**import this** — импортировать это ь это»

****Функция**** — это подпрограмма, выполняющая определённые действия. **Функция print()** — это встроенная в Python подпрограмма для вывода данных на экран (или для «печати на экране», на жаргоне программистов).

'Привет, Мир!' — это текст, который функция print() напечатает. **Функция print()** выводит на экран любые данные, указанные в скобках после названия функции.

Программисты называют такие данные **«**аргументы**»**; их (аргументы, а не программистов!) указывают в скобках после имени функции.

Объединение нескольких строк в одну называют «**конкатенацией**» (от лат. concatenatio, «присоединение, сцепление»).

А вот сложить число и строку так просто не удастся: складывать можно значения одинакового типа, иначе Python выдаст ошибку.

Преобразованием данных из одного типа в другой занимаются специальные функции. Так, для превращения числа в строку вызывают **функцию** **str()- string (строка)**

А преобразовать строку в целое число можно **функцией int()**.

Сообщения об ошибках называются ****traceback**** (по-русски говорят «трейсбэк»)

сообщение ****SyntaxError****, «ошибка синтаксиса»: при сложении забыли написать одно слагаемое.

Сообщение **invalid syntax** переводится как «недопустимый синтаксис».

**SyntaxError: EOL while scanning string literal**; сокращение ****EOL**** означает end of line, а полный текст сообщения переводится как «конец строки во время обработки текста». Python увидел открывающую кавычку, но строчка закончилась, а закрывающей кавычки не было. На этом Python сломался.

**Как выбрать имя переменной?**

Составляйте названия переменных из английских слов. Когда ваша программа вырастет до многих сотен строк, названия-слова помогут быстро сориентироваться в коде.

Сравните сами: переменную с названием урока можно назвать буквой l, а можно — словом lesson. Второй вариант лучше: ведь то, что когда-то разработчик сократил lesson до одной буквы, со временем забудут и он, и его коллеги. Код станет сложнее для чтения.

Не рекомендуется использовать русские слова в английской раскладке. Рано или поздно ваш код будет читать человек, не владеющий русским. Он может не понять, что к чему. Сразу называйте переменные по-английски: child, а не транслитерацией — rebyonok.

Запятая между аргументами по умолчанию заменяется на пробел.

Числа бывают целыми и дробными. Для десятичных дробей (иначе их называют «числа с плавающей запятой») в Python есть специальный тип данных — **float**.

PYTHON

Можно преобразовать строку в дробное число, для этого есть функция float():

Дробные числа приводят к целым функцией int(). Обратите внимание: int() **не округляет** числа по правилам арифметики, а просто отбрасывает дробную часть.

В англоязычной литературе запятая называется точкой, floating point, потому что десятичные дроби пишутся через точку.

# Списки

**Списки** — это последовательности чисел, строк или каких-то ещё значений.

Вы уже присваивали переменным числовые и строковые значения, точно так же переменная может содержать и список.

Содержимое списка пишется в квадратных скобках, элементы списка разделяются запятой:

**new\_list = [<элемент>, <элемент>, <элемент>, <элемент>, <элемент>]**

У каждого элемента есть свой порядковый номер — **индекс**. С помощью индекса можно получить значение элемента списка

Для подсчёта элементов списка есть стандартная функция len().

Оператор if может включать блок кода, который выполнится в том случае, если условие ложно. Для этого существует конструкция if / else.

Если условное выражение истинно, выполняется код из блока if, а если условие ложно — сработает код из блока else.

if <условие>:

<код, который выполнится, если условие истинно>

*# Внимание: этот блок имеет отступ в 4 пробела от начала строки*

else:

<код, который выполнится, если условие ложно>

*# Этот блок тоже отбит от начала строки 4 пробелами*

**elif** — это вложенная конструкция для if: «если условие для if не выполнено, но выполняется условие для elif — выполнить код в блоке elif»

Как и для оператора if, в elif должно быть установлено условие, логическое выражение, которое вернёт True или False.

# Логические выражения

Кроме простых операторов сравнения

* «равно» ==,
* «меньше» <,
* «больше» >,

есть и более сложные операторы, учитывающие сразу два условия:

* **больше или равно** >=,
* **меньше или равно** <=,
* **не равно** !=.