Программы на языке Питон представляют собой обычные текстовые файлы, в которых записана последовательность команд. Код легко читается и интуитивно понятен.

Например, программы выводящая Hello, world! записывается всего в одну строку:

```
print('Hello, world!')
```

В этой программе вызывается функция печати print, которой в качестве параметра передается строка, содержащая в себе фразу Hello, world!. Если мы хотим задать какую-то строку, то должны обрамлять её одинарными (') или двойными(") кавычками, иначе она будет интерпретироваться как код на языке Питон.

Кроме строк в сегодняшнем занятии мы рассмотрим целочисленный тип данных. Например, можно посчитать результат вычисления арифметического выражения 2 + 3 и вывести его с помощью такой однострочной программы на языке Питон:

```
print(2 + 3)
```

Такая программа выведет результат вычисления выражения, который будет равен 5. Если бы числа 2 и 3 были заключены в кавычки, то они интерпретировались бы как строки, а операция + проводила бы конкатенацию (склеивание) строк. Например, такой код:

```
print('2' + '3')
```

выведет 23 - строку, состоящую из склеенных символов '2' и '3'.

Функция print может принимать и несколько параметров, тогда они будут выводиться через пробел, причем параметры могут иметь различные типы. Если мы хотим получить вывод вида 2 + 3 = 5, то можем воспользоваться следующей программой:

```
print('2 + 3 =', 2 + 3)
```

Обратите внимание, что в строке '2 + 3 =' нет пробела после знака =. Пробел появляется автоматически между параметрами функции print. Что же делать, если хочется вывести строку вида 2+3=5 (без пробелов)? Для этого понадобится именованный параметр sep (separator, разделитель) для функции print. Та строка, которая передается в качестве параметра sep будет подставляться вместо пробела в качестве разделителя. В этой задаче мы будем использовать пустую строку в качестве разделителя. Пустая строка задается двумя подряд идущими кавычками.

```
print('2+3=', 2 + 3, sep='')
```

В качестве параметра sep можно использовать любую строку, в том числе состоящую из нескольких символов. Если нам нужно сделать несколько разных разделителей для разных частей строк, то не остается другого выбора, кроме как использовать несколько подряд идущих функций print. Например, если мы хотим вывести строку вида 1 + 2 + 3 + 4 = 10, то можем попробовать воспользоваться следующим кодом:

```
print(1, 2, 3, 4, sep = ' + ')

print(' = ', 1 + 2 + 3 + 4, sep = '')
```

Однако, вывод такого кода нас огорчит. Он будет выглядеть как:

1 + 2 + 3 + 4

= 10

Это связано с тем, что после каждой функции print по умолчанию осуществляется перевод строки. Для изменения того, что будет печататься после вывода всего, что есть в функции print можно использовать именованный параметр end. Например, в нашем случае после первого print мы не хотели бы печатать ничего. Правильный код выглядит следующим образом:

```
print(1, 2, 3, 4, sep=' + ', end='')

print(' = ', 1 + 2 + 3 + 4, sep='')
```

В качестве end также можно использовать абсолютно любую строку.