

Section VII: Листы

Section VII: Листы

Лист – это последовательность. Инициализация

```
>>> cheeses = ['Cheddar', 'Edam', 'Gouda']
```

```
>>> numbers = [17, 123]
```

```
>>> mixed = ['spam', 2.0, 5, [10, 20]]
```

```
>>> empty = []
```

```
>>> print(cheeses, numbers, mixed, empty)
['Cheddar', 'Edam', 'Gouda'] [17, 123] ['spam', 2.0, 5, [10, 20]] []
```

Section VII: Листы

Лист – изменяемый тип данных

```
>>> cheeses = ['Cheddar', 'Edam', 'Gouda']  
>>> print(cheeses[0])  
Cheddar
```

```
>>> numbers = [17, 123]  
>>> numbers[1] = 5  
>>> print(numbers)  
[17, 5]
```

Листы, как и строки:

- Имеют целочисленные индексы
- При обращении по несуществующему индексу – `IndexError`
- Отрицательные индексы начинаются с конца

Section VII: Листы

Итерации по листам

```
>>> cheeses = ['Cheddar', 'Edam', 'Gouda']  
>>> for cheese in cheeses:  
...     print(cheese)
```

```
>>> for idx in range(len(numbers)):  
...     numbers[idx] = numbers[idx] * 2
```

```
>>> for x in []:  
...     print('Will this ever happen?')
```

Section VII: Листы

Итерации по листам

```
>>> data = ['spam', 1, ['Brie', 'Roquefort', 'Pol le Veq'], [1, 2, 3]])
>>> for element in data:
...     print(element)
```

```
spam
1
['Brie', 'Roquefort', 'Pol le Veq']
[1, 2, 3]
```

Section VII: Листы

Операции, слайсы

```
>>> t = ['a', 'b', 'c', 'd', 'e', 'f']
>>> t[1:3]
['b', 'c']
>>> t[:4]
['a', 'b', 'c', 'd']
>>> t[3:]
['d', 'e', 'f']
```

```
>>> t[:]
['a', 'b', 'c', 'd', 'e', 'f']
```

```
>>> t = ['a', 'b', 'c', 'd', 'e', 'f']
>>> t[1:3] = ['x', 'y']
>>> print(t)
['a', 'x', 'y', 'd', 'e', 'f']
```

Section VII: Листы

Удаление элементов

```
>>> t = ['a', 'b', 'c']
>>> x = t.pop(1)
>>> print(t)
['a', 'c']
>>> print(x)
b
```

```
>>> t = ['a', 'b', 'c']
>>> del t[1]
>>> print(t)
['a', 'c']
```

```
>>> t = ['a', 'b', 'c']
>>> t.remove('b')
>>> print(t)
['a', 'c']
```

```
>>> t = ['a', 'b', 'c', 'd', 'e', 'f']
>>> del t[1:5]
>>> print t()
['a', 'f']
```

Section VII: Листы

Листы и строки

```
>>> s = 'spam'
>>> t = list(s)
>>> print(t)
['s', 'p', 'a', 'm']
```

```
>>> s = 'pining for the fjords'
>>> t = s.split()
>>> print(t)
['pining', 'for', 'the', 'fjords']
```

```
>>> t = ['pining', 'for', 'the', 'fjords']
>>> delimiter = ' '
>>> delimiter.join(t)
'pining for the fjords'
```

```
>>> s = 'spam-spam-spam'
>>> delimiter = '-'
>>> s.split(delimiter)
['spam', 'spam', 'spam']
```


Section VII: Листы

Методы

<code>append()</code>	Adds an element at the end of the list
<code>clear()</code>	Removes all the elements from the list
<code>copy()</code>	Returns a copy of the list
<code>count()</code>	Returns the number of elements with the specified value
<code>extend()</code>	Add the elements of a list (or any iterable), to the end of the current list
<code>index()</code>	Returns the index of the first element with the specified value
<code>insert()</code>	Adds an element at the specified position
<code>pop()</code>	Removes the element at the specified position
<code>remove()</code>	Removes the first item with the specified value
<code>reverse()</code>	Reverses the order of the list
<code>sort()</code>	Sorts the list

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

Section VII: Листы

Операции `map`, `reduce`, `filter`

```
1  def only_upper(t):
2      res = []
3      for s in t:
4          if s.isupper():
5              res.append(s)
6      return res
7
8  s = ['Cheddar', 'EDAM', 'Gouda']
9  print(only_upper(s))
```

```
['EDAM']
```

Section VII: Листы


Операции `map`, `reduce`, `filter`

```
1 def isupper(s):
2     if s.isupper() == True:
3         return True
4     else:
5         return False
```

```
5 s_upper = list(filter(isupper, s))
6 print(s_upper)
```

```
1 s = ['Cheddar', 'EDAM', 'Gouda']
2 s_upper = list(filter(str.isupper, s))
3 print(s_upper)
```

`list()` –
преобразование к листу
(инициализация)



Section VII: Листы

Операции `map`, `reduce`, `filter`

```
1  def capitalize_all(t):
2      res = []
3      for s in t:
4          res.append(s.capitalize())
5      return res
6
7  s = ['cheddar', 'edam', 'gouda']
8  print(capitalize_all(s))
```

```
['Cheddar', 'Edam', 'Gouda']
```

```
1  s_cap = list(map(str.capitalize, s))
2  print(s_cap)
```

```
['Cheddar', 'Edam', 'Gouda']
```

Section VII: Листы

Операции `map`, `reduce`, `filter`

```
1 def capitalize_all(t):
2     res = []
3     for s in t:
4         res.append(s.capitalize())
5     return res
```

```
1 s_cap = list(map(capitalize_all, s))
2 print(s_cap)
```

```
[['C', 'H', 'E', 'D', 'D', 'A', 'R'], ['E', 'D', 'A', 'M'], ['G', 'O', 'U', 'D', 'A']]
```

Section VII: Листы

Операции `map`, `reduce`, `filter`

```
1 circle_areas = [3.56773, 5.57668, 4.00914, 56.24241, 9.01344, 32.00013]
2
3 result = list(map(round, circle_areas, 2))
4
5 print(result)
```

`TypeError: 'int' object is not iterable`

```
1 circle_areas = [3.56773, 5.57668, 4.00914, 56.24241, 9.01344, 32.00013]
2 decimal_points = [2, 2, 2, 2, 2, 2]
3
4 result = list(map(round, circle_areas, decimal_points))
5
6 print(result)
```

```
[3.57, 5.58, 4.01, 56.24, 9.01, 32.0]
```

Section VII: Листы

Операции map, reduce, filter

```
1 def add_all(t):
2     total = 0
3     for x in t:
4         total += x
5     return total
```

```
1 t = [1, 2, 3]
2 print(sum(t))
```

6

```
1 from functools import reduce
2
3 def custom_sum(a, b):
4     return a + b
5
6 t = [1, 2, 3]
7 print(reduce(custom_sum, t))
```

Section VII: Листы

Объекты и значения

```
a = 'banana'  
b = 'banana'
```

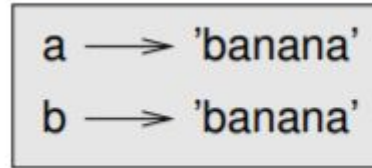
```
>>> a = 'banana'  
>>> b = 'banana'  
>>> a is b
```

True

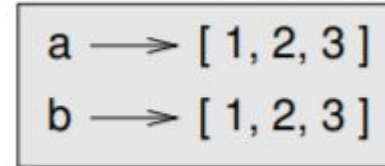
```
>>> a = [1, 2, 3]  
>>> b = [1, 2, 3]  
>>> a is b
```

False

Эквивалентность



Идентичность



Почему?

Section VII: Листы

Алиасы

```
>>> a = [1, 2, 3]
>>> b = [1, 2, 3]
>>> a is b
```

False

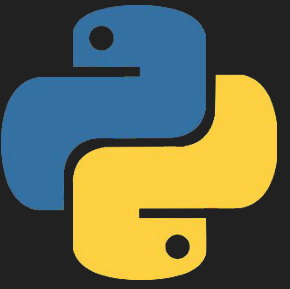
```
>>> a = [1, 2, 3]
>>> b = a
>>> a is b
```

True

```
>>> b[0] = 17
>>> print a
```

[17, 2, 3]





Section VIII: Словари

Section VIII: Словари

Словарь – отображение. Инициализация

```
>>> eng2ru = dict()
>>> print(eng2ru)
{}
```

```
>>> eng2ru['one'] = 'один'
```

```
>>> print(eng2ru)
{'one': 'один'}
```

```
eng2ru = {'one': 'один',
          'two': 'два',
          'three': 'три'}
```

```
>>> print(eng2ru)
{'one': 'один', 'two': 'два',
 'three': 'три'}
```

```
>>> print(eng2ru['one'])
один
```

```
>>> print(eng2ru['four'])
KeyError: 'four'
```

Section VIII: Словари

Словарь как набор счетчиков

```
1  def histogram(input_string):
2      count_dict = {}
3      for char in input_string:
4          if char not in count_dict:
5              count_dict[char] = 1
6          else:
7              count_dict[char] += 1
8      return count_dict
```

```
10  h = histogram('brontosaurus')
11  print(h)
```

```
{'b': 1, 'r': 2, 'o': 2, 'n': 1, 't': 1, 's': 2, 'a': 1, 'u': 2}
```

Section VIII: Словари

Словарь как набор счетчиков. Метод `get`

```
1 d = {'a': 1}
2 print(d.get('a', default))
```

1

```
1 def histogram(input_string):
2     count_dict = {}
3     for i in input_string:
4         count_dict[i] = count_dict.get(i, 0) + 1
5     return count_dict
6
7 h = histogram('brontosaurus')
8 print(h)
```

```
{'b': 1, 'r': 2, 'o': 2, 'n': 1, 't': 1, 's': 2, 'a': 1, 'u': 2}
```

Section VIII: Словари

Итерации по словарю

```
1  def print_dict(dictionary):  
2      for key in dictionary:  
3          print('{}: {}'.format(key, dictionary[key]))
```

```
1  def print_dict(dictionary):  
2      for key, value in dictionary.items():  
3          print('{}: {}'.format(key, value))
```

Section VIII: Словари

Методы

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

<u>clear()</u>	Removes all the elements from the dictionary
<u>copy()</u>	Returns a copy of the dictionary
<u>fromkeys()</u>	Returns a dictionary with the specified keys and value
<u>get()</u>	Returns the value of the specified key
<u>items()</u>	Returns a list containing a tuple for each key value pair
<u>keys()</u>	Returns a list containing the dictionary's keys
<u>pop()</u>	Removes the element with the specified key
<u>popitem()</u>	Removes the last inserted key-value pair
<u>setdefault()</u>	Returns the value of the specified key. If the key does not exist: insert the key, with the specified value
<u>update()</u>	Updates the dictionary with the specified key-value pairs
<u>values()</u>	Returns a list of all the values in the dictionary

Section VIII: Словари

Немного о глобальных переменных...

```
1 verbose = True
2
3 def example_1():
4     if verbose:
5         print('Running example_1.')
6
7 example_1()
```

Running example_1.

```
1 been_called = False
2
3 def example_2():
4     been_called = True
5
6 example_2()
7 print(been_called)
```

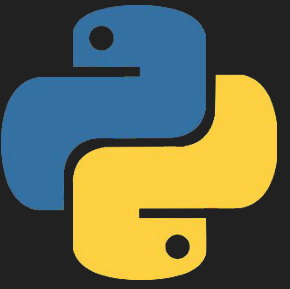
False

Section VIII: Словари

Немного о глобальных переменных...

```
1  been_called = False
2
3  def example_3():
4      global been_called
5      been_called = True
6
7  example_3()
8  print(been_called)
```

True



Section IX: Кортежи (tuples)

Section IX: Кортежи (tuples)

Кортежи: почти как листы, только неизменяемые (immutable)

```
>>> t = 'a', 'b', 'c', 'd', 'e'
```

```
>>> t = ('a', 'b', 'c', 'd', 'e')
```

```
>>> t = tuple('lupins')  
>>> print(t)  
( 'l', 'u', 'p', 'i', 'n', 's')
```

```
>>> t[0] = 'A'  
TypeError: object doesn't support item assignment
```

```
>>> t = ('A',) + t[1:]  
>>> print(t)  
( 'A', 'b', 'c', 'd', 'e')
```

Section IX: Кортежи (tuples)

Tuple Assignment

```
>>> temp = a
>>> a = b
>>> b = temp
```

```
>>> a, b = b, a
```

```
>>> a, b = 1, 2, 3
ValueError: too many values to unpack
```

```
>>> addr = 'monty@python.org'
>>> uname, domain = addr.split('@')
```

Swap integers without additional variable?

CHALLENGE ACCEPTED



```
a = a + b;
b = a - b;
a = a - b;
```

PLEASE



```
a,b = b,a
```

Section IX: Кортежи (tuples)

Кортеж как возвращаемое значение

```
>>> t = divmod(7, 3)
>>> print(t)
(2, 1)
```

```
>>> quot, rem = divmod(7, 3)
>>> print(quot)
2
>>> print(rem)
1
```

```
>>> def min_max(t):
...     return min(t), max(t)
```

Section IX: Кортежи (tuples)

Переменное количество аргументов

```
>>> def printall(*args):  
...     print(args)
```

```
>>> printall(1, 2.0, '3')  
(1, 2.0, '3')
```

```
>>> t = (7, 3)  
>>> divmod(t)  
TypeError: divmod expected 2  
arguments, got 1
```

```
>>> divmod(*t)  
(2, 1)
```

```
>>> def printall(**kwargs):  
...     print(kwargs)
```

```
>>> printall(**{'a': 1, 'b': 2, 'c': 3})  
{'a': 1, 'b': 2, 'c': 3}
```

Section IX: Кортежи (tuples)

Кортежи и листы. Функция `zip`

```
>>> s = 'abc'
>>> t = [0, 1, 2]
>>> zip(s, t)
[('a', 0), ('b', 1), ('c', 2)]
```

```
>>> zip('Anne', 'Elk')
[('A', 'E'), ('n', 'l'), ('n', 'k')]
```

```
>>> t = [('a', 0), ('b', 1), ('c', 2)]
>>> for letter, number in t:
...     print(number, letter)
```

Section IX: Кортежи (tuples)

Кортежи и листы. Функции `zip`, `enumerate`

```
1  def has_match(t1, t2):
2      for x, y in zip(t1, t2):
3          if x == y:
4              return True
5      return False
```

```
1  for index, element in enumerate('abc'):
2      print(index, element)
```

```
0 a
1 b
2 c
```


Section IX: Кортежи (tuples)

Кортежи и словари

```
>>> d = {'a':0, 'b':1, 'c':2}
>>> t = d.items()
>>> print(t)
[('a', 0), ('c', 2), ('b', 1)]
```

```
>>> t = [('a', 0), ('c', 2), ('b', 1)]
>>> d = dict(t)
>>> print(d)
{'a': 0, 'c': 2, 'b': 1}
```

```
>>> d = dict(zip('abc', range(3)))
>>> print(d)
{'a': 0, 'c': 2, 'b': 1}
```

Как с помощью zip и dict
создать словарь с
целочисленными ключами?