СОДЕРЖАНИЕ

Что такое вещественное число?

Вещественные числа, или числа с плавающей запятой, — это числа, которые могут содержать дробную часть. Например, 3.14, 0.5, -2.7 — это все вещественные числа. В отличие от целых чисел, которые не могут иметь дробную часть, вещественные числа позволяют более точно описывать величины, особенно в контексте вычислений, таких как измерение времени, расстояния и так далее.

Чтобы создать вещественное число в Python, достаточно использовать **точку** в записи числа:

```
number = 3.14
```

При работе с вещественными числами Python использует тип данных float, который хранит значения с определённой точностью (до 15-17 знаков после запятой). Этот тип подходит для большинства вычислений, связанных с вещественными числами.

Важный момент:

Когда вы решаете задачи с вещественными числами, при считывании данных с клавиатуры нужно использовать **float(input())**, а не int(input()), так как программа ожидает вещественное число. Если пользователь введёт целое число, оно автоматически преобразуется в вещественное, добавив дробную часть с нулём. Пример:

```
number = float(input())
```

Если используется int(input()), программа будет ожидать строго целое число, и если пользователь введёт вещественное, возникнет ошибка.

Встроенные математические функции

Python предоставляет несколько встроенных функций для работы с числами. Рассмотрим две полезные функции: round() и abs().

round() — округляет число до указанного количества знаков после запятой. Если количество знаков не указано, то округляется до ближайшего целого:

```
number = 3.14159
rounded_number = round(number, 2)  # Округление до двух знаков
print(rounded_number)  # Вывод: 3.14
```

abs() — возвращает абсолютное значение числа, то есть положительное число:

```
number = -5.7
positive_number = abs(number)
print(positive_number)  # Вывод: 5.7
```

PEP8: Скобки в вызовах функций round() и abs() пишутся без пробелов.

Что такое модуль и как использовать math

Модули — это библиотеки кода, которые содержат функции и переменные, упрощающие выполнение различных задач. Они позволяют не писать код с нуля, а использовать готовые решения. Для работы с расширенными математическими функциями Python предоставляет модуль **math**.

Чтобы использовать функции из этого модуля, его нужно сначала подключить:

```
import math
```

Теперь для вызова любой функции из этого модуля необходимо использовать синтаксис **math.<имя функции>**. Пробелы между именем модуля и точкой, а также между точкой и названием функции ставить нельзя:

```
result = math.sqrt(16)  # Извлечение квадратного корня из 16
print(result)  # Вывод: 4.0
```

Одной из популярных констант в модуле math является число пи. Оно представлено как math.pi и может быть использовано для вычислений:

```
radius = 5
area = math.pi * (radius ** 2)  # Площадь круга
print(area)  # Вывод: 78.53981633974483
```

Таблица функций модуля math

Функция	Описание	Пример в программе	Пример вычисления
math.pow(x, y)	Возводит число x в степень y	math.pow(2, 3)	2 ** 3 = 8
math.sqrt(x)	Возвращает квадратный корень из x	math.sqrt(25)	√25 = 5
math.sin(x)	Синус угла x (в радианах)	math.sin(math.pi / 2)	sin(π/2) = 1
math.cos(x)	Косинус угла x (в радианах)	math.cos(0)	cos(0) = 1
math.tan(x)	Тангенс угла x (в радианах)	math.tan(math.pi / 4)	tan(π/4) = 1

PEP8: Операции вызова функций должны быть написаны без пробелов между именем модуля, точкой и именем функции.

Работа с радианами и градусами

Функции sin(), cos() и tan() работают с углами в радианах, а не в градусах. Чтобы перевести градусы в радианы, можно использовать формулу:

$$\text{Радианы} = \text{Градусы} \times \frac{\pi}{180}$$

Обратная формула для перевода радианов в градусы:

$$\text{Градусы} = \text{Радианы} \times \frac{180}{\pi}$$

Python предоставляет удобные функции для этих преобразований:

- math.radians(углы_в_градусах) — переводит градусы в радианы.
- math.degrees(углы_в_радианах) — переводит радианы в градусы.

```
import math

degrees = 180
radians = math.radians(degrees)  # Перевод 180 градусов в радианы
print(radians)  # Вывод: 3.141592653589793

# Обратное преобразование
radians = math.pi
degrees = math.degrees(radians)  # Перевод радианов в градусы
print(degrees)  # Вывод: 180.0
```

Округление вещественного числа

Python позволяет округлять вещественные числа до определённого количества знаков после запятой с помощью функции **format()** или **f-строк**. Например:

```
number = 3.14159
formatted_number = format(number, '.2f')
print(formatted_number)  # Вывод: 3.14
```

Функция format() возвращает строку, поэтому её удобно использовать при выводе через print().

Функция format() возвращает строку (тип данных **str**), а не число, даже если она применяется к числовым значениям. Это значит, что результат работы функции format() подходит для вывода информации, но не для дальнейших математических операций. Если вам нужно продолжить вычисления с результатом, нужно использовать саму переменную с числом, а не форматированную строку.

```
number = 3.14159
formatted_number = format(number, '.2f')  # Получаем строку '3.14'
print(formatted_number)  # Вывод: 3.14

# Попробуем сложить форматированное число с другим числом
result = formatted_number + 1  # Ошибка: TypeError: can only concatenate
```

Как видно из примера, мы получаем ошибку, потому что formatted_number является строкой и не может участвовать в арифметических операциях. Поэтому для математических вычислений всегда используйте исходное число.

Пример с f-строкой:

```
number = 3.14159
print(f"{number:.2f}")  # Вывод: 3.14
```

Таким образом, format() и f-строки лучше использовать для вывода, а не для последующих расчётов.