Язык программирования Python

07 Ввод-вывод

Файловый режим

Для продолжения изучения основ Python перейдем к файловому режиму работы.

При работе в этом режиме весь код, находящийся в файле, запускается с первой строки до последней.

В интерактивном режиме результат написанного выражения выводился на следующей строке.

В файловом режиме для вывода данных необходимо использовать функции вывода.

Вывод данных

Для вывода данных используется функция print. С помощью print мы можем выводить значения выражений и переменных. То, что мы желаем напечатать на экране, помещаем внутри скобок.

```
a = 5
b = 10

print(100) # 100
print(a) # 5
print(b) # 10
print(a + b) # 15

print('hello') # hello
```

Вывод данных

Причем посредством функции print можно выводить и несколько значений. Для этого переменные и выражения надо перечислить внутри скобок через запятую.

```
a = 5
b = 10

print(100, 200) # 100 200
print(a, b) # 5 10
print(b, a) # 10 5
print(a + b, a – b, a * b, a / b) # 15 –5 50 0.5

print(‘hello’, ‘world’) # hello world
```

Разделитель

При печати нескольких значений, записанных через запятую, по умолчанию функция print разделит их пробелом. Мы можем изменить разделитель на любой другой символ или последовательность символов.

Для этого после перечисления данных для печати после запятой необходимо параметру sep (от англ. separator — разделитель) задать другое значение.

```
print(10, 20, 30, sep = '/') # 10/20/30

print('hello', 'world', sep = '\n')
hello
world
```

Конец строки

По умолчанию при печати с помощью функции print строка заканчивается переходом на новую строку (‘\n’).

Мы можем изменить и это.

```
print(10, 20, end = ‘end’) # 10 20end

print(‘hello’, end = ‘_world’) # hello_world
```

Преобразование типов

Возможность преобразовать значение одного типа в другой бывает очень полезной.

Полезно, например, число представить как строку и наоборот.

Для этого используются функции преобразования типов, названия которых совпадают с названиями типов.

```
int('100') # Преобразует строку '100' в целое число 100.

str(200) # Преобразует целое число 200 в строку '200'
```

Вывод данных

```
print() # Переход на новую строку

a = 1
b = 2
print('a + b = ', a + b) # a + b = 3

print(a, '+', b, '=', a + b) # 1 + 2 = 3

print(str(a) + ' + ' + str(b) + ' = ' + str(a + b)) # 10 +
20 = 30

print(str(2 + 2) * int('1' + '0')) # 4444444444
```

Ввод данных

Ввод данных позволяет очень многое. Он создает интерактив, не нужно лезть в код, чтобы исправлять параметры, необходимые для вычислений и еще много-много причин, почему он важен.

Для считывания вводимой нами строки используется функция input. Она считывает строку с клавиатуры и возвращает значение этой строки.

Возвращаемое значение можно сразу присвоить переменной.

```
message = input()
```

Введенные вами символы до нажатия Enter будут доступны через переменную message.

Ввод данных

Важно, что функция print всегда возвращает именно строку.

Даже если вы введете число, вернется строка. При попытке сложить, предположим 2 введенных вами таких «числа» произойдет конкатенация, а не сложение.

```
a = input() # a = ‘1’
b = input() # b = ‘9’

print(a + b) # ‘19’ (кавычки поставлены для понимания)
print(int(a) + int(b)) # 10
```

Ввод данных

Мы также можем объединить считывание строки и преобразование типов.

```
a = int(input()) # a = 1
b = int(input()) # b = 9
```

Теперь в переменных a и b будут храниться целые числа, а не строки.

Итоги

Мы познакомились с такими понятиями, как ввод данных, вывод данных, разделитель, конец строки, преобразование типов,

а также попрактиковались в работе с интерпретатором в файловом режиме.