Лабораторна робота № 8

Створення та використання підпрограм на Python. Використання підпрограм для опрацювання колекцій.

Мета роботи: *ознайомлення з прийомами процедурного програмування, структурою функцій користувача, способами передачі параметрів та опрацюванням списків за допомогою підпрограм.*

Короткі теоретичні відомості

В алгоритмах розв’язування різних задач нерідко можна зустріти певну послідовність команд, яку доводиться виконувати кілька разів, але для різних наборів даних. Така послідовність, зазвичай може бути описана як окремий алгоритм. Вона розв’язує деяку «підзадачу», тобто, задачу, що є частиною основної. Програму, що реалізує такий алгоритм, відповідно, називають ***підпрограмою***. До розв’язування великих та складних сучасних задач часто застосовують ***декомпозицію*** – підхід що полягає в розбитті основної задачі на окремі прості задачі. Тому більшість сучасних програм, насправді, є сукупностями великої кількості ***підпрограм***, що взаємодіють між собою.

Можливість створювати та використовувати підпрограми закладена в кожній мові програмування. В залежності від мови та технології програмування ***підпрограми*** можуть називати ***процедурами***, ***функціями*** чи ***методами***. Від першої з цих назв походить назва однієї з ***парадигм програмування***, – ***процедурне програмування***, що розглядає роботу програми як послідовність викликів відповідних процедур. У об'єктно-орієнтованому програмуванні підпрограми є членами класів, тут їх називають ***методами***.

Кожну ***підпрограму*** можна розробляти, налагоджувати та вдосконалювати незалежно від решти компонентів програми. Сьогодні використання ***декомпозиції*** належить до базових навичок розробника, а розуміння процедурного підходу важливе для інженера будь-якого фаху нарівні з розумінням алгоритмічних конструкцій.

Підпрограма розв’язує якусь певну частину загальної задачі, тому для своєї роботи повинна отримати необхідні вхідні дані від основної програми. Так само, обчисливши розв’язок поставленої їй задачі, підпрограма повинна повідомити результат своєї роботи основному коду. Перше завдання вирішується за допомогою механізму ***передачі параметрів***, друге, окрім передачі параметрів може використовувати також механізм ***повернення результату функції***.

Взаємодія між підпрограмами, очевидно, пов’язана з опрацюванням деяких спільних даних. Одним з основних способів обміну інформацією між частинами програми є механізм ***передачі параметрів*** (аргументів) у процедуру, функцію чи метод. Однак, з цією метою можна використовувати й інші засоби – ***глобальні змінні*** чи зовнішню пам'ять (***файли*** на диску). Проте

велика кількість глобальних змінних змушує програміста ретельно узгоджувати імена загальних змінних, збільшує залежність програмних одиниць одна від одної, накладає обмеження на вибір імен ***локальних змінних*** (власних змінних підпрограми). Крім того, загальнодоступність глобальних змінних може призвести до їх неузгодженої зміни різними частинами програмного коду, а використання зовнішньої пам'яті в ряді випадків призводить до значного сповільнення роботи програми.

Підпрограма, як окрема програмна одиниця, повинна оперувати деякими даними, які зберігають в коді за допомогою змінних (можливо, й констант). Для даних, переданих підпрограмі «ззовні», тобто параметрів, в її коді також оголошують деякі змінні, які називають ***формальними параметрами***. Їх оголошують як змінні в заголовку підпрограми при її описі, вони не містять даних, а лише показують, які саме дії виконуватиме підпрограма з отриманими даними. Під час виклику підпрограми основна програма на місце ***формальних параметрів*** передає дані для виконання підпрограми. З боку основної програми вони також представлені деякими змінними чи константами, їх називають ***фактичними параметрами***, бо вони містять дані, з якими фактично працюватиме підпрограма.

Для підпрограм на ***Python***, як і в ***C++*** використовується термін ***функція***. Тут можна описувати функції, що повертають певне значення та функції які нічого не повертають, а лише виконують закладену в них послідовність команд.

На правилах синтаксису та використання функцій в ***Python*** суттєво позначається трансляція шляхом інтерпретації та динамічна типізація:

- ✓ *оголошення функції є виконуваним оператором, його виконання прив'язує назву функції об'єкта функції, відповідно функція має бути реалізована до її виклику;*
- ✓ *функція має заголовок, що починається службовим словом def, за яким вказують її назву, список параметрів в круглих дужках;*
- ✓ *тіло функції слідує за заголовком і відділяється від нього двокрапкою та відповідним горизонтальним відступом з нового рядка;*
- ✓ *список формальних параметрів функції записують у круглих дужках одразу після її назви, розділяючи їх комами, тип параметра не вказується;*
- ✓ *до змінних, оголошених в тілі функції (локальні змінні) можуть звертатися лише оператори з тіла цієї функції;*
- ✓ *в Python реалізовано передачу параметрів виключно за адресою, проте для змінних простих типів (immutable) при передачі фактичних параметрів створюється копія;*
- ✓ *функція може повертати результат, його значення передається з тіла функції за допомогою оператора* **`return;`**
- ✓ *оператор* **`return`** *може повертати окреме значення, або список значень у вигляді кортежу.*

Завдання. 1.

Запрограмувати обчислення заданого виразу, розбивши його на частини однотипної структури. Створити підпрограми для обчислення значень окремих частин виразу. В програмі передбачити ввід вхідних даних та вивід результатів. Вхідні дані задати самостійно.

1. $z = \frac{|v^2-u^2|^{cos(x^2-y^2)}}{|x^2-y^2|^{sin(v^2-u^2)}}$;

2. $p = \frac{\sqrt{|x\cos\alpha t - y\cos\beta t|}}{\sqrt{|y\cos\beta t - x\cos\alpha t|}}$;

3. $y = \frac{\left(\sqrt[3]{v^2-u^2}\right)}{\left(\sqrt[3]{b^2-a^2}\right)} - \frac{\left(\sqrt[3]{b^2-v^2}\right)}{\left(\sqrt[3]{a^2-u^2}\right)}$;

4. $y = \frac{sin(\omega t+\gamma)}{sin(\gamma t+\beta)} - \frac{sin(\omega t+\beta)}{sin(\beta t+\gamma)}$;

5. $y = \frac{v^{sin\,p} - v^{sin\,q}}{v^{sin\,p} + v^{sin\,q}}$, $v > 0$;

6. $z = \frac{\sqrt[5]{x\sin\alpha t + y\sin\beta t}}{\sqrt[5]{y\sin\beta t - x\sin\alpha t}}$;

7. $y = \frac{\left(\sqrt{|a+b^3|} - \sqrt{|b+a^3|}\right)}{-\sqrt{|t+(a-b)^3|}}$;

8. $s = \frac{\left(\sqrt[3]{t^2+u^2}\right)}{\left(\sqrt[3]{b^2+a^2}\right)} - \frac{\left(\sqrt[3]{b^2+t^2}\right)}{\left(\sqrt[3]{a^2+u^2}\right)}$;

9. $y = \frac{e^{\sqrt{p+q}} - e^{-\sqrt{p+r}}}{e^{-\sqrt{p+r}} + e^{\sqrt{q+r}}}$, $p, q, r > 0$;

10. $t = \frac{\sqrt[v]{|sin\,p|} - \sqrt[v]{|sin\,q|}}{\sqrt[u]{|sin\,q|} + \sqrt[u]{|sin\,p|}}$;

11. $p = \frac{(v^2+u^2)^{ln(x^2+y^2)}}{(x^2+y^2)^{lg(v^2+u^2)}}$;

12. $z = \frac{(v^2+u^2)^{sin(x^2+y^2)}}{(x^2+y^2)^{cos(v^2+u^2)}}$;

13. $y = \frac{a\,sin(\omega t+\gamma)}{b\,sin(\omega t+\beta)} - \frac{b\,sin(\omega t+\beta)}{a\,sin(\omega t+\gamma)}$;

14. $z = \frac{|v^3+u^2|^{cos(x^3-y^2)}}{|x^3+y^2|^{sin(v^3-u^2)}}$;

15. $z = (x^2 + y^2 + 2)\,log_{(x^2+y^2+2)}(a^2 + b^2 + 2)$;

16. $p = \left(\sqrt[3]{yb^2} + \sqrt[3]{xa^2}\right) \cdot \left(\sqrt[3]{bx^2} - \sqrt[3]{ay^2}\right)$.

У звіт подати:

- умову задачі варіанта;
- блок-схему алгоритму розв’язування задачі;
- складений Python-код;
- результати обчислень для деякого набору вхідних даних.

Приклад виконання завдання 1

Варіант № 16. Умова задачі: Запрограмувати обчислення заданого виразу, розбивши його на частини однотипної структури. Створити підпрограми для обчислення значень окремих частин виразу. В програмі передбачити ввід вхідних даних та вивід результатів. Вхідні дані задати самостійно.

16. $p = \left(\sqrt[3]{yb^2} + \sqrt[3]{xa^2}\right) \cdot \left(\sqrt[3]{bx^2} - \sqrt[3]{ay^2}\right)$.

Тут завдання програми порахувати значення виразу при заданих значення змінних, що в нього входять. В нашому випадку для обчислення виразу потрібно задати значення величин a, b, x та y. Записавши оператор присвоєння та заданий вираз в його правій частині, отримаємо звичайну лінійну програму.

Запис такого виразу часто може виявитися доволі громіздким, особливо, враховуючи, що в деяких системах програмування відсутні штатні засоби для добування кореня кубічного чи піднесення до степеня. З іншого боку, розглянувши уважно структуру виразу можемо бачити, що в ньому кілька разів зустрічається одна і таж арифметична конструкція вигляду $\sqrt[3]{uv^2}$. Отже, вираз в умові можна записати $p = \big(f(y,b) + f(x,a)\big) \cdot \big(f(b,x) - f(a,y)\big)$, де $f(u,v) = \sqrt[3]{uv^2}$. Така підстановка, вказує програмну реалізацію обчислення, що використовуватиме підпрограму для обчислення $f(u,v) = \sqrt[3]{uv^2}$.

Що ж до виду підпрограми, то для обчислення виразів найзручніше використовувати функції, які повертають результат у місце свого виклику. У протилежному випадку доведеться використовувати додаткові змінні для передачі результату через параметри.

Код програми для розв'язування задачі на мові ***Python***:

```
import math

#Функція для обчислення значення радикала

def f(u,v):

  t = u * v * v

  res = -(-t) ** (1/3) if t < 0 else t ** (1/3)

  return res

a = float(input('Введіть значення a: '))

b = float(input('Введіть значення b: '))

x = float(input('Введіть значення x: '))

y = float(input('Введіть значення y: '))

p = (f(y, b) + f(y, b)) * (f(b, x) - f(a, y))

print('Значення виразу %10.3f' % p)
```

Завдання2.

Скласти програму розв'язання задачі відповідно до варіанта. У звіт подати:

- умову задачі варіанта;
- складений Python-код;
- результати обчислень для деякого набору вхідних даних.

Варіанти завдань:

1. Реалізувати підпрограму для створення та заповнення масиву з 15 дійсних чисел за правилом $sin(x^3) + 3x\,sin\,k$, $k = 1,2,\ldots,15$, x – деяке задане користувачем число. Використовуючи її утворити три масиви дійсних чисел $M(15)$, $P(15)$ та $Q(15)$. Вивести масиви на екран, в кожному масиві знайти півсуму максимального та мінімального елемента з використанням окремих, розроблених для цього підпрограм.

2. Три масиви дійсних чисел X, Y та W, складаються з 20 елементів кожен. Створити підпрограму для заповнення масиву введеними з клавіатури числами та використати її для вводу елементів заданих масивів. Вивести масиви на екран, в кожному масиві знайти різницю квадратів максимального та мінімального елементів, використовуючи окремі, розроблені для цього підпрограми.

3. Утворити масиви дійсних чисел A та B по 12 елементів кожен та заповнити їх випадковими числами з відрізка $[-50;\ 50]$ за допомогою створеної для цього підпрограми. Масив C утворити за правилом $C_k = max\{|A_k|;\ 2B_k\}$. Вивести масиви на екран, в кожному масиві знайти номер елемента, найближчого за величиною до 1. Вивід кожного масиву та пошук у ньому вказаної величини оформити у вигляді окремих підпрограм.

4. Створити і заповнити масиви дійсних чисел U, V та X, кожен по 16 елементів за допомогою підпрограми, яка заповняє масив випадковими числами з відрізка $[-2x;\ 3x]$ (число x задає користувач, окремо для кожного масиву). Вивести масиви на екран, в кожному масиві знайти мінімальний додатний елемент. Для виводу масиву та пошуку у ньому вказаної величини реалізувати окремі підпрограми.

5. Утворити масиви дійсних чисел M, P по 12 елементів кожен та заповнити їх введеними користувачем даними за допомогою окремої, розробленої для цього підпрограми. Масив S утворити з масиву P, замінивши в ньому всі додатні елементи мінімальним елементом масиву M. Вивести масиви на екран, в кожному масиві знайти номер максимального елемента, використовуючи окремі, розроблені для цього підпрограми.

6. Три масиви дійсних чисел A, B та C містять по 15 елементів кожен. Заповнити масиви A та B введеними користувачем даними за допомогою окремої, розробленої для цього підпрограми. Масив C утворити додаванням оберненого масиву A до масиву B, тобто $C_1 = A_{15} + B_1$, $C_2 = A_{14} + B_2$, $C_3 = A_{13} + B_3$, і т.д. Вивести масиви на екран, в кожному масиві знайти суму елементів, значення яких належать інтервалу (-2; 0). Вивід кожного масиву та пошук у ньому вказаної величини оформити у вигляді окремих підпрограм.

7. Утворити три масиви дійсних чисел M, P та Q, кожен містить по 15 елементів та заповнити їх за допомогою створеної для цього підпрограми випадковими числами з відрізка $[0;\ p]$ (число p задає користувач, окремо для кожного масиву). Вивести масиви на екран, в кожному масиві знайти різницю

максимального та мінімального елемента, використовуючи для цього окремі підпрограми.

8. Утворити три масиви дійсних чисел U, V та W, кожен по 11 елементів та заповнити їх елементами за допомогою підпрограми, яка заповняє масив елементами за правилом $x_k = (-1)^k\sqrt{(k+p)(k-q)^2}$, де p та q – деякі задані користувачем числа (різні для кожного з масивів), $k = 1,2,\ldots,11$. Вивести масиви на екран, створивши для цього окрему підпрограму. За допомогою ще однієї підпрограми в кожному масиві знайти різницю між максимальним елементом та середнім арифметичним усіх елементів.

9. Утворити чотири масиви дійсних чисел P, Q, R та S, по 17 елементів кожен, за допомогою підпрограми, яка заповняє масив введеними з клавіатури числами. Створити підпрограми для виведення масиву на екран та для обчислення в масиві відношення максимального його елемента до мінімального. Використовуючи створені підпрограми вивести на екран усі масиви та відношення максимального елемента до мінімального в кожному з них.

10. Реалізувати підпрограму для створення та заповнення масиву з 12 дійсних чисел елементами за правилом $sin(p) + 3x\,sin\,k$, де $k = 1,2,\ldots,12$, p та x – деякі числа, задані користувачем при створенні масиву. Виколисуючи її заповнити три масиви дійсних чисел A, B та C по 12 елементів кожен. Вивести масиви на екран, в кожному масиві знайти суму від'ємних елементів, розробивши для цього окремі підпрограми.

11. Утворити масиви дійсних чисел U, V та W по 14 елементів кожен. Для заповнення масивів U та V розробити підпрограму, яка заповнює масив введеними з клавіатури числами. Масив W утворити з масиву V, замінивши в ньому всі додатні елементи мінімальним елементом масиву U. Вивести масиви на екран, в кожному масиві знайти номер елемента, найближчого за величиною до 0, розробивши для цього окремі підпрограми.

12. Утворити три масиви дійсних чисел A, B та C, розробивши для цього підпрограму, яка заповнює масиву з 16 дійсних чисел елементами за правилом $\frac{(-1)^k(2k^2+p)}{p}$, де $k = 1,2,\ldots,16$, а p – деяке задане користувачем для кожного масиву окремо. Розробити ще дві підпрограми, за допомогою яких вивести масиви на екран, в кожному масиві знайти середнє арифметичне додатних елементів.

13. Утворити три масиви дійсних чисел U, V та W, кожен по 12 елементів та заповнити їх елементами за допомогою підпрограми за правилом $x_k = (-1)^k\sqrt{(k^2+p)(k+q^2)}$, де p та q – деякі задані користувачем числа (різні для кожного з масивів), $k = 1,2,\ldots,12$. Вивести масиви на екран, в кожному масиві знайти кількість елементів, більших за 1, розробивши для цього окремі підпрограми.

14. Три масиви дійсних чисел A, B та C, містять по 10 елементів кожен. Для заповнення масивів A та B створити підпрограму, що заповняє масив числами, які користувач вводить з клавіатури. Масив C утворити за правилом $C_k = 2A_k - 3B_k$. Вивести масиви на екран, в кожному масиві знайти суму додатних елементів, розробивши для цього окремі підпрограми.

15. Утворити масиви дійсних чисел U та V по 18 елементів кожен та заповнити їх випадковими числами з відрізка $[-9;\ 9]$ за допомогою створеної для цього підпрограми. Масив W утворити за правилом $W_k = max\left\{\frac{U_k}{v_k}; \frac{V_k}{U_k}\right\}$. Вивести масиви на екран, в кожному масиві знайти суму елементів, абсолютна величина яких менша за 1, розробивши для цього окремі підпрограми.

16. Створити за допомогою підпрограми два масиви A та B, по 12 елементів кожен, заповнивши їх дійсними числами за правилом $x_k = (-1)^k\sqrt{(k^2 + p)(k + q^2)}$, де p та q – деякі задані користувачем числа (різні для кожного з масивів), $k = 1,2,\ldots,12$.. Масив C утворити з масиву B, замінивши в ньому всі від'ємні елементи середнім значенням елементів масиву A. Вивести масиви на екран, в кожному масиві знайти елемент, значення якого найближче до числа 1, розробивши для цього окремі підпрограми.

Приклад виконання завдання 2

Варіант № 16. Умова задачі: Створити за допомогою підпрограми два масиви A та B, по 12 елементів кожен, заповнивши їх дійсними числами за правилом $x_k = (-1)^k\sqrt{(k^2 + p)(k + q^2)}$, де p та q – деякі задані користувачем числа (різні для кожного з масивів), $k = 1,2,\ldots,12$.. Масив C утворити з масиву B, замінивши в ньому всі від'ємні елементи середнім значенням елементів масиву A. Вивести масиви на екран, в кожному масиві знайти елемент, значення якого найближче до числа 1, розробивши для цього окремі підпрограми.

За умовою задачі програма повинна опрацьовувати три масиви чисел по 12 елементів. Для заповнення перших двох потрібно створити підпрограму, заповнення третього масиву C індивідуальне, за окремим правилом. Оскільки код, що заповнює масив C використовуватиметься лише один раз, то створювати для нього окрему підпрограму недоцільно. Нарешті, після утворення масивів, усі три масиви потрібно вивести на екран, та в кожному знайти елемент, значення якого найближче до 1. А для цього, за умовою, потрібно організувати ще пару підпрограм.

У підсумку наша програма повинна містити три підпрограми: для заповнення масиву за вказаним правилом, для виводу елементів масиву на екран та для пошуку в масиві елемента, значення якого найближче до 1.

Загальна схема роботи програми буде такою:

- *програма за допомогою створеної підпрограми заповнює масиви A та B;*
- *на основі створених масивів A та B програма за вказаним правилом утворює третій масив* C*;*
- *до кожного зі створених масивів програма застосовує підпрограму для виводу на екран, та підпрограму для пошуку елемента, значення якого найближче до 1, а знайдене значення виводить на екран.*

Щодо підпрограм для заповнення нового масиву за заданим правилом, то тут можливі два різні підходи: масив створюється в основній програмі і передається у підпрограму для заповнення даними як вхідно-вихідний параметр, або масив створюється і заповнюється даними безпосередньо у підпрограмі і повертається як результат її роботи у вигляді вихідного параметра. Перший дозволяє програмі контролювати створення об'єктів, проте потребує додаткової передачі даних. Другий, мінімізуючи передачу даних, навпаки позбавляє програму контролю за створенням власних об'єктів. На практиці вибір одного з цих двох підходів узгоджується з фізичним механізмом зберігання масивів і в тій чи іншій технології програмування.

В якості масивів в ***Python*** використовуються списки, що вже самі по собі є об'єктами. Більше того, динамічна типізація обумовлює передачу усіх параметрів функцій як об'єктів. Тому ідея створення масивів та заповнення їх елементами безпосередньо у підпрограмі виглядає більш природньо.

Інтерпретація коду та безпосереднє виконання інтерпретатором коду реалізації функції зобов'язує нас оголосити та описати підпрограми до основного коду з їх викликом.

Оголошення псевдоконстанти з розміром масивів має на меті хіба що контроль циклів зі створення масивів. Код програми для розв'язування задачі 2 на мові ***Python***:

```
import math
# псевдоконстанта для розміру масивів
SIZE= 12
# функції оголошуються до основної програми
def create_array():
  sign = -1 # -1^n при n = 1
  p = float(input("p = "))
  q = float(input("q = "))
  m = []
```

```
  for i in range(SIZE):
    m.append( sign * math.sqrt(((i + 1) * (i + 1) + p)* \
            (i + 1 + q * q)) )
    sign *= -1 # обчислення -1 ^ (k + 1)
  return m

def print_array(m):
  for a in m:
    print("%7.2f, "% a, end='')
  print()

def min_dist_el(m):
  min = m[0] # початкове наближення та його
  dist = abs(min - 1) # відстань до 1
  for a in m: # цикл по елементах масиву
    if abs(a - 1) < dist :
      # знайдено новий елемент найближчий до 1
      min = m[i]
      dist = abs(min - 1)
  return min
# основна программа
A = create_array()
B = create_array()
# заповнення третього масиву C
SumA = 0
for el in A:
  SumA += el
avg = SumA / SIZE
C = [] # оголошуємо список для масиву C
# і додаємо в нього елементи
```

```
for i in range(SIZE):
  if B[i] < 0 :
    C.append(avg) # записуємо середнє
  else :
    C.append(B[i]) # беремо з масиву B

print('Масив A:')
print_array(A) # вивід масиву
min_d1 = min_dist_el(A) # пошук найближчого до 1
print("Найближче до 1 %10.2f\n" % min_d1)

print('Масив B:')
print_array(B) # вивід масиву
min_d1 = min_dist_el(B) # пошук найближчого до 1
print("Найближче до 1 %10.2f\n" % min_d1)

print('Масив C:')
print_array(C) # вивід масиву
min_d1 = min_dist_el(C) # пошук найближчого до 1
print("Найближче до 1 %10.2f\n" % min_d1)
```

Запитання для самоконтролю

1. Для чого використовуються підпрограми?
2. Шо таке формальні параметри підпрограми?
3. Шо таке фактичні параметри підпрограми?
4. Які ситаксичні конструкції для створення підпрограм використовує ***Python***?
5. Як повернути результат роботи функції в основний код?
6. Який механіз передачі параметрів у функцію використовується в ***Python***?