python3语法速查

1. 技巧文章

2. 数组 list/tuple

python有可变数组和不可变数组

2.1 不可变数组tuple

创建：

```
array = ()
```

2.2 可变数组list

创建：

```
array = []
```

2.2.1 数组添加元素

添加一个元素到数组末尾：

```
array.append(insert_item)
```

添加另外一个数组的所有元素到一个数组：

```
dest_array.extend(insert_array)
```

执行效果□insert_array数组内的元素全部append到dest_array, 最终dest_array的数组元素变多。

```
s[i] = x

item i of s is replaced by x

s[i:j] = t

slice of s from i to j is replaced by the contents of the iterable t

del s[i:j]

same as s[i:j] = []

s[i:j:k] = t

the elements of s[i:j:k] are replaced by those of t

(1)

del s[i:j:k]

removes the elements of s[i:j:k] from the list

s.append(x)

appends x to the end of the sequence (same as s[len(s):len(s)] = [x])

s.clear()

removes all items from s (same as del s[:])

(5)

s.copy()

creates a shallow copy of s (same as s[:])
```

```
(5)

s.extend(t) or s += t

extends s with the contents of t (for the most part the same as
s[len(s):len(s)] = t)

s *= n

updates s with its contents repeated n times

(6)

s.insert(i, x)

inserts x into s at the index given by i (same as s[i:i] = [x])

s.pop([i])

retrieves the item at i and also removes it from s

(2)

s.remove(x)

remove the first item from s where s[i] is equal to x

(3)

s.reverse()

reverses the items of s in place

(4)
```

2.3 数组length

```
tmp_array = [1, 2]
ret_len = len(tmp_array)
```

3. dict字典

3.1 创建dict

```
#### 方式1
>>> a = dict(one=1, two=2, three=3)
```

```
>>> b = {'one': 1, 'two': 2, 'three': 3}
>>> c = dict(zip(['one', 'two', 'three'], [1, 2, 3]))
>>> d = dict([('two', 2), ('one', 1), ('three', 3)])
>>> e = dict({'three': 3, 'one': 1, 'two': 2})
>>> a == b == c == d == e
True

### 方式2
import collections

dict = collections.OrderedDict()
```

3.2 functions

```
list(d)
Return a list of all the keys used in the dictionary d.

len(d)
Return the number of items in the dictionary d.

d[key]
Return the item of d with key key. Raises a KeyError if key is not in the
map.

d[key] = value
Set d[key] to value.

del d[key]
Remove d[key] from d. Raises a KeyError if key is not in the map.

key in d
Return True if d has a key key, else False.

key not in d
Equivalent to not key in d.

keys()
Return a new view of the dictionary’s keys. See the documentation of view
objects.
```

3.3 使用举例

```
for item in a_dict.keys():
  print(f'key={item}')
```

4. string

```
class str(object='')
class str(object=b'', encoding='utf-8', errors='strict')
Return a string version of object. If object is not provided, returns the
empty string.
```

5. 文件操作

5.1 打开文件

fh = open('file.txt', 'r')

文件打开类型, r - 只读打开rb - 只读二进制文件打开 w - 只写文件打开 wb - 只写二进制文件打开, a - 追加写

5.1.1 读文件操作

```
fh = open('file.txt', 'r')

for line in fh.readlines():
    line = line.rstrip() #去除最右侧换行符
    print(f'line: {line}')

fh.close()
```

5.1.2 写文件操作

```
fh = open('file.txt', 'w', encoding='utf-8')

fh.write('haha\n')
fh.write(f'hello\n')

fh.close()
```

5.2 关闭文件

```
fh = open('file.txt', 'r', encoding='utf-8')

fh.close()
```

5.3 其它

5.3.1 os.path

```
import os

os.path.exists(path)

os.path.isfile(path)
os.path.isdir(path)
os.path.islink(path)

os.path.dirname(path)
os.path.basename(path)

os.path.abspath(path)
os.path.isabs(path)
```

5.3.2 os

```
import os


os.getenv(key, default=None)
os.putenv(key, value)
os.unsetenv(key)

os.chmod(path, mode, *, dir_fd=None, follow_symlinks=True)
os.chown(path, uid, gid, *, dir_fd=None, follow_symlinks=True)

os.link(src, dst, *, src_dir_fd=None, dst_dir_fd=None, follow_symlinks=True)
os.symlink(src, dst, target_is_directory=False, *, dir_fd=None)

os.remove(path, *, dir_fd=None)
os.unlink(path, *, dir_fd=None)
os.rename(src, dst, *, src_dir_fd=None, dst_dir_fd=None)

os.mkdir(path, mode=0o777, *, dir_fd=None)
os.rmdir(path, *, dir_fd=None)

os.listdir(path='.')
os.chdir(path)
os.getcwd()
```

6. base function

6.1 eval()

可以用来支持一些逻辑表达式的计算：

```
#!/usr/bin/python3

import re


A = '0xaa'
B = '0xcc'
C = '0xf0'


s_prem = 'A&(B|~C)'

# re.sub(pattern, repl, string, count=0, flags=0)
# flags=re.IGNORECASE

s_new_math = s_prem
s_new_math = re.sub(r'A', '0xaa', s_new_math)
s_new_math = re.sub(r'B', '0xcc', s_new_math)
s_new_math = re.sub(r'C', '0xf0', s_new_math)

print(f's_new_math = {s_new_math}')


try:
    ret = eval(s_new_math)
    print(f'ret = {hex(ret)}')
except SyntaxError as e:
    print(e)
```

6.2 getopt()

C-style parser for command line options

短argument

```
>>> import getopt
>>> args = '-a -b -cfoo -d bar a1 a2'.split()
>>> args
['-a', '-b', '-cfoo', '-d', 'bar', 'a1', 'a2']
>>> optlist, args = getopt.getopt(args, 'abc:d:')
>>> optlist
[('-a', ''), ('-b', ''), ('-c', 'foo'), ('-d', 'bar')]
>>> args
['a1', 'a2']
```

长argument

```
>>> s = '--condition=foo --testing --output-file abc.def -x a1 a2'
>>> args = s.split()
>>> args
['--condition=foo', '--testing', '--output-file', 'abc.def', '-x', 'a1',
'a2']
>>> optlist, args = getopt.getopt(args, 'x', [
    'condition=', 'output-file=', 'testing'])
>>> optlist
[('--condition', 'foo'), ('--testing', ''), ('--output-file', 'abc.def'),
('-x', '')]
>>> args
['a1', 'a2']
```

一个比较接近实用的例子

```
import getopt, sys

def main():
    try:
        opts, args = getopt.getopt(sys.argv[1:], "ho:v", ["help",
"output="])
    except getopt.GetoptError as err:
        # print help information and exit:
        print(err)  # will print something like "option -a not recognized"
        usage()
        sys.exit(2)
    output = None
    verbose = False
    for o, a in opts:
        if o == "-v":
            verbose = True
        elif o in ("-h", "--help"):
            usage()
            sys.exit()
        elif o in ("-o", "--output"):
            output = a
        else:
            assert False, "unhandled option"
    # ...

if __name__ == "__main__":
    main()
```

6.3 enumerate()

enumerate(iterable, start=0)

Return an enumerate object. iterable must be a sequence, an iterator, or some other object which supports iteration. The next() method of the iterator returned by enumerate() returns a tuple containing a count (from start which defaults to 0) and the values obtained from iterating over

iterable.

```
seasons = ['Spring', 'Summer', 'Fall', 'Winter']
list(enumerate(seasons))
[(0, 'Spring'), (1, 'Summer'), (2, 'Fall'), (3, 'Winter')]
list(enumerate(seasons, start=1))
[(1, 'Spring'), (2, 'Summer'), (3, 'Fall'), (4, 'Winter')]
```

7. function

7.1 默认参数

```
def my_sum (a, b=1):
  return a+b

def my_append (new_item, a_list = None):
  if a_list == None:
    a_list = []
  a_list.append(new_item)
  return a_list
```

总结：

- 默认值为不可变值（比如上述例子中b=1为常数不变值）时，使用不会有问题。
- 普通列表项目默认值为可变值，即为对象□list□dict等，最好默认值输入为None□ 在使用的时候显示将其初始化为空对象（比如上述使用的a_list = []□□这样在不同位置调用此function函数不会产生相互影响。

8. 条件判断

8.1 if

```
if (a == b) :
  pass
elif (a == c) :
  pass
else:
  pass


if (a == b) or (e == f) :
  pass
elif (a == c) not (i == j) :
  pass
elif (a == c) and (m == n) :
  pass
else:
```

```
  pass
```

9. 循环

9.1 for

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

for i in range(1,11):
    print(i)
    continue
```

9.2 range用法

```
 # range(stop)
 # range(start, stop[, step])

>>> list(range(10))
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
>>> list(range(1, 11))
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
>>> list(range(0, 30, 5))
[0, 5, 10, 15, 20, 25]
>>> list(range(0, 10, 3))
[0, 3, 6, 9]
>>> list(range(0, -10, -1))
[0, -1, -2, -3, -4, -5, -6, -7, -8, -9]
>>> list(range(0))
[]
>>> list(range(1, 0))
[]
```

9.3 while

```
>>> a=3
>>> while a > 0:
...  a = a -1
...  print(a)
...
2
1
0
>>>
```

10. 正则表达式

```
import re

#用上r''的方式，其正则表达式的内容跟perl比较想像

#可以从中间搜索
if re.search(r'(abc|def)', "abc"):
  print(f' matched!')

#从头开始匹配搜索
if re.match(r'(abc|def)', "abc"):
  print(f' matched!')


#不匹配,结果=None
if re.match(r'a', "dbc") == None:
  print(f' not matched!')
```

10.1 group

```
>>> m = re.match(r"(\w+) (\w+)", "Isaac Newton, physicist")
>>> m.group(0)       # The entire match
'Isaac Newton'
>>> m.group(1)       # The first parenthesized subgroup.
'Isaac'
>>> m.group(2)       # The second parenthesized subgroup.
'Newton'
>>> m.group(1, 2)    # Multiple arguments give us a tuple.
('Isaac', 'Newton')
```

11. python操作json

```
import collections
import json


dict0 = collections.OrderedDict()
dict1 = collections.OrderedDict()
dict0['line0'] = 'hha0'
dict0['line1'] = 'hha1'
dict0['line2'] = 'hha2'
dict0['line3'] = dict1

dict1['a'] = 0
dict1['b'] = 1
dict1['c'] = 2
```

```
#########################################################
### conver dict to json string
s_djs = json.dumps(dict0, indent=2)
print(f's_djs = {s_djs}')

### conver json string to dict
tmp_dict = json.loads(s_djs)
for key in tmp_dict.keys():
    print(key)


#########################################################
### dict dump to json file
hdl0 = open('w0.json', 'w')
json.dump(dict0, hdl0, indent=2)
hdl0.close()

### load json file to dict
hdl1 = open('w0.json', 'r')
tmp_dict = json.load(hdl1)
hdl1.close()
print(f'tmp_dict = {tmp_dict}')
```

运行结果：

```
s_djs = {
  "line0": "hha0",
  "line1": "hha1",
  "line2": "hha2",
  "line3": {
    "a": 0,
    "b": 1,
    "c": 2
  }
}
line0
line1
line2
line3
tmp_dict = {'line0': 'hha0', 'line1': 'hha1', 'line2': 'hha2', 'line3':
{'a': 0, 'b': 1, 'c': 2}}
```

12. 格式转换或输出

12.1 String Methods

```
str.lower() -- 转小写
```

```
str.upper() -- 转大写
```

12.2 字符串格式化

12.2.1 占位符%

- https://python-reference.readthedocs.io/en/latest/docs/str/formatting.html
- https://docs.python.org/3/library/stdtypes.html#printf-style-string-formatting

```
name = "Li hua"
age = 24
print("Hello "+name+", you are " + str(age) + " years old")


name = "Li hua"
age = 24
print("Hello %s, you are %d years old" % (name, age))


####  字符串
'%5s' % 's'     # 左留空，占5个字符宽度
'%-5s' % 's'     # 右留空，占5个字符宽度


#####数值
'%5d' % 200     # 左留空，占5个字符宽度
'%-5d' % 200     # 右留空，占5个字符宽度
```

12.2.2 printf样式

```
#### printf样式□  %s, %03d□只不过是直接把变量插在%号后面
print('%(language)s has %(number)03d quote types.' %
      {'language': "Python", "number": 2})
Python has 002 quote types.
```

12.2.3 python3.6之后 f"" 方法

在python里面双引号和单引号功能是一样的。

```
print(f'名字是□{name}□年龄是□{age}')
```

12.3 print to string

```
import io
import re
```

```
output = io.StringIO()
output.write('First line.\n')
print('Second line.', file=output)

# Retrieve file contents -- this will be
# 'First line.\nSecond line.\n'
contents = output.getvalue()
contents = re.sub(r'\n', '', contents, count=0)

# Close object and discard memory buffer --
# .getvalue() will now raise an exception.
output.close()

print(contents)
```

12.4 str.format()

Accessing arguments by position:

```
>>> '{0}, {1}, {2}'.format('a', 'b', 'c')
'a, b, c'
>>> '{}, {}, {}'.format('a', 'b', 'c')  # 3.1+ only
'a, b, c'
>>> '{2}, {1}, {0}'.format('a', 'b', 'c')
'c, b, a'
>>> '{2}, {1}, {0}'.format(*'abc')      # unpacking argument sequence
'c, b, a'
>>> '{0}{1}{0}'.format('abra', 'cad')   # arguments' indices can be repeated
'abracadabra'



>>> print('{:<8} {:<20} {:<20}'.format(data0, data1, data2))
```

Accessing arguments by name:

```
>>> 'Coordinates: {latitude}, {longitude}'.format(latitude='37.24N',
longitude='-115.81W')
'Coordinates: 37.24N, -115.81W'
>>> coord = {'latitude': '37.24N', 'longitude': '-115.81W'}
>>> 'Coordinates: {latitude}, {longitude}'.format(**coord)  #意思是将dict字典
展开
'Coordinates: 37.24N, -115.81W'
```

Accessing arguments' items:

```
>>> coord = (3, 5)
>>> 'X: {0[0]};  Y: {0[1]}'.format(coord)
'X: 3;  Y: 5'
```

Aligning the text and specifying a width:

```
>>> '{:<30}'.format('left aligned')
'left aligned                  '
>>> '{:>30}'.format('right aligned')
'                 right aligned'
>>> '{:^30}'.format('centered')
'           centered           '
>>> '{:*^30}'.format('centered')  # use '*' as a fill char
'***********centered***********'
```

Replacing %x and %o and converting the value to different bases:

```
>>> # format also supports binary numbers
>>> "int: {0:d};  hex: {0:x};  oct: {0:o};  bin: {0:b}".format(42)
'int: 42;  hex: 2a;  oct: 52;  bin: 101010'
>>> # with 0x, 0o, or 0b as prefix:
>>> "int: {0:d};  hex: {0:#x};  oct: {0:#o};  bin: {0:#b}".format(42)
'int: 42;  hex: 0x2a;  oct: 0o52;  bin: 0b101010'
```

12.5 format数值格式转换

12.5.1 float保留2位小数

```
>>> i = 503
>>> j = 100
>>> format(i/j, ".2f")
'5.03'
```

12.5.2 转换为二进制数(字符串)

Convert an integer number to a binary string prefixed with “0b”.

```
>>> bin(3)
'0b11'

#如果不想要0b开头，可以使用format 'b'的方式
>>> format(14, '#b'), format(14, 'b')
('0b1110', '1110')
>>> f'{14:#b}', f'{14:b}'
('0b1110', '1110')
```