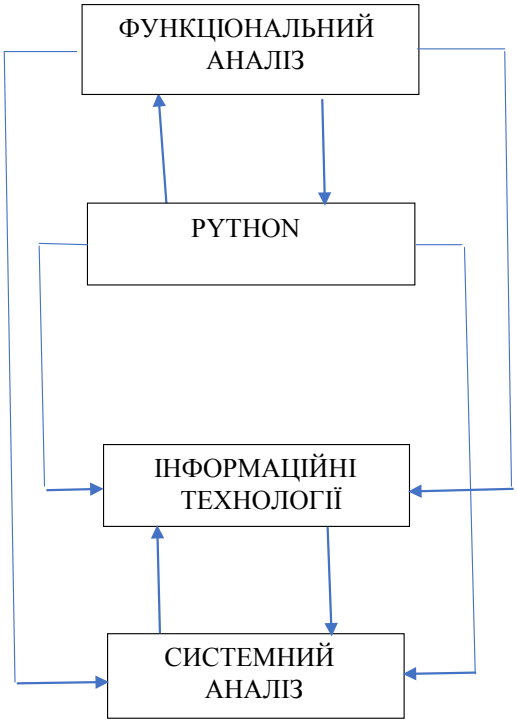
**Б. І. МОКІН, В. Б. МОКІН, О.Б. МОКІН**

# НАВЧАЛЬНИЙ ПОСІБНИК

## для опанування студентами способів розв'язання задач з функціонального аналізу мовою Python

**Частина 1**




Замовити книжку: https://press.vntu.edu.ua/index.php/vntu/catalog/book/689

Міністерство освіти і науки України
Вінницький національний технічний університет

Б. І. Мокін, В. Б. Мокін, О. Б. Мокін

# НАВЧАЛЬНИЙ ПОСІБНИК
## для опанування студентами способів розв'язання задач з функціонального аналізу мовою Python

Частина 1

Вінниця
ВНТУ
2022

**УДК 517.98**
**М74**

Рекомендовано до друку Вченою радою Вінницького національного технічного університету Міністерства освіти і науки України як навчальний посібник для студентів закладів вищої освіти, що спеціалізуються в галузі інформаційних технологій
(протокол № 7 від «31» березня 2022 р.)

**Рецензенти:**
**В. Я. Данилов**, доктор технічних наук, професор (НТУУ «КПІ ім. Сікорського»)
**В. І. Клочко**, доктор педагогічних наук, професор (ВНТУ)
**О. С. Макаренко**, доктор фіз.-мат наук, професор (НТУУ «КПІ ім. Сікорського»)

**Мокін, Б. І.**

М 74 Навчальний посібник для опанування студентами способів розв'язання задач з функціонального аналізу мовою Python. Частина 1 / Б. І. Мокін, В. Б. Мокін, О. Б. Мокін. – Вінниця : ВНТУ, 2022. – 124 с.

ISBN 978-966-641-892-3

В навчальному посібнику викладено способи розв'язання задач з функціонального аналізу, адаптованого до прикладних проблем в галузі інформаційних технологій, у відповідності зі змістом однойменного навчального посібника цих же авторів, а також викладені основи програмування мовою Python і програми реалізації способів розв'язання даного класу задач цією мовою. Частина 1 охоплює задачі з теорії множин, метричних просторів, теорії міри, інтегралів Рімана, Стілтьєса та Лебега, а також задачі з дослідження функціоналів на екстремум.

Навчальний посібник рекомендується для студентів та аспірантів, що спеціалізуються в ІТ-галузі за спеціальностями 124 – «Системний аналіз» та 126 – «Інформаційні системи та технології»

**УДК 517.98**

**ISBN 978-966-641-892-3**

**ЗМІСТ**

# ВСТУП

У 2020 році ми опублікували навчальний посібник «Функціональний аналіз, адаптований до прикладних задач в галузі інформаційних технологій» [1], з використанням якого студенти, що навчаються у ВНТУ за спеціальністю 124 – «Системний аналіз», опановують навчальну дисципліну «Функціональний аналіз», яка, згідно з освітньо-професійною програмою цієї спеціальності входить до переліку обов'язкових.

Уже після першого року використання в навчальному процесі нашого навчального посібника з функціонального аналізу ми зрозуміли, що студентам важко засвоювати практичні навички з розв'язання прикладних задач з цієї навчальної дисципліни, користуючись лише навчальним посібником, в якому викладені її теоретичні основи, особливо ж в умовах реалізації навчального процесу в онлайн-режимі, зумовленому карантинними обмеженнями щодо перебування студентів групами в аудиторіях, викликаними світовою пандемією коронавірусної хвороби. А тому в нашому авторському колективі виникла задумка написати з функціонального аналізу ще один навчальний посібник, в якому викласти практичні способи розв'язання задач з функціонального аналізу, характерних для ІТ-галузі, аби цей другий навчальний посібник доповнив перший практикою застосування теоретичних основ функціонального аналізу, викладених у першому навчальному посібнику, при розв'язанні прикладних задач, характерних для ІТ-галузі. Особливо важливим аспектом цього другого навчального посібника, на наш погляд, має стати доведення етапу розв'язання прикладних задач функціонального аналізу до застосування сучасної мови програмування Python, усі дистрибутиви якої викладені в інтернеті для вільного доступу у вигляді програмного середовища Anaconda [2]. Саме ця задумка і реалізована нами у навчальному посібнику, зміст якого подається нижче. І оскільки цей навчальний посібник покликаний доповнювати практикою попередній, присвячений теорії функціонального аналізу, то і розділи його збігаються за назвами і змістом з відповідними розділами свого теоретичного попередника.

Характерною особливістю нашого навчального посібника, зміст якого подається нижче, є те, що ми спочатку демонструємо способи і алгоритми «ручного» розв'язання прикладних задач функціонального аналізу на конкретних простих прикладах, а уже потім показуємо як ці задачі розв'язувати за допомогою мови Python з використанням дистрибутивів і підпрограм, що допускають їх комп'ютерну реалізацію в консолі Spyder.

Такий підхід обумовлений тим, що, як показала практика роботи зі студентами під час вивчення функціонального аналізу, в разі, якщо для розв'язання задачі використати готову программу, написану мовою Python, то студенти, використовуючи цю програму, отримують розв'язок задачі, не розуміючи суті закладених в програму алгоритмів, що, по-перше, надовго не залишається в пам'яті, а по-друге, не сприяє осмисленому застосуванню цих же алгоритмів при розв'язанні інших задач, умови яких в чомусь відрізняються.

Але, оскільки у цьому навчальному посібнику ми використовуватимемо мову програмування Python, то спочатку, з посиланням на популярний [3] та науковий [4] варіанти її викладення, здійснимо екскурс в історію створення цієї мови та в загальних рисах розглянемо основні дистрибутиви, що входять до її структури.

І почнемо зі звернення уваги на те, що розробники мови Python, метою яких було створення безкоштовної, доступної для всіх, мови програмування високого рівня, назвали цю мову не в честь змії і читати цю назву потрібно не «пітон», тобто не в прямому перекладі літер англійської мови, а читати назву мови потрібно «Пайтон», бо назвали розробники її в честь циркового коміка Пайтона, який створив популярне у 70-х роках минулого сторіччя як в Англії, так і у Британії в цілому, комедійне шоу «Летючий цирк Монті Пайтона», яке дуже подобалось розробникам нової мови програмування. Але, щоб задовольнити і тих, кому

подобалось вкладати в слово Python зміст слова «пітон», усі програми, які створювались цією мовою уже в 64-бітовому варіанті, їх розробники розмістили і продовжують розміщувати в програмному середовищі, названому на честь найдовшої у світі змії Anaconda.

Тим нашим читачам, яких цікавить лише використання програм мовою Python для розв'язання прикладних задач, ми радимо детально ознайомитись з навчальним посібником [4], в якому викладено основи мови Python та приклади її застосування для розв'язання наукових задач, а тим, кого цікавить і технологія створення програм мовою Python, варто ознайомитись ще й з навчальним посібником [3], в якому ця технологія у спрощеному вигляді з використанням інтегрованого 32-бітового середовища IDLE викладена так, щоб бути зрозумілою і дітям шкільного віку. Але одразу ж попереджаємо, що в пакет прикладних програм Anaconda, для яких використовується виключно 64-бітове середовище, 32-бітове середовище IDLE не входить, тож IDLE в Anakonda шукати не варто, але його можна викликати за окремим посиланням [5].

Ми згадали про інтегроване середовище IDLE для повноти історичної довідки, але користуватись ним у цьому навчальному посібнику не будемо, адже 32-бітове інтегроване програмне середовище, яким є IDLE, нині вже неактуальне, тому в подальшому на нього посилатись не будемо, а сконцентруємо увагу на 64-бітовому інтегрованому середовищі Anaconda, на яке ви можете зустріти посилання в різних літературних джерелах і як на дистрибутив, і як на пакет прикладних програм (ППП) мовою Python. Встановлювати на своєму комп'ютері дистрибутив Anaconda вигідно ще й тому, що одночасно і автоматично на вашому комп'ютері встановлюється і бібліотека пакетів прикладних програм, які використовуються для наукових досліджень і носять назви: numpy,sympy, scipy, matplotlib.

ППП numpy (або NumPy – числовий Пайтон) – це пакет програм, який використовується для числових розрахунків.

ППП sympy (або SymPy – символьний Пайтон) – це пакет програм, який використовується для всіляких перетворень виразів, заданих у символьній формі (зокрема і для взяття похідних та невизначених інтегралів від складних функцій).

ППП scipy ( або SciPy – науковий Пайтон) – це пакет програм, який зручно застосовувати при обробленні результатів наукових досліджень у сукупності з пакетами sympy та numpy.

ППП matplotlib – це пакет програм для одно-, дво- і тривимірних графічних зображень результатів розрахунків, виконаних з застосуванням пакетів numpy, sympy та scipy, тобто, це графічний редактор, пристосований до роботи з програмами, написаними мовою Python.

Зміст усіх цих ППП та порядок їх застосування ми будемо розкривати в процесі їх використання для розв'язання задач функціонального аналізу, а у цій вступній частині навчального посібника згадаємо ще лише про те, що реалізовувати ці ППП в разі, якщо ви встановили на своєму комп'ютері дистрибутив Anaconda, можна за допомогою інтерпретаторів (або, що одне і те ж, консолей) IPython, IPython Notebook, який ще називають Jupyter Notebook, та Spyder, головне вікно останнього з яких на екрані розділене на три частини, причому ліва частина призначена для набору і редагування програм у вигляді файлів, які можна або виконувати або відправляти в пам'ять з можливістю виклику в подальшому, права нижня частина призначена для набору програм за допомогою командного рядка, їх тестування та виведення результатів їх дії і результатів тестування на екран, а права верхня частина призначена для виведення на екран рисунків, що супроводжують обчислення.

Ми в нашому навчальному посібнику будемо використовувати саме консоль Spyder, назва якої є абревіатурою від Scientific PYthon Development EnviRonment, що перекладається як «Наукового Пайтону середовище розвитку», і яка є найбільш узагальненою, оскільки інтегрує в собі основні риси інших двох інтерпретаторів, згаданих вище.

На цьому ми вступну частину нашого навчального посібника закінчуємо і приступаємо до конкретизації його змісту.

## Розділ 1 МНОЖИНИ І МЕТРИЧНІ ПРОСТОРИ ТА ЇХ ХАРАКТЕРИСТИКИ (В ПРИКЛАДАХ І ПРОГРАМАХ)

### 1.1 Множини та їх характеристики

У кінці однойменного підрозділу у базовому навчальному посібнику [1] серед інших стосовно множин сформульовані і такі запитання.

1. Який зміст вкладається в поняття «множина» та які основні характеристики множин і їх приклади ви знаєте?
2. Що собою являють сума, переріз і різниця множин?

Тож із відповідей на ці запитання ми і розпочнемо викладення змісту нашого «Навчального посібника для опанування студентами способів розв'язання задач з функціонального аналізу мовою Python»

Що стосується першого питання, то його розкриття почнемо з повторення означення множини та її характеристик, які дані в роботі [1].

Отже, ***множина*** - це сукупність об'єктів якоїсь природи, які прийнято називати елементами. ***Множина*** вважається заданою, якщо є відомими всі її елементи та правило, згідно з яким ці елементи відносять до цієї ***множини. Множина*** зі скінченною кількістю елементів називається ***скінченною множиною,*** а ***множина*** з нескінченною кількістю елементів називається ***нескінченною множиною***. Прикладом ***скінченної множини*** є ***множина*** автомобілів, зареєстрованих в Україні, а прикладом ***нескінченної множини*** є ***множина*** дійсних чисел на відрізку [0,1] числової осі. ***Множина,*** яка не містить жодного елемента, називається ***пустою.*** Якщо ***множини*** $A$ **і** $B$ складаються з однакових елементів, то вони вважаються ***рівними*** між собою, про що свідчить запис $A = B$ . Якщо ж у ***множину*** $A$ входять не всі елементи ***множини*** $B$ , то ***множину*** $A$ називають ***підмножиною множини*** $B$, про що свідчить запис $A \subset B$ . Наприклад, на числовій осі ***множина*** раціональних чисел $R$ є ***підмножиною множини*** дійсних чисел $Z$, а ***множина*** автомобілів марки «Хонда» є ***підмножиною множини*** автомобілів, зареєстрованих в Україні.

Важливою характеристикою ***множин*** є їх ***еквівалентність***, згідно з якою ***множини*** $A$, $B$ вважаються ***еквівалентними,*** якщо кожному елементу $a \in A$ за якимось, заздалегідь обумовленим правилом, ставиться у відповідність один-єдиний елемент $b \in B$ , а кожному елементу $b \in B$ ставиться у відповідність один-єдиний елемент $a \in A$. Наприклад, еквівалентними є ***множина*** $A$ легкових автомобілів приватної власності, кожний з яких зареєстровано лише на одного власника у певному населеному пункті, і ***множина*** $B$ людей, які є власниками цих автомобілів. Правилом, за яким встановлюється ***еквівалентність*** цих множин, є запис органом реєстрації автомобілів прізвища власника в паспорті автомобіля.

А для того, щоб порівнювати між собою ***нееквівалентні множини***, вводиться поняття їх ***потужності***, яку будемо встановлювати за чимось спільним, що має місце в усіх ***множинах, еквівалентних*** тій, яку ми розглядаємо. Очевидно, що спільним у ***скінченних множинах*** різної природи є лише кількість їх елементів, а тому, якщо ***множина*** $A$ ***має*** $n$ елементів, а ***множина*** $B$ ***має*** $m$ елементів і при цьому $n > m$, то ми констатуємо, що ***множина*** $A$ ***має потужність більшу, ніж множина*** $B$**.** Але виникає запитання: «А як порівнювати між собою за ***потужністю нескінченні множини***, кожна з яких має нескінченну кількість елементів?».

Досліджуючи на числовій осі різні нескінченні числові послідовності, математики встановили, що з усіх нескінченних послідовностей найшвидше наближається до нескінченності ***натуральний ряд*** $N$, оскільки кожне його наступне число дорівнює попередньому, збільшеному на одиницю, і при цьому при формуванні цього ряду

пропускаються всі дійсні числа, які містяться на числовій осі у кожній такій одиниці. А тому домовились вважати ***натуральний ряд***, який являє собою нескінченний ряд чисел, ***нескінченною множиною найменшої потужності***, яку домовились позначати символічно малою латинською літерою ***a,*** і всі інші ***нескінченні множини*** порівнювати між собою, виходячи з того, як вони співвідносяться за ***потужністю*** з ***потужністю натурального ряду***, а всі ***нескінченні множини*** з ***потужністю натурального ряду*** називати ***зліченними множинами***, оскільки кожному їх елементу можна приписати індекс, який дорівнює відповідному числу натурального ряду, за рахунок чого кожен їх елемент можна ***полічити.***

І першим фактом, який встановили математики після цієї домовленості стосовно ***потужності натурального ряду***, є те, що ***потужність множини Z дійсних чисел, заданих на відрізку [0,1], є більшою за a .***

***Потужність нескінченної множини дійсних чисел на відрізку [0,1]*** математики ***вирішили назвати потужністю континууму***, а символічно ***позначати малою латинською літерою c,*** отже, справедливою є нерівність **c > a.** Більше того, математики встановили, що ***для потужностей*** **c, a** ***справедливою є рівність*** $c = 2^a$ . А оскільки ***множина дійсних чисел Z на відрізку [0, 1]*** числової осі, є сумою ***підмножини R раціональних чисел та підмножини ірраціональних чисел***, заданих на цьому ж відрізку, а кожне раціональне число можна подати у вигляді відношення двох цілих чисел, які є елементами натурального ряду, який є зліченною множиною потужності ***a*** , то ці числа і в чисельнику, і в знаменнику можна полічити, а тому ***підмножина раціональних чисел на відрізку [0, 1] числової осі теж є зліченною множиною потужності a .*** А з цього факту витікає ***два наслідки***, ***перший*** з яких засвідчує, що ***підмножина ірраціональних чисел на вказаному відрізку є нескінченною множиною потужності континууму с ,*** бо лише за рахунок цієї підмножини множина дійсних чисел на вказаному відрізку матиме потужність **c,** а ***другим наслідком є твердження, що в разі додавання до множини потужності континууму будь-якої зліченної підмножини потужність їх суми залишається рівною с*** . Тож на основі цього твердження можна зробити ***важливий висновок, що і вся вісь дійсних чисел є множиною потужності континууму с .***

А тепер у такому ж сконцентрованому варіанті, скориставшись теоретичним матеріалом, наведеним у нашому навчальному посібнику [1], дамо відповідь на друге з наведених вище запитань.

При об'єднанні двох ***множин*** *A* і *B* утворюється нова ***множина*** *M*, яку називають їх ***сумою*** і яка містить у собі всі елементи обох цих ***множин***, причому кожен однаковий елемент обох ***множин*** в їх ***суму*** *M* входить як один елемент, символічно записується це так:

$$M = A \cup B. \tag{1.1}$$

Наприклад, якщо *A* і *B* – це числові ***множини***,

де $$A = \{1,2,3,4,5\},\ B = \{4,5,6,7,8\}, \tag{1.2}$$

то, згідно з (1.1), матимемо

$$M = A \cup B = \{1,2,3,4,5\} \cup \{4,5,6,7,8\} = \{1,2,3,4,5,6,7,8\} \tag{1.3}$$

При перетині двох ***множин*** *A* і *B* утворюється нова ***множина*** *P*, яку називають їх ***перерізом*** і яка містить у собі лише ті елементи обох цих ***множин***, які є однаковими, причому кожен із цих однакових елементів обох ***множин*** в їх ***переріз*** *P* входить як один елемент, символічно записується це так:

$$P = A \cap B. \tag{1.4}$$

Для наведених в умовах попереднього прикладу числових ***множин*** (1.2), згідно з (1.4), матимемо

$$P = A \cap B = \{1,2,3,4,5\} \cap \{4,5,6,7,8\} = \{4,5\}. \tag{1.5}$$

***А множина*** $Q$, яка складається з елементів ***множини*** $A$, що не входять у ***множину*** $B$, називається ***різницею*** цих ***множин*** і позначається як $A - B$ або $A \setminus B$ , тобто,

$$Q = A - B = A \setminus B. \tag{1.6}$$

Цілком очевидно, що в загальному випадку

$$\mathrm{A} - \mathrm{B} \neq \mathrm{B} - \mathrm{A}. \tag{1.7}$$

Наприклад, для числових ***множин*** (1.2)

$$A - B = \{1,2,3\}, \tag{1.8}$$

$$B - A = \{6,7,8\}. \tag{1.9}$$

На завершення цього підрозділу розглянемо операцію над множинами, яку, як правило, не розглядають у класичному функціональному аналізі, а тому їй не знайшлося місця і у нашому навчальному посібнику [1], але яка теж може зустрітись в задачах системного аналізу, що використовують операції над множинами. Це ***операція об'єднання множин А та В, яку позначимо R, з вилученням спільних елементів***, тобто операція

$$R = \overline{A} \cup \overline{B}, \tag{1.10}$$

за умови, що

$$A \subset S, \quad B \subset S, \quad A \cup B = S \tag{1.11}$$

Виходячи з цих умов, для числових ***множин*** (1.2) матимемо

$$R = \overline{A} \cup \overline{B} = \{8,7,6,3,2,1\} \tag{1.12}$$

### 1.2 Початкові відомості з мови програмування Python, достатні для розв'язання задач, пов'язаних з операціями над множинами

Для того, щоб продемонструвати, як наведені у попередньому підрозділі операції над множинами виконувати ***в програмному середовищі Python***, спочатку, орієнтуючись на наведені в списку використаної літератури джерела [2]–[5], нагадаємо деякі початкові відомості з технології використання цієї мови програмування.

Отже, виходимо з того, що ви вже викликали на свій комп'ютер ***дистрибутив Anaconda*** і вибрали та встановили ***консоль Spyder***. В результаті цих дій ви вже матимете на своєму комп'ютері ***інтерактивну оболонку IPython***, в якій уже можна виконувати певні операції мовою Python і без виклику спеціалізованих ***пакетів*** програм типу: ***numpy, sympy, scipy, matplotlib.***

Після встановлення ***консолі Spyder екран*** вашого комп'ютера буде розділений ***на три частини, в лівій*** з яких ви зможете ***набирати*** потрібні вам програми ***мовою Python*** у вигляді цілісних ***файлів***, які, надавши їм назву і занісши в пам'ять, можна буде викликати з пам'яті за потреби їх використання. ***В правій нижній частині*** екранного ***вікна консолі Spyder*** ви зможете набирати потрібні вам ***програми в*** інтерактивній ***оболонці IPython***, використовуючи

***командні рядки***, кожен із яких починається символом ***In[m]*** де ***m*** – ***номер*** командного рядка, а ***перехід*** від поточного рядка до наступного здійснюється натискуванням ***клавіші Enter***. А ***в правій верхній частині*** цього екранного ***вікна*** будуть відображатись ***рисунки і графіки*** функцій, якими супроводжуватиметься виконання заданих вами програм, якщо ви програмно вказуватимете на потребу їх виведення на екран.

***В пояснювальному прикладі № 1***, поданому нижче, продемонстровано, як набирати команди в командних рядках для виконання операцій ***множення*** з символом «*», ***піднесення до степеня*** з символом «**», ***додавання*** з символом «+», ***віднімання*** з символом «-», ***ділення*** з символом «/», ***використання результату попередньої операції*** з символом «_» (одинарне підкреслення), ***використання результату операції, попередньої попередній*** з символом «__» (подвійне підкреслення). Звертаємо увагу на те, що ***після командного*** рядка під номером ***[m],*** в якому програмується безпосереднє виконання якоїсь операції, ***під символом «Out[m]» з'являється рядок***, в якому наводиться ***результат*** виконання цієї ***операції.*** В разі ж, якщо ви в командному рядку записали щось таке, що не відповідає позначенням чи змісту, прийнятному для мови Python, то в наступному рядку після даного, з'явиться англійською мовою текст роз'яснення суті допущеної вами помилки. Виправивши цю помилку, ви можете продовжити набирання подальших команд вашої програми. Звертаємо увагу на те, що, якщо в командному рядку подається лише одна вихідна величина з конкретизованим її значенням, то після неї не ставляться ніякі інші знаки, а якщо в одному командному рядку подається кілька вихідних величин з конкретизованими значеннями, то між ними ставиться знак «;» (кома з крапкою). Цей же знак ставиться і між символьно записаною операцією в командному рядку і символом вказаного у цьому ж рядку результату цієї операції. ***Звертаємо увагу*** також і на те, що, як і у роботі [4], при формуванні команд у кожній програмі мовою Python, для кращого розуміння суті кожної команди, ми у кожному ***командному рядку***, в якому записуватиметься одна команда, ***справа від цієї команди після умовного символа «#» розміщуватимемо пояснення*** дії, яка реалізується даною командою.

**Пояснювальний приклад № 1**

```
In[1]: x=2                          # Внесення змінної x та її значення
In[2]: y=4                          # Внесення змінної y та її значення
In[3]: z=x*y; z                     # Отримання добутку xy змінних
Out[3]: 8
In[4]: z1=x**y; z1                  # Піднесення змінної x до степеня y
Out[4]: 16
In[5]: z2=x**y-10; z2               # Обчислення виразу зі степенем і різницею
Out[5]: 6
In[6]: z3=_*2; z3                   # Перемноження на 2 попереднього числа
Out[6]: 12
In[7]: z4=__*3; z4                  # Перемноження на 3 числа з рядка [5]
Out[7]: 18
In[8]: z5=y**0.5; z5                # Обчислення кореня квадратного з y
Out[8]: 2.0
In[9]: z6=(x+y)*2; z6               # Перемноження на 2 суми x та y
Out[9]: 12
In[10]: z7=_/3; z7                  # Ділення на 3 попереднього числа
Out[10]: 4
```

**Кінець пояснювального прикладу № 1.**

Оскільки жодна програма не може виконуватись, якщо для запрограмованої задачі не задані вихідні дані, то зупинимось коротко на тому, яким чином задаються ***вихідні дані у програмах мовою Python.***

У цій мові вихідні дані ***можуть задаватись у вигляді стрічок (str), списків (list), кортежів (tuple) та словників (dict), елементами яких можуть бути літери певного алфавіту або слова, складені з цих літер, дійсні (float), комплексні (complex) або цілі (int) числа, а також логічні змінні (bool) зі змістом «Правда» (True), яка в числовому еквіваленті означає «1», та зі змістом «Неправда» (False), яка в числовому еквіваленті означає «0».***

***Стрічки (str)*** – наголошую, для програм мовою Python саме стрічки, а не рядки – ***утворюються взяттям їх елементів в одинарні (типу апострофа) або подвійні лапки***, як показано в пояснювальному прикладі № 2. Але, якщо ви наберете, наприклад, стрічку "Добрий день!" в рядку під номером [m], то після натискання клавіші Enter в наступному рядку під цим же номером буде надруковано Out[m]: 'Добрий день!' – тобто стрічка буде повторена, але з тим же номером і в одинарних лапках. Якщо ж ви хочете в наступному рядку вивести стрічку «в чистому вигляді», то її треба ***виводити на екран через операцію print( )*** – обидва ці варіанти виведення стрічки наведені в пояснювальному прикладі № 2 як з використанням одинарних, так і подвійних лапок. Відзначимо також, що ***індексування елементів в стрічці починається з нуля***, а за індексом елемента в стрічці, взятим в квадратні дужки, можна викликати його значення; що елементи в стрічці не дозволяється змінювати; що операцією «in» можна перевірити, чи даний елемент є в стрічці; що операцією «+» дві стрічки можна об'єднувати в одну (конкатенація); що за допомогою операції «*» стрічку можна повторити стільки разів, на скільки вказує число, яке стоїть перед символом цієї операції, утворивши у такий спосіб стрічку, в стільки ж разів довшу; що ***за допомогою функції len( ) можна підрахувати, скільки елементів входить в стрічку*** – усі ці особливості стрічок теж наведені в пояснювальному прикладі № 2.

**Пояснювальний приклад № 2**

```
In [1]: "Добрий день!"                  # Внесення стрічки у формі "."
Out[1]: 'Добрий день!'
In [2]: 'Добрий день!'                  # Внесення стрічки у формі ' '
Out[2]: 'Добрий день!'
In [3]: print("Добрий день!")           # Виведення чистої стрічки на екран
Добрий день!
In [4]: print('Добрий день!')          # Виведення чистої стрічки на екран
Добрий день!
In [5]: s1="Добрий день,"               # Внесення стрічки у формі s1
In [6]: s2='шановні студенти!'          # Внесення стрічки  у формі s2
In [7]: s1+s2                           # Формування суми стрічок s1 та s2
Out[7]: 'Добрий день, шановні студенти!'
In [8]: s3="Ура!"                       # Внесення стрічки у формі s3
In [9]: 4*s3                            # Стрічка з чотирьох s3
Out[9]: 'Ура!Ура!Ура!Ура!'
In [10]: "р" in s3                      # З'ясування чи є літера «р» в s3
Out[10]: True
In [11]: "к" in s2                      # З'ясування чи є літера «к» в s3
Out[11]: False
In [12]: s1[0]                          # Виклик із s1 елемента з індексом 0
Out[12]: 'Д'
In [13]: s1[2]                          # Виклик із s1 елемента з індексом 2
Out[13]: 'б'
In [14]: len(s1)                        # Визначення кількості членів в s1
Out[14]: 12
```

**Кінець пояснювального прикладу № 2.**

## СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Мокін Б. І., Мокін В. Б., Мокін О. Б. Функціональний аналіз, адаптований до прикладних задач в галузі інформаційних технологій : навчальний посібник. Вінниця : ВНТУ, 2020. 192 с.

2. Python. [Електронний ресурс]. Режим доступу : https://www.python.org//downloads//.

3. Бріггс Джейсон Р. Python для дітей (веселий вступ до програмування). / перекладачка з англійської Олександра Гордійчук. Львів : Видавництво старого Лева, 2019. 400 с.

4. Доля П. Г. Введение в научный Python. Харків : ХНУ ім. Каразіна, 2016. 265 с.

5. https://www.python.org/getit

6. Мокін Б. І., Мокін О. Б., Шалагай Д. О. Про один із підходів наближеного обчислення інтегралів Стілтьєса і Лебега на мові Python в задачах системного аналізу з дискретними моделями. *Вісник Вінницького політехнічного інституту*, 2021. № 3. С. 61–68.

---

*Навчальне видання*

**Мокін Борис Івович**
**Мокін Віталій Борисович**
**Мокін Олександр Борисович**

**НАВЧАЛЬНИЙ ПОСІБНИК**
**для опанування студентами способів розв'язання задач з функціонального аналізу мовою Python**

**Частина 1**

Редактор В. Дружиніна

Оригінал-макет підготовлено Б. Мокіним

Підписано до друку 15.06.2022. Формат 29,7×42¼. Папір офсетний.
Ґарнітура Times New Roman. Ум. друк. арк. 14,78.
Наклад 35 пр. Зам. № 2022-057.

Видавець та виготовлювач –
Вінницький національний технічний університет,
редакційно-видавничий відділ.
21021, м. Вінниця, Хмельницьке шосе, 95,
ВНТУ, ГНК, к. 114.
Тел. (0432) 65-18-06.
Свідоцтво суб'єкта видавничої справи
серія ДК № 3516 від 01.07.2009 р.