# Лекция 1

## Установка

Начнем с установки. Для начала нам потребуется скачать питон с официального сайта. После чего запустить скаченный файл и САМОЕ важно это поставить галочку add python \_\_ to PATH. Затем нажимаем Install now.

Мы можем работать с питоновскими файлами и без среды разработки иначе называемой IDE, но это не совсем удобно, для запуска файла с расширение .py в терминале(в винде это командная строка или же cmd) необходимо после перехода в директорию с файлом написать следующее python \_\_\_.py после чего в консоли вы увидите весь результат программы.

Но мы прибегнем к более легкому способу, который требует для своего запуска просто нажать на одну кнопку. И вариант этот называется PyCharm. Скачать можно с офф сайта <https://www.jetbrains.com/pycharm/download/> для нас нужна именно Community версия, она полностью бесплатна и почти не имеет, для нас по крайней мере обрезанного функционала, так как разрабатывать мы будем только на питоне, пока что. А профессиональная версия включает в себя еще расширенные инструменты для разработки на HTML, JS и баз данных SQL.

<https://www.tutorialspoint.com/execute_python_online.php> это лютая херь, пишется в ней только на старом питоне 2.7

## Арифметические операции, Переменные и синтаксис

Начну я свой рассказ с самых азов, а именно с простейших арифметических операций и том, как их правильно писать.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Сложение | Вычитание | Умножение | Возведение в степень | Деление | Остаток от деления | Целая часть деления |
| + (+=) | - (-=) | \* (\*=) | \*\* | / (/=) | // | % |
| 3 + 2 | 3 - 2 | 3 \* 2 | 3 \*\* 2 | 7 / 3 | 7 // 3 | 7 % 3 |
| 5 | 1 | 6 | 9 | 2.33333… | 2 | 1 |

Ну мы скажем что это все так или иначе или логично или просто легко и продолжим дальше. Python у нас умный язык и мы можем не говорить какого именно типа мы запихиваем в переменную данные и если я напишу x = 1.35, а потом попрошу это вывести он без зазрения совести выведет это как тип ***FLOAT***.

Но иногда язык наоборот показывает крайнюю степень тупости и делает всякие прикольные вещи. Давайте вместе подумаем, что выведется если я напишу вот так?

x = input() y = input() print(x+y) при введении чисел 123 и 354. 477? Нет, 123354. Всё это связанно с динамической типизацией и тем, что input выдает строковое значение, чтобы преобразить переменную одного типа в другой мы можем использовать следующие функции:

str(х) - переводит переменную х в строковую

int(х) – переводит переменную х в целочисленную

float(х) – переводит переменную х в переменную с плавающей точкой

Есть еще часто забиваемый всеми тип это bool, переменные логического значение, то есть принимающие значение True(правда) и False(ложь)

Ко всем переменным можно просто обращаться по имени и получать их значение, но строковые переменные особенные и к ним можно обращаться по индексу символа. Индексация в строке происходит с нуля. Вывести длину строковой переменной можно командой len(s). Пример обращения к строке по индексу:

S=’spam’

print(S[0]) >> s

print(S[2]) >> a

print(S[-2]) >>a

Строки можно умножать на числа, складывать между собой.

S1 = 'spam'

S2 = 'eggs'

print(S1 + S2) >>'spameggs'

print('spam' \* 3) >>spamspamspam

Строковые переменные можно доставать «срезами», делается это так:

название\_переменной[X:Y]. X – начало среза, а Y – окончание;

символ с номером Y в срез не входит. По умолчанию первый индекс равен 0, а второй - длине строки

s = 'spameggs'

>>> s[3:5] >>'me'

>>> s[2:-2] >>'ameg'

>>> s[:6] >>'spameg'

>>> s[1:] >>'pameggs'

Функции для перевода всей строки в верхний регистр \_.upper(), а для перевода всей строки в нижний регистр \_.lower(), есть еще \_.title() это перевод начальных символов всех слов в верхний регистр, \_.capitalize() только самый первый символ делает заглавным. Сбор строки из массива осуществляется так переменная = “способ связи”.join(название массива) или вместо массива можно указать название строковой переменной для разделения между каждым символом указанным способом связи.

## Условные операторы

Начнём с синтаксиса условных операторов if elif else:

if условия 1:

действие 1

elif условие 2:

действие 2

else:

действие 3

Блок elif является не обязательным и конструкцию if elif else можно построить обычным образом если-иначе if else. Elif может быть сколько угодно.

Есть укороченная версия if-else, но она сложна как по мне в читабельности, но рассказать про нее надо бы. Вместо того чтобы писать так:

if X:

A=Y

else:

A=Z

Можно написать все немного иначе A=Y if X else Z. В литературе этот прием называется трехместное выражение if/else.

## Алгебра логики

Для if любое число не равное 0 или иные не пустые объекты – истина, числа равные 0 или пустые объекты и значения None – логическая ложь. Сравнение возвращает логическое значение.

Рассмотрим основные логические операторы: X and Y – логическое И выдает истину только если И Х И Y истина, X or Y – логическое ИЛИ выдает истину если хотя бы 1 из правда, X xor Y – взаимоисключающее или является истиной если только одно из истина, not X – логическое отрицание(инверсия).

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| А | В | Not A | A and B | A or B | A xor B |
| 0 | 0 | 1 | 0 | 0 | 0 |
| 0 | 1 | 1 | 0 | 1 | 1 |
| 1 | 0 | 0 | 0 | 1 | 1 |
| 1 | 1 | 0 | 1 | 1 | 0 |