Переменные, выражения и операторы

Глава 2

Python для всех

Константы

- Фиксированные значения, такие как числа, буквы, и строки, называются “константами” потому, что их значение не может быть изменено
- Числовые константы такие, как вы вводите
- Строковые константы используют одинарные('), либо двойные(") кавычки

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

Зарезервированные слова

Вы не можете использовать зарезервированные слова в качестве названий/идентификаторов имён переменных:

```
False   class   return  is      finally
None    if      for     lambda  continue
True    def     from    while   nonlocal
and     del     global  not     with
as      elif    try     or      yield
assert  else    import  pass
break   except  in      raise
```

Переменные

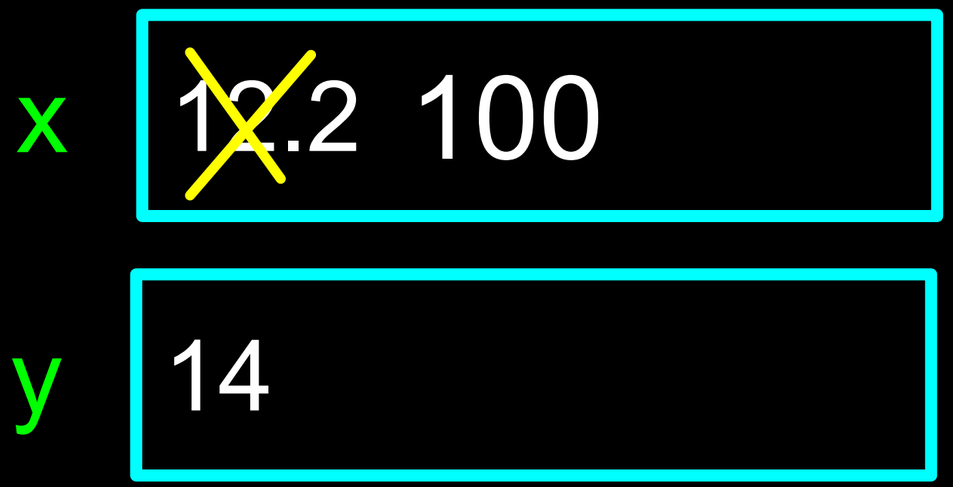
- Переменная - это именованное место в памяти, где программист может хранить данные, а затем извлекать данные, используя переменную «имя».
- Программисты могут выбирать имена переменных
- Вы можете изменить содержимое переменной в более поздней инструкции

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

Переменные

- Переменная - это именованное место в памяти, где программист может хранить данные, а затем извлекать данные, используя переменную «имя».
- Программисты могут выбирать имена переменных
- Вы можете изменить содержимое переменной в более поздней инструкции

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



Python Правила имён переменных

- Должен начинаться с буквы или подчеркивания _
- Должен состоять из букв, цифр и знаков подчеркивания
- Чувствительны к регистру (Регистрозависимые переменные)

```
Good:       spam     eggs    spam23      _speed
Bad:        23spam       #sign   var.12
Different:      spam    Spam    SPAM
```

Мнемоника имён переменных

- Поскольку нам, программистам, предоставляется выбор в выборе имён переменных, есть несколько «хороших практик».
- Мы называем переменные, чтобы помочь нам запомнить, что мы собираемся в них хранить («мнемоника» = «помощь в запоминании»).
- Это может сбить с толку начинающих студентов, потому что хорошо названные переменные часто «звучат» настолько хорошо, что должны быть ключевыми словами.

https://ru.wikipedia.org/wiki/Мнемоника

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

Что за хрень тут
происходит?

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

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

Что за хрень тут
происходит?

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

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

Почему нельзя
сделать так? ->

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

Предложения или строки

```
x = 2
x = x + 2
print(x)
```


Оператор присваивания


Присвоение с выражением


Вывод в консоль

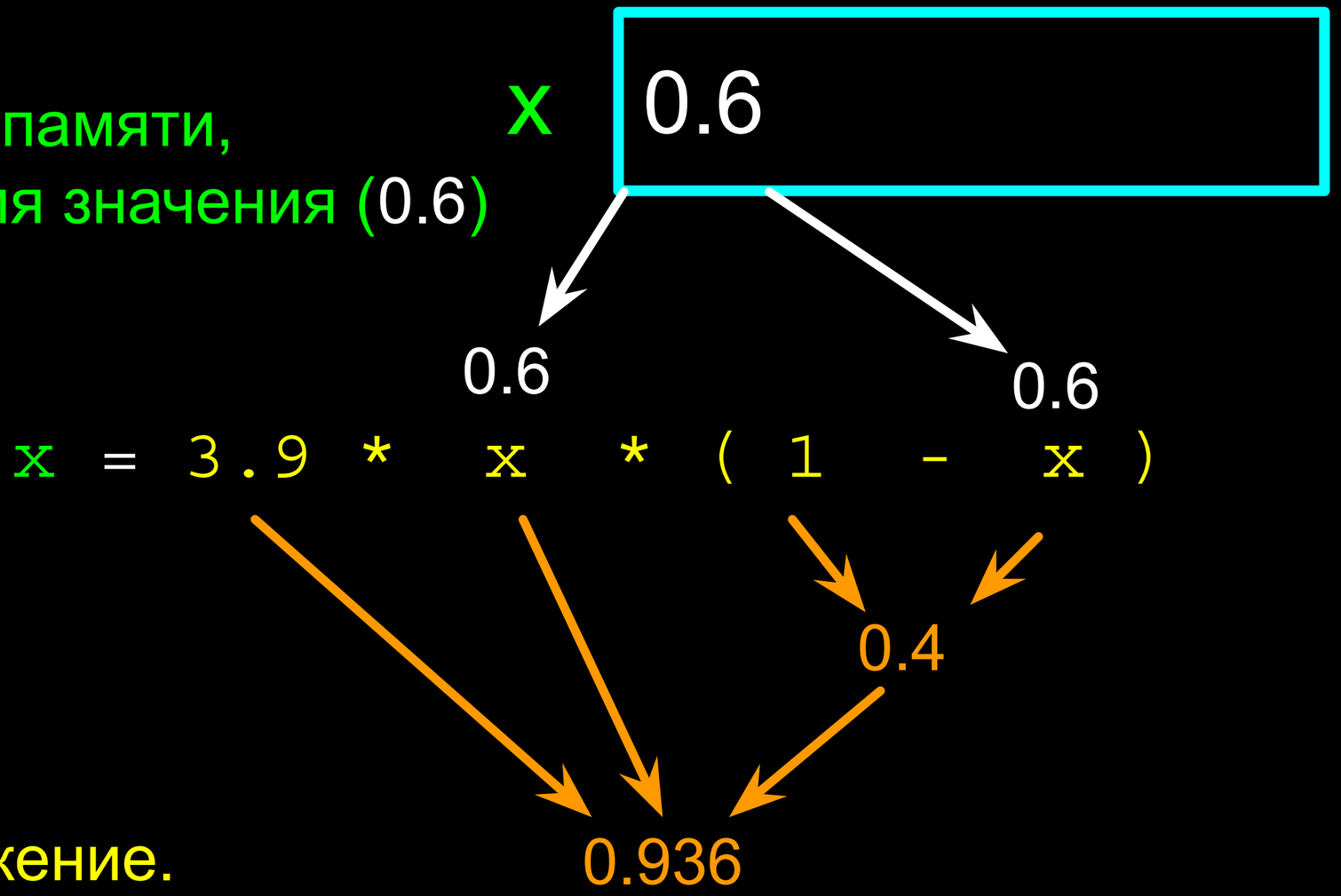
Переменная Оператор Константа Функция

Оператор присваивания

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

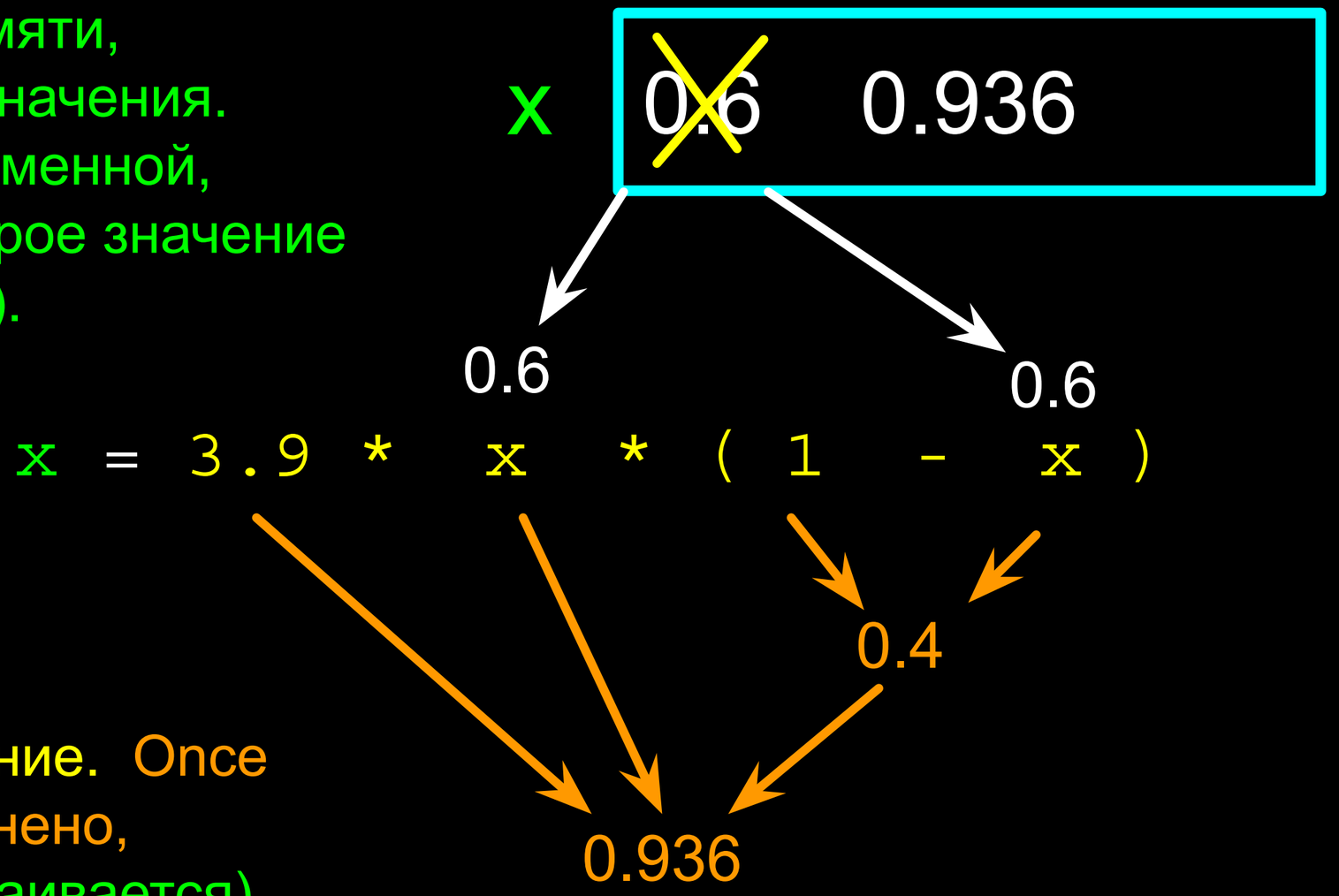
```
x = 3.9 * x * ( 1 - x )
```

Переменная - это область памяти,
используемая для хранения значения (0.6)



Правая сторона - это выражение.
Как только выражение выполнено, результат помещается в (присваивается) x.

Переменная - это область памяти, используемая для хранения значения. Значение, хранящееся в переменной, можно обновить, заменив старое значение (0,6) новым значением (0,936).



Правая сторона - это выражение. Once Как только выражение выполнено, результат помещается (присваивается) переменной слева (у нас это x).

Выражения…

Числовые выражения

- Из-за отсутствия математических символов на компьютерных клавиатурах мы используем «компьютерный язык» для выражения классических математических операций.
- Звездочка - это умножение
- Возведение в степень выглядит иначе, чем в математике

Оператор	Операция
+	Сумма
-	Разность
*	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)
5.28
```

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

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

Оператор	Операция
+	Сумма
-	Разность
*	Умножение
/	Деление
**	Степень
%	Остаток

Приоритет операторов

- Когда мы объединяем операторы вместе - Python должен знать, какой из них делать в первую очередь
- Это называется “приоритет операторов”
- Какой оператор «имеет приоритет» над другими?

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

Правила приоритета операторов

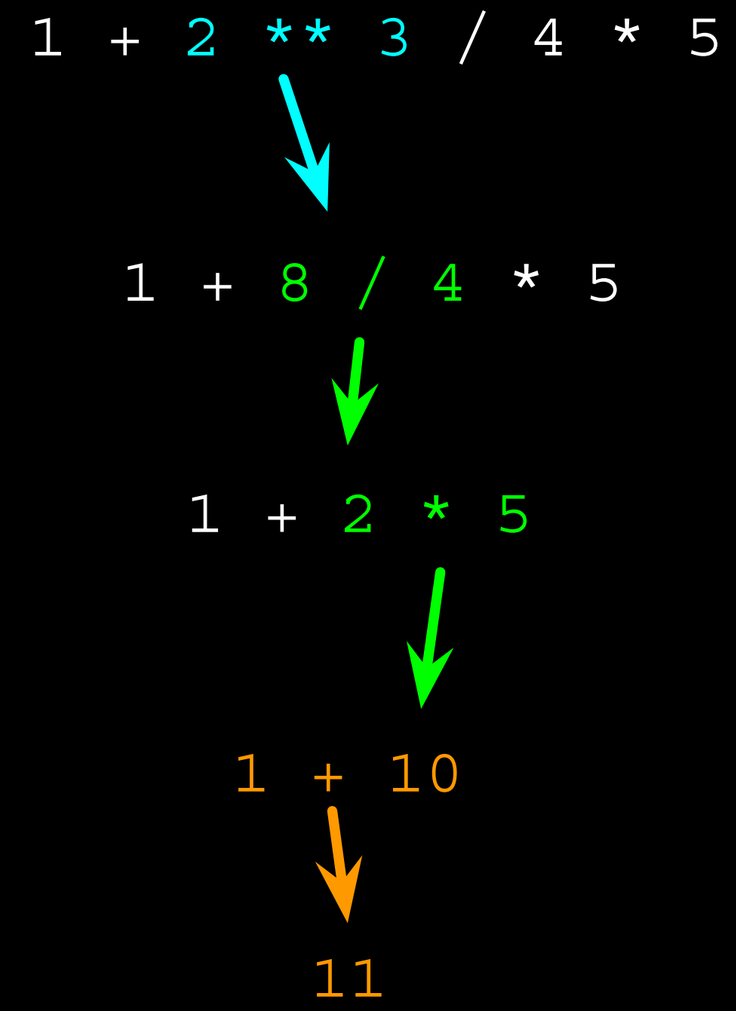
Правило наивысшего приоритета к правилу низшего приоритета:

- Круглые скобки всегда соблюдаются
- Возведение в степень
- Умножение, деление и остаток
- Сложение и вычитание
- Слева на право

Скобки
Степень
Умножение
Сложение
Слева на право

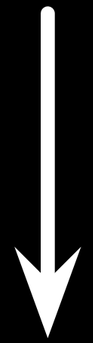
```
>>> x = 1 + 2 ** 3 / 4 * 5
>>> print(x)
11.0
>>>
```

Скобки
Степень
Умножение
Сложение
Слева на право



Приоритет операторов

Скобки
Степень
Умножение
Сложение
Слева на право



- Помните правила сверху вниз
- При написании кода – используйте круглые скобки
- При написании кода - делайте математические выражения достаточно простыми, чтобы их было легко понять.
- Разбейте длинные серии математических операций, чтобы сделать их более понятными

Что такое «тип» данных?

- В Python переменные, литералы и константы имеют «тип»
- Python знает разницу между целым числом и строкой
- Например, «+» означает «сложение», если что-то является числом, и так же он означает «объединить», если что-то является строкой.

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

Конкатенация = соединить строку

Type Matters

- Python знает, что такое «тип»
- Некоторые операции запрещены
- К примеру вы не можете добавить число (“ + 1”) к строке
- Мы можем спросить Python, что это за тип, используя функцию type()

```
>>> eee = 'hello ' + 'there'
>>> eee = eee + 1
Traceback (most recent call last):
File "<stdin>", line 1, in
<module>TypeError: Can't convert
'int' object to str implicitly
>>> type(eee)
<class'str'>
>>> type('hello')
<class'str'>
>>> type(1)
<class'int'>
>>>
```

Несколько типов чисел

- У чисел есть два основных типа

-Integer это все целые числа:
-14, -2, 0, 1, 100, 401233

- Floating Point Numbers (Число с плавающеё точкой, float) имеет десятичную часть: -2.5 , 0.0, 98.6, 14.0

- Есть и другие типы чисел - это вариации чисел с плавающей запятой и целых чисел.

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

Преобразования типов

- Когда вы помещаете в выражение целое число и число с плавающей запятой, целое число неявно преобразуется в число с плавающей запятой.
- Вы можете контролировать это с помощью встроенных функций int() и float().

```
>>> print(float(99) + 100)
199.0
>>> i = 42
>>> type(i)
<class'int'>
>>> f = float(i)
>>> print(f)
42.0
>>> type(f)
<class'float'>
>>>
```

Целочисленное деление

Целочисленное деление дает результат с плавающей запятой

```
>>> print(10 / 2)
5.0
>>> print(9 / 2)
4.5
>>> print(99 / 100)
0.99
>>> print(10.0 / 2.0)
5.0
>>> print(99.0 / 100.0)
0.99
```

В Python 2.x все было иначе.

Преобразования строк

- Вы также можете использовать int() и float() для преобразования строк в целые числа.
- Вы получите сообщение об ошибке, если строка не содержит числовых символов.

```
>>> sval = '123'
>>> type(sval)
<class 'str'>
>>> print(sval + 1)
Traceback (most recent call last):
File "<stdin>", line 1, in <module>
TypeError: Can't convert 'int' object
to str implicitly
>>> ival = int(sval)
>>> type(ival)
<class '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()
with base 10: 'x'
```

Пользовательский ввод

- Мы можем указать приостановить выполнение программы и попросить пользователя ввести данные через input ().
- Функция input() всегда возвращает строку

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

Введи своё имя? Chuck
Привет, Chuck!

Конвертирование ввода

- Если мы хотим прочитать число от пользователя, мы должны преобразовать его из строки в число, используя функцию преобразования типа .
- Позже мы разберемся с неверными входными данными

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

Сколько этажей? 0
У вас 1 этаж

Комментарии в Python

- Все, что находится после #, игнорируется Python
- Зачем нужны комментарии?

  - Опишите, что будет происходить в коде

  - Документируйте, кто написал код или другую вспомогательную информацию

  - Отключите строку кода - возможно, временно

```
# Получает название файла и открываем его
name = input('Enter file:')
handle = open(name, 'r')

# Подсчёт частоты слов
counts = dict()
for line in handle:
    words = line.split()
    for word in words:
        counts[word] = counts.get(word,0) + 1

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

# Всё завершено
print(bigword, bigcount)
```

Итог Модуль 1

- Типы
- Зарезервированные слова
- Переменные (мнемоника)
- Операторы
- Приоритет операторов
- Челочисленное деление
- Конвертирование типов
- Пользовательский ввод
- Комментарии (#)

Упражнения

Упражнения по модулю 1:

https://repl.it/@nesterenkov/PythonHomeWork#module1/tasks.txt