Основы Python

Словари

Словарь (Dictionary)

Словари – это изменяемый, неупорядоченный тип данных.

Словарь (ассоциативный массив, хеш-таблица):

- данные в словаре это пары "ключ:значение"
- доступ к значениям осуществляется по ключу, а не по номеру, как в списках
- словари неупорядоченны, поэтому не стоит полагаться на порядок элементов словаря
- так как словари изменяемы, то элементы словаря можно менять, добавлять, удалять
- ключ должен быть объектом неизменяемого типа:
    - число
    - строка
    - кортеж
- значение может быть данными любого типа

Словарь (Dictionary)

Пример словаря:

```
london = {'name': 'London1', 'location': 'London Str',
'vendor': 'Cisco', 'model': '4451', 'IOS': '15.4'}
```

Можно записывать и так:

```
london = {
        'id': 1,
        'name':'London',
        'IT_VLAN':320,
        'User_VLAN':1010,
        'Mngmt_VLAN':99,
        'to_name': None,
        'to_id': None,
        'port':'G1/0/11'
}
```

Словарь (Dictionary)

Для того чтобы получить значение из словаря, надо обратиться по ключу, таким же образом, как это было в списках, только вместо номера, будет использоваться ключ:

```
In [1]: london = {'name': 'London1', 'location': 'London Str'}

In [2]: london['name']
Out[2]: 'London1'

In [3]: london['location']
Out[3]: 'London Str'
```

Аналогичным образом можно добавить новую пару ключ:значение:

```
In [4]: london['vendor'] = 'Cisco'

In [5]: print london
{'vendor': 'Cisco', 'name': 'London1', 'location': 'London
Str'}
```

Словарь (Dictionary)

В словаре в качестве значения можно использовать словарь:

```
london_co = {
    'r1' : {
        'hostname': 'london_r1',
        'location': '21 New Globe Walk',
        'vendor': 'Cisco',
        'model': '4451',
        'IOS': '15.4',
        'IP': '10.255.0.1'
        },
    'sw1' : {
        'hostname': 'london_sw1',
        'location': '21 New Globe Walk',
        'vendor': 'Cisco',
        'model': '3850',
        'IOS': '3.6.XE',
        'IP': '10.255.0.101'
        }}
```

Получить значения из вложенного словаря можно так:

```
In [7]: london_co['r1']['IOS']
Out[7]: '15.4'

In [8]: london_co['r1']['model']
Out[8]: '4451'

In [9]: london_co['sw1']['IP']
Out[9]: '10.255.0.101'
```

Полезные методы для работы со словарями

Метод clear() позволяет очистить словарь:

```
In [1]: london = {'name': 'London1', 'location': 'London Str', 'vendor':
'Cisco', 'model': '4451', 'IOS': '15.4'}

In [2]: london.clear()

In [3]: london
Out[3]: {}
```

Метод copy() позволяет создать полную копию словаря.

```
In [10]: london = {'name': 'London1', 'location': 'London Str',
'vendor': 'Cisco'}

In [11]: london2 = london.copy()

In [12]: id(london)
Out[12]: 25524512

In [13]: id(london2)
Out[13]: 25563296

In [14]: london['vendor'] = 'Juniper'

In [15]: london2['vendor']
Out[15]: 'Cisco'
```

Полезные методы для работы со словарями

Если при обращении к словарю указывается ключ, которого нет в словаре, мы получаем ошибку:

```
In [16]: london = {'name': 'London1', 'location': 'London Str',
'vendor': 'Cisco'}

In [17]: london['IOS']
---------------------------------------------------------------------
---
KeyError                                  Traceback (most recent call last)
<ipython-input-17-b4fae8480b21> in <module>()
----> 1 london['IOS']

KeyError: 'IOS'
```

Метод get() позволяет запросить значение, но если его нет, вместо ошибки возвращается указанное значение (по умолчанию возвращается None):

```
In [18]: london = {'name': 'London1', 'location': 'London Str',
'vendor': 'Cisco'}

In [19]: print london.get('IOS')
None

In [20]: print london.get('IOS', 'Ooops')
Ooops
```

Полезные методы для работы со словарями

Методы keys(), values(), items():

```
In [24]: london = {'name': 'London1', 'location': 'London Str',
'vendor': 'Cisco'}

In [25]: london.keys()
Out[25]: ['vendor', 'name', 'location']

In [26]: london.values()
Out[26]: ['Cisco', 'London1', 'London Str']

In [27]: london.items()
Out[27]: [('vendor', 'Cisco'), ('name', 'London1'), ('location', 'London
Str')]
```

Удалить ключ и значение:

```
In [28]: london = {'name': 'London1', 'location': 'London Str',
'vendor': 'Cisco'}

In [29]: del(london['name'])

In [30]: london
Out[30]: {'location': 'London Str', 'vendor': 'Cisco'}
```

Варианты создания словаря

Словарь можно создать с помощью литерала:

```
In [1]: r1 = {'model': '4451', 'IOS': '15.4'}
```

Конструктор dict позволяет создавать словарь несколькими способами.

Если в роли ключей используются строки, можно использовать такой вариант создания словаря:

```
In [2]: r1 = dict(model='4451', IOS='15.4')

In [3]: r1
Out[3]: {'IOS': '15.4', 'model': '4451'}
```

Второй вариант создания словаря с помощью dict:

```
In [4]: r1 = dict([('model','4451'), ('IOS','15.4')])

In [5]: r1
Out[5]: {'IOS': '15.4', 'model': '4451'}
```

Варианты создания словаря

Если необходимо создать словарь с известными ключами, но, пока что, пустыми значениями (или одинаковыми значениями), очень удобен метод fromkeys():

```
In [5]: d_keys = ['hostname', 'location', 'vendor', 'model', 'IOS', 'IP']

In [6]: r1 = dict.fromkeys(d_keys, None)

In [7]: r1
Out[7]:
{'IOS': None,
 'IP': None,
 'hostname': None,
 'location': None,
 'model': None,
 'vendor': None}
```

Генераторы словарей. Сгенерируем словарь с нулевыми значениями, как в предыдущем примере:

```
In [16]: d_keys = ['hostname', 'location', 'vendor', 'model', 'IOS', 'IP']

In [17]: d = {x: None for x in d_keys}

In [18]: d
Out[18]:
{'IOS': None,
 'IP': None,
 'hostname': None,
 'location': None,
 'model': None,
 'vendor': None}
```

Словарь из двух списков

```
In [4]: d_keys = ['hostname', 'location', 'vendor', 'model', 'IOS', 'IP']
In [5]: d_values = ['london_r1', '21 New Globe Walk', 'Cisco', '4451', '15.4',
'10.255.0.1']

In [6]: zip(d_keys,d_values)
Out[6]:
[('hostname', 'london_r1'),
 ('location', '21 New Globe Walk'),
 ('vendor', 'Cisco'),
 ('model', '4451'),
 ('IOS', '15.4'),
 ('IP', '10.255.0.1')]

In [7]: dict(zip(d_keys,d_values))
Out[7]:
{'IOS': '15.4',
 'IP': '10.255.0.1',
 'hostname': 'london_r1',
 'location': '21 New Globe Walk',
 'model': '4451',
 'vendor': 'Cisco'}
In [8]: r1 = dict(zip(d_keys,d_values))

In [9]: r1
Out[9]:
{'IOS': '15.4',
 'IP': '10.255.0.1',
 'hostname': 'london_r1',
 'location': '21 New Globe Walk',
 'model': '4451',
 'vendor': 'Cisco'}
```

Словарь из двух списков

```
In [10]: d_keys = ['hostname', 'location', 'vendor', 'model', 'IOS', 'IP']

In [11]: data = {
   ....: 'r1': ['london_r1', '21 New Globe Walk', 'Cisco', '4451', '15.4', '10.255.0.1'],
   ....: 'r2': ['london_r2', '21 New Globe Walk', 'Cisco', '4451', '15.4', '10.255.0.2'],
   ....: 'sw1': ['london_sw1', '21 New Globe Walk', 'Cisco', '3850', '3.6.XE', '10.255.0.101']
   ....: }

In [12]: london_co = {}

In [13]: for k in data.keys():
   ....:     london_co[k] = dict(zip(d_keys,data[k]))
   ....:

In [14]: london_co
Out[14]:
{'r1': {'IOS': '15.4',
  'IP': '10.255.0.1',
  'hostname': 'london_r1',
  'location': '21 New Globe Walk',
  'model': '4451',
  'vendor': 'Cisco'},
 'r2': {'IOS': '15.4',
  'IP': '10.255.0.2',
  'hostname': 'london_r2',
  'location': '21 New Globe Walk',
  'model': '4451',
  'vendor': 'Cisco'},
 'sw1': {'IOS': '3.6.XE',
  'IP': '10.255.0.101',
  'hostname': 'london_sw1',
  'location': '21 New Globe Walk',
  'model': '3850',
  'vendor': 'Cisco'}}
```

Python для сетевых инженеров

Автор курса: Наташа Самойленко
nataliya.samoylenko@gmail.com