Переменные, выражения и инструкции

Глава 2

Python for Informatics: Exploring Information
www.pythonlearn.com

Константы

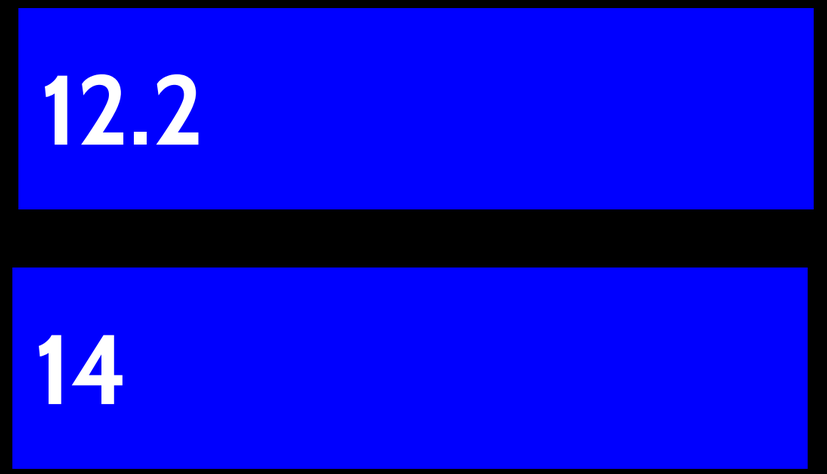
- Фиксированные значения, такие как числа, буквы и строки называются “константами”, потому что их значение не изменяется
- Числовые константы вводятся как обычные числа
- Строковые константы вводятся в одинарных (') или двойных (") кавычках

```
>>> print 123
123
>>> print 98.6
98.6
>>> print 'Hello world'
Hello world
```

Переменные

- Переменная - это определенное место в памяти компьютера, в котором программа хранит данные для последующего их извлечения по “названию” заданной переменной
- Программисты сами выбирают названия переменных
- Позже содержимое переменной можно изменить

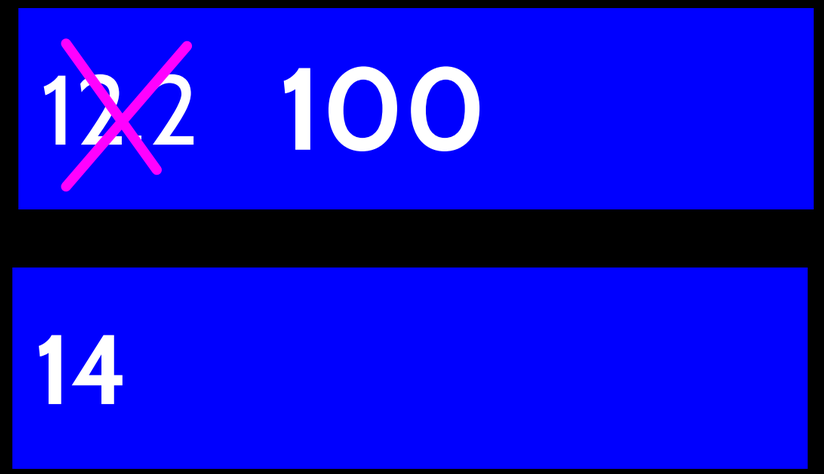
```
x = 12.2
y = 14
```



Переменные

- Переменная - это определенное место в памяти компьютера, в котором программа хранит данные для последующего их извлечения по “названию” заданной переменной
- Программисты сами выбирают названия переменных
- Позже содержимое переменной можно изменить

```
x = 12.2
y = 14
x = 100
```



Правила Python для названия переменных

- Название переменной должно начинаться с буквы или нижнего подчеркивания _
- Должно состоять из букв и может включать числа и нижние подчеркивания
- Регистр имеет значение
- Хорошие названия: spam eggs spam23 _speed
- Плохие названия: 23spam #sign var.12
- Отличающиеся названия: spam Spam SPAM

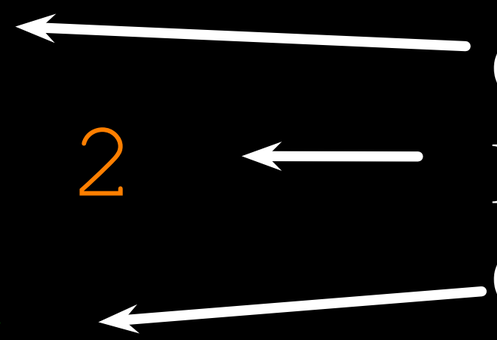
Зарезервированные слова

- Использование зарезервированных слов не допускается в названии переменных/идентификаторов

and del for is raise assert elif
from lambda return break else
global not try class except if or
while continue exec import pass
yield def finally in print as with

Строки

```
x = 2
x = x + 2
print x
```



Оператор присваивания

Присваивание с выражением

Оператор печати

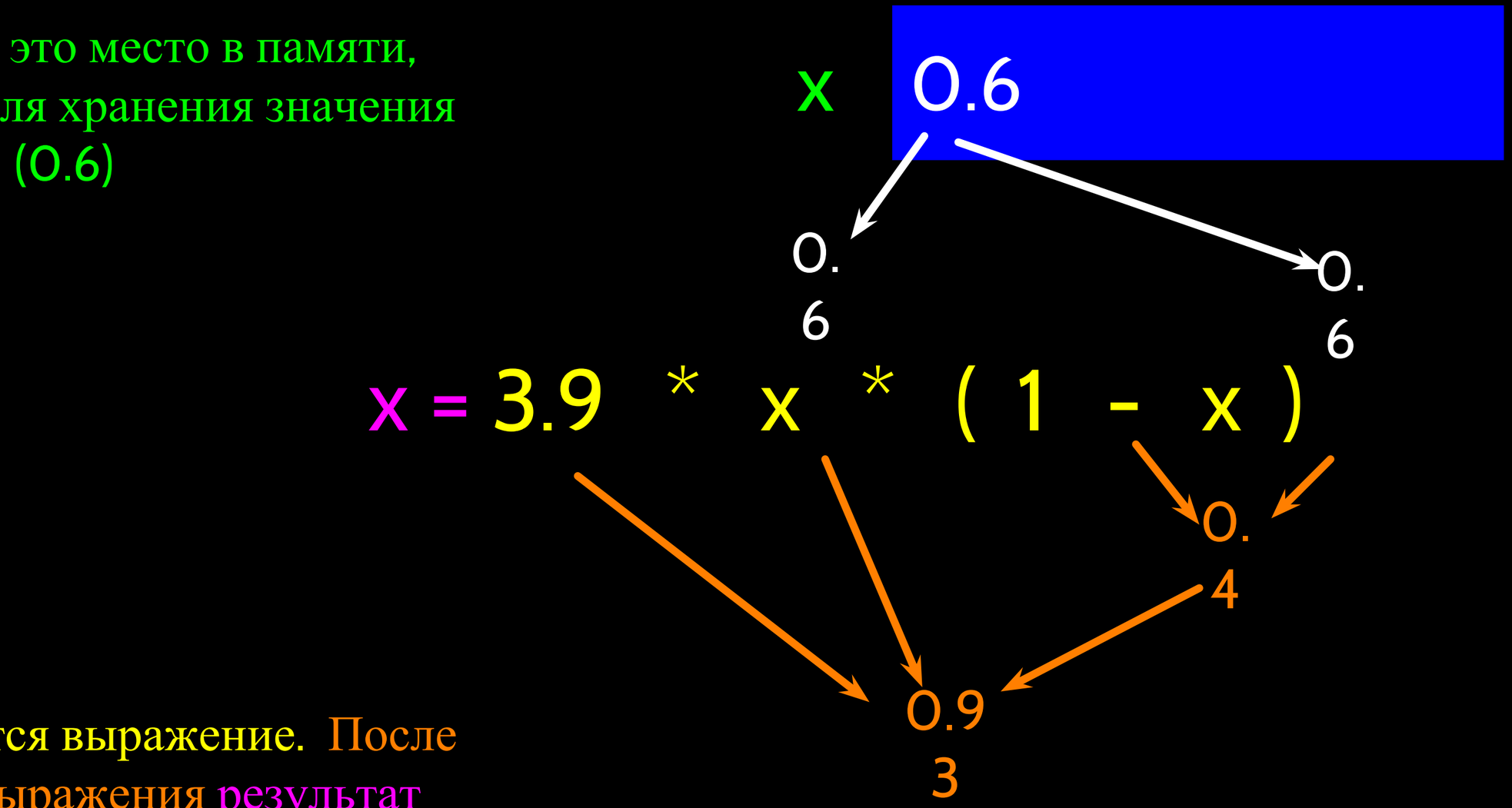
Переменная Оператор Константа Зарезервированное слово

Операторы присваивания

- Мы присваиваем переменной значение с помощью оператора присваивания (=)
- Оператор присваивания состоит из выражения справа и переменной для хранения результата

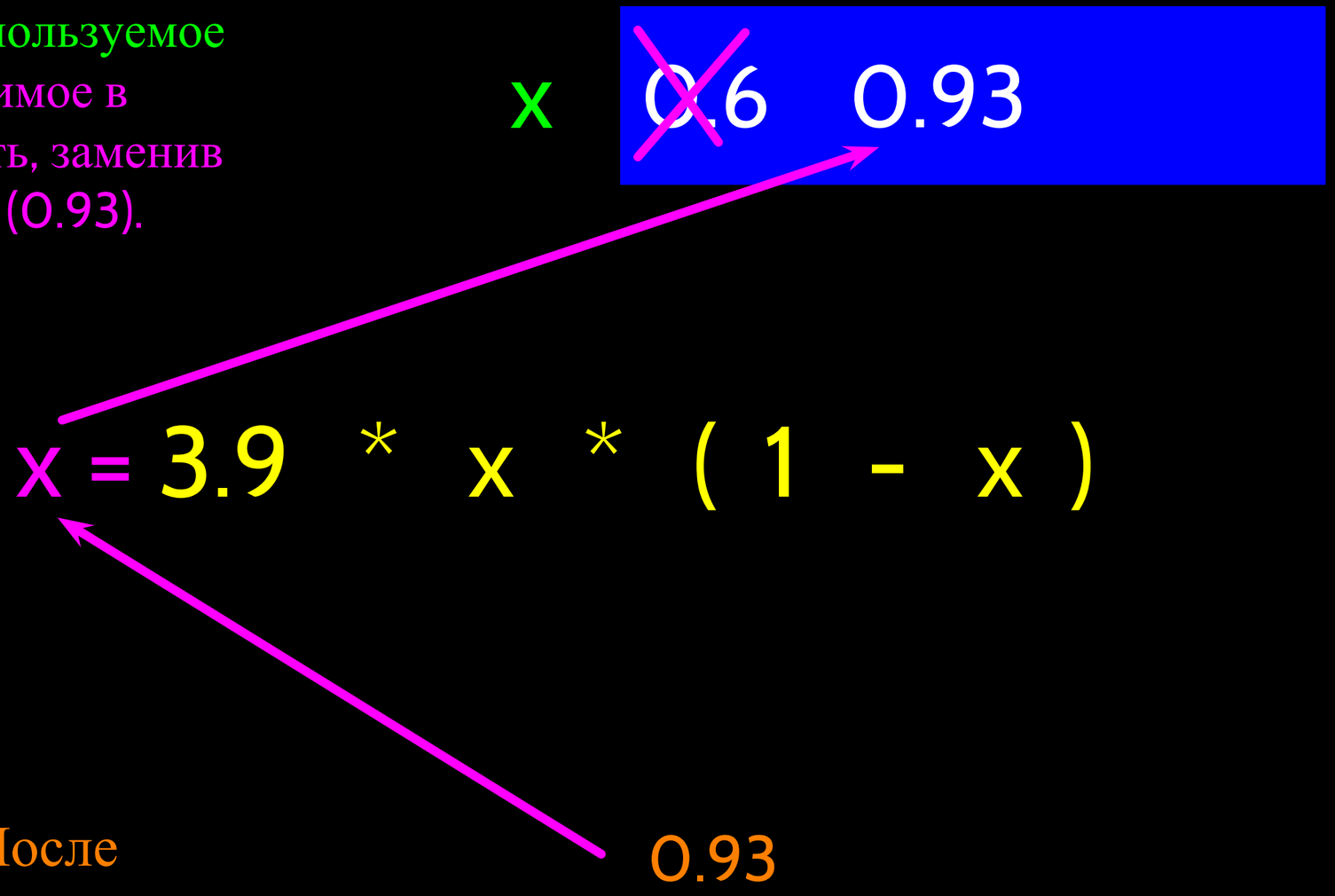
```
x = 3.9 * x * ( 1 - x )
```

Переменная - это место в памяти, используемое для хранения значения (0.6)



Справа находится выражение. После вычисления выражения результат помещается в (присваивается) x.

Переменная - это место в памяти, используемое для хранения значения. Содержимое в переменной значение можно обновить, заменив исходное значение (0.6) на новое (0.93).



Справа находится выражение. После вычисления выражения результат помещается в (присваивается) x.

Числовые выражения

- Из-за отсутствия на компьютерной клавиатуре математических символов для обозначения математических операций мы используем "компьютерные символы"
- Звездочкой обозначается умножение
- Возведение в степень в математике обозначается по-другому

Оператор	Операция
+	Сложение
-	Вычитание
*	Умножение
/	Деление
**	Степень
%	Остаток

Числовые выражения

```
>>> xx = 2
>>> xx = xx + 2
>>> print xx
4
>>> yy = 440 * 12
>>> print yy
5280
>>> zz = yy / 1000
>>> print zz
5
```

```
>>> jj = 23
>>> kk = jj % 5
>>> print kk
3
>>> print 4 ** 3
64
```

```
   4 R 3
  ------
5 | 23
    20
   ---
    3
```

Оператор	Операция
+	Сложение
-	Вычитание
*	Умножение
/	Деление
**	Степень
%	Остаток

Порядок вычисления

- Когда в одной строке используется несколько операторов, Python должен знать порядок вычисления
- Это понятие называется “приоритетом операторов”
- Какой оператор является “первостепенным”?

```
x = 1 + 2 * 3 - 4 / 5 ** 6
```

Правила приоритетов операторов

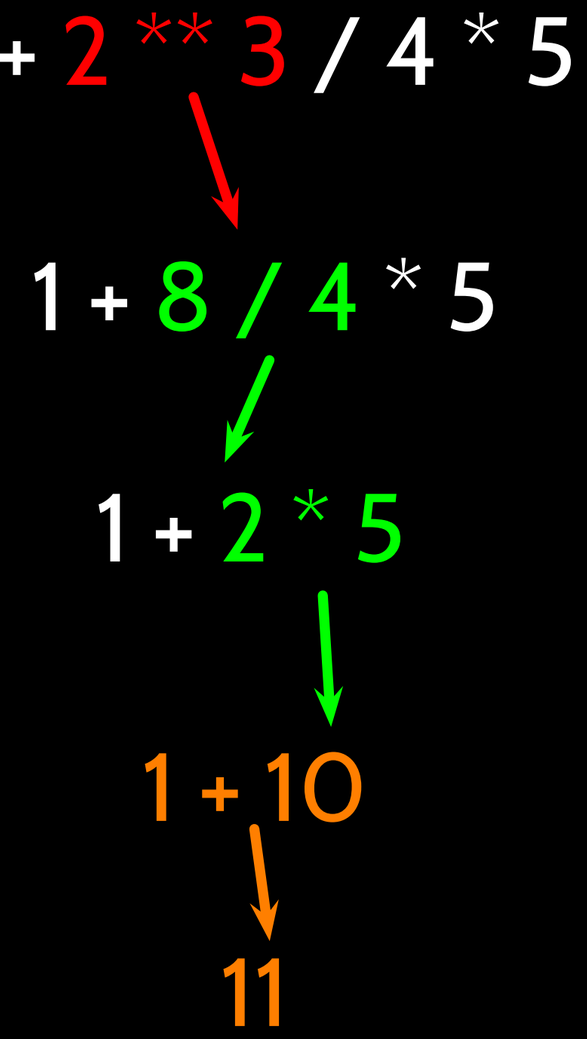
От наивысшего приоритета к наименьшему:

- Выражения в скобках всегда выполняются первыми
- Возведение в степень
- Умножение, деление и остаток
- Сложение и вычитание
- Слева направо

Скобки
Степень
Умножение
Сложение
Слева направо

```
>>> x = 1 + 2 ** 3 / 4 * 5
>>> print x
11
>>>
```

Скобки
Степень
Умножение
Сложение
Слева направо



Приоритет операторов

Скобки
Степень
Умножение
Сложение
Слева направо

- Запомните правила сверху вниз
- При написании кода используйте скобки
- При написании кода используйте наиболее простые математические выражения для легкости понимания
- Разбивайте длинные математические выражения на короткие для четкости восприятия

Тест: `x = 1 + 2 * 3 - 4 / 5`

Деление на языке Python выполняется странно!

- При делении целых чисел дробная часть результата отсекается
- Деление значений с плавающей точкой возвращает значение с плавающей точкой

```
>>> print 10 / 2
5
>>> print 9 / 2
4
>>> print 99 / 100
0
>>> print 10.0 / 2.0
5.0
>>> print 99.0 / 100.0
0.99
```

Это правило изменилось в версии Python 3.0

Использование целых чисел и значений с плавающей точкой

- Если один операнд является целым числом, а другой - значением с плавающей точкой, результатом является значение с плавающей точкой
- Перед выполнением операции целое число преобразуется в значение с плавающей точкой

```
>>> print 99 / 100
0
>>> print 99 / 100.0
0.99
>>> print 99.0 / 100
0.99
>>> print 1 + 2 * 3 /
4.0 - 5
-2.5
>>>
```

Что такое “тип”?

- В Python переменные, литералы и константы имеют “тип” данных
- Python знает разницу между целым числом и строкой
- Например “+” означает “сложение” при использовании с числами и “конкатенацию” - со строками

```
>>> ddd = 1 + 4
>>> print ddd
5
>>> eee = 'hello ' +
'there'
>>> print eee
hello there
```

конкатенировать = соединить

Тип имеет значение

- Python знает “тип” всех используемых данных
- Некоторые операции запрещены
- К строке нельзя “прибавить 1”
- Воспользуйтесь функцией type(), чтобы узнать тип данных

```
>>> eee = 'hello ' + 'there'
>>> eee = eee + 1
Traceback (most recent call last):
  File "<stdin>", line 1, in
<module>
TypeError: cannot concatenate 'str'
and 'int' objects
>>> type(eee)
<type 'str'>
>>> type('hello')
<type 'str'>
>>> type(1)
<type 'int'>
>>>
```

Несколько типов чисел

- Числа имеют два основных типа
  - Целые числа:
    -14, -2, 0, 1, 100, 401233
  - Значения с плавающей точкой имеют дробную часть: -2.5 , 0.0, 98.6, 14.0
- Остальные типы чисел являются комбинацией целых чисел и значений с плавающей точкой

```
>>> xx = 1
>>> type (xx)
<type 'int'>
>>> temp = 98.6
>>> type(temp)
<type 'float'>
>>> type(1)
<type 'int'>
>>> type(1.0)
<type 'float'>
>>>
```

Преобразование из одного типа в другой

- При использовании в выражении целого числа и значения с плавающей точкой, целое число косвенно преобразуется в значение с плавающей точкой
- Такое преобразование выполняется с помощью встроенных функций int() и float()

```
>>> print float(99) / 100
0.99
>>> i = 42
>>> type(i)
<type 'int'>
>>> f = float(i)
>>> print f
42.0
>>> type(f)
<type 'float'>
>>> print 1 + 2 * float(3) / 4 - 5
-2.5
>>>
```

Преобразование строк

- Функции int() и float() можно также использовать для преобразования строк в целые числа и наоборот
- Если строка состоит не из числовых значений, Python выдаст ошибку

```
>>> sval = '123'
>>> type(sval)
<type 'str'>
>>> print sval + 1
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: cannot concatenate 'str' and
'int'
>>> ival = int(sval)
>>> type(ival)
<type 'int'>
>>> print ival + 1
124
>>> nsv = 'hello bob'
>>> niv = int(nsv)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
ValueError: invalid literal for int()
```

Ввод данных пользователем

- Функция raw_input() позволяет Python читать данные, введенные пользователем
- Функция raw_input() возвращает строку

```
nam = raw_input('Who are you?')
print 'Welcome', nam
```

Who are you? Chuck
Welcome Chuck

Преобразование введенных пользователем данных

- Чтобы прочитать введенные пользователем данные, их необходимо преобразовать из строки в число с помощью функции преобразования типа
- К проблеме ввода неверных данных мы обратимся позднее

```
inp = raw_input('Europe floor?')
usf = int(inp) + 1
print 'US floor', usf
```

```
Europe floor? 0
US floor 1
```

Комментарии в Python

- Python игнорирует всю строку после символа решетки #
- Для чего нужны комментарии?
  - Чтобы пояснить последующую часть кода
  - Задокументировать информацию об авторе кода или другую вспомогательную информацию
  - Отключить строку кода, возможно, временно

```
# Получить название файла и открыть его
name = raw_input('Enter file:')
handle = open(name, 'r')
text = handle.read()
words = text.split()

# Посчитать частоту использования слов
counts = dict()
for word in words:
   counts[word] = counts.get(word,0) + 1

# Найти наиболее часто используемое слово
bigcount = None
bigword = None
for word,count in counts.items():
    if bigcount is None or count > bigcount:
        bigword = word
        bigcount = count

# Конец
print bigword, bigcount
```

Операции со строками

- Некоторые операции применимы к строкам
  - + означает “конкатенацию”
  - * означает “несколько конкатенаций”
- Python знает разницу между строкой и числом и в соответствии с этим выполняет все операции

```
>>> print 'abc' + '123’
abc123
>>> print 'Hi' * 5
HiHiHiHiHi
>>>
```

Символические имена переменных

- Так как программисты сами выбирают имена переменных, есть несколько общих “правил” их написания
- Мы даем переменным имена, помогающие помнить назначение данных, на которые они ссылаются (“символические имена” - “памятка”)
- Переменные с хорошим именем могут смутить новичков, потому что могут напоминать зарезервированные слова

http://en.wikipedia.org/wiki/Mnemonic

```
x1q3z9ocd = 35.0
x1q3z9afd = 12.50
x1q3p9afd = x1q3z9ocd * x1q3z9afd
print x1q3p9afd
```

```
a = 35.0
b = 12.50
c = a * b
print c
```

Что делает этот код?

```
hours = 35.0
rate = 12.50
pay = hours * rate
print pay
```

Упражнение

Напишите программу, которая предлагает пользователю ввести отработанные часы и ставку заработной платы для расчета суммы заработной платы.

Введите часы: 35
Введите ставку: 2.75
Заработная плата: 96.25

Обзор

- Тип
- Зарезервированные слова
- Переменные (символические имена)
- Операторы
- Приоритет операторов
- Деление целых чисел
- Преобразование типов данных
- Ввод данных пользователем
- Комментарии (#)

Благодарность / Содействие

Первоначальная разработка: Чарльз Северанс, Школа информации Мичиганского университета

Здесь впишите дополнительных авторов...

...