Функции print() и input(). Преобразование строк в числа int() и float()

На этом занятии речь пойдет о двух распространенных функциях:

- print() – вывод данных в консоль;
- input() – ввод данных из стандартного входного потока (часто клавиатуры).

Функция print()

О функции print() мы с вами уже немного говорили, и вы видели ее использование в самых простых ситуациях, например:

```
print(1)
```
или
```
print(3, 5, 7)
```

выведет указанные значения через пробел. Также можно выводить отдельные переменные:

```
a = -6.84
print(a)
```

результаты арифметических операций:
```
print(a * 2 + 3)
```

или функций:
```
print(abs(a * 2 + 3))
```

Но у этой функции имеется два необязательных именованных параметра, которые довольно часто используются в практике программирования:

sep – разделитель между данными;
end – завершающий символ или строка.

Давайте я покажу на примере работу этих параметров. Предположим, мы выводим три переменные:

```
a = -6.84
b = 7
c = 25.6
print(a, b, c)
```

Как видите, между ними автоматически добавляется пробел. Но этот символ можно поменять через параметр sep, например, так:
```
print(a, b, c, sep=" | ")
```

Соответственно, чтобы параметр sep «сработал», необходимо хотя бы два аргумента в функции print(). Если указать, например, один:
```
print(a, sep=" | ")
```

то здесь разделять нечего и он будет проигнорирован. А вот для двух переменных уже появится вертикальная черта:
```
print(a, b, sep=" | ")
```

Второй параметр end задает окончание строки вывода и по умолчанию: end = '\n' переводу на следующую строку. Это такой спецсимвол, о которых мы также еще будем говорить. Так вот, благодаря такому параметру end два последовательных вызова функции print() напечатают текст с новой строки:

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

Но если в первой функции print() добавить этот параметр с пробелом:

```
print("Hello", end=' ')
print("World!")
```

то увидим в одной строчке оба слова. Разумеется, здесь параметр end с пробелом применяется только к первому print(). У второго он уже берется по умолчанию с переносом строки.

Последнее, что я хочу рассказать о функции print() – это способ вывода форматированной информации в консоль. Давайте предположим, что у нас имеются две переменные (координаты точки) x и y:

```
x = 5.76
y = -8
```

И мы хотим вывести их в формате: «Координаты точки: x = 5.76; y = -8»

Сделать это можно несколькими способами. Первый, самый очевидный, записать все через запятую: print("Координаты точки: x = ", x, '; y = ', y)

Но, начиная с версии Python 3.6, появилась возможность использовать специальные F-строки. О них мы также еще будем говорить, но здесь я приведу простой пример и вы уже сейчас сможете применять этот механизм в своих программах. Запишем функцию print(), следующим образом:

```
print(f"Координаты точки: x = {x}; y = {y}")
```

Смотрите, здесь перед строкой ставится специальный символ f, указывающий, что это будет F-строка. А, в самой строчке внутри фигурных скобок мы можем записывать любые конструкции языка Python. В данном случае, я просто указал переменные x и y. Видите, как это просто, наглядно и удобно. Сейчас практически всегда используются F-строки для форматированного вывода информации.

Вторая функция input() служит для ввода информации, как правило, с клавиатуры. В самом простом варианте ее можно вызвать так:

```
a = input()
```

И это важный момент: функция input() всегда возвращает строку. На что это может повлиять? Например, мы хотим вычислить модуль введенного числа:

```
a = input()
b = abs(a)
```

При вызове функции abs() возникнет ошибка, так как в качестве ее аргумента должно быть число, а не строка. Как решить эту проблему? Очень просто. Если мы знаем, что пользователь должен ввести число, предположим, целое число, то можно воспользоваться функцией:

```
a = "54"
b = int(a)
```

Теперь b будет ссылаться на число 54, а не строку. И наша программа примет вид:

```
a = input()
a = int(a)
b = abs(a)
print(b)
```

Здесь первые две строчки можно объединить и записать их так:

```
a = int(input())
```

Но функция int() преобразовывает только целые числа. Если в строке будет хотя бы один не цифровой символ: `int("64.56")`

возникнет ошибка. То есть, в нашей программе пользователь обязательно должен вводить целые числа. А как тогда преобразовывать вещественные значения? Для этого есть другая функция: `float("64.56")`

Поэтому, когда на входе ожидаются вещественные данные, то=следует использовать именно ее:

```
a = float(input())
b = abs(a)
print(b)
```

Давайте напишем программу для вычисления периметра= прямоугольника. Пользователь будет вводить два числа (стороны= прямоугольника), а мы, затем, вычислим периметр:

```
a = float(input())
b = float(input())
print("Периметр:", 2 * (a + b))
```

Как видите, все достаточно просто. Мы вводим первое число,=нажимаем Enter и вводим второе число. Вводить два числа через=пробел здесь нельзя, иначе получится строка с двумя числами,=разделенные пробелом и ее нельзя будет преобразовать функциями=int() и float().

Также в этой программе пользователю совершенно непонятно, что=нужно вводить. Давайте добавим подсказки. Для этого в функции=input() первым аргументом передается строка:

```
a = float(input("Введите длину пр
b = float(input("Введите ширину п
print("Периметр:", 2 * (a + b))
```