目录

第 1 章 总体规划

1.1 要学的知识清单

要学的		知识清单	进度
numpy		1.tutorialspoint numpy	finished 17.12.30-18.1.2
系统调用 os			finished 18.1.10-18.1.10
编译器调用 sys			finished 18.1.10-18.1.10
hash 算法 hashlib			finished 18.1.10-18.1.10
数学函数 math			finished 2018.2.20
装饰器 decorator			finished 2018.2.22
itertools		廖雪峰的官方网站--itertools	finished 18.1.10
Python 默认双下划线函数			finished
Pythonic		做一个 Pythonic [bar,baz,qux,qot] http://lovesoo.org/pythonic-interpretation-and-example.html http://lovesoo.org/pythonic-python-programming.html	
cmath			
pylab			不作学习
爬虫()			
泛泛而谈	JavaScript		
	json		finished
	xml		
	MATLAB		
	机器学习		ing
	HTML		
	矩阵		ing
	算法导论		
	LINUX 运维		
	Java		
	C/C++		
	Caffe		不作学习

Python 命令行参数	sys.argv		finished 18.1.24-18.1.24
	getopt		finished 18.1.24-18.1.24
	argparse	官网教程: https://docs.python.org/2/howto/argparse.html	ing-suspend 待整理
图像绘制和处理	PIL (pillow)	http://pillow.readthedocs.io/en/latest/handbook/tutorial.html	ing
	opencv		ing-suspend
	Matplotlib		ing-suspend
DL 框架	tensorflow		finished
	Keras	https://morvanzhou.github.io/tutorials/machine-learning/keras/	
pandas			ing
高级文件操作 shutil			
持久化存储对象	pickle		finished
	numpy io	https://docs.scipy.org/doc/numpy/reference/routines.io.html https://www.cnblogs.com/AllStarGIS/p/3784937.html	
	shelve		
	h5py	http://www.h5py.org/	
日志 logging			
获取系统信息 psutil			
2.x-3.x lib2to3		https://docs.python.org/3/library/2to3.html#module-lib2to3	了解
PEP 系列			
virtual environments			
描述符 descriptor		https://docs.python.org/3.5/howto/descriptor.html#descriptor-howto-guide https://docs.python.org/3/reference/datamodel.html#descriptors	
MRO		官方文档: https://www.python.org/download/releases/2.3/mro/	
scipy		https://www.tutorialspoint.com/scipy/index.htm	
scikit-learn			
skimage			

glob		http://python.jobbole.com/81552/	
Jupyter magic		http://ipython.readthedocs.io/en/stable/interactive/magics.html#built-in-magic-commands	
collections		https://docs.python.org/3/library/collections.html#module-collections	ing
statsmodels		http://www.statsmodels.org/stable/index.html	
functools			
tokenize		返回多个迭代器	
multitask		貌似已经被抛弃	
协程 greenlet		https://docs.python.org/3.5/reference/datamodel.html#coroutine-objects	
contextlib		https://www.liaoxuefeng.com/wiki/00143160895572764a6b348958f449949df42a6d3a2e542c000/0014786517706626de401ff1c0d94f379774cabd842222ff000	
python web	flask		
	django		
operator		https://docs.python.org/3.7/library/operator.html#module-operator	
pymouse			
pykeyboard			
warnings			
Python 调试			
datetime			
time			
calendar			
PySide			
协程(Coroutine)			
weakref			
tarfile		.tar 文件模块	
zipfile			
xml			

1.2 要做的后续工作

项目		
整理 argparse	https://docs.python.org/2/howto/argparse.html#argparse-tutorial	
为开源做贡献		
Python	http://python.jobbole.com/87266/	

NaN 和 None		
	pd.read_csv('iris.csv')　读取 cvs	
	pd 的 plot 操作	
不能理解	http://pandas.pydata.org/pandas-docs/stable/dsintro.html In[81]到 In[85]	
-df,dtype=bool	取反操作	
opencv 和 PIL 格式互转	http://blog.csdn.net/dcrmg/article/details/78147219 http://www.mamicode.com/info-detail-1777172.html	
bound method	https://www.cnblogs.com/feihe/archive/2011/02/01/1947062.html	
functools	http://pandas.pydata.org/pandas-docs/stable/basics.html#custom-describe	
chardet		
类变量和实例变量	https://www.cnblogs.com/20150705-yilushangyouni-Jacksu/p/6238187.html	
@property	https://www.liaoxuefeng.com/wiki/001374738125095c955c1e6d8bb493182103fac9270762a000/001386820062641f3bcc60a4b164f8d91df476445697b9e000	
一些 object	__closure__ 返回 cell object __code__ 返回 code object	
元类 metaclass	https://www.liaoxuefeng.com/wiki/001374738125095c955c1e6d8bb493182103fac9270762a000/001386820064557c69858840b4c48d2b8411bc2ea9099ba000 http://blog.jobbole.com/21351/	
函数式编程		
array	https://docs.python.org/2/library/array.html#module-array	
其他 mapping 类型	dbm:https://docs.python.org/2/library/dbm.html#module-dbm gdbm:https://docs.python.org/2/library/gdbm.html#module-gdbm bsddb:https://docs.python.org/2/library/bsddb.html#module-bsddb	
关于修改类变量的 __dict__	https://www.cnblogs.com/duanv/p/5947525.html	
easydict		

2.1 descriptors(更新中...)

1. Python 对 Descriptors 的理解[python __set__ , __get__ 等解释]
参考网址:
http://blog.csdn.net/leafage_m/article/details/54960432
http://nbviewer.jupyter.org/urls/gist.github.com/ChrisBeaumont/5758381/raw/descriptor_writeup.ipynb#Python-Descriptors-Demystified

查找属性时,如obj.attr时,Python发现这个属性attr有一个__get__方法,则返回attr中__get__方法的返回值.
只有在类的__dict__中查找属性attr时,Python才会去查询attr是否为descriptor,即查询attr中是否包含__get__.否则,Python不会理会attr是否包含__get__,直接返回attr本身.
对于类Descriptor:

```
class Descriptor(object):
    def __get__(self, obj, type=None):
        return 'get', self, obj, type
    def __set__(self, obj, val):
        print 'set', self, obj, val
    def __delete__(self, obj):
        print 'delete', self, obj

class T(object):
    d = Descriptor()

t = T()
```

self 表示描述符类的实例t.
obj 表示拥有描述符实例属性值的对象,实例访问descriptor时,obj为实例t本身.直接用类T访问descriptor,obj为None.
type 是obj的类型.实例访问descriptor时,type为t.__clas__.类T访问descriptor时,type为T
val 在设置属性时,t.d=value实际上调用d.__set__(t,value)

```
>>> t.d         #t.d，返回的实际是d.__get__(t, T)
('get', <__main__.Descriptor object at 0x00CD9450>, <__main__.T object at 0x00CD0E50>,
 <class '__main__.T'>)
>>> T.d        #T.d，返回的实际是d.__get__(None, T)，所以obj的位置为None
('get', <__main__.Descriptor object at 0x00CD9450>, None, <class '__main__.T'>)
```

```
>>> t.d = 'hello'    #在实例上对 descriptor 设置值。要注意的是，现在显示不是返回值，而是__set__方法中 print 语句输出的。
set <__main__.Descriptor object at 0x00CD9450> <__main__.T object at 0x00CD0E50> hello

>>> t.d          #可见，调用了 Python 调用了__set__方法，并没有改变 t.d 的值
('get', <__main__.Descriptor object at 0x00CD9450>, <__main__.T object at 0x00CD0E50>, <class '__main__.T'>)
>>> T.d = 'hello'    #没有调用__set__方法
>>> T.d                 #确实改变了 T.d 的值
'hello'
>>> t.d                  #t.d 的值也变了，这可以理解，按我们上面说的属性查找策略，t.d 是从 T.__dict__中得到的
                       #T.__dict__['d']的值是'hello'，t.d 当然也是'hello'
'hello'
```

当类 Descriptor 为 non-data descriptor,即类 Descriptor 中未定义__set__,执行 t.d = ‘hello’, 则直接把 t.d 值变为’hello’

```
class Descriptor(object):
    def __get__(self, obj, type=None):
            return 'get', self, obj, type
class T(object):
        d = Descriptor()
t = T()
```

执行结果如下:

```
>>> t.d
('get', <__main__.Descriptor object at 0x00CD9550>, <__main__.T object at 0x00CD9510>, <class '__main__.T'>)
>>> t.d = 'hello'
>>> t.d
'hello'
```

属性 obj.attr 查找策略(obj 可以是类或实例):
1.如果 attr 是一个 Python 预定义属性,如(__doc__,__name__等),结束.否则执行 2.
2.查找 obj.__class__.___dict__

2.2 format 的用法和 % 格式化输出

1. format 的用法

参考链接:
https://www.cnblogs.com/chunlaipiupiupiu/p/7978669.html

- 通过关键字格式化

```
print('{name} is {age} years old!'.format(name='xiaoming',age=10))
```

- 通过位置格式化

```
print('{0} is {1} years old'.format('xiaohua','21'))
```

- 填充和对齐^<>分别表示居中、左对齐、右对齐，后面带宽度

```
print('*'*20)
print('{:^20}'.format('@@'))
print('{:>20}'.format('@@'))
print('{:<14}'.format('@@'))
print('{:$<14}'.format('OO'))
print('{:%>14}'.format('UU'))
# 输出
********************
         @@
                  @@
@@
OO$$$$$$$$$$$$
%%%%%%%%%%%%UU
```

- 精度常和 f 一起使用

```
print('{:.1f}'.format(4.234324525254))
print('{:.4f}'.format(4.1))
# 输出
4.2
4.1000
```

- 进制转化，b o d x 分别表示二、八、十、十六进制

```
# 不带前缀
print('不带前缀')
print('{:b}'.format(256))
print('{:o}'.format(256))
print('{:d}'.format(256))
print('{:x}'.format(256))
# 带前缀
print('带前缀')
print('{:#b}'.format(256))
print('{:#o}'.format(256))
print('{:#d}'.format(256))
print('{:#x}'.format(256))
# 输出
```

```
不带前缀
100000000
400
256
100
带前缀
0b100000000
0o400
256
0x100
```

亦可使用:

```
>>> format(256,'#o')
0o400
>>> format(256,'o')
400
```

- 千位分隔符，这种情况只针对与数字

```
print('{:,}'.format(100000000))
print('{:,}'.format(235445.234235))
# 输出
100,000,000
235,445.234235
```

2. %--格式化输出

◆ 整数输出

%o — oct 八进制

%d — dec 十进制

%x — hex 十六进制

```
>>> print('%o' % 20)
24
>>> print('%d' % 20)
20
>>> print('%x' % 20)
14
```

加 “#” 有前缀:

```
>>> print('%#o' % 20)
0o24
>>> print('%#d' % 20)
20
>>> print('%#x' % 20)
```

```
0x14
```

◆ 浮点数输出

%f —保留小数点后面六位有效数字

%.3f —保留小数点后面三位有效数字

%e —保留小数点后面六位有效数字，科学计数法输出

%.3e —保留 3 位小数位，科学计数法输出

%g —在保证六位有效数字的前提下，使用小数方式，否则使用科学计数法

%.3g 保留 3 位有效数字，使用小数或科学计数法

APIs	作用/用法
os.path.abspath(filename)	获得 filename 绝对路径
os.path.basename(name)	返回最后的文件名(不会判断 name 文件是否存在)
os.path.dirname(dirname)	返回路径名
os.path.getsize(filename)	获得文件大小,单位字节
os.path.isdir(name)	判断 name 是否为路径
os.path.isfile(name)	判断 name 是否为文件
os.path.join(dirname,name)	连接目录与文件名
os.path.normpath(path)	规范 path 路径字符串形式
os.path.split(name)	分割文件名与路径(不会判断 name 文件是否存在)
os.path.splitext(name)	分割文件名与扩展名
os.access(path,mode)	验证路径权限
os.chdir(dirname)	改变当前路径到 dirname
os.curdir()	返回当前目录('.')
os.fdopen()	
os.getcwd()	获得当前工作路径
os.listdir(dirname)	列出 dirname 目录下的文件
os.isatty(fd)	判断如果文件描述符 fd 是打开的，同时与 tty(-like) 设备相连，则返回 true，否则 False。 http://www.runoob.com/python3/python3-os-isatty.html
os.lstat(name)	获得文件或路径的信息
os.lseek(fd, pos, how)	设置文件描述符 fd 当前位置为 pos，how 方式修改。
os.major(device)	http://www.runoob.com/python3/python3-os-major.html
os.makedev(major, minor)	http://www.runoob.com/python3/python3-os-makedev.html
os.makedirs(path,mode)	递归创建目录。像 mkdir()，但创建的所有 intermediate-level 文件夹需要包含子目录。
os.open()	http://www.runoob.com/python3/python3-os-open.html
os.popen()	
os.read(fd,n)	从文件描述符 fd 中读取最多 n 个字节，返回包含读取字节的字符串，文件描述符 fd 对应文件已达到结尾，返回一个空字符串。
os.remove(path)	删除指定路径的文件。如果指定的路径是一个目录，将抛出 OSError。
os.removedirs(path)	递归删除目录。像 rmdir()，如果子文件夹成功删除，removedirs()才尝试它们的父文件夹,直到抛出一个 error(它基本上被忽略,因为它一般意味着你文件夹不

	为空)。
os.rename(src, dst)	文件或文件夹重命名，从 src 到 dst,如果 dst 是一个存在的目录，将抛出 OSError。
os.renames(old,new)	递归重命名目录或文件。类似 rename()。
os.rmdir(path)	删除空目录，复制抛出异常
os.stat(path)	获取 path 指定的路径的信息，功能等同于 C API 中的 stat()系统调用。
os.stat_float_times([newvalue])	newvalue -- 如果 True，调用 stat() 返回 floats,如果 False，调用 stat 返回 ints。如果没有该参数返回当前设置。
os.unlink(path)	于删除文件,如果文件是一个目录则返回一个错误。
os.utime(path, times)	设置指定路径文件最后的修改和访问时间。 http://www.runoob.com/python3/python3-os-utime.html
os.write(fd, str)	写入字符串到文件描述符 fd 中. 返回实际写入的字符串长度。
os.walk(dirname)	返回一个对象，他的每个部分都是一个三元组,('目录 x'，[目录 x 下的目录 list]，目录 x 下面的文件)

3.2 sys 模块

APIs	作用/用法
sys.argv	获取当前正在执行的命令行参数的参数列表(list)。
sys.builtin_module_names	获取内建模块名称
sys.byteorder	请求本地主机字节顺序(大端 or 小端)
sys.exc_info()	获取当前正在处理的异常类,exc_type、exc_value、exc_traceback 当前处理的异常详细信息
sys.exit()	一般来说 os._exit() 用于在线程中退出

	sys.exit() 用于在主线程中退出。 os._exit()会直接将 python 程序终止，之后的所有代码都不会继续执行。 exit(0)：无错误退出 exit(1)：有错误退出
sys.hash_info	
sys.modules	返回系统导入的模块字段，key 是模块名，value 是模块
sys.maxsize	类型 Py_ssize_t 可以 take 的最大整数,(64 位机器为 2**63-1)
sys.modules.keys()	返回所有已经导入的模块名称列表
sys.modules.values()	返回所有已经导入的模块地址列表
sys.path	指定模块搜索路径的字符串列表。 在 python 启动时，sys.path 根据内建规则、PYTHONPATH 变量进行初始化。
sys.path.insert(0, module)	向 path 中插入指定模块,可以直接 import
sys.platform	获取当前平台环境
sys.stdin.read()	从标准输入读取数据

sys.stdout	print 会调用 sys.stdout 的 write 方法
	下边两行结果是一样的：
	`sys.stdout.write('hello'+'\n')`
	`print('hello')`
	sys.stdout 指向控制台，如果将文件对象的引用赋值给 sys.stdout，那么就会输出到文件。如果输出到文件之后还想在控制台输出内容，那么应该将控制台的对象引用保存下来。
	输入到文件如下：
	`import sys`
	`with open('test.txt','w') as f:`
	`  sys.stdout = f`
	`  print('hhh')`

sys.version	获取编译器版本

3.3 Numpy 模块

numpy dtype		
No.	简称	类型
1	i, i1, i2, i4	i=int32; i1,i2,i4,i8=int8,int16,int32,int64
2	b	int64
3	b1	bool
4	u1, u2, u4, u8	uint8,uint16,uint32,uint64
5	f,f2,f4,f8	float32,float16,float32,float64
6	c	S1
7	c8,c16	complex64,complex128
8	m,m8	timedelta64
9	M,M8	datetime64
10	O,O8	object
11	S,S1,S2,...	|S0, |S1, |S3, ...
12	a,a1,a2,...	|S0, |S1, |S3, ...
13	U,U1,U2,...	unicode,<U1,U2,...
14	V,V1,V2,...	unicode,<U1,U2,...
15		
Numpy 重要函数		
	ndarray.all()	
	np.angle	返回复数的角度
	ndarray.any()	
	np.argsort	返回将对数组进行排序的索引.
	ndarray.astype	类型转换
	np.cast	类型转换,np.cast['f'](np.pi)
	np.concatenate	沿现有轴链接一系列序列。
	np.imag	返回虚部
	np.iscomplex	复数判断,元素级的
	np.iscomplexobj	复数判断,参数是复数或参数中至少有一个复数返回 True
	np.isreal	实数判断,元素级的
	np.isrealobj	实数判断,参数均为实数返回 True
	np.isscalar	标量判断
	np.linalg.cholesky	Cholesky 分解
	np.linalg.det	求行列式
	np.linalg.eig	方阵的特征值和特征向量
	np.linalg.heig	Hermitian 矩阵特征值和特征向量
	np.linalg.eigvals	方阵的特征值
	np.linalg.eigvalsh	Hermitian 矩阵特征值
	np.linalg.inv	矩阵的逆
	np.linalg.lstsq	最小二乘法
	np.linalg.norm	矩阵范数
	np.linalg.pinv	矩阵的伪逆

	np.linalg.qr	QR 分解
	np.linalg.solve	求解线性方程组
	np.linalg.svd	svd
	np.linspace	linspace
	np.logspace	
	np.mgrid	创建一个多维的 mashgrid
	np.ogrid	和 np.mgrid 区别?
	np.ploy1d	
	np.real	返回实部
	np.real_if_close	虚部接近 0 时,返回实部
	np.select	根据条件返回从 choicelist 中的元素中抽取的数组。
	np.squeeze	移除一维 entries
	np.random.choice	从 1-D array 中选择任意样本
	np.unique	
	np.unwrap	
	np.where(con,x,y)	满足条件为 x, 否则为 y 若不指定 x,y,则返回对应的索引元组,元组的第一个元素为行索引向量,第二个元素为列索引向量

Numpy 子模块		
1	ploy1d	from numpy import ploy1d
2		
3		
4		
5		

tips

1.numpy 列向量乘行向量

若 x = array([1, 2, 3, 4, 5, 6]),x.T 亦为 array([1, 2, 3, 4, 5, 6]),因此若进行列向量乘行向量,可执行如下代码:

```
np.dot(x.reshape(-1,1),x[np.newaxis])
```

fromkeys	seq = ('name', 'age', 'sex') dict1 = dict1.fromkeys(seq) print(dict1) dict1 = dict1.fromkeys(seq, 10) print(dict1) {'age': None, 'name': None, 'sex': None} {'age': 10, 'name': 10, 'sex': 10}
dict.setdefault	指定的 key 不存在,则设为默认值

3.5 itertools 模块

参考网址：http://python.jobbole.com/87455/

count(start=0,step=1)	产生初始值为 start，步长为 step 的自然数序列，类似于 generator 可迭代
cycle(builtins.object)	把传入的一个序列无限重复下去
repeat(builtins.object)	把传入的元素无限重复下去

takewhile(predicate, iterable)	返回 predicate 条件为 True 的迭代器,如:
	`naturals = itertools.count(1)`
	`ns = itertools.takewhile(lambda x: x<=10, naturals)`
	`for i in ns:`
	`  print(i)`

chain(*iterables)	将多个迭代器连接起来形成更大的迭代器

groupby(iterable[, keyfunc])	将迭代器中的相邻的重复元素挑出来放在一起,见
	`for key, group in itertools.groupby('ABCAADAE'):`
	`  print (key, list(group))`
	以及
	`for key, group in itertools.groupby('AaaBBbcCAAa', lambda c: c.upper()):`
	`  print key, list(group)`

islice(*kw)	对迭代器进行分片,见 help()
accumulate	前项累加
combinations(iter,n)	iter 中 n 个不重复元素的全部组合
combinations_with_replacement(iter,n)	iter 中 n 个可重复元素的全部组合
permutations(iter,n)	iter 中 n 个元素的全部排列
compress	按照真值表筛选元素,如 `itertools.compress(range(5), (True, False, True, True, False))`
dropwhile	按照真值函数丢弃掉列表和迭代器前面的元素 `itertools.dropwhile(lambda e: e < 5, range(10))`
filterfalse	保留对应真值为 False 的元素(与 dropwhile 相反)
starmap	同内建函数 map,如: `itertools.starmap(str.islower, 'aBCDefGhI')`
tee	一个序列之上运行多个迭代器
zip_longest	同内建函数 zip,除了以长迭代器的为准

3.6 内建函数(Built-in Function)

参考网址：http://www.runoob.com/python/python-built-in-functions.html

abs()	返回绝对值
all()	判断给定的可迭代参数 iterable 中的所有元素是否不为 0、''、False 或者 iterable 为空，如果是返回 True，否则返回 False。
any()	判断给定的可迭代参数 iterable 是否全部为空对象，如果都为空、0、false，则返回 False，如果不都为空、0、false，则返回 True。
ascii()	ascii() 函数类似 repr() 函数，返回一个表示对象的字符串，但是对于字符串中的非 ASCII 字符则返回通过 repr() 函数使用 \x, \u 或 \U 编码的字符。 生成字符串类似 Python2 版本中 repr() 函数的返回值。
bin()	转换为二进制格式
bool()	返回 bool 值
bytearray	返回一个新字节数组。这个数组里的元素是可变的，并且每个元素的值范围: 0 <= x < 256。
bytes	bytes 函数返回一个新的 bytes 对象，该对象是一个 0 <= x < 256 区间内的整数不可变序列。它是 bytearray 的不可变版本。
callable	callable() 函数用于检查一个对象是否是可调用的。如果返回 True，object 仍然可能调用失败；但如果返回 False，调用对象 ojbect 绝对不会成功。 对于函数，方法，lambda 函式，类，以及实现了 __call__ 方法的类实例，它都返回 True。
chr	返回整数对应的 string 表示
classmethod()	

compile	将一个字符串编译为字节代码。如:
	`c = compile('for x in range(10):print(x)', '', 'exec')`
	`exec(c)`
	见:
	http://www.runoob.com/python/python-func-compile.html

complex()	返回复数
delattr()	删除 object 中的属性,根据实验(Python3.5)来看, object 为类时删除类属性，为实例是只能删除实例动态添加的属性.
dict()	创建字典
dir()	没有参数时，返回局部命名空间的变量,有参数时,尝试返回有效的 object 属性.如果定义__dir__(), 会__dir__的返回值.当参数是类时,不会返回 metaclass 属性.
divmod()	返回包含商和余数的元组(a // b, a % b)。
enumerate()	返回枚举对象.
eval()	用来执行一个字符串表达式，并返回表达式的值。
exec()	执行储存在字符串或文件中的 Python 语句，相比于 eval，exec 可以执行更复杂的 Python 代码。

filter	过滤掉真值函数为 False 的序列,如 filter(lambda x:x%2==0,range(30))
float()	转换为 float,参数可以是字符串
format()	value 格式化
frozenset	冻结的集合,不能添加和删除任何元素.
getattr()	返回 object 指定属性的 value,如 getattr(x,'foobar')等价于 x.foobar
globals()	以字典的形式返回当前位置的全局变量
hasattr()	判断对象是否包含对应的属性。
hash()	返回 object hash 值
help()	调用内建帮助系统
hex()	十进制转十六进制,字符串表示
id()	返回 object id
input()	程序输入
int()	转换为 int,可以转换 2 进制,8 进制,16 进制等,如 int('0x16',base=16)
isinstance()	isinstance() 与 type() 区别: type() 不会认为子类是一种父类类型，不考虑继承关系。 isinstance() 会认为子类是一种父类类型，考虑继承关系。 如果要判断两个类型是否相同推荐使用 isinstance()。
issubclass(class1, class2)	判断参数 class1 是否是类型参数 class2 的子类。
iter(obj,[,sentinel])	返回迭代器,一个有价值应用是读取文件的行. 见 https://docs.python.org/3/library/functions.html#iter
len()	返回 object 长度
list()	转换为 list
locals	更新并返回表示当前本地符号表的字典。在函数块中调用时，由 locals() 返回自由变量，但不在类块中调用。 在函数中调用 locals()返回函数中的自由变量.
map()	根据提供的函数对指定序列做映射。
max()	返回 iterable 中的最大项或两个或更多个参数中最大的项。
memoryview	返回给定参数的内存查看对象(Momory view)。 所谓内存查看对象，是指对支持缓冲区协议的数据进行包装，在不需要复制对象基础上允许 Python 代码访问。 见 http://www.runoob.com/python/python-func-memoryview.html
min()	返回 iterable 中的最小项或两个或更多个参数中最小的项。
next	调用__next__()获得 iterator 的下一个 item
object()	返回无任何特征的对象.该对象无__dict__，因此不能给 object class 的实例赋任何属性.
oct()	将整数转换为 8 进制
open()	打开文件,并返回文件对象
ord	以一个字符（长度为 1 的字符串）作为参数，返回对应的 ASCII 数值，或者 Unicode 数值，如果所给的 Unicode 字符超出了你的 Python 定义范围，则会引发一个 TypeError 的异常。是 chr()反操作.

pow(x,y[,z])	pow(100,2) #返回 int math.pow(100,2) #返回 float 是提供第三个参数会对 z 取模,速度快于 pow(x,y)%z
print	输出语句
property	返回 property 属性.
range	range(start, stop[,step])
repr	将对象转化为供解释器读取的形式。返回对象的 str 格式,函数中需定义 __repr__
reversed	返回一个反转的迭代器.
round	round(number[,ndigits])
set	set
setattr()	设置属性,如: setattr(x, 'foobar', 123) 等价于 x.foobar = 123. 见: http://www.runoob.com/python/python-func-setattr.html
slice	slice(start, stop[,step])
sorted	对所有可迭代的对象进行排序操作。见: https://docs.python.org/3/howto/sorting.html#sortinghowto http://www.runoob.com/python/python-func-sorted.html
staticmethod	将方法转换为 static method
str	转换为字符串
sum	求和,区别于 math.fsum
super	用于调用父类(超类)的一个方法。 super 是用来解决多重继承问题的,直接用类名调用父类方法在使用单继承的时候没问题,但是如果使用多继承,会涉及到查找顺序(MRO)、重复调用(钻石继承)等种种问题。 见: http://python.jobbole.com/86787/
tuple	元组
type	type() 函数如果你只有第一个参数则返回对象的类型,三个参数返回新的类型对象。
slice()	实现切片对象,主要用在切片操作函数里的参数传递。如: my_slice = slice(0,10,2) list(range(30)[my_slice]) [0, 2, 4, 6, 8]
vars	返回对象 object 的属性和属性值的字典对象。要有__dict__属性.没有参数的话,vars()类似于 locals()
zip	zip
__import__()	这个函数被 import 语句 imvoke. 用于动态加载类和函数. 如果一个模块经常变化就可以使用 __import__() 来动态载入。

3.7 collections(更新中...)

Counter	为 hashable 对象计数，是字典的子类。
Counter(a=1,b=2) Counter(dict) Counter(str)	初始化方式
clear()	清空 Counter
elements()	返回 Counter 的迭代对象,chain??
copy()	返回浅拷贝
fromkeys(*args)	同 dict.fromkeys()
get(key)	同 dict.get(key)
Items()	同 dict.items()
keys()	获得所有的键,同 dict.keys()
most_common()	返回按 values 排序后的列表
pop(key)	弹出某个 key-value,同 dict.pop(key)
popitem()	按默认顺序弹出第一个 key-value,同 dict.popitem()
update(*a,**k)	添加元素,类似于 dict.update()
subtract(*a,**k)	删除元素，如果为 0,只把相应 value 设为 0, 类似于 dict.update()
setdefault	同 dict.setdefault()
defaultdict	
deque	
OrderedDict	
namedtuple()	

3.8 Integer,Float 类型的一些方法

int.bit_length()	返回表示二进制整数所需的位数，不包括符号和前导零
int.to_bytes	返回表示整数的字节数组。

	https://docs.python.org/3.5/library/stdtypes.html#int.to_bytes
int.from_bytes	返回给定字节数组所表示的整数。int.to_bytes 反操作. https://docs.python.org/3.5/library/stdtypes.html#int.from_bytes
float.as_integer_ratio()	返回整数元组(m,n),其中 m/n 等于浮点数.
float.is_integer	小数部分是否为 0
float.hex	以十六进制字符串的形式返回浮点数的表示形式。 对于有限的浮点数，这种表示将始终包含一个前导 0x 和一个尾随 p 和指数。
classmethod float.fromhex()	返回由十六进制字符串 s 表示的 float。 字符串 s 可能有前导和尾随空白。float.hex 反操作. 参考: https://docs.python.org/3.5/library/stdtypes.html#float.fromhex

3.9 String 方法

参考网址:

https://docs.python.org/3.5/library/stdtypes.html#string-methods

str.capitalize()	首字母大写
str.casefold()	转为小写,区别于 str.lower.
str.center	返回一个新字符串,写原字符串居中,空白或一个字符填充.
str.count	返回指定范围内子字符串出现的次数.
str.encode	返回字符串对应的指定编码(默认'utf-8')的 bytes
str.endswith	如果字符串以指定的后缀结尾，则返回 True，否则返回 False。

str.expandtabs	看代码吧:
	`>>> '01\t012\t0123\t01234'.expandtabs()`
	`'01      012     0123    01234'`
	`>>> '01\t012\t0123\t01234'.expandtabs(4)`
	`'01  012 0123    01234'`

str.find	返回指定范围找到的 substring 的最小索引
str.format	格式化字符串
str.format_map(mapping)	类似于 str.format(**mapping),除了直接使用 mapping 并且不复制到 dict.(看官网吧....)
str.index	类似于 str.find,除了当未发现 substring 时,抛出错误
str.isalnum	如果字符串中的所有字符都是文字字母数字(alphanumeric)并

	且至少有一个字符，则返回 true，否则返回 false。
str.isalpha	如果字符串中的所有字符都是字母(alphabetic)并且至少有一个字符，则返回 true，否则返回 false。
str.decimal	如果字符串中的所有字符都是十进制字符(纯数字)并且至少有一个字符，则返回 true，否则返回 false。
str.isdigit	如果字符串中的所有字符都是数字并且至少有一个字符，则返回 true，否则返回 false。
str.isidentifier	根据语言定义，如果字符串是有效的标识符，则返回 true
str.islower	如果字符串中的所有字符都是小写字母，并且至少有一个小写的的字符，则返回 true，否则返回 false。
str.isnumeric	如果字符串中的所有字符都是数字字符，并且至少有一个字符，则返回 true，否则返回 false。 Numeric characters 包括 digit characters 和所有具有 Unicode numeric value 属性的 character
str.isprintable	如果字符串中的所有字符都可打印或字符串为空，则返回 true，否则返回 false。不可打印字符如'\n','\t'等
str.isspace	如果字符串中只有空格字符，并且至少有一个字符，则返回 true，否则返回 false。
str.istitle	如果字符串是一个标题字符串并且至少有一个字符，则返回 true，例如，大写字符只能跟在 uncased 字符之后，而小写字母只能在 cased 字符之后。 否则返回 false。 title case 参考网址: http://www.grammar-monster.com/lessons/capital_letters_title_case.htm
str.isupper	如果字符串中的所有 cased 字符都是大写且至少有一个 cased 字符，则返回 true，否则返回 false。
str.join	字符串拼接.
str.ljuest	以指定 width 的字符串返回左对齐(left justified)的字符串。
str.lower	返回小写字符串
str.lstrip	返回删除前导字符的字符串副本。 chars 参数不是一个前缀；相反，其值的所有组合都被 strip.
static str.maketrans	看例子吧: http://www.runoob.com/python3/python3-string-maketrans.html
str.partition	在 sep 第一次出现时拆分字符串，并返回包含分隔符之前的部分，分隔符本身和分隔符之后的部分的三元组。 如果未找到分隔符，则返回包含该字符串本身的三元组，然后返回两个空字符串。
str.replace	返回所有出现的旧字符串替换为新字符串的副本。如果给出可选的参数 count，则仅替换前 count 个事件.
str.rfind()	返回指定范围找到 substring 子字符串的最高索引.
str.rindex	类似于 str.rfind,除了未找到 substring 会跑出 ValueError
str.rjust	以指定 width 的字符串返回右对齐(right justified)的字符

	串。
str.rpartition	在 sep 最后出现时拆分字符串，其余同 str.partition.
str.rsplit	如果使用参数 maxsplit=-1，用法同 str.split，否则，split maxsplit 个从右往左的子字符串.
str.rsplit	如果使用参数 maxsplit=-1，用法同 str.rsplit，否则，split maxsplit 个从左往右的子字符串.
str.splitline	参见官网和 https://www.yiibai.com/python3/string_splitlines.html
str.startswith	如果字符串以指定 prefix 开头，则返回 True，否则返回 False。

str.strip	返回删除前导字符和尾随字符的字符串副本。其余用法同 str.lstrip
	注意：从前端删除字符，直到达到字符集中不包含的字符串。如
	`>>> comment_string = '#....... Section 3.2.1 Issue #32 .......'`
	`>>> comment_string.strip('.#! ')`
	`'Section 3.2.1 Issue #32'`

str.swapcase	返回大写字符转换为小写字符串的副本，反之亦然。 请注意，s.swapcase().swapcase() == s 不一定是正确的。
str.title	转换为 title case，
str.translate	看例子吧：http://www.runoob.com/python3/python3-string-maketrans.html
str.upper	转换为大写.注意:str.upper().isupper()或许为 False，如果 s 包含 uncased 字符或结果字符的 Unicode 类别不是“Lu”（Letter，uppercase），但例如，“Lt”(Letter，titlecase)。
str.zfill	返回左侧填充 ASCII'0'数字的字符串的副本，以生成一个长度宽度的字符串。

3.10 关于 bytes 和 bytearray 的知识汇总

参考网址：
https://docs.python.org/3/library/stdtypes.html#binary-sequence-types-bytes-bytearray-memoryview

- bytes.fromhex 和 bytes.hexs 是一对反操作
- b[0]返回 Integer,b[0:1]返回 bytes
- b’...’通常比 bytes([46,46,46])更有用,可以使用 list(b)将 bytes 转换为 list
- b.replace

bytes/bytearray 方法

count	参考 str.count
decode	参考 str.decode
endswith	参考 str.endswith
find	参考 str.find
index	参考 str.index
join	参考 str.join
maketrans	参考 str.maketrans
partition	参考 str.partition
replace	参考 str.replace
rfind	参考 str.rfind
rindex	参考 str.rindex
rpartition	参考 str.rpartition
startswith	参考 str.startwith
translate	参考 str.translate
center	参考 str.center
ljust	参考 str.ljust
lstrip	参考 str.lstrip
rjust	参考 str.rjust
rsplit	参考 str.rsplit
rstrip	参考 str.rstrip
split	参考 str.split
strip	参考 str.strip
capitalize	参考 str.capitalize
expandtabs	参考 str.expandtabs
isalnum	参考 str.isalnum
isalpha	参考 str.isalpha
isdigit	参考 str.isdigit
islower	参考 str.islower
isspace	参考 str.isspace
istitle	参考 str.istitle
isupper	参考 str.isupper
lower	参考 str.lower
splitlines	参考 str.splitlines
swapcase	参考 str.swapcase
title	参考 str.title
upper	参考 str.upper
zfill	参考 str.zfill

3.11 set 和 frozenset 的操作

set 和 frozenset 的共有操作	
len(s)	集合 set 中元素的个数
x (not) in s	x 在(不在)集合 s 中
isdisjoint	两集合是否相交
issubset	子集测试
set<=other	set 是否为 other 子集
set<other	set 是否为 other 真子集
issuperset	超级测试
set>=other	other 是否为 set 子集
set>other	other 是否为 set 真子集
union(*other)	并集
set|other|...	
intersection(*other)	交集
set&other&...	
difference	差集
set-other-...	
symmetric_difference	并集 - 交集
set^other^...	
copy	返回浅拷贝

set 和独有操作	
update(*other)	求并集并更新集合
set |= other | ...	
intersection_update	求交集并更新集合
set &= other &	
difference_update	求差集并更新集合
set -= other|...	
symmetric_difference_update	求”对等差分”,并更新集合
set^=other	
add	添加元素
remove	删除元素,如果元素不存在抛出 KeyError 错误
discard	删除元素,如果元素不存不做任何操作
pop	pop 任意元素
clear	清空 set

注 意 :union(),intersection(),difference(),symmeric_difference(),issubset 和 issupperset()的非运算符版本可接受任何 iterable 参数.相反,它们对应的操作符要求参数是 sets.这样就避免了像 set('abc')&'cbs'这样容易出错的结构,而采用更易读的 set('abc').intersection('cbs')。

- set 和 frozenset 支持 set to set 比较(如’==’).
- 混有 set 和 frozenset 的二元操作返回第一个操作数的内容.

```
'''保存为.json'''
import json
param_dict = {'train':10000, 'validation':3000, 'test':3000}
with open('param_dict.json','w') as f:
    json.dump(param_dict, f)

with open('param_dict.json','r') as f1:
    A = json.load(f1)
```

4.1 做一个 Pythonic

参考网址：http://lovesoo.org/pythonic-interpretation-and-example.html

1. 使用 List Comprehension

```
[expr for iter_var in iterable]
[expr for iter_var in iterable if cond_expr]
```

2. 使用 Generator Expression

```
(expr for iter_var in iterable)
(expr for iter_var in iterable if cond_expr)
```

3. 使用 .format 代替 %

```
name = 'Joo'
age = 25
print('%s is %s years old. '%(name, age))
print('{0} is {1}  years old. '.format(name, age))
```

4. 字符串拼接

‘ ’ + join 代替 str1 += str2 因为 join ,保证这个过程的时间复杂度为线性的，效率更高.

5. None 判断

判断一个变量是否为空的时候，应该总是用 is 或者 is not，而不要使用 =

6. 对象类型判断

对象类型的比较应该始终用 isinstance()代替直接比较类型，如使用 if isinstance(obj, int)代替 if type(obj) is type(1)

7. 字符串前后缀判断

在检查前缀或后缀时，用 startswith()和 endswith()代替对字符串进行切片，如使用 if foo.startswith('bar')替代 if foo[:3] == 'bar'

8．变量值交换

使用 a,b = b, a 代替 t=a; a=b; b=t

9． 判断 dict 中是否有 key

使用 dict.has_key(key)代替 key in dict

10．使用 zip 将 2 个有对应关系的 list 构造一个 dict

```
name=['Tom','Jim']
age=[23,40]
print(dict(zip(name,age))
```

11．使用 set 去掉 list 中重复数据

```
list(set(oldlist))
```

12． 使用 with 进行文件操作

```
with open('a.txt','r') as f:
    for line in f:
    print(line)
```

13．输出数组的 index 和值

```
list1 = ['a','b','c']
for index, value in enumerate(list):
    print(index,value)
```

14．实现 a?b:c

```
return_value = True if a else False
```

4.2 Python 特殊属性:双下划线方法(更新中...)

参考网址:

https://docs.python.org/2/reference/datamodel.html

https://docs.python.org/3.7/library/inspect.html

方法	作用
__abs__	abs()
__aenter__	协程中的,在语义上与__enter__()相似，只不过它必须返回一个

	awaitable。
__aexit__	协程中的,在语义上与__exit__()相似，只不过它必须返回一个awaitable。
__add__	+
__and__	“&” 运算
__annotations__	函数注释,见 http://www.jb51.net/article/89436.htm
__array__	
__base__	返回类遵循 MRO 规则的第一个父类
__bases__	返回所有父类
__bool__	内置函数 bool()调用来实现真值测试.实例未定义__bool__时,调用__len__,两个函数均未定义时,实例被认为 ture
__builtin__	
__builtins__	
__bytes__	由 bytes()调用,计算对象的 byte-string 表示 https://yq.aliyun.com/articles/93442 https://docs.python.org/3.5/reference/datamodel.html#object.__bytes__
__call__	emulating callable objects,常见于 function 类
__class__	__class__是类的属性的方式来获取类的类型
__closure__	闭包,见 https://segmentfault.com/a/1190000007321972
__code__	表示编译函数体的代码对象 不是很明白 Code objects: https://docs.python.org/3.5/reference/datamodel.html#the-standard-type-hierarchy
__complex__	complex()
__contains__	当使用 in，not in 对象的时候调用
__defaults__	位置参数默认值,可修改
__del__	销毁实例. del x 不能直接调用 x.__del__()，前者将 x 的引用计数减 1,后者仅当 x 的引用计数等于 0 的时候调用.
__delattr__	删除实例属性
__delete__	删除实例属性
__delitem__	调用以实现删除 self[key]
__dict__	Python 的实例有自己的__dict__，它对应的类也有自己的__dict__. 一个对象的属性查找顺序遵循首先查找实例对象自己，然后是类，接着是类的父类。 https://www.cnblogs.com/duanv/p/5947525.html https://www.jianshu.com/p/cf8450b4b537
__dir__	同 dir()

__divmod__	divmod()
__doc__	文档说明书
__enter__	输入与此对象相关的运行时上下文。 with 语句会将此方法的返回值绑定到语句的 as 子句中指定的目标(如果有).
__eq__	x==y
__exit__	退出与此对象相关的运行时上下文。 参数描述导致上下文退出的异常。 如果上下文没有异常退出，则所有三个参数都将为 None。
__file__	如果模块从文件加载,则__file__表示该文件路径
__float__	float()
__floordiv__	“//” 运算
__format__	没有看懂 https://docs.python.org/3.5/reference/datamodel.html#object.__format__
__func__	方法的底层方法
__ge__	x>=y
__get__	
__getattr__	
__getattribute__	
__getitem__	使类具有’[i]’或 slice 功能,即索引功能. http://www.jb51.net/article/87447.htm
__getnewargs__	
__globals__	定义函数模块的全局命名空间的变量. 我的理解是函数所能使用的全局变量
__gt__	x>y
__hash__	内置函数 hash()调用
__iadd__	“+=”
__iand__	“&=”
__ifloordiv__	“//=”
__ilshift__	“<<=”
__imatmul__	“@=”
__imod__	“%=”
__imul__	“*=”
__index__	operator.index()
__int__	int()
__init__	初始化,在__new__之后,返回给调用者之前
__instancecheck__	
__invert__	一元运算 “~”
__ior__	“|=”
__ipow__	“**=”
__irshift__	“>>=”
__isub__	“-=”
__iter__	通过 container 进行高效的迭代
__itruediv__	“/=”

__ixor__	“^=”
__kwdefaults__	关键字参数默认值
__le__	x<=y
__len__	len()
__length_hint__	实现 operator.length_hint(),返回 object 的估计长度.此方法纯粹是一种优化,不要求正确性.
__lshift__	“<<” 运算
__lt__	x<y
__matmul__	“@” 运算
__missing__	当 key 不在字典中时,由 dict.__getitem__调用,为 dict 子类实现 self[key] 可以理解为:dict 子类实现了__missing__,当调用子类 self[key]时,若 key 不存在,则会调用__missing__
__mod__	“%” 运算
__module__	返回函数所在的模块(module)
__mul__	*, 乘法
__name__	类名称
__ne__	x!=y
__neg__	一元运算 “-”
__new__	创建实例
__prepare__	
__pos__	一元运算 “+”
__pow__	pow(),”**” 运算
__qualname__	
__radd__	
__rand__	
__rdivmod__	
__reduce__	
__reduce_ex__	
__repr__	对象的’官方’字符串表示,常用于 debug print 打印__str__的输出 pprint 打印__repr__的输出
__reversed__	由 reversed 调用
__rfloordiv__	
__rlshift__	
__rmatmul__	
__rmod__	
__rmul__	
__ror__	
__round__	round()
__rpow__	
__rrshift__	
__rshift__	“>>” 运算

__rsub__	
__rtruediv__	
__rxor__	
__self__	实例方法调用,表示实例本身
__set__	
__setattr__	
__setitem__	调用以实现为 self[key]赋值
__sizeof__()	实例调用,占用空间大小
__slots__	限制 class 实例添加变量,只对当前类有效,对子类无效. 见廖雪峰网站 __slots__ 更多详情见官网 __slots__
__str__	对象的'非正式'和可打印字符串形式,返回值必须是一个字符串表示. print 打印__str__的输出 pprint 打印__repr__的输出
__sub__	-
__subclasshook__	
__truediv__	"/"运算
__or__	"|" 运算
__xor__	"^" 运算

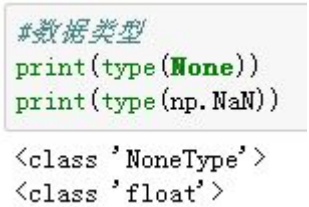
4.3 Python Options and arguments(更新中...)

选项和参数	作用/用法
-b	
-B	
-c cmd	
-d	
-E	
-h	
-i	
-I	
-m mod	
-O	
-OO	
-q	
-s	
-S	
-u	

-v	
-V	
-W arg	
-x	
-X opt	
file	
-	
arg...	
PYTHONSTARTUP	
PYTHONPATH	
PYTHONHOME	
PYTHONCASEOK	
PYTHONIOENCODING	
PYTHONFAULTHANDLER	
PYTHONHASHSEED	

4.4 None VS np.NaN

参考链接：http://python.jobbole.com/87266/

<table>
<tr><td>1.数据类型不一致</td><td> 
</td></tr>
<tr><td>2.均可作为 dict 的 key 和 value</td><td></td></tr>
<tr><td rowspan="2">3.pandas 中的 None</td><td>3.1 将 None 视为 np.NaN
(1)当数据中有数字时
(2)</td></tr>
<tr><td>3.2 None 不变
(1) 数据含有 str, bool
(2)数据均为 None</td></tr>
<tr><td>4.等值判断</td><td>http://python.jobbole.com/87266/</td></tr>
</table>

4.5 Jupyter 魔法命令

No.	魔法命令	作用
	!	执行 cmd 命令
	%%!或%%cmd	执行 cmd 语法
	%%bash	执行 bash 语法
	%env	
	%%HTML 或%%html	执行 html 语法
	%%js 或%%javascript	执行 js 语法
	%%latex	执行 latex 语法
	%ls dir	列出 dir 路径下的文件
	%matplotlib inline	在 notebok 里画图
	%magic	magic 函数帮助
	%%perl	执行 perl 语法
	%pycat xxx.py	把外部文件语法高亮显示（以弹出窗方式）
	%%time	计算 cell 的运行时间
	%%timeit	使用了 Python 的 timeit 模块，该模块运行某语句 100，000 次（默认值），然后提供最快的 3 次的平均值作为结果。
	%%writefile xxx.py	将 cell 保存为 xxx.py 文件
	%pdb	调试
	%prun fun()	给你一个按顺序排列的表格，显示每个内部函数的耗时情况，每次调用函数的耗时情况，以及累计耗时。
	%%python %%python2 %%python3	选择相应的 Python 编译器
	%%ruby	执行 ruby 语法
	%run	可以运行.py 格式的 python 代码,也可以运行其它的 jupyter notebook 文件.
	%store	可以在两个 notebook 文件之间传递变量。
	%%SVG	执行 SVG 语法
	%who	列出所有的全局变量。加上参数 str 将只列出字符串型的全局变量。

	%	
	%	
	%	
	%	
	%	
	%	
	%	
	%	

4.6 __getattribute__, __getattr__, __get__的区别

参考网址:http://luozhaoyu.iteye.com/blog/1506426

__getattribute__:

通过实例访问属性(包括类属性和实例属性)时,无条件被调用,但类访问类属性时不调用.

- 调用存在的属性:

```
'''定义类'''
class D:
    a= 'abc'
    def __init__(self):
        self.b = 2
    def __getattribute__(self, *args, **kwargs):
        print("__getattribute__() is called")
        return object.__getattribute__(self, *args, **kwargs)
    def __getattr__(self, name):
        print('__getattr__ is called')
        print(self)
        print(name)
```

实例调用类属性:

```
d = D()
d.a
```

```
__getattribute__() is called
'abc'
```

实例调用实例属性:

```
d = D()
d.b
```

```
__getattribute__() is called

2
```

均调用了__getattribute__.

- 当调用不存在的属性时

```
print(d.bb)
```

```
__getattribute__() is called
__getattr__ is called
<__main__.D object at 0x0000001EC6088D68>
bb
None
```

可以看出，先调用了__getattribute__,然后在 object.__getattribute__中没有发现属性,就调用__getattr__,最后在__getattribute__返回.

4.7 对类和实例__dict__的理解

参考网址 1:https://www.cnblogs.com/duanv/p/5947525.html

Python 的类和对应的实例都有自己的__dict__.
类的__dict__的 type 为 mappingproxy,其 dir()为 dict 子集
实例的__dict__的 type 为 dict

一个对象的属性查找顺序遵循首先查找实例对象自己，然后是类，接着是类的父类。

如,对于

```
class A:
    #类变量
    cls1 = 1
    def __init__(self):
        #实例变量
        self.ins1 = 2

a1 = A()
a2 = A()
```

初始情况下
A 的__dict__为:

```
mappingproxy({'__dict__': <attribute '__dict__' of 'A' objects>,
```

```
            '__doc__': None,
            '__init__': <function __main__.A.__init__>,
            '__module__': '__main__',
            '__weakref__': <attribute '__weakref__' of 'A' objects>,
            'cls1': 1})
```

a1 和 a2 的__dict__为:

```
{'ins1': 2}
```

也就是说实例的__dict__中并不包含类变量.根据属性查找顺序,当输出的 a1.cls1 或 a2.cls1 值为 1,即为初始值.

```
In [69]: print(A.cls1)
         print(a1.cls1)
         print(a2.cls1)

1
1
1
```

当修改 A.cls1 的值时,根据属性查找顺序,因为 a1 和 a2 中没有'cls1'属性,所以 a1.cls1 和 a2.cls1 均等于 A.cls1.

```
In [70]: A.cls1 = 3
         print(A.cls1)
         print(a1.cls1)
         print(a2.cls1)

3
3
3
```

执行如下两个 cell:

```
In [72]: a1.cls1 = 2
         print(A.cls1)
         print(a1.cls1)
         print(a2.cls1)

3
2
3
```

```
In [73]: print(a1.__dict__)
         print(a2.__dict__)

{'ins1': 2, 'cls1': 2}
{'ins1': 2}
```

可以发现 a1.__dict__新增了'cls1'属性,并遵循属性查找顺序.

4.8 Numpy 中的 math 函数和 math 库对比

numpy 直接使用 np.math 调用 math 和 import math 效果是一样的

math	numpy	作用
数论与表示函数		
math.**ceil**	np.**ceil**	向上取整.
math.**copysign**(x,y)	np.copysign(x,y)	x 与 y 保持同符号. 对于 np, x,y 是 array-like 类型,会求 x 和 y 相对应的值.
math.**fabs**	np.abs	求绝对值. 作用同内建函数 abs
math.**factorial**	np.prod(a,axis)	求阶乘. np.prod 求 a 得按元素的乘积,用 np.prod(range(1,n+1)) 求 n 阶乘
math.**fmod**	np.mod	求余 对于 np, x,y 是 array-like 类型,会求 x 和 y 相对应的值. 作用类 % 内建函数 divmod
math.**frexp**(x)		将浮点数 x 分解为尾数 ret 和指数 exp:

		x = ret*2^exp
math.**ldexp**(x, i)		返回 x * (2**i). 本质上与 math.frexp 相反
math.**fsum**(iterable)	np.sum	求和 前者为一维求和; 后者可以多维,可以指定 axis 内奸函数 sum
math.**isinf**(x)	np.isfinite	判断 x 是否为无穷数
math.**isnan**(x)	np.isnan	判断 x 是否非空 math.nan 和 np.nan 均为 float
math.**modf**(x)	np.modf	返回 x 的小数部分和整数部分,保留符号.
math.**trunc**(x)	np.trunc	返回整数部分
幂函数与对数函数		
math.**exp**(x)	np.exp	返回 e**x
math.**expm1**(x)	np.expm1	返回 e**x-1, 比 math.exp 精度高
math.**log**(x[,base])	np.log	求对数,默认自然对数
math.**log10**(x)	np.log10	求以 10 为底数的对数
math.**log1p**(x)	math.log1p	计算 log(1+x)
math.**pow**(x, y)	np.power	计算阶乘 x**y 内建函数 pow
math.**sqrt**(x)	np.sqrt	求平方根
三角函数,角度转换和双曲函数		
math.acos	np.arccos	反余弦
math.asin	np.arcsin	反正弦
math.atan	np.arctan	反正切
math.atan2(x,y)	np.arctan2	返回 atan(x/y)
math.cos	np.cos	余弦
math.sin	np.sin	正弦
math.tan	np.tan	正切
math.hypot	np.hypot	计算欧几里得范数
math.degrees	np.degrees	弧度-度
math.radians	np.radians	度-弧度
math.cosh	np.cosh	双曲余弦
math.sinh	np.sinh	双曲正弦
math.tanh	np.tanh	双曲正切
math.acosh	np.arccosh	反双曲余弦
math.asinh	np.arcsinh	反双曲正弦
math.atanh	np.arctanh	反双曲正切
特殊函数和常值		
math.erf		误差函数 没咋懂 https://www.programcreek.com

		/python/example/58378/math.erf
math.erfc		误差补函数
math.gamma		伽马函数
math.lgamma		伽马函数自然对数
math.pi	np.pi	π
math.e	np.e	自然对数底数

4.9 numpy.r_用法(更新中...)

参考网址:

https://docs.scipy.org/doc/numpy/reference/generated/numpy.r_.html