Section IV: Условные выражения

Section IV: Условные выражения

Логические выражения (Boolean expression)

```
>>> 5 == 5
True

>>> 5 == 6
False
```

```
>>> type(True)
<type 'bool'>

>>> type(False)
<type 'bool'>
```

Операторы отношений:

- `x == y`
- `x != y`
- `x > y`
- `x < y`
- `x >= y`
- `x <= y`

~~`x = y`~~

~~`x =< y`~~

~~`x => y`~~

Section IV: Условные выражения

Оператор ветвления `if-else`. Логические операторы `and`, `or`, `not`

```
if x > 0:
    print('x is positive')
```

```
if x % 2 == 0:
    print('x is even')
else:
    print('x is odd')
```

```
if x % 2 == 0 and x % 5 == 0:
    print('x is divided by 10')

if x % 2 == 0 or x % 5 == 0:
    print('x is divided by 2 or 5')

if not x % 2 == 0:
    print('x is not even')
```

плохой пример использования

Section IV: Условные выражения

Оператор ветвления `if-else`. Логические операторы `and`, `or`, `not`

```
>>> if 17:
...     print("It is true!")
... else:
...     print("It is false!")
```

```
It is true!
```

```
>>> if 0:
...     print("It is true!")
... else:
...     print("It is false!")
```

```
It is false!
```

```
>>> if -1:
...     print("It is true!")
... else:
...     print("It is false!")
```

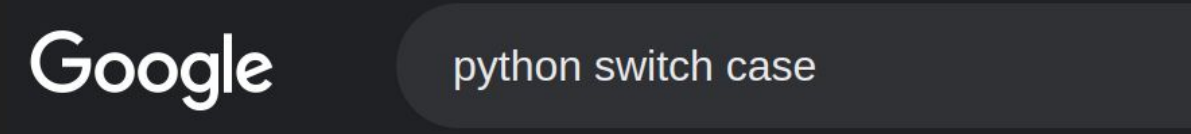
```
It is true!
```

Section IV: Условные выражения

Множественное ветвление `if-elif-else`

```
if x == y:
    print 'x and y are equal'
else:
    if x < y:
        print 'x is less than y'
    else: print 'x is greater than y'
```

```
if x == y:
    print 'x and y are equal'
elif x < y:
    print 'x is less than y'
else: print 'x is greater than y'
```



Section IV: Условные выражения

Немного рекурсии…

```
def recursion():
    recursion()
```

```
>>> recursion()
```

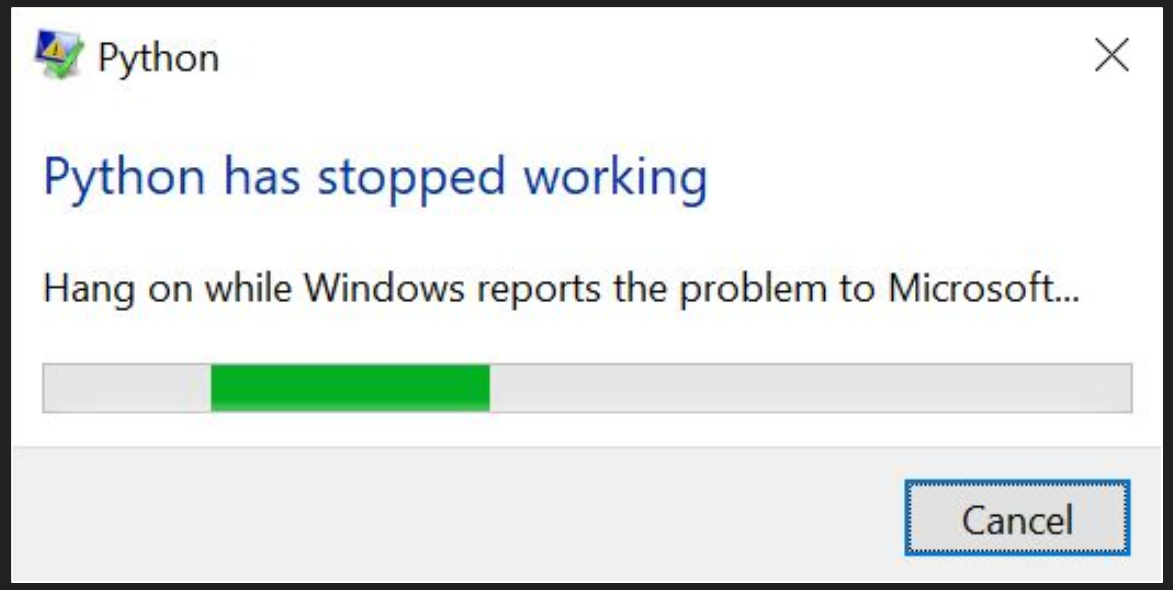
```
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
  File "<stdin>", line 2, in recursion
  File "<stdin>", line 2, in recursion
  File "<stdin>", line 2, in recursion
  [Previous line repeated 996 more times]
RecursionError: maximum recursion depth exceeded
```

Section IV: Условные выражения

~~Не~~много рекурсии…

```
import sys
sys.setrecursionlimit(10**6)
```

```
>>> recursion()
```



Section V: Итерации

Section V: Итерации

Обновление значения переменных. Операторы присваивания

```
>>> x = x + 1
```

```
NameError: name 'x' is not defined
```

```
>>> x = 0
>>> x = x + 1
```

```
>>> x += 1
```

- `+=`
- `-=`
- `*=`
- `/=`
- `%=`
- `**=`
- `//=`

Section V: Итерации

Цикл `while`

```
1  def countdown(n):
2      while n > 0:
3          print(n)
4          n = n-1
5      print('Blastoff!')
```

```
1  while True:
2      pass
```

`pass` – нулевой (null) оператор.

Section V: Итерации

Цикл `while`. Операторы `break` и `continue`

```
1  while True:
2      line = input('> ')
3      if line == 'done':
4          break
5      print(line)
6  print('Done!')
```

```
1  i = 0
2  while i < 6:
3      i += 1
4      if i == 3:
5          print('Skipping i=3...')
6          continue
7      print(i)
```

`input()` – метод, запрашивающий ввод строки.

Section V: Итерации

Цикл `for` – iterator-based или collection-based цикл

C++

```
for (i = 1; i <= 10; i++)
    <loop body>
```

Python

```
for i in <collection>:
    <loop body>
```

Примеры коллекций: `list`, `tuple`, `dict`, `string`...

Section V: Итерации

Цикл `for` – iterator-based или collection-based цикл

C++

```cpp
for (i = 0; i < 10; i++)
    cout << i
```

```cpp
for (i = 2; i < 6; i++)
    cout << i
```

```cpp
for (i = 2; i < 30; i+=3)
    cout << i
```

Python

```python
for i in range(10):
    print(i)
```

```python
for i in range(2, 6):
    print(i)
```

```python
for i in range(2, 30, 3):
    print(i)
```

Section V: Итерации

Цикл `for` – iterator-based или collection-based цикл

```python
fruits = ["apple", "banana", "cherry"]
for x in fruits:
    print(x)
```

```
apple
banana
cherry
```

```python
fruits = ["apple", "banana", "cherry"]
for x in fruits:
    if x == "banana":
        break
    print(x)
```

Section V: Итерации

Волшебное слово `else`

```
1   fruits = ["apple", "banana", "cherry"]
2   for x in fruits:
3       if x == "banana":
4           break
5   else:
6       print('There is no banana here!')
```

```
1   i = 1
2   while i < 6:
3       print(i)
4       i += 1
5   else:
6       print("i is no longer less than 6")
```

Section VI: Строки

Section VI: Строки

Строка – это последовательность. Оператор `len`

```
>>> fruit = 'banana'
>>> letter = fruit[1]
>>> print(letter)
```

```
a
```

```
>>> type(letter)
```

```
<class 'str'>
```

```
>>> letter = fruit[1.5]
TypeError: string indices must be integers
```

Section VI: Строки

Строка – это последовательность. Оператор `len`

```
>>> fruit = 'banana'
>>> len(fruit)
```

```
6
```

```
>>> length = len(fruit)
>>> last_char = fruit[length]
```

```
IndexError: string index out of range
```

```
>>> last_char = fruit[length - 1]
```

```
>>> last_char = fruit[-1]
```

Section VI: Строки

Итерирование по строке

```
1   fruit = 'banana'
2   index = 0
3   while index < len(fruit):
4       letter = fruit[index]
5       print(letter, end='')
6       index += 1
```

```
1   for char in fruit:
2       print(char, end='')
```

```
banana
```

`print(x, end='\n')` – параметр end определяет конец строки.

Section VI: Строки

Ключевое слово `in`

```
>>> 'a' in 'banana'
True
>>> 'seed' in 'banana'
False
```

```
1  def in_both(word1, word2):
2      for letter in word1:
3          if letter in word2:
4              print letter
```

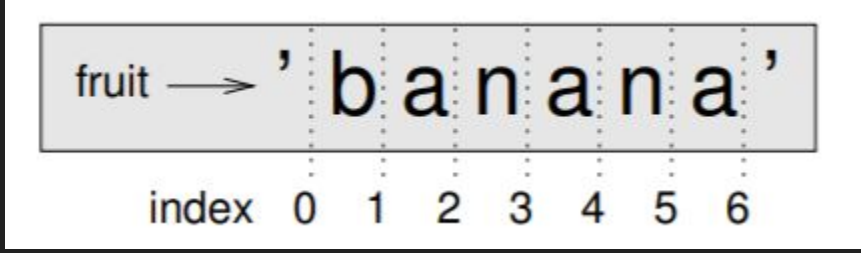
Section VI: Строки

Слайсы

```
>>> s = 'Monty Python'
>>> print(s[0:5])
Monty
>>> print(s[6:12])
Python
```

```
>>> fruit = 'banana'
>>> fruit[:3]
'ban'
>>> fruit[3:]
'ana'
```

```
>>> fruit[3:3]
''
```



Section VI: Строки

Слайсы

```
>>> fruit = 'banana'
>>> print(fruit[:])
```

```
banana
```

```
>>> print(fruit[-1:])
```

```
a
```

```
>>> print(fruit[:-1])
```

```
banan
```

```
>>> print(fruit[::-1])
```

Section VI: Строки

Слайсы с шагом

```
>>> fruit = 'banana'
>>> print(fruit[::-1])
```

```
ananab
```

```
>>> start_idx = 1
>>> end_idx = 4
>>> step = 2
>>> print(fruit[start_idx:end_idx:step])
```

```
aa
```

```
>>> step = -2
```

Section VI: Строки

Строки – неизменяемый тип данных (immutable)

```
>>> greeting = 'Hello, world!'
>>> greeting[0] = 'J'
TypeError: 'str' object does not support item
assignment
```

```
>>> greeting = 'Hello, world!'
>>> new_greeting = 'J' + greeting[1:]
>>> print new_greeting
Jello, world!
```

Section VI: Строки

Поиск подстроки

```
1  def find(word, letter):
2      index = 0
3      while index < len(word):
4          if word[index] == letter:
5              return index
6          index = index + 1
7      return -1
```

Section VI: Строки

Строковые методы

```
>>> word = 'banana'
>>> new_word = word.upper()
>>> print(new_word)
BANANA
```

```
>>> word = 'banana'
>>> index = word.find('a')
>>> print(index)
1
```

```
>>> word.find('na')
2
```

```
>>> word.find('na', 3, 5)
4
```

Section VI: Строки

Строковые методы

count()	Returns the number of times a specified value occurs in a string
endswith()	Returns true if the string ends with the specified value
find()	Searches the string for a specified value and returns the position of where it was found
strip()	Returns a trimmed version of the string

join()	Joins the elements of an iterable to the end of the string
split()	Splits the string at the specified separator, and returns a list
rsplit()	Splits the string at the specified separator, and returns a list
replace()	Returns a string where a specified value is replaced with a specified value

https://www.w3schools.com/python/python_ref_string.asp

Section VI: Строки

Строковые методы: `format`

Old

```
'%s %s' % ('one', 'two')
```

New

```
'{} {}'.format('one', 'two')
```

Output

```
one two
```

Old

```
'%d %d' % (1, 2)
```

New

```
'{} {}'.format(1, 2)
```

Output

```
1 2
```

Section VI: Строки

Строковые методы: `format`

New

```
'{1} {0}'.format('one', 'two')
```

Output

```
t w o   o n e
```

This operation is not available with old-style formatting.

Section VI: Строки

Строковые методы: `format`

Old

```
'%10s' % ('test',)
```

New

```
'{:>10}'.format('test')
```

Output

```
      test
```

Old

```
'%-10s' % ('test',)
```

New

```
'{:10}'.format('test')
```

Output

```
test
```

Section VI: Строки

Строковые методы: `format`

Old

```
'%06.2f' % (3.141592653589793,)
```

New

```
'{:06.2f}'.format(3.141592653589793)
```

Output

```
003.14
```

Section VI: Строки

Строковые методы: `format`

Setup

```
from datetime import datetime
```

New

```
'{:%Y-%m-%d %H:%M}'.format(datetime(2001, 2, 3, 4, 5))
```

Output

```
2 0 0 1 - 0 2 - 0 3   0 4 : 0 5
```

https://pyformat.info/

Section VI: Строки

Сравнение строк

```
1   word = 'Pineapple'
2   if word == 'banana':
3       print('All right, bananas.')
4   if word < 'banana':
5       print('Your word, {}, comes before banana.'.format(word))
6   elif word > 'banana':
7       print('Your word, {}, comes after banana.'.format(word))
8   else:
9       print('All right, bananas.')
```

```
Your word, Pineapple, comes before banana.
```