

Питон показывает язык



Справочник по языку Python 3.4.4
с картинками

Петербург
04.07.2024

Данная книжка является не учебником, а справочником.

Часть I (Важный Питон) подборка важных особенностей языка (стр. 03-22)

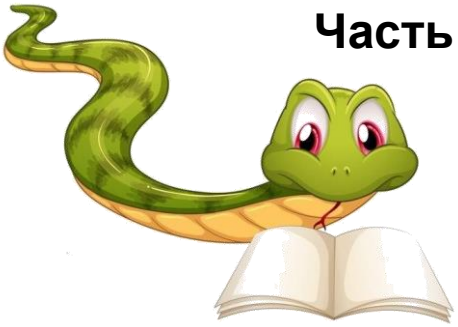
Часть II (Голодный Питон) минимально необходимый арсенал языка (стр. 23-42)

Часть III (Занятный Питон) некоторые выражения языка из Интернета (стр. 43-62)



Это красивый и местами загадочный язык. И даже зная его весьма неплохо, рано или поздно находишь для себя нечто такое, что раньше не использовал...

Условные обозначения:
Жирным шрифтом обозначены слова обязательного написания.
В квадратной рамочке [ ] - необязательные конструкции.
После символа → будут даваться значения выражений.



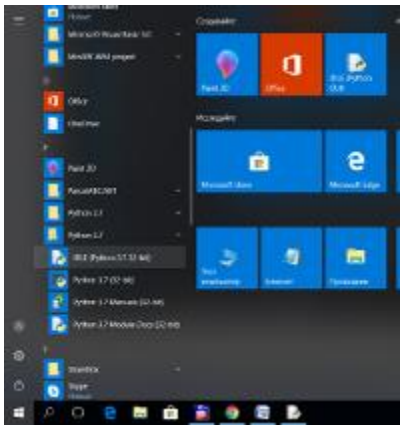
Часть I Важный Питон

При изучении кое-чего,
важно иметь в виду кое-что.

Запуск Питона

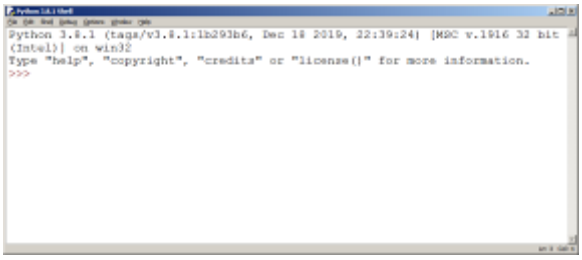
Важно!

Запуск Питона производится через оболочку IDLE.





Оболочка IDLE состоит из двух окон Shell и Edit.
Вначале открывается окно Shell



Важно!

Проверяем в строке заголовка - точно ли это не Python 2.7.xx ?

Терминология

Оператор, операция, операнды, функция, метод, выражение

Оператор - лаконичное высказывание с ключевым словом на языке Питон
Операция - арифметическая, символьная, логическая.
Знак операции - либо в виде символа, либо в виде ключевого слова.
Пример: 2 + 2
Операнды - элементы, с которыми выполняется операция
Функция - действие, которое изображается так:
сначала имя функции, а затем, аргументы в скобках, через запятую
Пример: sin(0.523)
Метод - действие, которое изображается так:
объект, точка, имя метода, аргументы в скобках, через запятую
Пример: s.strip("\n")
Выражение - некоторое количество операций
Принимаемые значения - те аргументы, которые мы передаём функции или методу, и которые они в состоянии правильно обработать.
Возвращаемое значение - результат той или иной операции, функции, метода.
Это значение невидимо возникает в Питоне, и может быть использовано в присваивании, печати, или иных операциях.
В Shell возвращаемое значение последнего выражения хранится в переменной _ .

В справочной литературе используются формулировки, типа:

Функция **pow**(*base*, *exp*, *mod*) возвращает результат возведения числа base в степень *exp*, с опциональным делением по модулю *mod*. Она принимает два параметра: *base* – какое число возводить и *exp* – в какую степень возводить, и *mod* – число, на которое требуется произвести деление по модулю (необязательно)

Типы Питона

Те или иные объекты в Питоне ведут себя по разному, в зависимости от типа.
Будем оперировать следующими типами:

Тип	Имя типа	Пример константы
Числа		
Целый	**int**	5
Вещественный	**float**	5.5
Логический	**bool**	**True** **False**
Комплексный	**complex**	(5+4j)
Последовательности		
Строка	**str**	'мама '
Список	**list**	[1, 3, 6, (5+4j),"мама", 2.3]
Кортеж	**tuple**	(1, 3, 6, (5+4j),"мама", 2.3)

Числа (int, float, bool, complex)

Важно!

Shell, - это, в том числе, и большой калькулятор!

Shell - это для разового использования. Текст, записанный в Shell, - не программа!

Важно!

При численных вычислениях будем оперировать целыми и вещественными числами.

Целыми числами считаем целые штуки. Вещественными - вещество.

Целые вычисления производятся точно, вещественные - примерно.

Вещественные числа, это те, у которых есть (либо может быть) дробная часть. Иначе: так называемые числа с плавающей запятой.

Важно!

Целые числа принципиально нужны только тогда, когда имеет место быть...

1. Задачи из теории чисел, в частности, использование сравнения на равенство
2. Дублирование строк и списков
3. Использование в качестве индексов и параметров **range**
4. Указание степени при возведении отрицательного числа
5. Использование в битовых операциях
6. Подсчёт уж очень большого количества штук, с точностью до одной

Во всех прочих случаях применяются вещественные числа.

Важно!

Десятичный разделитель вещественных чисел в Питоне - точка.

Ни в коем случае не запятая, иначе результат будет катастрофическим!

Целые и вещественные числа в Питоне различаются наличием точки.

5 – Целое (без точки) 5.0 – Вещественное (с точкой)

Важно!

Незначащие нули целых чисел писать запрещено!

Незначащие нули вещественных чисел можно писать, а можно не писать.

Научная запись числа (... на 10 в степени ...) – через букву "e".

Пример:

Число Авогадро $N_A = 6{,}022 \cdot 10^{23}$ записывается так: 6.022e23

Важно!

Число в научной записи всегда вещественное.

Численное значение **True** → 1

Численное значение **False** → 0

Комплексные числа, это только для тех, кто что-то понимает в Комплексных числах, но знать об их существовании необходимо.

Арифметические операции

A + B	— сумма A и B	A - B	— разность A и B
A * B	— произведение A на B	A / B	— деление A на B
A ** B	— A в степени B		
A // B	— целая часть от деления A на B	A % B	— остаток от деления A на B

Примеры:

2 + 2 → 4	10 - 9 → 1	
5 * 5 → 25	3 ** 2 → 9	14 // 4 → 3
1 / 2 → 0.5	1 // 2 → 0	14 % 4 → 2

Важно!

Если в арифметическом выражении все операнды целые, то и результат целый.
Если хоть один операнд вещественный, то результат - вещественный.

5+0 – это целое число. 5+0. – это вещественное число.

Исключение – операция деления Результат всегда вещественный.
Т.е., если есть хотя бы одно деление, то результат - вещественный.
Если число с минусом возводится в степень, то минус остается, т.е.:

-2**2 → -4 а (-2)**2 → 4

Важно!

10 A ^ B - это **НЕ** возведение в степень, а другая операция (побитовое исключающее *или*).

Важно!

Арифметические выражения записываются в одну строчку, т.е.– числители, знаменатели, показатели степени, аргументы функций – всё нужно брать в скобки:

Математическая запись	Запись на Питоне
$\frac{ab}{c}$	`a * b / c`
$\frac{a}{bc}$	`a / ( b * c )` или `a / b / c`
$\frac{a^3+b^3}{a+b}$	`( a**3 + b**3 ) / ( a + b )`

$\frac{-b+\sqrt{b^2-4ac}}{2a}$	`( -b + math.sqrt(b**2 - 4*a*c)) / ( 2 * a )`
	или
	`( -b + (b**2 - 4*a*c)**.5) / ( 2 * a )`

$\frac{a}{1+\frac{a^{2+b}}{3+(2a)^2}}$	`a /(1 + a**(2+b)/(3 + (2*a)**3))`

Важно!

Если, перед выражением открыта скобка (круглая, квадратная или фигурная), и она не закрыта в этой строке, то Питон будет просматривать последующие строки, пока не найдет соответствующую закрывающую скобку.

20 ***Важно! Важно! Важно!***

Если, случайно, скобка не закрыта, то, Питон, не найдя закрывающую скобку, ставит ошибку на оператор, расположенный на следующей строке.

Важно!

Если отрицательное число, случайно, возводится в нецелую степень, то всё выражение, молча, становится комплексным.

(-25)**(1/2) → (3.061616997868383e-16+5j)

Кстати, при использовании функции **math.sqrt** Питон выдаёт ошибку:

```
>>> import math
>>> math.sqrt(-25)
Traceback (most recent call last):
  File "<pyshell#3>", line 1, in <module>
    math.sqrt(-25)
ValueError: math domain error
```

Математические функции

Встроенных полезных функций в Питоне немного:

Функция **abs**(x) возвращает модуль числа
Функция **round**(x,2) округляет вещественное число (первый аргумент)
до некоторого знака после запятой (второй аргумент).
Если второй аргумент не задан, то округляется до целого, и тип становится целым,
а если задан, пусть даже и 0, то - вещественным.
Функция **pow**(x,5) возводит в степень. Первое число – что возводим,
второе – в какую степень возводим. Полный аналог x ** 5
Функция **max**(x1, x2, x3) находит максимальное значение перечисленных чисел.
Функция **min**(x1, x2, x3) находит минимальное значение перечисленных чисел.
Функция **sum**((x1, x2, x3)) находит сумму. (Обратите внимание на количество скобок)

Функции с именем типа преобразуют аргумент в этот тип.

Для использования прочих математических функций, надо подключать модуль **math**.

import math

math.ceil(x) - округление до ближайшего большего числа
math.floor(x) - округление до ближайшего меньшего числа
math.trunc(x) - отбрасывание дробной части
math.sqrt(x) - квадратный корень
math.sin(x) - синус. Аргумент указывается в радианах
math.cos(x) - косинус. Аргумент указывается в радианах
math.tan(x) - тангенс. Аргумент указывается в радианах
math.atan(x) - арктангенс. Результат получается в радианах
math.atan2(y, x) - арктангенс y/x. Результат в радианах. С учетом четверти
math.pi - π = 3,14159258…
math.radians(x) - конвертация градусов в радианы
math.degrees(x) - конвертация радиан в градусы
math.e - *e* = 2,718281828...
math.exp(x) - *e* в степени x
math.expm1(x) - $e^x - 1$. При маленьких x точнее, чем **math**.**exp**(x) - 1
math.log(x, n) - логарифм от x по основанию n . Без n - натуральный логарифм
math.log1p(x) - натуральный логарифм от (1 + x). Точнее, чем **math**.**log**(1 + x)
math.log10(x) - логарифм от x по основанию 10
math.log2(x) - логарифм от x по основанию 2
math.fmod(x, y) - остаток от деления x на y
math.copysign(x, y) - число, имеющее значение модуля числа x, а знак, как у числа y
math.modf(x) - два числа - дробная и целая часть. Оба они с тем же знаком, что и x

Пример:
import math
y = **math**.**sin**(**math**.**radians**(30)) → 0.5

Строки (str)

Важно!

Текстовые константы (в том числе и на русском языке) должны быть обрамлены.
Обрамление - кавычки. Они могут быть, как ординарными ('), так и двойными (").

Важно!

Главное - одинаково.
Это сделано для того, чтобы внутри строки использовать кавычки, как символы.

Важно!

Две ординарные кавычки двойными не являются.

Важно!

Кавычка - это (' или ") только из-под IDLE или Блокнота.
Прочие кавычки из под текстовых процессоров, как то: ’ ‘ ″ ′ ‚ ’ ‹ › ´ ` ” „
кавычками не являются.

Строковые операции

S[index] — получение элемента строки (буквы) по индексу;
S + B — слипание (конкатенация) двух строк;
S * N — дублирование строки N раз (N должно быть целое).

Примеры: 'мама'[1] → а 'ма'+'ма' → мама 'ма'*2 → мама

В Питоне используются отрицательные индексы. При этом нумерация идёт с конца.
Есть понятие "срез", которое будет подробно рассмотрено в Голодном Питоне

Важно!

Для нас важны следующие частные случаи срезов:
S[0] – самая начальная буква строки S S[:3] – первые три буквы строки S.
S[-1] – самая последняя буква строки S S[-3:] – последние три буквы строки S.
S[5] – шестая от начала буква строки S. Номера букв начинаются от 0.

Важно!

Индексы нумеруются от 0.
Индексы, и параметры срезов должны быть целыми.

Важно!

При слипании, с обеих сторон должны быть строки.

Важно!

При дублировании, с одной стороны должна быть строка, с другой – целое число.

Важно!

В Питоне строки являются неизменяемыми.
S = "Питон это мультипарадигмальный язык"
Нам захотелось в строке S слово "мультипарадигмальный" поменять на "хороший"?
Если попытаемся подстроку изменить, то получим:
TypeError: 'str' object does not support item assignment
Решение:
S = S[:10] + "хороший" + S[-5:]

Встроенные функции строк

Нам интересна только одна функция — **len**(S).
Она возвращает количество символов в строке S.

Преобразование простых типов

Напоминаем, что функция преобразования в к/л тип совпадает с именем этого типа.

Функция **int()** преобразует в целое число в десятичной системе счисления:

- любое число, с отбрасыванием дробной части
- строку, изображающую целое число
- строку, изображающую целое число в иной системе счисления до 36

Пример:

int() → 0
int(12.4) → 12
int("12") → 12
int("12", 5) → 7

Важно!

Строку, изображающую число с плавающей точкой, функция **int** не обработает.

int("12.78") → ValueError: invalid literal **for int**() with base 10: '12.78'

Чтобы получить целое значение из строки, изображающей число с плавающей точкой, нужно его предварительно обработать функцией **float** (см. чуть ниже)

int(**float**("12.78")) → 12

Функция **float()** преобразует любое число или строку, изображающую любое число в число с плавающей точкой.

Пример:

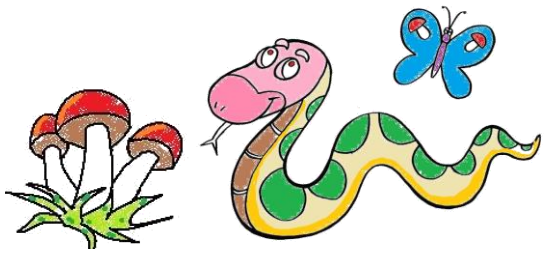
float(12) → 12.0
float("12") → 12.0
float("12.78") → 12.78

Функция **str()** перерабатывает что угодно в строку.

str(12) → '12'
str(12 * 2.5) → '30.0'

Важно!

str([1,2]) → '[1, 2]'
str(**range**(5)) → '**range**(0, 5)'



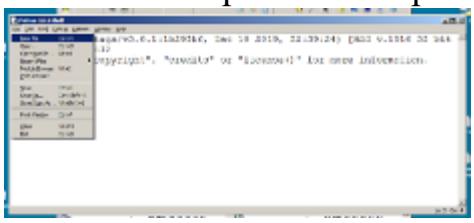
Преобразование сложных типов (**list, tuple**) будет рассмотрено в Голодном Питоне.

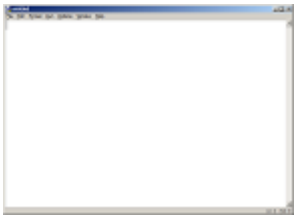
Программирование

Важно!

Программу пишут в окне Edit.

Окно Edit открывается через пункт меню окна Shell File – New file





Запускают набранную программу через пункт меню Run – Run module либо F5.

Важно!

При первом запуске, система предложит Вам сохранить программу. Сохраните её в осмысленном месте (или, куда укажет преподаватель). Куда попало - не годится!

Важно!

Результат выполнения программы Питон помещает в окно Shell.

Важно!

Если программа зависла-зациклилась, то можно просто закрыть окно Shell.

Важно!

Небезвредно запускать, и перезапускать программу сколько угодно раз.

Важно!

Если в программе есть русские буквы, то, в первой строкой полезно иметь:

 # -*- coding: cp1251 -*-

Важно!

Если мы запустили программу, то активным окном автоматически становится Shell.

И, когда программа завершится, то активным окном Shell так и останется.

Значит, чтобы перезапустить программу, надо вернуться в окно Edit, и там нажать F5.

Кстати, после завершения программы Shell знает все последние значения переменных.

Это можно использовать для посмертного анализа.

Признаком того, что мы, таки, перезапустили программу, будет фраза:

========== RESTART: C:/Users/User/Desktop/33.py ==========

Программа – это совокупность операторов, последовательное выполнение которых решает нашу задачу. Основные операторы это:

- оператор присваивания
- операторы обмена информацией (в двух ипостасях - туда, и сюда)
- условный оператор
- операторы цикла (в двух ипостасях - определённое и неопределенное число шагов)

Согласно принципам структурного программирования, любой алгоритм можно изобразить с помощью этих основных операторов.

Каждый оператор пишется на отдельной строке.

Почти все служебные слова в Питоне пишутся маленькими буквами.



Служебные слова в Питоне окрашиваются в оранжевый цвет. Значит, если Вами написанное служебное слово стало оранжевым, значит, Вы написали его исправно.

С другой стороны, если Вы дали имя, а оно вдруг окрасилось в оранжевый цвет, то это имя употреблять нельзя!

Имена встроенных функций окрашиваются в фиолетовый цвет.

Оператор присваивания



Переменные в Питоне появляются в результате действия оператора присваивания. Оператор присваивания одновременно рождает переменную, описывает её и присваивает ей значение. Отдельного оператора описания переменных в Питоне нет. Описание переменных выполняется через присваивание. Что присвоим, такой и тип. Одной и той же переменной в разных местах можно присваивать разные типы, т.е переродить их заново.

Значения переменным присваиваются при помощи оператора присваивания.
Ключевой символ оператора присваивания

=

Важно!

Слева от знака равенства пишем имя переменной,
Имя придумывает программист, а не сосед программиста и не преподаватель на доске.
Справа – выражение, которое вычисляет значение этой переменной.

```
x = 100 + 50
```

это значит, что мы описали переменную x как целую и присвоили ей значение 150.

Важно!

Слева пишем имя переменной, и только имя переменной.
Идентификаторы (имена переменных) состоят из букв, цифр и знака подчеркивания.
Они должны начинаться с буквы. Размер играет роль.
Можно использовать русские буквы (но не рекомендуется).
Рекомендация: делать осмысленные имена.
Можно использовать подчеркивание, чтобы образовать осмысленное имя из двух слов.

Важно!

Нельзя, чтобы имя переменной совпало со служебным словом. Если придуманное Вами слово стало оранжевым или фиолетовым, то нужно придумать другое имя.

Важно!

x и X – это две разные переменные!

Пример:

```
x = 5
y = X + 5
```

```
Traceback (most recent call last):
File "C:/Users/1/Desktop/test.py", line 2, in <module>
y = X + 5
NameError: name 'X' is not defined
```

Дополнительно:

Можно нескольким переменным присвоить одно и то же значение (каскадное присваивание)

Пример:

```
x = y = z = 100
```

Всем этим переменным будет присвоено 100
Можно нескольким переменным присвоить разные значения (множественное присваивание)

Пример:

```
x, y = 5, 10
```

Иксу будет присвоено 5, а игреку 10

Диалог с пользователем

Вывод сообщений и данных

Операции вывода применяются тогда, когда хотим что-то сообщить пользователю.
Информация выводится при помощи инструкции **print**().
То, что хотим вывести, записывается после слова **print** в скобках.
Важно!
Если выводим несколько значений, то их нужно записывать через запятую.
Можно выводить как значения переменных, так и значения выражений.
Важно!
Информация выводится в окно Shell.
Важно!
В Питоне 3, инструкция **print** реализована как функция. Не вдаваясь в подробности, что это означает, переходим к выводам, которые из этого следуют:
1. Слово **print** изображается фиолетовым, а не оранжевым цветом
2. Элементы вывода необходимо писать в скобках
3. Чтобы отменить перевод строки после вывода, в конце надо добавить (... , **end**="")
4. Чтобы удалить разделяющий пробел, надо добавить (... , **sep**="")
5. Чтобы каждое из значений выводилось на новой строке, надо либо использовать либо новый **print**, либо добавить (... , **sep**="\n")

Важно!
Параметры **end** и **sep** должны располагаться в конце перечисления элементов вывода.

Пример:
Пусть x=4 y=5

Код	Результат
print(x)	4
print(x,y)	4 5
print(x,y, **sep**="")	45
print(x,y, **sep**="\n")	4 5
print(x) **print**(y)	4 5
print(x, **end**="") **print**(y)	4 5
print(x,y, **sep**="-")	4-5
print(x, **end**="-") **print**(y)	4-5

Есть еще параметр **file**. О нём будет сказано позже.

Ввод данных от пользователя

Операции ввода применяются тогда, когда надо что-то получить от пользователя.
В Питоне ввод осуществляется при помощи оператора присваивания.
Некоторой переменной присваивается значение функции ввода.

Важно!

В Питоне есть только одна функция ввода, и она назначена только для ввода строки:
X = **input** ("*Строка-пояснение*")

Необязательный параметр этой функции - строка-пояснение для пользователя.
Важно!
В качестве строки-пояснения может быть только одна строка.
В крайнем случае, строковое выражение, приводимое к одной строке.

Пример:
S = **input**("Как Вас зовут?")

В окне Shell появится надпись "Как Вас зовут?". Машина замрёт в ожидании,
пока пользователь не введет какой-нибудь текст и не нажмёт Enter.
Важно!
Если необязательного параметра - строки-пояснения - не указать, то с краю окна Shell будет сиротливо мерцать одинокая слабозаметная вертикальная чёрточка. И всё.

Для того, чтобы ввести целое число...
Важно!
...нужно эту строку преобразовать в целое число при помощи функции **int()**

Пример:

K = **int**(**input**("Сколько у Вас яблок?"))
Важно!
Пользователь аккуратно должен ввести целое число, иначе получит ошибку.
Важно!
Для того, чтобы ввести вещественное число...
Важно!
...нужно эту строку преобразовать в вещественное число при помощи функции **float()**

Пример:

X = **float**(**input**("Каков вес Ваших яблок?"))

