Функція – це фрагмент коду, призначений для розв'язання певної задачі, до якого можна багаторазово звертатись з різних місць основної програми. Вона повертає у місце виклику певне значення, що є результатом її виконання.

Отже, будь-яка функція має такий загальний вигляд:

def <назва> ([параметри]):
<блок команд>
[return <результат>]

Параметр (також називають аргументом) – це змінна, яка отримує конкретне значення під час звернення до функції. Параметри вказувати не обов'язково, але при цьому круглі дужки опускати не можна.

Інструкція return повертає результат виконання функції в основну програму. Наведемо приклад використання нижче.

Функція:

```
def заощадження(кишенькові, зароблені, витрачені):
  return кишенькові + зароблені - витрачені
```

Виклик функції у межах основної програми:

```
print(заощадження(50, 100, 20))
```

Результат:

130

Змінну, яка використовується у межах функції, не можна використати ще раз у межах основної програми, адже вона втратить свою видимість. Тому розрізняють два види змінних: локальні і глобальні.

Локальна змінна – це змінна, яка оголошена всередині функції. Локальні змінні доступні тільки всередині функції, у якій їх було створено.

Глобальна змінна – це змінна, яка оголошена в основній програмі, поза межами функцій. Глобальні змінні є видимими для будь-якої частини програми, зокрема, всередині функцій.

Зауваження: Локальну змінну, що оголошена в середині функції, можна зробити видимою для усією програми, написавши перед нею службове слово *global*. Тоді значення цієї змінної збережеться і після завершення роботи функції, і не буде знищено, як це стається з усіма локальними змінними.

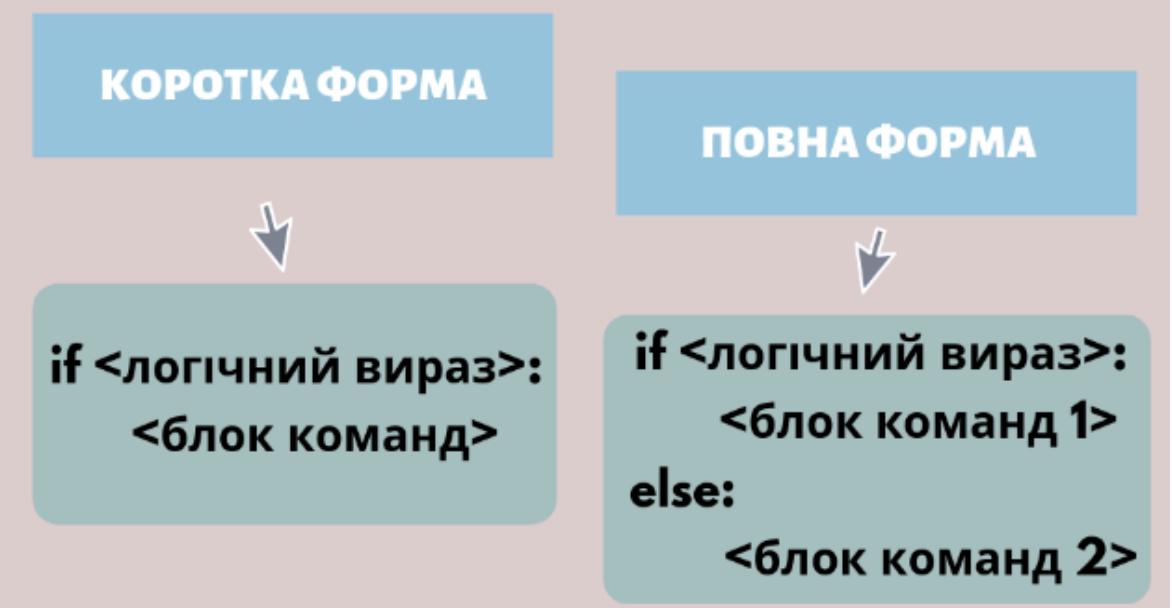
У мові Python розрізняють функції з довільною кількістю аргументів, з обов'язковими і необов'язковими параметрами, з позиційними та ключовими параметрами, лямбда-функції.

return зупиняє роботу функції і повертає вказане значення як її результат.

Бувають задачі, для розв’язання яких недостатньо лінійних конструкцій. Команди повинні виконуватись лише при певному значенні конкретної умови. Для розв’язування таких задач використовують **алгоритмічну структуру розгалуження.**

В алгоритмах з розгалуженням використовується оператор **if.**

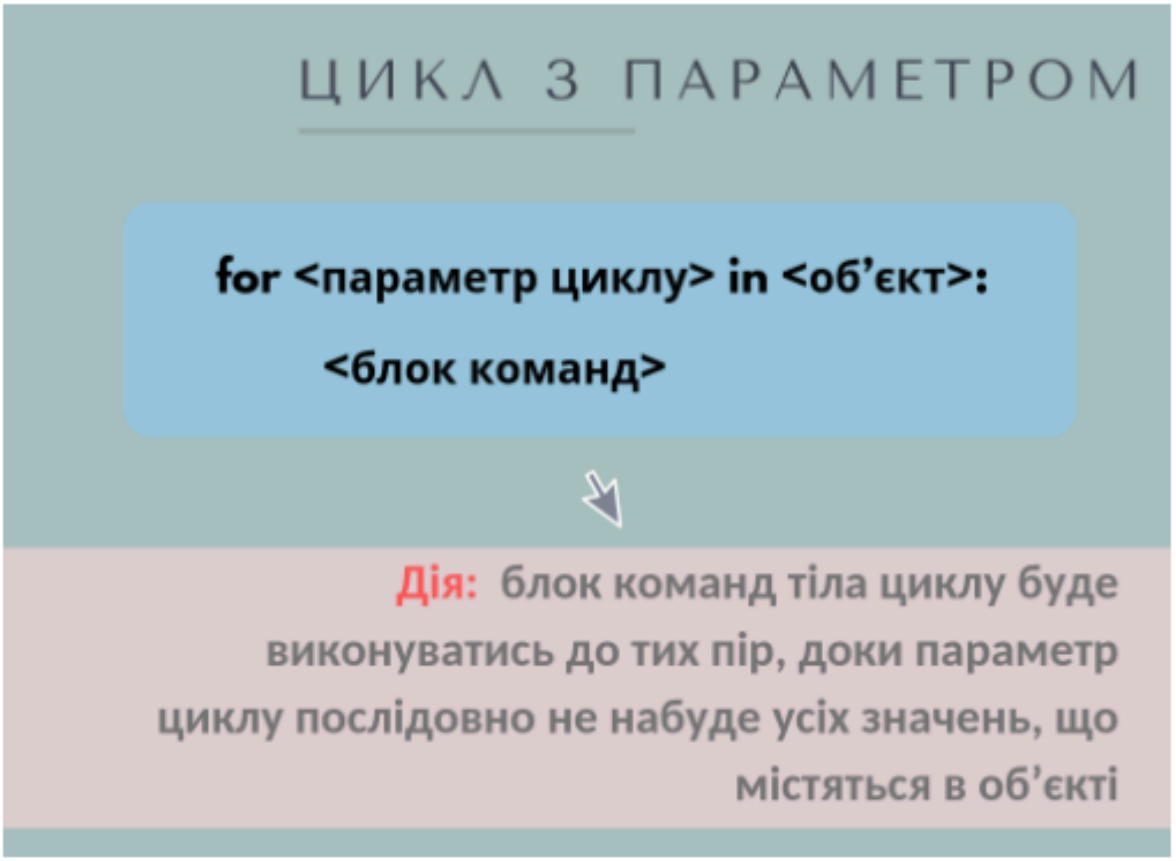
Його застосування має кілька форм.



1. Напишіть програму, яка знаходить абсолютне значення числа (модуль числа).

```
x = int(input('Введіть число: '))   #  вводимо число за допомогою команди input
if x > 0:                           #  команда розгалуження! перевіряємо, чи число додатне
  print(x)                  #  виводимо число без зміни
else:                       #  інакше — виводимо число з протилежним знаком
  print(-x)
```

Цикли



Цикл з параметром for

Для організації циклічних алгоритмів можна використати цикл з параметром for.

for <параметр циклу> in <об'єкт>

<блок команд>

Дія: блок команд тіла циклу буде виконуватись до тих пір, доки параметр циклу послідовно не набуде усіх значень, що містяться в об'єкті.

<об'єкт> може бути рядком, списком, словником тощо.

У циклічних алгоритмах часто застосовують функцію range - діапазон.

Приклад:

```
for i in range(0, 5):
    print ('Hello')
```

Результат:

Hello

Hello

Hello

Hello

Hello

Спершу змінна і набуває значення 0 і друкується слово "Hello". Далі відбувається повернення у цикл: змінна набуває значення 1 та знову друкується слово "Hello", і так далі. Останнім значенням змінної і буде 4 і знову надрукується слово "Hello". Таким чином, після виконання циклу на екрані з'явиться привітання 5 разів (від 0 до 4).

Приклад:

```
food = ['pizza', 'soup', 'butter']
for i in food:
   print (i)
   print (i)
```

Результат:

pizza

pizza

soup

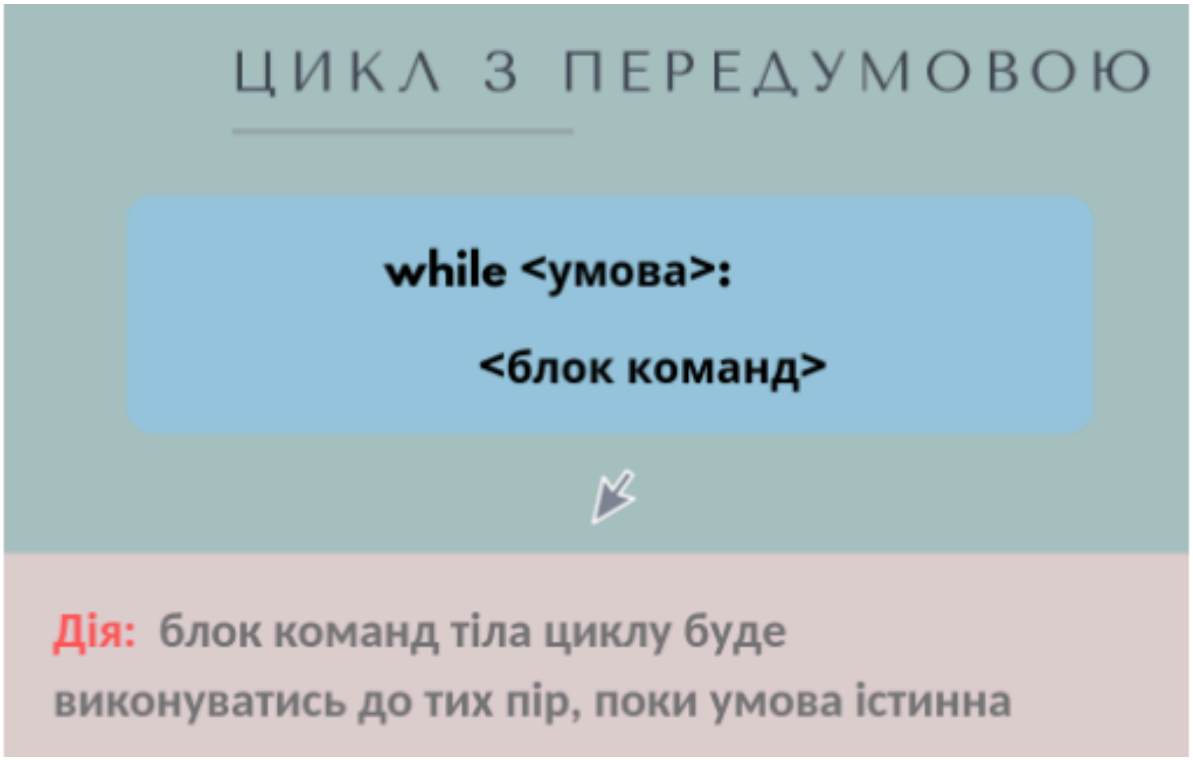
soup

butter

butter

Змінна і спершу набуває першого значення зі списку food — 'pizza'. Далі виконується двічі команда print(i), що друкує

значення змінної і, тобто на екрані двічі з'явиться слово pizza. Після цього відбувається повернення у цикл: змінна і стає словом soup, яке теж друкується на екрані двічі. І т. д.



Цикл з передумовою while

Для організації циклів з передумовою використовують оператор while.

```
while <умова>:
    <блок команд>
```

Дія: блок команд тіла циклу буде виконуватись до тих пір, поки умова істинна.

Наприклад, щоб вивести слово “Hello” 5 разів, можна написати такий скрипт:

```
i = 5
while i > 0:
  print ('Hello')
   i -= 1              # скорочений запис команди i = i - 1 (зменшення числа на одиницю)
```

Результат:

```
Hello
```

Hello

Hello

Hello

Hello

!!! Змінна, задіяна в умові, обов'язково повинна змінюватись у тілі циклу. Без цього команди можуть «зациклитись», тобто виконуватимуться безупинно.

У даному прикладі у першому рядку змінна і набуває значення 5. Цикл розпочинається службовим слово while: відбувається перевірка умови i > 0. Очевидно, що 5 > 0, тому команди у тілі циклу виконуватись будуть. Тіло циклу складатиметься з двох команд print ('Hello') та i -= 1, бо вони відділені від початку рядка однаковою кількістю пробілів. Отже, буде надруковано слово "Hello" та змінна і зменшиться на одиницю. Тепер змінна і дорівнює 4. Повертаємося у цикл та знову перевіряємо умову i > 0: очевидно, що 4 > 0, тому знову надрукується слово "Hello" та змінна і стане 3-кою. І т. д. Цикл припиниться аж тоді, коли і стане нулем, бо умова i > 0 при такому значення є хибною.

Цикл з післяумовою

Щоб створити цикл з післяумовою, можна використати таку конструкцію:

```
while True:
```

<блок команд>

if <умова>: break

Вбудовані функції

Часто у програмі потрібно перетворити величини одного типу у інший тип даних. Для цього використовують такі функції:

bool() - перетворення об'єктів на логічний тип даних;

int() - перетворення у цілий числовий тип;

float() - перетворення цілого числа (або рядка) у дійсний тип даних;

str() - перетворення у рядок;

list() - перетворення у список.

Варто зауважити, що ***інформація про тип зберігається в об'єкті, а не в змінній***.

Арифметичні оператори: + (додавання), - (віднімання), * (множення), / (ділення), // (цілочисельне ділення), % (остача від ділення), ** (піднесення до степеня).

Для роботи з числами можна використовувати стандартні математичні функції, які можна імпортувати у проект за допомогою команди *import math*. Цей модуль містить тригонометричні функції (sin(), cos(), tan(), asin(), acos(), atan()), функції перетворення радіан на градуси та градусів на радіани (degrees(), radians()), експоненту (exp()).

Крім того, найчастіше вживаними є такі функції:

Функція	Призначення
sqrt()	квадратний корінь
ceil()	найближче ціле (більше)
log10()	десятковий логарифм
floor()	найближче ціле (менше)
pow(a, n)	піднесення числа a до степеня n
factorial()	факторіал
fmod(a, n)	остача від ділення числа a на число n

Введення та виведення даних

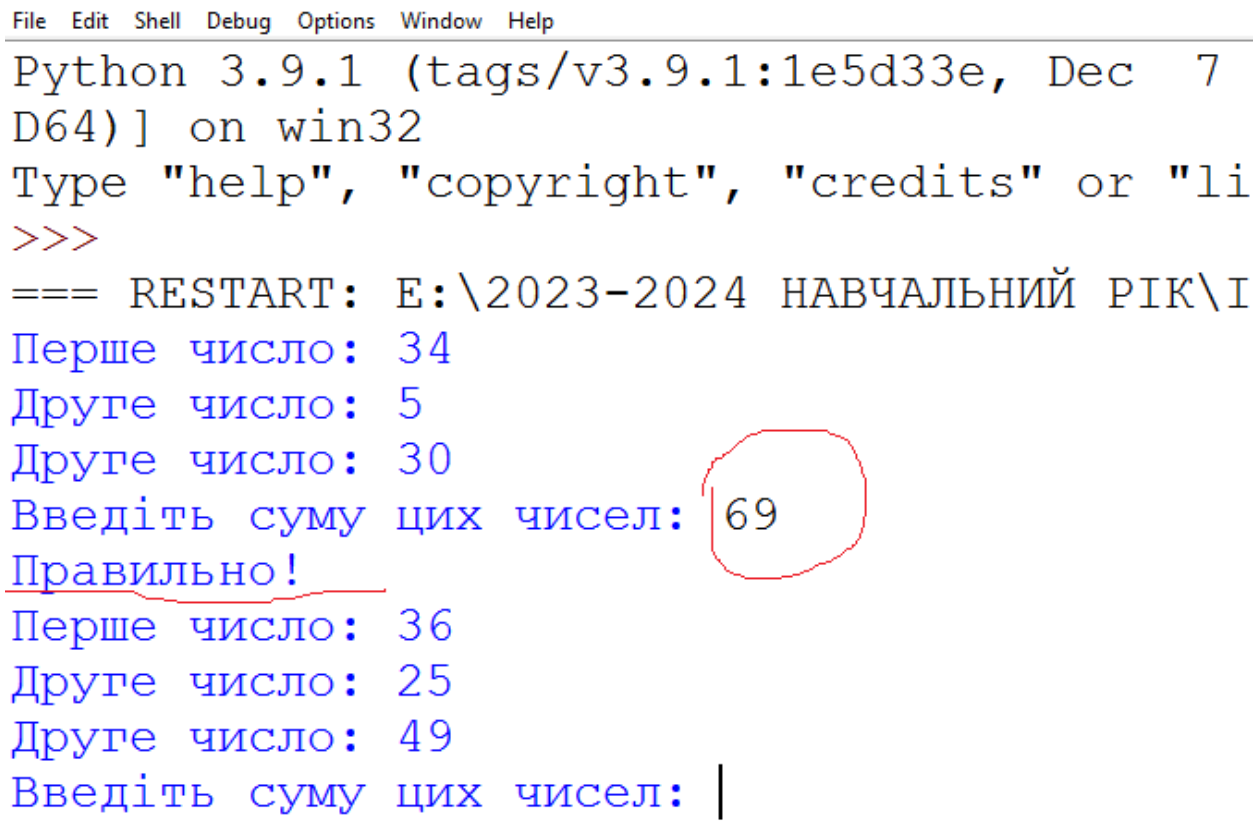
Вбудована функція **input** – це засіб отримання результату введення з клавіатури.

Вона може виводити підказку, текст якої міститься в необов'язковому аргументі-рядку, і повертає введену користувачем відповідь у вигляді рядка.
Наприклад:
n = int(input('Введіть число: '))

Для виведення даних на екран використовують команду **print.**
Створити програму, яка за допомогою функції random буде виводити три числа в діапазоні від 0 до 50. Потім пропонує користувачу ввести результат додавання тих трьох чисел, тобто їх суму.

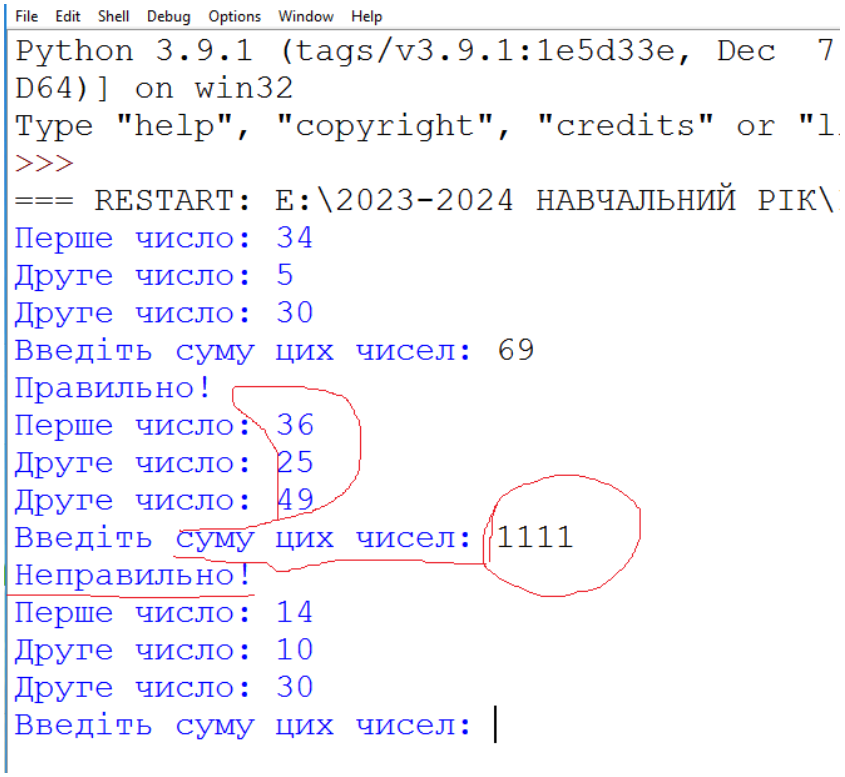
- Якщо користувач ввів правильно, то програма виводить фразу «правильно», у іншому випадку – «неправильно».
- Після виведення фрази (не важливо «правильно» чи «неправильно» програма знову генерує нові числа і знову просить користувача ввести їх суму, тобто вона працює безкінечно)

Якщо користувач ввів **суму** *правильно,*
то програма виводить фразу «правильно»

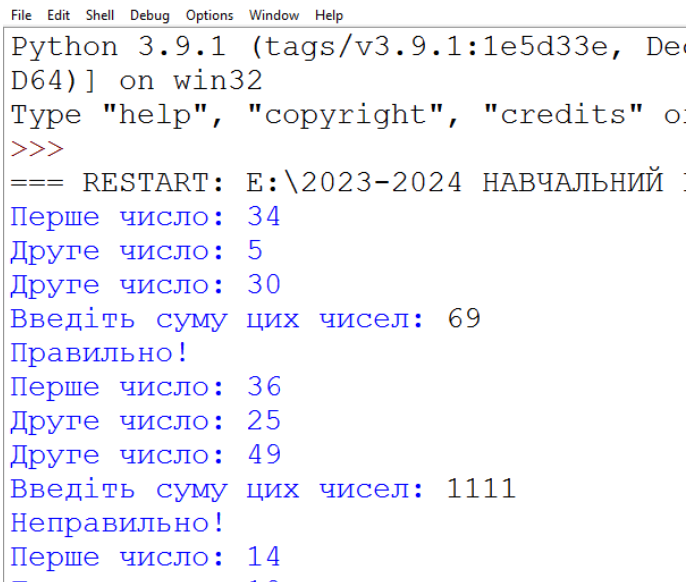


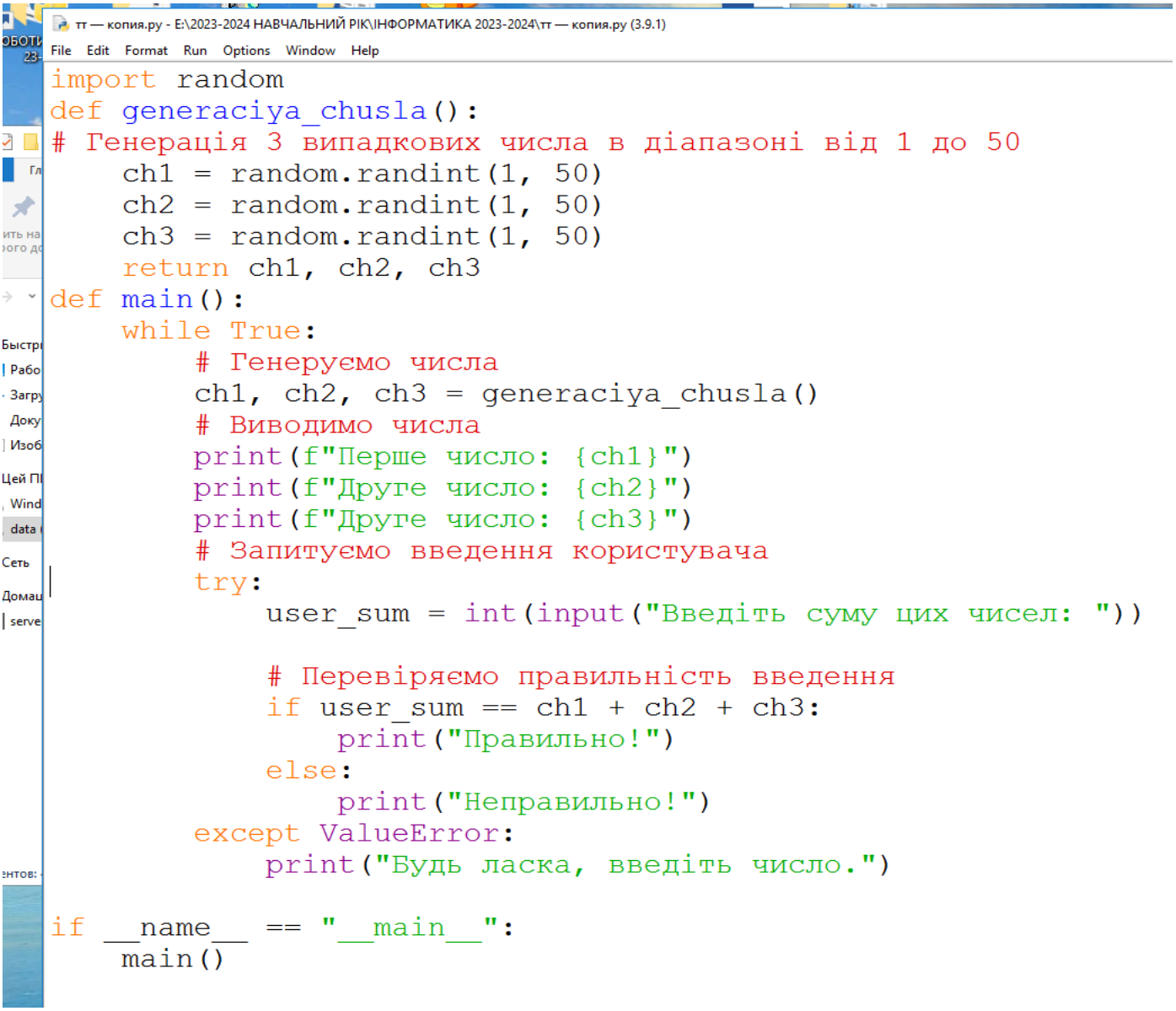
Якщо користувач ввів **суму** *неправильно,*

то програма виводить фразу «неправильно»



програма знову генерує нові числа і знову просить користувача ввести їх суму, тобто вона працює безкінечно

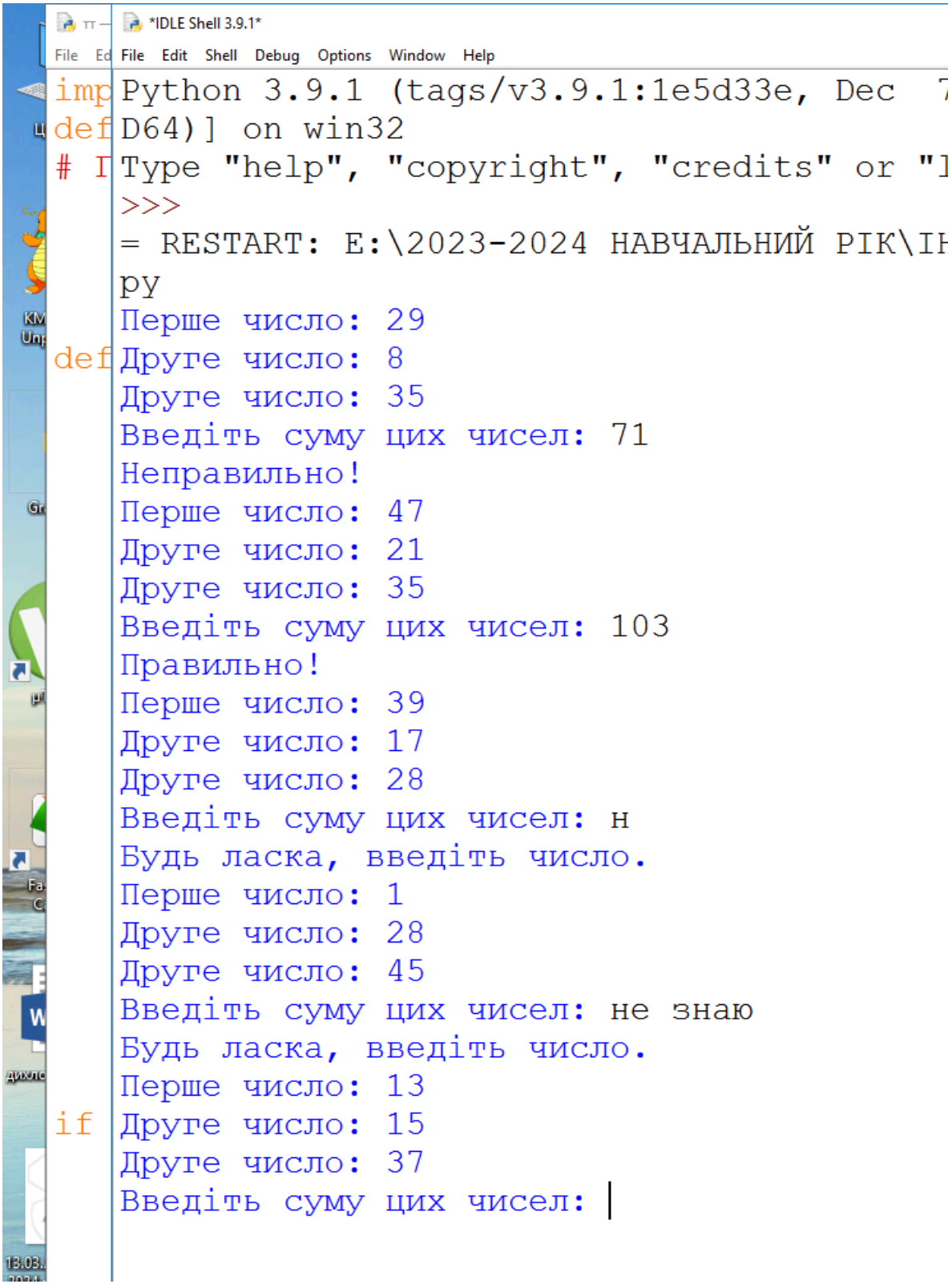




except ValueError:

print("Будь ласка, введіть число.")

```
Перше число: 27
Друге число: 12
Друге число: 3
Введіть суму цих чисел: p
Будь ласка, введіть число.
```


IDLE Shell 3.9.1
File Edit Shell Debug Options Window Help
Python 3.9.1 (tags/v3.9.1:1e5d33e, Dec
D64)] on win32
Type "help", "copyright", "credits" or "l
>>>
= RESTART: E:\2023-2024 НАВЧАЛЬНИЙ РІК\І
ру
Перше число: 29
Друге число: 8
Друге число: 35
Введіть суму цих чисел: 71
Неправильно!
Перше число: 47
Друге число: 21
Друге число: 35
Введіть суму цих чисел: 103
Правильно!
Перше число: 39
Друге число: 17
Друге число: 28
Введіть суму цих чисел: н
Будь ласка, введіть число.
Перше число: 1
Друге число: 28
Друге число: 45
Введіть суму цих чисел: не знаю
Будь ласка, введіть число.
Перше число: 13
Друге число: 15
Друге число: 37
Введіть суму цих чисел:

Створити програму, яка за допомогою функції **random** буде виводити три числа в діапазоні від 0 до 50. Потім пропонує користувачу ввести результат додавання тих трьох чисел, тобто їх суму та результат віднімання цих трьох чисел, тобто їх різницю.

- *Якщо користувач ввів правильно*, то програма виводить фразу «правильно», у іншому випадку – «неправильно».
- Після виведення фрази (не важливо «правильно» чи «неправильно» *програма знову генерує нові числа* і знову просить користувача ввести їх суму, тобто вона працює безкінечно)