bool 0,空集,None为False,其余为True

.isnull()

x if 0>1 else y # => y (三目运算)

immutable: int, float, ~~char~~ str, tuple, bool
mutable: list, dict, set (可变!)

赋值时会改变,需要

变量都是 immutable.

list

li=[]

li.append(x) 加入x   .pop() 删掉最后一个

li[首:尾:步长]   li.remove(x) 删掉第一个出现的x,若无则报错

li.insert(index, x) 在 index 处加入 x   li.sort()

li + other_li (不变)   li.append(other_li)   li 变化↓

li.sorted([, ,], key= lambda x: x[1])

Tuple: tup(1,2,3)   可加
tup[0] # => 1

~~可加~~ a, b, c = (1, 2, 3) (可以去掉括号)
a, b = b, a   交换

dictionary

dict = {"a": 1, "b": 2, "c": 3} or dict = {("a", "b", "c"): [1,2,3]}

key (immutable types)   value

list[dict.keys()] : ["a", "b", "c"]

"a" in dict: True
False

直接用 dict["d"] 会报错
用 dict.get("d", 4) 会返回4

添加字典: dict["d"]=4
或 dict.update({:})

文件:
```
with open("file.txt", "w+") as file:
    file.write(json.dumps(contents))  / str(contents)
with open("file.txt", "r+") as file
    contents = json.load(file)   / file.read()
```

set

set = [1, 2, 3, 4]   不重复, ~~变量类型需一致~~

set.add(x) 加入值   set1 & set2   set1 | set2   set1 - set2

```
if ... :
    ...
elif ... :
    ...
else:
    ...
```

```
for x in [ , , ]
    print("{} is a ...".format(x))
for i in range(4)
for i, value in enumerate(animals):
    print(i, value)
```

```
while x<4:
    ...
```

```
try:
    raise IndexError("This is an index error")
except IndexError as e:
    pass
except (TypeError, NameError):
    pass
else:
    print("All good!")
finally:
    print("We can clean up resources here")
```

function

```
def add(x, y):
add(5, 6)   add(y=6, x=5)
def varargs(*args):   (任意数量变量)
    return args
def keyword_args(**kwargs)
    return kwargs   (任意数量键值)
keyword_args(a=3, b=4)
```

(lambda x, y : x**2 + y**2)(2, 1) # => 5

list(map(max, [1,2,3], [4,2,1])) # => [4, 2, 3]

[x for x in [3,4,5,6,7] if x>5] # => [6, 7]

modules

```
import math
print(math.sqrt(16))
from math import sqrt
sqrt(16)
```

"".join(["1", "2", "3"]) # => '123'

splitlines()

x.split() 切片   rstrip, lstrip

x.strip() 去除头尾空格

string: len(str) 长度   str.upper() 大写   str1+str2   5*str

str.title() 标题   str.capitalize() 首字母大写

~~str.line~~ str.center(30) 居中   .ljust()   .rjust() 靠左/右

'435'.rjust(10, '\n') # => \n\n\n\n\n\n\n435

'435'.zfill(10) # => 0000000435

str.find('abc') 找不到 -1   str.index('abc') 找不到报错   第一个的索引

str.endswith('...')   str.startswith('...')   str.replace('a', 'b')

Iterater

```
I = iter([2, 4, 6, 8, 10])
print(next(I)) : 2
print(next(I)) : 4
print(*range(10))
# => 0 1 2 3 4 ... 9
```

zip

```
L = [2, 4, 6, 8, 10]
R = [3, 6, 9, 12, 15]
list(zip(L, R, R, R, R)) : ~~[...]~~
# => [(2,3,3,3,3), (4,6,6,6,6), ...]
dict(zip(L, R)) # => {2:3, 4:6, ...}
```

map

```
square = lambda x: x**2
list(map(square, [1,2,3,4,5])) # => [1, 4, 9, 16, 25]
```

filter

```
for val in filter(is_even, range(10))
    print(val, end=' ')
```


skewed right

mean ± z SDs is at least $1-1/z^2$ (Chebyshev's Bounds) 0, 75, 88.8

正态分布: 68-95-99.7

~~Z-scores~~: SD:

population $\sigma^2 = \frac{\Sigma(x-\mu)^2}{N}$

sample $s^2 = \frac{\Sigma(x-\bar{x})^2}{n-1}$

Z-score: $z = \frac{\text{value} - \text{average}}{SD}$

负的z: 比平均小
正 大

z接近0, 接近平均
z接近±1, 接近SD.

Pandas

data = pd.read_csv("file.csv")

.columns：列名 .shape 大小 .values 值

data.groupby(["City","State"]).count()

data.loc[:, "sex_01"] = data.loc[:, "sex"].map({"male":0, "female":1})

创建df： df = pd.Dataframe({'a':[1,2,3], 'b':[4,5,6]}) 或 df = pd.Dataframe([{'a':1,'b':2,'c':3},{'a':4,'b':5,'c':6}])

改名： .rename(columns={"col1":"col2"})

.head() 取前几行

.set_index() 设置索引

data.[["col1","col2"]] 两列+索引 (Df)

data.["col"] 一列 (Series) + .to_frame()

data.[["col"]] 一列 (Df)

data[0:3] 前三行

.loc[[ ],[ ]] 选取特定行和列，: 为全部 (有单个就是Series) (也可以用election)

.iloc 列索引也可用数字

~~.values~~

sum()：各行或各列的总和

count()：各行或各列的非空值个数

value_counts()：某一列中每个值一位的出现次数 for series

unique()：去重

Selection 选择

data[(data["col1"]>=1) & (data["col2"]>=2)] (一个dataframe)

data["col1"]>=1：一个bool的Series

data['party'].isin(['repu..', 'demo..'])

max(data['col']) .idxmax() 找出最大值的index

np.mean(data['col'])

.sort_values("Count", ascending=False)

data.apply(lambda x: x**2)

增删 column:

data['density'] = data['pp']/data['area'] / = data['pp'].apply(f)

data.drop(["density"], axis=1, inplace=True)

pd.concat([df1, df2]) 合并, axis=0 默认, 纵向.

merge

df1.merge(df2) pd.merge(df1, df2, on='...')

left_on='...', right_on="..." (用于两个df列名不同)

how='inner' (交) how='outer' (并) left: 保留左边, 右边已有的

left_index=True 选取index 相当于 left_on

aggregate (agg)

df.groupby(['key']).agg([max, np.mean, np.std, lambda x: list(x)])

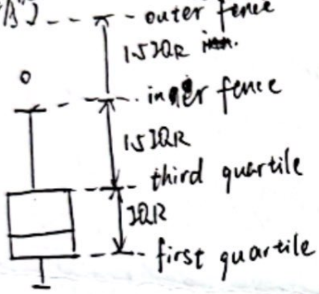
agg({data1: [...]})

.filter() 筛选

.transform() 改变 .apply()

g = df.groupby('key')["data1"].sum()

g.get_group("A")



numpy

np.array([1,2,3,4])

np.array([range(i,i+3) for i in [2,4,6]])

np.ndim() 维数 np.shape() 形状 np.size() 大小

np.arange(0,10,1) 0~9 间隔为1

np.linspace(0,10,25) 连续空间取

np.logspace(0,1,5,base=10) (默认为10)

np.random.rand(5,5) 5×5 的随机(0,1)矩阵

.randn(5,5) 正态分布

.randint(1,4) [1,3] 中

np.where(mask)

np.percentile(data,[25,50,75])

np.nan == np.nan false

np.sum() np.prod() 求积

np.reshape((a,b))

linear algebra:

A*A 数值乘

np.dot(A, A.T) 点积

np.matrix(A)

for index, row in df.iterrows():
sex = row.sex
younger = row.younger
...
df.loc[index, "col"] = ...

matplotlib

fig = plt.figure(figsize=(4,3))

axes = fig.add_axes([0,0,1,1])

axes.plot(x, y, color='r') — bar / scatter / pie / hist (bins) / boxplot

axes.set_xlabel('X')

axes.set_ylabel('Y')

axes.set_title('title')

axes.set_xticks(categories) axes.legend() 图例

axes.set_xticklabels(("Male", "Female"))

.plot(kind="bar")

.corr() correlation

sm.qqplot(data1, data2, fit=true, line='45', ax=axes)

plot fit:

slope, intercept, r_value, _, _ = stats.linregress(x, y)

line = slope * x + intercept

axes.plot(x, line, label="fit")

使用子图:

fig, axes = plt.subplots(figure=(8,4))

axes.boxplot(data1, positions=[0])

axes.set_xticks([0,1])

plt.tight_layout() 调整布局