Проте після запуску програми і введення значень, наприклад 2 та 3, в результаті виконання програми отримаємо 23, хоча напевне нами очікувалося 5. Це пов'язане з тим, що за функцією `input()` повертається значення рядкового типу. А вказана операція «+» для змінних рядкового типу виконує їх об'єднання (детальна робота з рядковими змінними буде розглянута пізніше).

Отже необхідно отримані від функції `input()` рядкові значення перетворити в цілі або дійсні числа. Для цього можна скористатися функціями `int()` або `float()` відповідно.

Тоді програма матиме вигляд:

```
print('Введіть два числа')
a = int(input())
b = int(input())
s = a + b
print(s)
```

Якщо потрібно зчитати два числа записаних в рядку через пропуск, можна скористатися наступними командами:

```
a, b = input('Введіть два числа').split()
a, b = int(a), int(b)
```

або:

```
a, b = map(int, input('Введіть два числа').split())
```

3. Числові дані

Як було сказано раніше, в Python існує три вбудованих числових типи даних:

- цілі числа. Наприклад: 2, 3, 5. Цілі числа можуть бути довільної довжини, тобто у мові Python з цілими числами працюють в режимі довгої арифметики (можна обчислити значення 2^{2019});
- дійсні числа. Наприклад: 3.23, 52.3E-4 (E вказує степінь числа 10, в даному випадку $52.3*10^{-4}$);
- комплексні числа. Наприклад: (-5+4j), (2.3 - 4.6j).

3.1. Робота з цілими та дійсними числами

3.1.1. Математичні та бітові операції

Математичні операції

Символ операції	Призначення	Використання	Приклад
+	Додавання	x + y	2 + 3 результат 5 При a=3 та b=2 a + b результат 5
-	Віднімання	x – y	3 - 2 результат 1
*	Множення	x * y	2 * 3 результат 6
/	Ділення	x / y	4 / 2 результат 2.0 5 / 2 результат 2.5
//	Знаходження цілої частини від цілочисельного ділення	x // y	7 // 3 результат 2 7 // -3 результат -3 -7 // 3 результат -3 -7 // -3 результат 2
%	Знаходження залишку від цілочисельного ділення	x % y	7 % 3 результат 1 7 % -3 результат -2 -7 %3 результат 2 -7 % -3 результат -1 *Пояснення:* *c = a // b* *z = a % b, z має той же знак що і b таким чином, щоб a = c * b + z*
**	Піднесення до степеня	x**y	2**3 результат 8 4**0.5 результат 2.0

Якщо в якості операндів деякого арифметичного виразу використовуються тільки цілі числа, то результат теж буде ціле число, винятком є операція ділення, результатом якої буде дійсне число. При спільному використанні цілих і дійсних чисел результатом буде дійсне число.

Бітові операції над цілими числами

В мові Python передбачені бітові (побітові) операції над цілими числами. Такі операції називаються бітовими, тому що виконуються послідовно над окремими бітами операндів (побітово).

Символ операції	Призначення	Використання	Приклад
&	Бітове «І» (AND)	x & y	9&3 результат 1 *Пояснення:* $9_{10}=1001_2$ $3_{10}=0011_2$ *Після бітового* «AND» *отримаємо:* $0001_2=1_{10}$
|	Бітове «АБО» (OR)	x | y	9 | 3 результат 11 *Пояснення:* $9_{10}=1001_2$ $3_{10}=0011_2$ *Після бітового* «OR» *отримаємо:* $1011_2=11_{10}$
^	Бітове ВИКЛЮЧАЮЧЕ АБО (XOR)	x ^ y	9 ^ 3 результат 10 *Пояснення:* $9_{10}=1001_2$ $3_{10}=0011_2$ *Після бітового* «XOR» *отримаємо:* $1010_2=10_{10}$
~	Бітова операція НІ (NOT). Для числа x відповідає -(x+1)	~x	~9 результат -10
<<	Зсув вправо на кількість біт. До числа в двійковому записі справа дописується вказана кількість нулів	x << n	9 << 1 результат 18 *Пояснення* $9_{10}=1001_2$ *Після дописування справа одного 0 отримаємо:* $10010_2=18_{10}$
>>	Зсув вліво. З числа в двійковому записі забирається вказана кількість розрядів	x >> n	9 >> 1 результат 4 *Пояснення* $9_{10}=1001_2$ *Після прибирання справа одного символу отримаємо:* $100_2=4_{10}$

Короткий запис математичних та бітових операцій

Досить часто результат проведення певної математичної або бітової операції необхідно присвоїти змінній, над якою ця операція проводилася. Тобто необхідно модифікувати (змінити значення) змінну, нове значення якої буде залежати від її попереднього значення. Для цього в мові Python існують короткі форми запису виразів, тобто вираз виду «змінна = змінна операція вираз» прийме вигляд «змінна операція = вираз».

Так вираз: `a = a + 3`, перепишеться в вигляді `a += 3`.

Така коротка форма може бути застосовна до всіх зазначених вище математичних та бітових операцій.

3.1.2. Порядок обчислення операцій

Якщо маємо вираз виду 2 + 3 * 4, то з шкільного курсу математики відомо, що спочатку виконується операція множення, а вже потім операція додавання, оскільки операція множення має вищий пріоритет, ніж операція додавання.

Так само і в операціях мови Python, спершу обчислюються оператори і вирази з вищим пріоритетом, а потім поступово за спаданням пріоритету.

В таблиці наведено пріоритет операторів мови Python, починаючи з самого низького (зверху таблиці) і до найвищого (внизу таблиці).

Оператор	Опис
lambda	Лямбда-вираз (Лямбда-функція)
or	Логічне «АБО»
and	Логічне «І»
not x	Логічне «НІ»
in, not in	Перевірка приналежності
is, is not	Перевірка тотожності
<, <=, >, >=, !=, ==	Оператори порівняння
|	Бітове «АБО»
^	Бітове «ВИКЛЮЧАЮЧЕ АБО»
&	Бітове «І»
<<, >>	Зсуви
+, -	Додавання та віднімання
*, /, //, %	Множення, ділення, цілочисельне ділення та залишок від ділення
+x, -x	Додатне, від'ємне
~x	Бітове «НІ»

**	Піднесення до степеня
x.attribute	Посилання на атрибут
x[індекс]	Звернення за індексом
x[індекс1:індекс2]	Зріз
f(аргументи ...)	Виклик функції
(вираз, ...)	Кортеж (tuple)
[вираз, ...]	Список (list)
{ключ:дані, ...}	Словник (dict)

В таблиці оператори з рівним пріоритетом розміщені в одному рядку (наприклад, + та – мають рівний пріоритет).

Оператори, які ще не були розглянуті, будуть описані в наступних розділах.

Так само як і в математиці, в мові Python при формуванні математичних виразів для зміни порядку обчислень можуть використовуватися дужки. Наприклад (2 + 3) * 4.

3.1.3. Вбудовані функції цілих і дійсних чисел

В мові Python поряд з вбудованими типами є вбудовані функції, які містяться в стандартній бібліотеці і доступні без будь-яких додаткових вказівок. Розглянемо функції, що можуть бути застосовані до цілих та дійсних чисел.

`abs(X)` – повертає абсолютне значення (модуль) числа.

`divmod(A, B)` – повертає пару чисел (`P, R`), які є цілою частиною `P` та остачею `R` при виконанні цілочисельного ділення. Для цілих чисел результат буде таким самим, як і при (`A // B, A % B`). Для дійсних чисел результатом є (`Q, A % B`), де `Q` зазвичай `math.floor(A / B)`, але може бути і на 1 менше. Незважаючи на це значення за виразом `Q * B + A % B` дуже близьке до `A`, якщо `A % B` не рівне нулю, то має такий самий знак, як і `B`, `0 <= abs(A%B) < abs(B)`.

```
>>> divmod(7,2)
(3, 1)
```

`pow(X, Y[, Z])` – повертає `X` в степені `Y` за модулем `Z` (обчислюється більш ефективно, чим `pow(X, Y) % Z)`. Двоаргументна форма `pow(X, Y)` – еквівалент використання оператора піднесення до

степеня: `X**Y`. Якщо параметр `Z` заданий, то `X` та `Y` мають бути цілочисельними, окрім того `Y` має бути невід'ємним.

```
>>> pow(2,3)
8
>>> pow(2,3,3)
2
```

`round(number[, ndigits])` – повертає число `number`, округлене до `ndigits` знаків після десяткової точки.

Прете поведінка `round()` для дійсних чисел може бути несподіваною, це пов'язане з особливостями зберігання таких чисел в пам'яті в комп'ютера.

```
>>> round(2.65, 1)
2.6
>>> round(2.75, 1)
2.8
>>> round(2.85, 1)
2.9
>>> round (2.665, 2)
2.67
>>> round (2.675, 2)
2.67
```

Якщо параметр `ndigits` не заданий, то округлення відбувається до найближчого цілого числа. Проте, якщо дробова частина = 0.5, то виконується «Банківське округлення», тобто округлення до найближчого парного числа.

```
>>> round (2.5, 2)
2
>>> round (3.5, 2)
4
```

`max(arg1, arg2, *args, *[, key=func])` – повертає найбільше значення з двох чи більше аргументів. Ця функція має більш широкі можливості, але про них пізніше.

min(arg1, arg2, *args, *[, key=func]) – повертає найменше значення з двох чи більше аргументів. Ця функція має більш широкі можливості, але про них пізніше.

int([object], [osn]) – повертає перетворене значення `object` до цілого десяткового числа. `osn` визначає систему числення задання `object` (`osn` від 2 до 36 включно). За замовчуванням `object=0`, `osn=10`.

```
>>> int(4.9)
4
>>> int('11')
11
>>> int('11',2)
3
```

float([X]) – повертає перетворене `X` до дійсного числа.

```
>>> float('1.23')
1.23
>>> float('1e-003')
0.001
>>> float(3)
3.0
```

bin(X) – повертає рядковий запис цілого числа `X` в двійковій формі.

```
>>> bin(5)
'0b101'
```

hex(X) – повертає рядковий запис цілого числа `X` в шістнадцятковій формі. Для перетворення дійсного числа використовується метод `float.hex(X)`.

```
>>> hex(255)
'0xff'
>>> float.hex(3.4)
'0x1.b333333333333p+1'
```

oct(X) – повертає рядковий запис цілого числа `X` у вісімковій формі.

```
>>> oct(12)
'0o14'
```

bool([X]) – повертає приведене значення `X` до логічного типу (bool), використовуючи стандартну процедуру перевірки істинності. Повертає логічне значення True або False.

```
>>> bool(1)
True
>>> bool(15)
True
>>> bool(0)
False
```

3.1.4. Модуль math

Окрім стандартної бібліотеки з досить великим набором функцій, мова Python містить велику кількість додаткових бібліотек, які можуть бути використанні при написанні програм. Так однією з бібліотек, яка містить математичні функцій і призначена для роботи з числовими даними, є бібліотека (модуль) math.

Для роботи з цим модулем його попередньо необхідно імпортувати (підключити), виконавши команду import math. Більш детально про імпорт та під'єднання модулів подано в додатку 1.

Розглянемо константи та функції, що містяться в бібліотеці math.

Константи:

math.pi. Число π (`math.pi≈3.141592653589793`).

math.e. Число e (`math.e≈2.718281828459045`).

math.tau. Математична константа кола, рівна 2π.

math.inf. Додатна нескінченність (дійсне значення). Еквівалентно результату `float('inf')`. Для від'ємної нескінченності використовується: `-math.inf`.

math.nan. "не число" ("not a number" (NaN)). Еквівалент результату `float('nan')`.

Перевірка значень:

math.isclose(A, B, *, rel_tol=1e-09, abs_tol=0.0). Повертає `True`, якщо дійсні числа `A` та `B` близькі одне до одного, і `False` в

іншому випадку. Чи є числа близькими, визначається в відповідності з даними абсолютним і відносним відхиленням.

`rel_tol` (відносне відхилення) - це максимальна допустима різниця між `A` та `B`, відносно більшої абсолютної величини `A` або `B`. Наприклад, щоб встановити відхилення в 5%, вказується `rel_tol=0.05`. Відхилення за замовчуванням дорівнює `1e-09`, це гарантує, що два значення однакові приблизно в 9 десяткових цифрах. `rel_tol` має бути більше нуля.

`abs_tol` (абсолютне відхилення) - це мінімальне абсолютне відхилення. Використовується для порівнянь значень близьких до нуля. `abs_tol` має бути більше або рівне нулю.

Логічний вираз функції можна представити в вигляді:
`abs(a-b)<=max(rel_tol*max(abs(a), abs(b)), abs_tol).`

Значення `NaN` не вважається близьким до будь-якого іншого значення (навіть до себе). Значення `math.inf` та `-math.inf` вважаються близькими лише до себе.

`math.isfinite(X)`. Повертає `True`, якщо `X` не є ні нескінченністю (`math.inf` та `-math.inf`) ні `NaN`, і `False` в іншому випадку. Зауважимо, що `0.0` вважається скінченним.

`math.isinf(X)`. Повертає `True`, якщо `X` є нескінченністю, і `False` в іншому випадку.

`math.isnan(X)`. Повертає `True`, якщо `X` є `NaN`, і `False` в іншому випадку.

Функції округлення чисел:

`math.ceil(X)`. Повертає найближче ціле число, більше чим `X` (округлення вгору).

```
>>> math.ceil(3.1)
4
>>> math.ceil(-3.1)
-3
```

`math.floor(X)`. Повертає найближче ціле число, менше чим `X` (округлення вниз).

```
>>> math.floor(3.8)
3
```

```
>>> math.floor(-3.8)
-4
```

math.trunc(X). Повертає усічене значення X до цілого.

```
>>> math.ceil(3.8)
4
>>> math.ceil(-3.8)
-3
```

Функції представлення та теоретико-числові функції:

math.copysign(X, Y). Повертає дійсне число, що має абсолютне значення таке ж, як і у числа X, а знак - як у числа Y.

```
>>> math.copysign(3, -5)
-3.0
>>> math.copysign(-3, 5)
3.0
```

math.fabs(X). Повертає абсолютне значення числа.

```
>>> math.fabs(-7.3)
7.3
```

math.frexp(X). Повертає мантису і експоненту числа у вигляді пари (M, E), таким чином, що, X=M*2E. M – дійсне число (0,5 <= abs(M) < 1), а E - ціле число.

```
>>> math.frexp(7.3)
(0.9125, 3)
```

math.ldexp(X, I) – повертає X*2I. Функція, зворотна функції math.frexp().

```
>>> math.ldexp(0.9125, 3)
7.3
```

math.modf(X). Повертає дробову і цілу частину числа X у вигляді пари (D, C). Обидва числа є дійсними і мають той же знак, що і X.

```
>>> math.modf(5)
(0.0, 5.0)
>>> math.modf(4.5)
(0.5, 4.0)
```

```
>>> math.modf(-4.5)
(-0.5, -4.0)
```

math.fmod(X, Y). Повертає залишок від ділення X на Y. Для додатних чисел аналогічно оператору X % Y. Якщо серед чисел X, Y є від'ємні, то залишок обчислюється для абсолютних величин чисел X та Y, а результат матиме такий же знак, що і X. Окрім того, більш точно працює з дійсними числами.

```
>>> math.fmod(7.5, 3)
1.5
>>> math.fmod(-7.5, 3)
-1.5
>>> math.fmod(7.5, -3)
1.5
>>> math.fmod(-7.5, -3)
-1.5
```

math.factorial(X). Повертає факторіал числа X.

```
>>> math.factorial(5)
120
```

math.fsum(iterable). Повертає суму всіх членів з iterable. Аналог вбудованої функції sum(), але math.fsum() більш точна для дійсних чисел.

```
>>>sum([.1, .1, .1, .1, .1, .1, .1, .1, .1, .1])
0.9999999999999999
>>>math.fsum([.1, .1, .1, .1, .1, .1, .1, .1, .1,
.1])
1.0
```

math.gcd(A, B). Повертає найбільший спільний дільник цілих чисел A та B (найбільше натуральне число, на яке ці числа діляться без остачі). math.gcd(0, 0) повертає 0.

```
>>>math.gcd(6,8)
2
>>>math.gcd(5,7)
1
```

Степеневі та логарифмічні функції:

math.exp(X). Повертає значення e^X. Аналогічно до `math.e**X`.

math.expm1(X). Повертає значення $e^X - 1$. При $X \to 0$ точніше, ніж `math.exp(X)-1`.

math.log(X, [base]). Повертає значення логарифму числа `X` за основою `base`. Якщо `base` не вказано, обчислюється значення натурального логарифму.

```
>>> math.log(8)
2.0794415416798357
>>> math.log(8,2)
3.0
```

math.log10(X). Повертає значення логарифму числа `X` за основою `10`. Аналогічно до `log(x, 10)`.

```
>>> math.log10(8)
0.9030899869919435
```

math.log2(X). Повертає значення логарифму числа `X` за основою `2`. Аналогічно до `log(x, 2)`.

```
>>> math.log2(8)
3.0
```

math.log1p(X). Повертає значення натурального логарифму для `(1 + X)`. При $X \to 0$ точніше, ніж `math.log(1 + X)`.

math.pow(X, Y). Повертає дійсне значення X^Y. Для `pow(1.0, A)` та `pow(A, 0.0)` завжди повертається `1.0`, навіть якщо `A` рівне нулю або `NaN`. Якщо `X` та `Y` скінченні, `X` від'ємне, а `Y` не є цілим числом, тоді результат `pow(X, Y)` невизначений.

```
>>> math.pow(3, 2)
9.0
>>> math.pow(-3, 3)
-27.0
```

math.sqrt(X). Повертає значення квадратного кореня числа `X`.

Функції кутових перетворень:

math.degrees(X). Конвертує радіани в градуси.

```
>>> math.degrees(math.pi/4)
```

```
45.0
```

math.radians(X). Конвертує градуси в радіани.

```
>>> math.radians(45)
0.7853981633974483
```

Тригонометричні функції:

math.cos(X). Повертає значення косинусу числа (X вказується в радіанах).

math.sin(X). Повертає значення синусу числа (X вказується в радіанах).

math.tan(X). Повертає значення тангенсу числа (X вказується в радіанах).

math.acos(X). Повертає значення арккосинусу числа в радіанах.

math.asin(X). Повертає значення арксинусу числа в радіанах.

math.atan(X). Повертає значення арктангенсу числа в радіанах.

math.atan2(Y, X). Повертає значення арктангенсу Y/X в радіанах. З урахуванням чверті, в якій знаходиться точка (X, Y), тобто результат в межах від –pi до pi.

```
>>> math.degrees(math.atan2(1, 1))
45.0
>>> math.degrees(math.atan2(-1, 1))
-45.0
>>> math.degrees(math.atan2(-1, -1))
-135.0
>>> math.degrees(math.atan2(1, -1))
135.0
```

math.hypot(X, Y). Повертає значення гіпотенузи трикутника з катетами X та Y (аналогічно до math.sqrt(x * x + y * y)).

Гіперболічні функції:

math.cosh(X). Повертає значення гіперболічного косинусу числа.

math.sinh(X). Повертає значення гіперболічного синусу числа.

math.tanh(X). Повертає значення гіперболічного тангенсу числа.

math.acosh(X). Повертає значення оберненого гіперболічного косинусу числа.

math.asinh(X). Повертає значення оберненого гіперболічного синусу числа.

math.atanh(X). Повертає значення оберненого гіперболічного тангенсу числа.

Спеціальні функції:

math.erf(X). Повертає значення функції помилок для X. Функція помилок - це неелементарна функція, що використовується в теорії ймовірності, статистиці і математичній фізиці. Вона визначається як: $erf(x) = \frac{2}{\sqrt{\pi}} \int_0^x e^{-t^2} dt$.

math.erfc(X). Повертає значення додаткової функції помилок для X (аналогічно до `1 - math.erf(X)`).

math.gamma(X). Повертає значення гамма-функції для X. Гамма-функція визначається формулою: $\Gamma(x) = \int_0^\infty s^{x-1} e^{-s} ds = \int_0^1 \left(\ln \frac{1}{s} \right)^{x-1} ds$.

math.lgamma(X). Повертає значення натурального логарифму гамма-функції для X.

Більш детально з функціями модуля math можна ознайомитися на сторінці офіційної документації:
https://docs.python.org/3/library/math.html?highlight=math#angular-conversion

3.2. Приклади розв'язування задач

Приклад. Написати програму для розрахунку ідеальної ваги чоловіка та жінки за формулою Брокка.
Ідеальна вага для чоловіка = (зріст в сантиметрах - 100) * 1,15.
Ідеальна вага для жінки = (зріст в сантиметрах - 110) * 1,15.

```
zrist=int(input('Ваш зріст в сантиметрах:'))
v_men=(zrist-100)*1.15
v_women=(zrist-110)*1.15
```

```
print('Ідеальна вага для чоловіка = ', v_men)
print('Ідеальна вага для жінки = ', v_women)
```

Приклад. Написати програму обчислення значення виразу $e^{x+y} + \frac{5}{\cos(y-x)+3}$.

```
import math
x=float(input('X= '))
y=float(input('Y= '))
res=math.exp(x+y)+5/(math.cos(y-x)+3)
print('Знаяення виразу = ', res)
```

3.3. Робота з комплексними числами

Для задання комплексного числа можна використати запис A + Bj.

```
>>>z=1+2j
>>>print(z)
(1+2j)
```

Інший спосіб задання комплексного числа – це використання функції `complex([real[, imag]])`, за якою формується число з значенням `real + imag*j`. Якщо перший параметр є рядком (другий параметр має бути відсутній), то він буде вважатися рядковим записом комплексного числа, і за функцією `complex()` буде сформоване комплексне число, яке відповідатиме даному рядковому значенню. При перетворенні із рядка, він не має містити пропусків навколо центрального оператора + або -.

```
>>>x=complex(3, 2)
>>>x
(3+2j)
>>>complex('3-2j')
(3+2j)
```

Над комплексними числами можна виконувати операції: додавання, віднімання, множення, ділення і піднесення до степеня.

```
>>> x+z
(4+4j)
>>> x-z
```