Язык программирования Python

Числа и строки

Составил преподаватель
Жданович А.Е.

В феврале 1991 года был разработан язык *Python* сотрудником голландского института CWI **Гвидо ван Россумом**. Разработка языка Python была начата в конце 1980-х г.

Автор назвал язык в честь популярного британского комедийного телешоу 1970-х «*Летающий цирк Монти Пайтона*». Впрочем, всё равно название языка чаще связывают именно со змеёй, нежели с передачей.

О Python лучше всего говорит создатель этого языка программирования, голландец Гвидо ван Россум:

"Python - интерпретируемый, объектно-ориентированный высокоуровневый язык программирования с динамической семантикой. *Встроенные высокоуровневые структуры данных в сочетании с динамической типизацией и связыванием делают язык привлекательным для быстрой разработки приложений (RAD, Rapid Application Development). Кроме того, его можно использовать в качестве сценарного языка для связи программных компонентов. Синтаксис Python прост в изучении, в нем придается особое значение читаемости кода, а это сокращает затраты на сопровождение программных продуктов. Python поддерживает модули и пакеты, поощряя модульность и повторное использование кода. Интерпретатор Python и большая стандартная библиотека доступны бесплатно в виде исходных и исполняемых кодов для всех основных платформ и могут свободно распространяться."*

Типы данных

Простейшие встроенные в Python типы данных:

- *булевы значения* (имеют значение True или False)
- *целые числа int* (например, 35 или 1 000 000)
- *число с плавающей точкой float* (числа с десятичной запятой)
- *строка str* (последовательности текстовых символов).

Python поддерживает динамическую типизацию, то есть тип переменной определяется только во время исполнения.

Переменные - это имена объектов, которые содержат данные.

Имя переменных может содержать любые буквы, цифры и нижнее подчеркивание (_), но не могут начинаться с цифры.

Пример:

```
>>> a = 7          # значение 7 присваивается имени a
>>> print (a)      # вывод на экран значения переменной a
>>> b = a          # значение с именем a присваивается переменной с именем b
>>> print (b)      # вывод на экран значения переменной b
```

Чтобы узнать тип какого-то объекта (переменной или значения) надо использовать конструкцию **type (объект)**

Пример:

```
>>> type (a)        #укажет тип переменной a
>>> type (58)       #укажет тип значения 58
```

>>> type (99.9) #укажет тип значения 99,9

>>> type (‘zxc’) #укажет тип значения ‘zxc’

Класс (class) – это определение объекта.

Работа с числами

Любая последовательность цифр является целым числом:

Пример ввода непосредственно чисел:

введите число **5** – для получения результата нажмите Enter,

введите **0** - для получения результата нажмите Enter,

введите **05** - для получения результата нажмите Enter.

Здесь ошибка говорит о том, что число нельзя начинать с 0

Операторы арифметических действий

Оператор	Описание	Примеры
+	Сложение	15 + 5 в результате будет 20
-	Вычитание	15 - 5 в результате будет 10
*	Умножение	5 * 5 в результате будет 25
/	Деление	7 /2 в результате будет 3,5
%	Деление по модулю – вычисление остатка	6 % 2 в результате будет 0 7 % 2 в результате будет 1
**	Возведение в степень	5 ** 2 в результате будет 25
//	Целочисленное деление - часть после запятой отбрасывается.	12 // 5 в результате будет 2 4 // 3 в результате будет 1

Пример ввода переменных, которым присвоены числовые значения:

```
>>> a=5
>>> a

>>> a-3
>>> a
```

Чтобы изменить значение переменной **a** надо, например:

```
>>> a = a-3
>>> a
```

Можно совместить арифметические операторы с присваиванием (=), размещая оператор арифметического действия перед знаком =. Например, выражение a=a-3 можно записать a-=3.

Примеры:

```
>>> a=95
>>> a-=3
>>> a

>>> a+=8
>>> a

>>> a*=2
>>> a


>>> a=13
>>> a//=4
>>> a
```

Преобразование типов

Чтобы изменить другие типы данных на целочисленный тип надо использовать функцию **int()**.

Примеры:

Значениями **булевых переменных** могут быть только ***True*** или ***False.*** При преобразовании в целые числа они представляют собой соответственно 1 и 0:

```
>>> int(True)
>>> int(False)
```

Преобразование числа с плавающей точкой в целое число:

```
>>> int(98.6)
```

Преобразование текстовой строки, которая содержит только цифры и знаки + - (строковые значения записываются в кавычках):

```
>>> int('99')
>>> int('-23')
```

Функция int() не обрабатывает строки, содержащие числа с плавающей точкой:

```
>>> int('98.8')
```

Автоматически преобразуются смешанные числовые значения:

```
>>> 4+7.0
>>> True+2
>>> False+2.0
```

Чтобы изменить другие типы данных на числа с плавающей точкой надо использовать функцию **float()**.

Примеры:

```
>>> float(True)
>>> float(False)
>>> float(98)
>>> float('99')
>>> float('99.9')
>>> float('-1.5')
```

Строки

Строка – это последовательность символов. В Python строки являются неизменяемыми.

Строка создается заключением символов в кавычки одинарные ‘ ’ или двойные “ “

Примеры:

>>> ‘Hello’

>>> “Hello”

>>> 'Возьмите в руки "мышь".‘

Можно использовать тройные одинарные или тройные двойные кавычки. Обычно их используют для создания многострочных строк (например, для стихов)

>>> '''Наша Таня

громко плачет'''

В полученном результате управляющий символ \n показывает форматирование – переход на новую строку.

Для вывода строки можно использовать функцию print():

```
>>> poem = '''Наша Таня
громко плачет'''
>>> print (poem)
```

Строка может быть пустой:

```
>>> ‘ ’
>>> “ ”
```

Преобразование типов

Чтобы изменить другие типы данных в строки надо использовать функцию **str()**.

Примеры:

```
>>> str(98.8)
>>> str(98)
>>> str(True)
```

Использование управляющих символов при создании строки

\n - переход на новую строку.

\t - табуляция (для выравнивания текста).

Примеры:

```
>>> poem2 = 'Наша Таня \nгромко плачет'
>>> print (poem2)

>>> poem2 = '\tНаша Таня \n\tгромко плачет'
>>> print (poem2)

>>> print (‘a\tbc’)

>>> print (‘ab\t c’)
```

Объединение строк

```
>>> 'Привет!' 'Как дела?‘
```

или

```
>>> 'Привет!' + 'Как дела?'
```

Размножение строк с помощью символа *

```
>>> start = 'Na ' *4+'\n'
>>> middle = 'Hey ' *3+ '\n'
>>> end= 'Goodbye.'
>>> print(start + start + middle +end)
```

Извлечение символа из строки

Символы в строке нумеруются (индексируются):

слева направо - начиная с 0

справа налево - начиная с -1

А	Б	В	Г	Д	Е	Й	к	а
0	1	2	3	4	5	6	7	8

А	Б	В	Г	Д	Е	Й	к	а
-9	-8	-7	-6	-5	-4	-3	-2	-1

Примеры:

```
>>> letters = 'АБВГДЕЙка'
>>> letters[0]
>>> letters[2]
>>> letters[-1]
>>> letters[-3]
```

Так как строки неизменяемы, то присвоить другое значение какому-либо символу в строке нельзя.

Например, мы хотим заменить слово *стол* на слово *стул*:

```
>>> term = ‘стол’
>>> term[2] = ‘у’
```

Извлечение подстроки с помощью оператора [start:end:step] - *СРЕЗЫ*

- **[:]** ***– извлекает всю последовательность***

```
>>> letters = 'АБВГДЕЙка'
>>> letters[:]
```

- **[start:]** ***– извлекает от заданной точки и до конца.***

```
>>> letters[2:]
>>> letters[-3:]
```

- **[:end]** ***– извлекает от начала до заданной точки***

```
>>> letters[:2]
```

- **[start:end]** ***- извлекает от заданной точки и до заданной точки***

```
>>> letters[2:4]
>>> letters[1: 7: 2]
>>> letters[: : 2]
>>> letters[: : -1]
```

Встроенные функции Python

Функция – это именованный фрагмент кода, который выполняет определенные операции.

Функция **len()** – подсчитывает символы в строке:

```
>>> len(letters)

>>> a=””
>>> len(a)
```

Функции для работы со строками

Чтобы применить строковую функцию надо использовать следующий синтаксис:

имя строки.функция (аргументы)

Разделение строки на список небольших строк с помощью функции split()

```
>>> a='Привет! Рад видеть!'
>>> a.split('!')
```

a – имя строки; split – функция; ! – аргумент.

Если аргумент не указан, то разделение будет по пробелу или табуляции, или по символу новой строки.

```
>>> a='Привет! Рад видеть!'
>>> a.split()
```

Объединение строки с помощью функции join()

```
>>> wname = ['Anna' , 'Julia' , 'Paula']
#Дан список строк (см. кавычки)с женскими именами
>>> name = ' , '.join(wname)
#Объединение этого списка в одну строку с перечислением через ,
>>> print("women's names:", name)
#Печать полученной строки
```

Замена символа с помощью функции replace()

Заменяет одну подстроку другой.
Аргументом функции являются:

replace (*старая подстрока, новая подстрока, количество включений,* которое нужно заменить***)***

Если количество включений не указано, то заменятся все включения.

```
>>> a=’Мама мыла раму’
>>> a.replace(‘Мама мыла’ , ‘Папа мыл’)
```

Строковые методы

Методы – это функции, доступные для данного типа объектов.

Изменение регистра:

```
>>> a=’мама мыла раму’
>>> a.capitalize()
>>> a.upper()
>>> a.lower()
>>> a.title()
```

Поиск:

```
>>> a.count('м')
>>> a.find('м')
>>> a.endswith('му')
```

Форматирование:

```
>>> a.center(30)     # 30 – это количество пробелов
>>> a.rjust(30)
>>> a.ljust(30)
```