Введение в язык программирования Python

О языке Python

- Язык программирования Python был создан к 1991 году голландцем Гвидо ван Россумом
- После того, как Россум разработал язык, он выложил его в Интернет, где сообщество программистов присоединилось к его улучшению
- Официальный сайт поддержки языка – https://www.python.org.

Основные особенности языка

- Python – это полноценный во многом универсальный язык программирования, используемый в различных сферах
- Основная, но не единственная, поддерживаемая им установка, – объектно-ориентированное программирование
- Однако в данном курсе мы будем изучать структурное программирование, так как оно является базой
- Без знания основных типов данных, ветвлений, циклов, функций нет смысла изучать более сложные парадигмы, т. к. в них все это используется

Основные особенности языка

- Python – интерпретируемый язык программирования. Это значит, что исходный код частями преобразуется в машинный в процессе его чтения специальной программой – интерпретатором
- Python характеризуется ясным синтаксисом, т. к. в мало используются такие вспомогательные синтаксические элементы как скобки, точки с запятыми
- Правила языка заставляют программистов делать отступы для обозначения вложенных конструкций

Основные особенности языка

- Интерпретаторы Python распространяется свободно на основании лицензии

Дзен Питона

Красивое лучше уродливого
Явное лучше неявного
Простое лучше сложного
Сложное лучше усложнённого
Плоское лучше вложенного
Разрежённое лучше плотного
Удобочитаемость важна
Частные случаи не настолько существенны, чтобы нарушать правила
Однако практичность важнее чистоты

Ошибки никогда не должны замалчиваться
За исключением замалчивания, которое задано явно
Перед лицом неоднозначности сопротивляйтесь искушению угадать
Должен существовать один — и, желательно, только один — очевидный способ сделать это
Хотя он может быть с первого взгляда не очевиден, если ты не голландец
Сейчас лучше, чем никогда
Однако, никогда чаще лучше, чем прямо сейчас
Если реализацию сложно объяснить — это плохая идея
Если реализацию легко объяснить — это может быть хорошая идея
Пространства имён — прекрасная идея, давайте делать их больше!

Как писать программы на Python

- Интерактивный режим
- Сценарный режим

Основы синтаксиса Python

- Сценарии исходного кода Python состоят из так называемых *логических строк*, каждая из которых в свою очередь состоит из *физических строк*
- Для обозначения комментариев используется символ #
- Комментарии и пустые строки интерпретатор игнорирует

Основы синтаксиса Python

- Физические строки разделяются самим символом конца строки, но если выражение слишком длинное для одной строки, то две физических строки можно объединить в одну логическую
- Для этого необходимо в конце первой строки ввести символ обратного слеша (\), и тогда следующую строку интерпретатор будет трактовать как продолжение первой
- Нельзя, чтобы на первой строке за символом \ находились бы другие символы, например, комментарий с #

Основы синтаксиса Python

- Для выделения блоков кода используются исключительно отступ
- Логические строки с одинаковым размером отступа формируют блок, и заканчивается блок в том случае, когда появляется логическая строка с отступом меньшего размера
- Именно поэтому первая строка в сценарии Python не должна иметь отступа
- Усвоение этих несложных правил поможет избежать большинства ошибок, связанных с освоением нового языка

Основы синтаксиса Python

- В Python нет символа, который бы отвечал за отделение выражений друг от друга в исходном коде, как, например, точка с запятой (;)
- Также отсутствует такая конструкция, как фигурные скобки {}, позволяющая объединить группу инструкций в единый блок.

Типы данных

- В языке Python выделяют несколько типов данных:
  - целые числа,
  - числа с плавающей точкой (вещественные),
  - строки,
  - логический тип
- Тип каждой переменной может динамически изменяться по ходу выполнения программы. Определить, какой тип имеет переменная, можно с помощью команды type()

Наборы данных (коллекции)

В Python определены три типа коллекций для хранения наборов данных:

- кортеж (tuple);
- список (list);
- словарь (dictionary).

Кортеж

- Кортеж представляет собой неизменяемую упорядоченную последовательность данных
- В нем могут содержаться элементы различных типов, например другие кортежи
- Кортеж определяется в круглых скобках, а его элементы разделяются запятыми
- Специальная встроенная функция tuple() позволяет создавать кортежи из представленной последовательности данных

Список

- Список – это изменяемая упорядоченная последовательность элементов
- Элементы списка также разделяются запятыми, но задаются уже в квадратных скобках
- Для создания списков предлагается функция list()

Словарь

- Словарь является хеш-таблицей, сохраняющей элемент вместе с его идентификатором-ключом
- Последующий доступ к элементам выполняется тоже по ключу, поэтому единица хранения в словаре – это пара объект-ключ и связанный с ним объект-значение
- Словарь – это изменяемая, но не упорядоченная коллекция, так что порядок элементов в словаре может меняться со временем
- Задается словарь в фигурных скобках, ключ отделяется от значения двоеточием, а сами пары ключ/значение разделяются запятыми
- Для создания словарей доступна функция dict().

Виды коллекций

(‘w’,‘o’,‘r’,‘l’,‘d’) # кортеж из пяти элементов

(2.62,) # кортеж из одного элемента

[“test”,'me'] # список из двух элементов

[ ] # пустой список

{ 5:‘отл’, 4:‘хор’, 3:‘удов’ }

словарь из трех элементов с ключами типа int

Переменная

- Переменная — это простейшая именованная структура данных, в которой может быть сохранён промежуточный или конечный результат работы программы.
- Переменную в Python создать очень просто — нужно присвоить некоторому идентификатору значение при помощи оператора присваивания «=».

Переменная

- ПРИМЕР

a = 10

b = 3.1415926

c = “Hello”

d = [1, 2, 3]

  - В этом примере используются четыре переменные:
  - переменная **a** хранит значение типа **int** (целое число),
  - переменная **b** — типа **float** (действительное число),
  - переменная **c** — типа **str** (строка),
  - переменная **d** — типа **list** (список, в данном случае из трех целых чисел).

Переменная

- Никакого специального объявления переменных не требуется, первое присваивание переменной значения и является ее объявлением
- Идентификатор в Python является «ссылкой» на хранимые в памяти данные
- Python — язык с динамической типизацией:
  - каждая переменная в каждый момент времени имеет определенный тип, но этот тип может меняться по ходу выполнения программы, достаточно просто присвоить ей новое значение другого типа

Операторы в Python

В языке существуют следующие типы операторов:

- Арифметические операторы
- Операторы сравнения (реляционные)
- Операторы присваивания
- Логические операторы
- Операторы членства

Арифметические операторы

Оператор	Описание	Примеры
+	Сложение	15 + 5 в результате будет 20
-	Вычитание	13.4 - 7 в результате будет 6.4
*	Умножение	5 * 5 в результате будет 25
/	Деление	15 / 5 в результате будет 3 5.0 / 2 в результате будет 2.5
%	Деление по модулю (возвращает остаток)	6 % 2 в результате будет 0 7 % 2 в результате будет 1 13.2 % 5 в результате 3.2
**	Возведение в степень	-3 ** 2 в результате будет -9
//	Целочисленное деление	4 // 3 в результате будет 1 25 // 6 в результате будет 4

Операторы сравнения

Оператор	Описание	Примеры
= =	Если равны оба операнда, то условие становится истинным	True == False в результате будет False "hello" == "hello" в результате будет True
!=	Если не равны оба операнда, то условие становится истинным	12 != 5 в результате будет True False != False в результате будет False "hi" != "Hi" в результате будет True
<>		12 <> 5 в результате будет True
>		5 > 2 в результате будет True
<		3 < 5 в результате будет True. "A" < "B" в результате будет True.
>=		23 >= 3.2 в результате будет True. "C" >= "D" в результате будет False

Операторы присваивания

Оператор	Описание	Примеры
=		c = 23
+=	Прибавит значение правого операнда к левому и присвоит эту сумму левому операнду	c = 5 a = 2 c += a или: c = c + a Значение c - 7
-=	Отнимает значение правого операнда от левого и присваивает результат левому операнду	c = 5 a = 2 c -= a или: c = c - a Значение c - 3
*=	Умножает правый операнд с левым и присваивает результат левому операнду	c = 5 a = 2 c *= a или: c = c * a Значение c - 10
/=	Делит левый операнд на правый и присваивает результат левому операнду	c = 10 a = 2 c /= a или: c = c / a Значение c - 5

Операторы присваивания

Оператор	Описание	Примеры
%=	Делит по модулю операнды и присваивает результат левому	c = 5 a = 2 c %= a или: c = c % a Значение c - 1
**=	Возводит в левый операнд в степень правого и присваивает результат левому операнду	c = 3 a = 2 c **= a или: c = c ** a Значение c - 9
//=	Производит целочисленное деление левого операнда на правый и присваивает результат левому операнду	c = 11 a = 2 c //= a или: c = c // a Значение c - 5

Логические операторы

Оператор	Описание
and	Логический оператор "И". Условие будет истинным, если оба операнда истина
or	Логический оператор "ИЛИ". Если хотя бы один из операндов истинный, то и все выражение будет истинным
not	Логический оператор "НЕ". Изменяет логическое значение операнда на противоположное

Операторы членства

Операторы членства предназначены для проверки на наличие элемента в составных типах данных, таких, как строки, списки, кортежи или словари

Оператор	Описание	Примеры
in	Возвращает истину, если элемент присутствует в последовательности, иначе возвращает ложь.	"cad" in "cadillac" вернет True. 1 in [2,3,1,6] вернет True. "hi" in {"hi":2,"bye":1} вернет True. 2 in {"hi":2,"bye":1} вернет False (в словарях проверяется наличие в ключах, а не в значениях).
not in	Возвращает истину если элемента нет в последовательности.	Результаты противоположны результатам оператора in.

Приоритет операторов

Оператор	Описание
**	Возведение в степень
* / % //	Умножение, деление, деление по модулю, целочисленное деление
+ -	Сложение и вычитание
<= < > >=	Операторы сравнения
<> == !=	Операторы равенства
= %= /= //= -= += *= **=	Операторы присваивания
in not in	Операторы членства
not or and	Логические операторы

Строки. Основные операции

Строки в Python имеют тип str.

Строкой называется последовательность символов: букв, цифр, знаков ы препинания и т.д.

A + B — конкатенация (строка B приписывается к строке A);

A * n — повторение n раз, значение n должно быть целого типа

Преобразование типов в Python

- Иногда бывает полезно целое число записать как строку
- И, наоборот, если строка состоит из цифр, то полезно эту строку представить в виде числа, чтобы дальше можно было выполнять арифметические операции с ней
- Для этого используются функции, название которых совпадает с именем типа, то есть int, float, str

Преобразование типов в Python

Например:

int(‘123’) вернет целое число 123

str(123) вернет строку ‘123’

print(str(2 + 2) * int('2' + '2'))

инструкция выведет символ “4” 22 раза

функция int(12.3) или int(-12.3)

превращает дробное число в целое

list('abc')

преобразование строки в список букв['a', 'b', 'c']

Ввод данных

- Для считывания строки со стандартного ввода используется функция input(), которая считывает строку с клавиатуры и возвращает значение считанной строки, которое сразу же можно присвоить переменным:

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

- Правда, функция input возвращает текстовую строку. Если нужно сделать так, чтобы переменные имели целочисленные значения, то сразу же после считывания выполним преобразование типов при помощи фунцкии int:

```
a = int(a)
b = int(b)
```

- Можно объединить считывание строк и преобразование типов, если вызывать функцию int для того значения, которое вернет функция input():

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

Ввод данных

- Сложнее считать значения переменных, если они записаны в отдельной строке
- Здесь нужно применить к считанной строке метод split(), который разделяет строку на части по одному или двум пробелам
- Затем результат выполнения этой функции присваивается кортежу из двух или нескольких чисел.
- Например, если в строке вводятся два числа через пробел, то считать их можно так:

```
a, b = input().split()
a = int(a)
b = int(b)
```

Ввод данных

- *Аналогично, три переменные можно считать, записав слева от оператора присваивания кортеж из трех переменных:*

```
a, b, c = input().split()
```

Вывод данных

- Для вывода данных используется функция print
- Выводит значения переменных, значения любых выражений
- Например:

```
print(2 + 2 ** 2)
a = 1
b = 2
 print (a, '+', b, '=', a + b)
```

- Выводимые значение разделяются одним пробелом

Вывод данных

- Можно разделять выводимые значения двумя пробелами, любым другим символом, любой другой строкой, выводить их в отдельных строках или не разделять
- Для этого функции print передают специальный именованный параметр, называемый sep (аббревиатура от слова separator, т.е. разделитель)
- По умолчанию параметр sep равен строке из одного пробела
- Параметр sep указывается в списке вывода:

```
print(a, b, c, sep = ‘ __:__ ‘)
```

Вывод данных

- Чтобы совсем убрать разделитель при выводе нужно передать параметр sep, равный пустой строке:

```
print (a, '+', b, '=', a + b, sep = '')
```

- Для того, чтобы значения выводились с новой строке, нужно

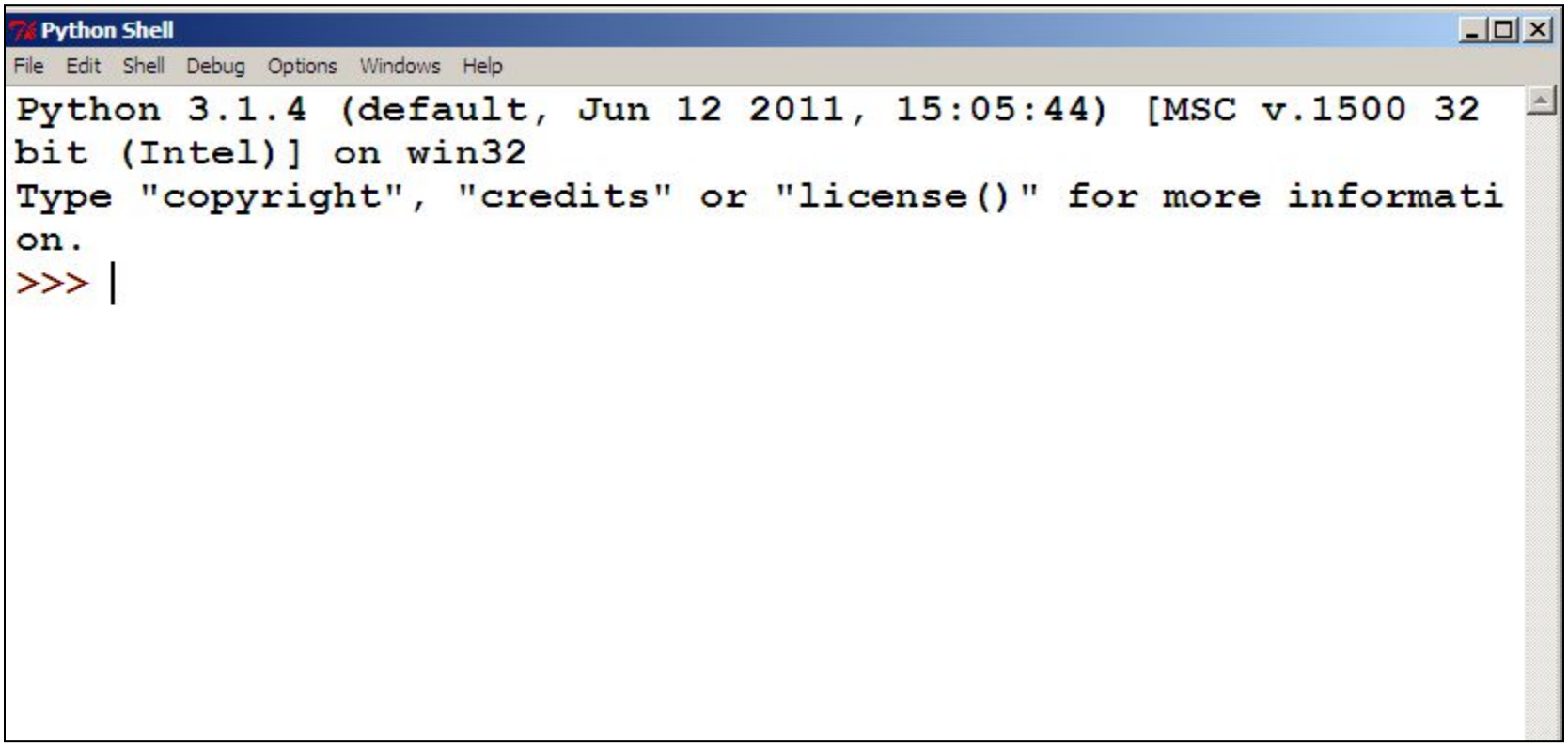
```
print (a, b, sep = '\n')
```

- Для вывода обратного слэша нужно повторить его два раза: ‘\\’

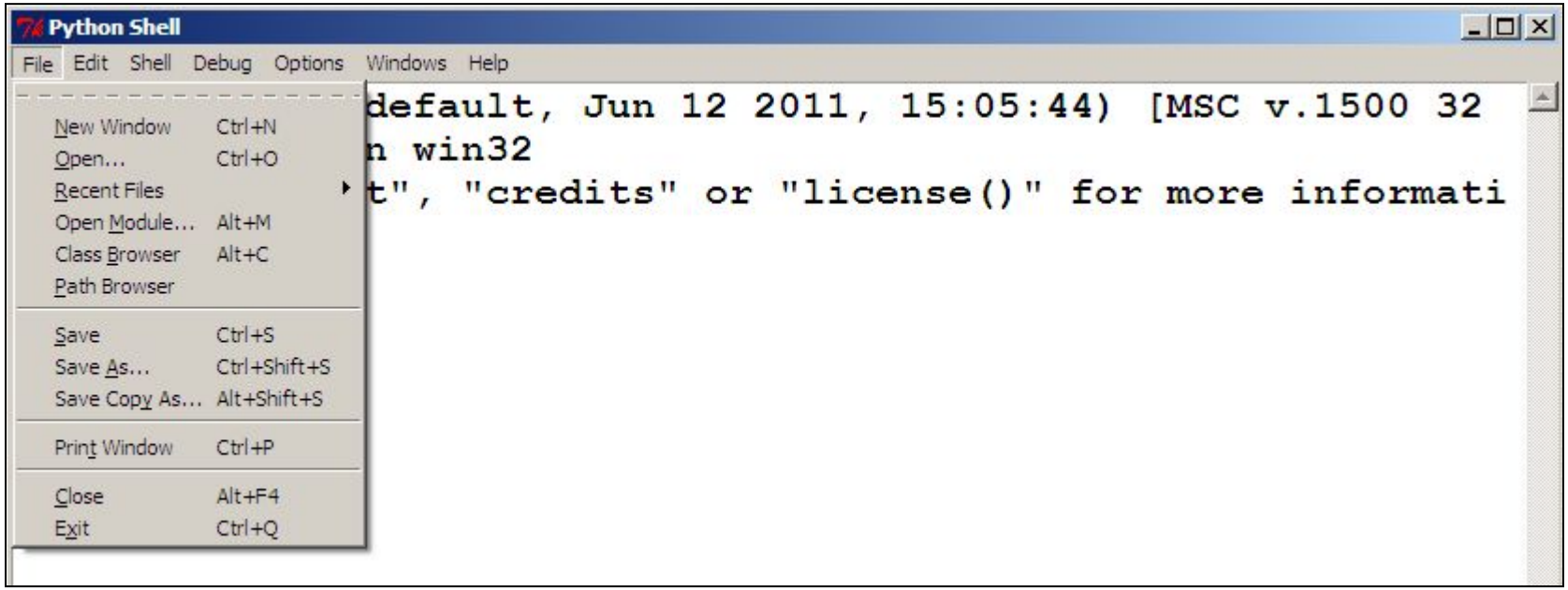
```
print("ПН", "ВТ", "СР", "ЧТ", "ПТ", "СБ", "ВС", sep=" - ")
```

ПН - ВТ - СР - ЧТ – ПТ – СБ - ВС

Программирование Python



Программирование Python



Программирование Python