The Second Topic

Learning via official docs, stack overflow, google, etc.

Mission_1

Across the Great Wall, we can reach every corner in the world

Step_1: Install *shadowsocks-qt5* on ubuntu

1. Open a terminal（Ctrl + alt + T）
2. sudo add-apt-repository ppa:hzwhuang/ss-qt5
3. sudo apt-get update
4. sudo apt-get install shadowsocks-qt5

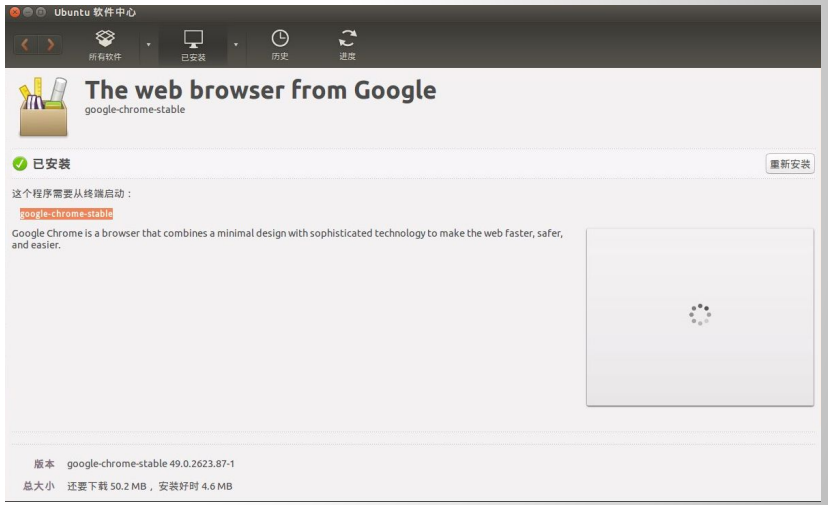
(https://github.com/shadowsocks/shadowsocks-qt5/wiki/%E5%AE%89%E8%A3%85%E6%8C%87%E5%8D%97

You will see

```
升级了 0 个软件包，新安装了 1 个软件包，要卸载 0 个软件包，有 503 个软件包未被升级。
需要下载 0 B/283 kB 的软件包。
解压缩后会消耗掉 742 kB 的额外空间。
正在选中未选择的软件包 shadowsocks-qt5。
(正在读取数据库 ... 系统当前共安装有 267598 个文件和目录。)
正准备解包 .../shadowsocks-qt5_2.7.0-1~ubuntu14.04.1_amd64.deb  ...
正在解包 shadowsocks-qt5 (2.7.0-1~ubuntu14.04.1) ...
正在处理用于 hicolor-icon-theme (0.13-1) 的触发器 ...
正在处理用于 gnome-menus (3.10.1-0ubuntu2) 的触发器 ...
正在处理用于 desktop-file-utils (0.22-1ubuntu1) 的触发器 ...
正在处理用于 bamfdaemon (0.5.1+14.04.20140409-0ubuntu1) 的触发器 ...
Rebuilding /usr/share/applications/bamf-2.index...
正在处理用于 mime-support (3.54ubuntu1.1) 的触发器 ...
正在设置 shadowsocks-qt5 (2.7.0-1~ubuntu14.04.1) ...
```

Step_2: Set Browser extensions

- The package has been uploaded to ftp://zhouzean.tk/
- Follow the steps if you know nothing. But after this class you must know you should Google before asking.
- Other Browsers such as firefox can do this, too.



Set up extensions

- Click the upper right corner

=> settings

=>extensions

- Drag the SwitchyOmega.crx into the browser
- Click import/export

=>Restore form file

=>OmegaOptions.bak

- Click auto switch

Or you can also set it system wide under linux

- ensure your proxy is connected
- install GenPAC (or gfwlist2pac)
  - Proxy Auto-config tools based on shadowsocks
  - using online gfwlist to produce autoproxy.pac file
  - System settings > Network > Network Proxy
  - Set Configuration URL to autoproxy.pac
- For details, search on Google
- So how to search?

Some skills for google.

- "mysql foreign key”
- site:http://zhihu.com/ 阿里月饼
- site:http://zhuanlan.zhihu.com/ 编程*
- d3js 2015..2016
- 肖申克的救赎 –avi
- stackoverflow.com bug
- Filetype:pdf 编程之美
- Exact hits
- Search in the site
- Using Wildcard
- Limited date
- Without something
- Using tab (using web’s self engine)
- Search for the filetype


39503 1
1UP=1
ARMS BOMB
200 3
53
PRESS START
MISSION 1
COMPLETE!
CREDITS 39
CREDIT 00

Mission_2:Learning via official docs

- There are many good resources in the homepage
- If you have bugs, or some syntax problems, you can find the answer to most of them
- The official docs are more reliable and clearer.

An example of how to use official docs.

- One question on piazza is ***in python3.4, about round()***We can find the answer ourselves via official docs.
- http://shtech.org/doc/manual/python3/library/index.html The Python Standard Library
- Search the function round()

round(*number*[, *ndigits*])

Return the floating point value *number* rounded to *ndigits* digits after the decimal point. If *ndigits* is omitted, it defaults to zero. Delegates to `number.__round__(ndigits)`.

For the built-in types supporting `round()`, values are rounded to the closest multiple of 10 to the power minus *ndigits*; if two multiples are equally close, rounding is done toward the even choice (so, for example, both `round(0.5)` and `round(-0.5)` are `0`, and `round(1.5)` is `2`). The return value is an integer if called with one argument, otherwise of the same type as *number*.

Note: The behavior of `round()` for floats can be surprising: for example, `round(2.675, 2)` gives `2.67` instead of the expected `2.68`. This is not a bug: it's a result of the fact that most decimal fractions can't be represented exactly as a float. See *Floating Point Arithmetic: Issues and Limitations* for more information.

Another question : ***about "lambda"***

- Just the same as the last question we can search in the PSL
- Then you will find the last one is about the *lambda* we want
- Click the ***lambda***(link) to the page we want

6.12. Lambdas

```
lambda_expr        ::=  "lambda" [parameter_list]: expression
lambda_expr_nocond ::=  "lambda" [parameter_list]: expression_nocond
```

Lambda expressions (sometimes called lambda forms) are used to create anonymous functions. The expression `lambda arguments: expression` yields a function object. The unnamed object behaves like a function object defined with

```
def <lambda>(arguments):
    return expression
```

See section *Function definitions* for the syntax of parameter lists. Note that functions created with lambda expressions cannot contain statements or annotations.

More questions... Just can not understand

- ***parameter_list*** and ***expression***

```
parameter_list ::=  (defparameter ",")*
                    | "*" [parameter] ("," defparameter)* ["," "**" parameter]
                    | "**" parameter
                    | defparameter [","] )
parameter      ::=  identifier [":" expression]
defparameter   ::=  parameter ["=" expression]
```

```
expression             ::=  conditional_expression | lambda_expr
expression_nocond      ::=  or_test | lambda_expr_nocond
```

```
or_test  ::=  and_test | or_test "or" and_test
and_test ::=  not_test | and_test "and" not_test
not_test ::=  comparison | "not" not_test
```

More questions... Just can not understand

- ***parameter_list*** and ***expression***

```
parameter_list ::=  (defparameter ",")*
                    | "*" [parameter] ("," defparameter)* ["," "**" parameter]
                    | "**" parameter
                    | defparameter [","] )
parameter      ::=  identifier [":" expression]
defparameter   ::=  parameter ["=" expression]
```

- There are some ***BNF*** hard for beginers
- Learn *https://en.wikipedia.org/wiki/Backus%E2%80%93Naur_Form*

```
def f1(a, b, c=0, *args, **kw):
    print('a =', a, 'b =', b, 'c =', c, 'args =', args, 'kw =', kw)
```

More questions... Just can not understand

- ***parameter_list***
- You will learn more

Function call semantics are described in more detail in section *Calls*. A function call always assigns values to all parameters mentioned in the parameter list, either from position arguments, from keyword arguments, or from default values. If the form "`*identifier`" is present, it is initialized to a tuple receiving any excess positional parameters, defaulting to the empty tuple. If the form "`**identifier`" is present, it is initialized to a new dictionary receiving any excess keyword arguments, defaulting to a new empty dictionary. Parameters after "`*`" or "`*identifier`" are keyword-only parameters and may only be passed used keyword arguments.

More questions... Just solved part of them

- ***expression***

```
conditional_expression ::=  or_test ["if" or_test "else" expression]
expression             ::=  conditional_expression | lambda_expr
expression_nocond      ::=  or_test | lambda_expr_nocond
```

- Then there comes more details
- And finally... ... It stoped

```
or_test  ::=  and_test | or_test "or" and_test
and_test ::=  not_test | and_test "and" not_test
not_test ::=  comparison | "not" not_test
```

```
comparison    ::=  or_expr ( comp_operator or_expr )*
comp_operator ::=  "<" | ">" | "==" | ">=" | "<=" | "!="
                   | "is" ["not"] | ["not"] "in"
```

What we get?

- You begin a question with “lambda”
- You know the lambda is an expression
- You will see (If you take a glance of the docs) other expressions such as expression_list
- Then learn about the parameter_list
- When you keep learning like this you will find you will learn faster than others.


1649270
1UP=4
ARMS BOMB
∞ 0
28
INSERT COIN
MISSION 2
COMPLETE!
CREDIT 00

Final Mission

Using docs, stack overflow, google, etc

If you want to know about ***Dict.values()***

- First you will go to PSL and find the explanation

values()

Return a new view of the dictionary's values. See the *documentation of view objects*.

- Ok, we will see

iter(dictview)

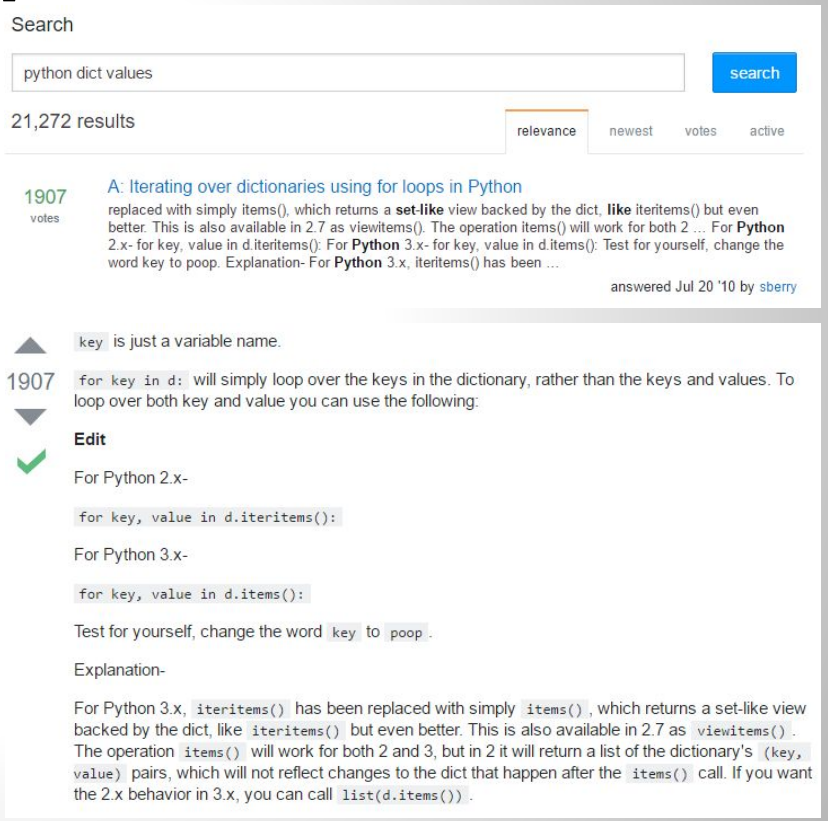
Return an iterator over the keys, values or items (represented as tuples of `(key, value)`) in the dictionary.

Keys and values are iterated over in an arbitrary order which is non-random, varies across Python implementations, and depends on the dictionary's history of insertions and deletions. If keys, values and items views are iterated over with no intervening modifications to the dictionary, the order of items will directly correspond. This allows the creation of `(value, key)` pairs using `zip()`: `pairs = zip(d.values(), d.keys())`. Another way to create the same list is `pairs = [(v, k) for (k, v) in d.items()]`.

Iterating views while adding or deleting entries in the dictionary may raise a `RuntimeError` or fail to iterate over all entries.

Go to **stack overflow**

- Search for *dict order*
- Choose one post –most time we choose the one with thousands agrees
- Try to understand it . Maybe it is not the answer but it might be useful
- Here is examples:



Find the answer or ask

Question solved?

Ordinary dictionaries are constructed from a hash table and do not preserve ordering. Use `OrderedDict` from the standard library `collections` module instead.

You can google for the hash table if you want

If you still have question go to piazza.

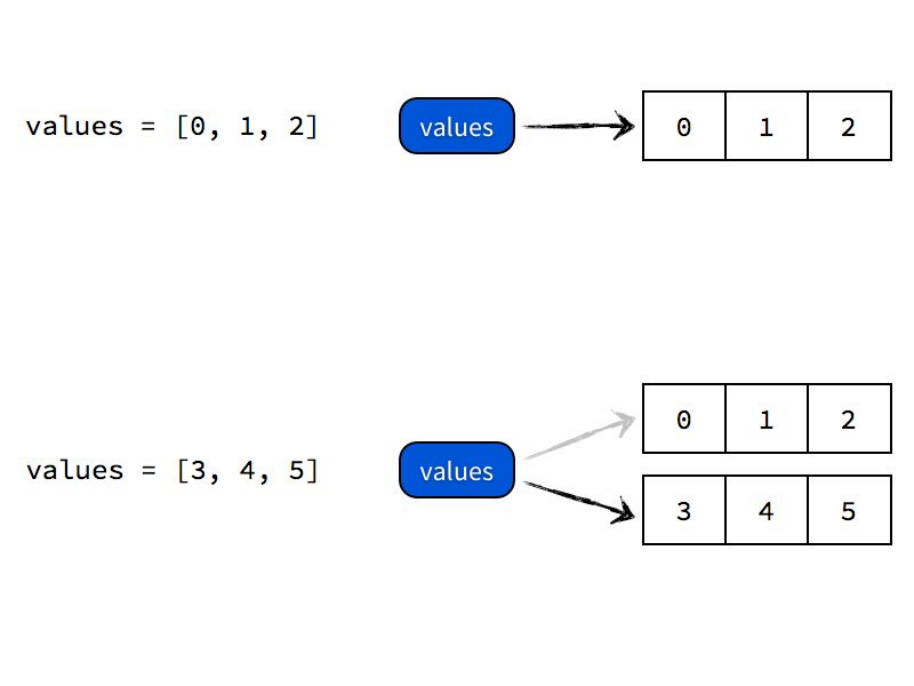

5437860
1UP=2
ARMS BOMB
200 12
25
PLEASE WAIT
FINAL
MISSION
COMPLETE!
CREDIT 0

Others

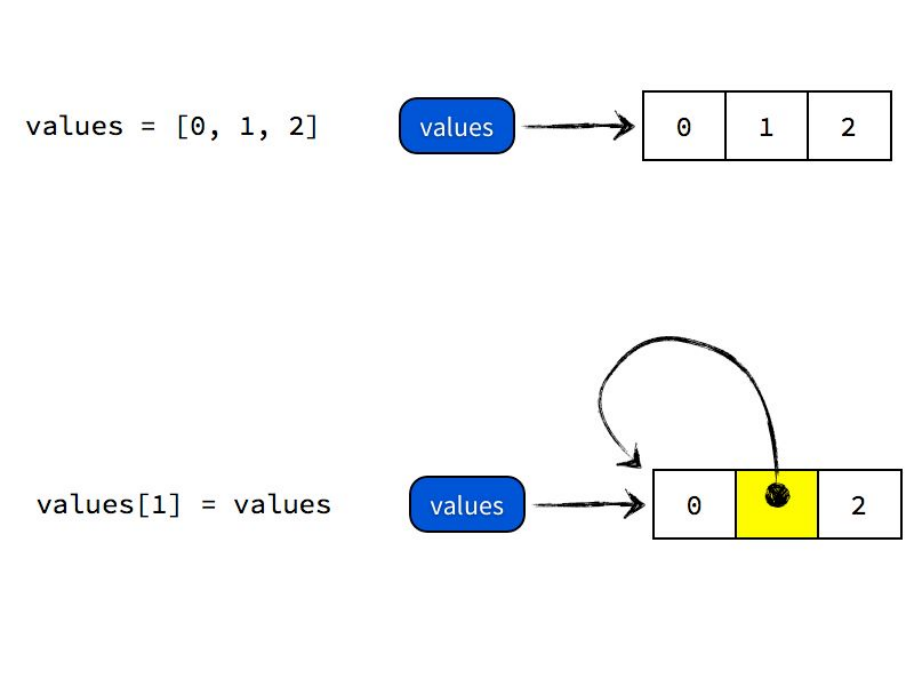
- You can alse learn via zhihu.
- Try this:

```
values = [0, 1, 2]

values[1] = values

print(values)
```

Others



Others



Others

- Search for the answer

```
a = [0, [1, 2], 3]
b = a[:]
a[0] = 8
a[1][1] = 9
print(a,b)
```

Others

```
T=[]

L=[[x for x in range(2,5)],[y for y in range(6,9)]]

for i in range(3):
    T.append((l[i] for l in L))

print(list(T[0]))

print(list((l[0] for l in L)))
```