

Ввод и вывод данных

01

print()

Напишем нашу первую программу. Наша первая программа будет выводить на экран надпись "Hello World!". Для того, чтобы вывести какую либо информацию в Python, необходимо воспользоваться встроенной функцией print()

Функция print() принимает, одно или несколько значений, поэтому в качестве аргумента внутри круглых скобок через запятую мы указываем то, что хотим вывести на экран. Если хотим вывести текст, указываем его в кавычках. Кавычки могут быть одинарными или двойными, но текст должны открывать и закрывать кавычки одного вида

Пример

```
...  
print("Hello World!")
```

Результат работы программы

```
...  
Hello World!
```

Выведем сразу несколько значений в одной строке

Пример

```
...  
name = "John"  
age = 25  
print("My name is", name, "and I am", age, "years old.")
```

Вывод

```
...  
My name is John and I am 25 years old
```

Также у функции print() есть два именованных аргумента - sep и end. При помощи аргумента sep (separate) можно настроить разделитель между значениями. При помощи аргумента end можно настроить, что должно быть выведено после последнего значения

Рассмотрим функцию print(), которая позволяет выводить на экран несколько значений, перечисленных через запятую

Пример

```
...  
name = "user"  
print("Hello, ", name, "!")
```

Вывод

```
...  
Hello, user !
```

При выполнении кода мы замечаем, что между именем пользователя и восклицательным знаком появляется лишний пробел. Это происходит потому, что по умолчанию между значениями ставится пробел. Чтобы избежать этого, мы можем использовать именованный аргумент `sep`, указав в качестве разделителя пустую строку (после "Hello, " внутри кавычек ставим пробел, иначе "Hello," и "user" будет написано слитно)

Пример

```
...  
name = "user"  
print("Hello, ", name, "!", sep="")
```

Вывод

```
...  
Hello, user!
```

Теперь в консоли выводится сообщение без лишних пробелов

Аргумент `end` используется встроенной функцией `print()` для указания, какой символ должен быть добавлен в конце строки после вывода значения

Пример

```
...  
print("Hello", end=" ")  
print("world!", end=" ")  
print("Welcome to", end=" ")  
print("Python programming!")
```

Вывод

```
...  
Hello, world! Welcome to Python programming!
```

02

input()

Команда `input()` запрашивает от пользователя ввод значения с клавиатуры. Ввод завершается нажатием клавиши `Enter`

Пример

```
...  
name = input("What is your name?")
```

Данная команда попросит пользователя ввести с клавиатуры свое имя и сохранит его в переменную `name`. Функция `input()` всегда возвращает строку, поэтому если нужно получить число от пользователя, необходимо использовать функцию `int()` или `float()` для преобразования строки в число

03

Переменные и оператор присваивания

Переменная

это величина, имеющая имя, тип и значение.
Значение переменной можно изменять
во время работы программы

Имя переменной должно отражать ее смысловое назначение. Имя может состоять из латинских букв, цифр и символа подчеркивания. Имя не может начинаться с цифры. В предыдущем примере в качестве имени переменной используется слово `name`

Значение переменной

то, что сохраняет в себе переменная

Оператор присваивания обозначается знаком “`=`”. Оператор присваивания присваивает переменной значение, которое находится справой стороны от него после знака “`=`”

Пример

```
...  
c = 14  
print(c)
```

Вывод

```
...  
14
```

В данном примере переменной `c` было присвоено значение `14` и выведено на экран. На экран можно выводить не только строки, но числовые значения

В Python комментарии используются для добавления пояснений, описания или заметок в коде, которые не будут выполняться как часть программы. Комментарии помогают программистам понять код, объясняют логику или функционал определенных частей программы и облегчают сопровождение и отладку кода.

В Python есть два способа создания комментариев:

01 Однострочные комментарии (# ...)

Они используются для создания комментариев, занимающих только одну строку. Однострочные комментарии начинаются с символа решетки (#) и после него идет текст комментария

Пример

```
...  
# Это однострочный комментарий  
x = 5 # Этот комментарий объясняет значение переменной x
```

02 Многострочные комментарии (""" ... """)

Они используются для создания комментариев, занимающих несколько строк. Многострочные комментарии окружаются тройными кавычками (можно использовать как одинарные, так и двойные кавычки) в верхней и нижней части комментария

Пример

```
...  
"""  
Это многострочный комментарий, который может занимать несколько строк.  
Здесь можно добавить подробное описание кода или заметки.  
"""  
x = 10  
"""  
Этот комментарий объясняет значение переменной x, которое равно 10.  
"""
```

Комментарии играют важную роль для других программистов, которые могут читать, поддерживать или сотрудничать над вашим кодом. Рекомендуется добавлять комментарии к сложным участкам кода или критическим моментам, чтобы облегчить понимание вашей логики и намерений

Тип данных

это специальная характеристика переменной, которая определяет, какую конкретную информацию можно хранить в ней и какие операции можно выполнять с этой информацией

Например, тип данных "число" указывает программе, что это число, а тип данных "строка" указывает, что это текстовая информация. Разные типы данных имеют разные свойства и методы работы, поэтому это важно учитывать при написании программ

Основные типы данных в Python

01 Числа (int, float, complex)

целые числа, числа с плавающей точкой и комплексные числа

Пример

```
...
x = 5    # целое число (int)
y = 3.14  # число с плавающей точкой (float)
z = 2 + 3j # комплексное число (complex)
```

02 Строки (str)

последовательности символов

Пример

```
...
name = "John" # строка (str)
message = 'Hello, World!' # также строка (str)
```

02 Логический тип данных "bool" и имеет только два возможных значения: "True" (истина) и "False" (ложь)

Логический тип данных обычно используется для логических операций или для проверки условий в программе

Пример

```
...
x = True # истина (bool)
y = False # ложь (bool)
```

В Python есть и другие типы данных, такие как: списки, кортежи, множества, словари. Поговорим о них позже

Для определения типа данных можно использовать функцию `type()`. Пример использования этой функции:

Пример

```
...
x = 5 # целое число (int)
y = 3.14 # число с плавающей точкой (float)
z = 2 + 3j # комплексное число (complex)
```

Вывод

```
...
<class 'int'>
```

06

Изменение значений переменных

В Python значения переменных можно изменять путем присваивания нового значения существующей переменной. Рассмотрим несколько примеров изменения значений переменных в Python

Примеры

```
...
x = 5 # Инициализация переменной x со значением 5
x = 10 # Изменение значения переменной x на 10
print(x) # Выводит 10

y = "Hello" # Инициализация переменной y со значением "Hello"
y = y + " world!" # Конкатенация строки " world!" с текущим значением переменной y
print(y) # Выводит "Hello world!"

z = 5 # Инициализация переменной z со значением 5
z += 3 # Увеличение значения переменной z на 3 (аналогично z = z + 3)
print(z) # Выводит 8

w = 7 # Инициализация переменной w со значением 7
w -= 2 # Уменьшение значения переменной w на 2 (аналогично w = w - 2)
print(w) # Выводит 5
```

Обратите внимание, что при изменении значения переменной, старое значение полностью заменяется новым значением

В Python вы можете выполнять арифметические выражения с помощью операторов и функций, предоставляемых языком

Пример

...	
a = 5 + 2	# Сложение, результат: 7
b = 4 - 1	# Вычитание, результат: 3
c = 3 * 2	# Умножение, результат: 6
d = 10 / 3	# Деление, результат: 3.333333333333335
e = 10 // 3	# Целочисленное деление, результат: 3
f = 10 % 3	# Остаток от деления, результат: 1
g = 2 ** 3	# Возведение в степень, результат: 8

()	Скобки
**	Возведение в степень
*, /, //, %	Умножение, деления, взятие остатка
()	Скобки
**	Возведение в степень
+, -	Сложение и вычитание
==, !=, >, ≥, <, ≤, in, not in	Сравнение, проверка вхождения
not	Логическое НЕ
and	Логическое И
or	Логическое ИЛИ

Пример

...	
a = (5 + 3) * 2	# Результат: 16, выполнит сначала сложение, а потом умножение
b = 10 / (4 - 2)	# Результат: 5.0, выполнит сначала вычитание, а затем деление

Библиотека `math` для математических вычислений

Библиотека `math` в Python предоставляет функции для математических операций и операций с числами. Она входит в стандартную библиотеку Python и не требует отдельной установки.

Для использования библиотеки `math` в Python, вам нужно сначала импортировать ее с помощью команды `import math`. Рассмотрим некоторые функции этой библиотеки.

Пример

Математические функции

```
...
import math
x = math.sqrt(16)          # Квадратный корень, результат: 4.0
y = math.sin(math.pi / 2)  # Синус, результат: 1.0
z = math.cos(math.pi)      # Косинус, результат: -1.0
```

Математические константы

```
...
import math
pi = math.pi               # Число Пи, результат: 3.141592653589793
e = math.e                  # Число е (основание натурального логарифма),
                           # результат: 2.718281828459045
```

Другие функции

```
...
import math
x = math.ceil(4.1)          # Округление вверх, результат: 5
y = math.floor(4.9)         # Округление вниз, результат: 4
z = math.factorial(5)       # Факториал числа, результат: 120
```

Библиотека `math` предоставляет множество других функций и констант для выполнения различных математических операций и вычислений. Документация Python содержит полный список функций и подробное описание их использования.

Итог

В данной теме мы разобрали способы ввода и вывода информации, понятие переменной, виды типов данных, рассмотрели способы работы с переменными.