Python B -- Course Content Summary

# Module B

## 1.1 Linux Environment Setting

### 1.1.1 wsl and Ubuntu installation

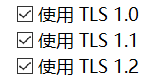
1）[Windows Subsystem for Linux Documentation | Microsoft Learn](https://learn.microsoft.com/en-us/windows/wsl/)

Note:

* In the item of Windows功能, please tick



* In the item of Internet选项，please tick



1. [Working across file systems | Microsoft Learn](https://learn.microsoft.com/en-us/windows/wsl/filesystems" \l "file-storage-and-performance-across-file-systems)

Note:

* In windows file system to check the Ubuntu file structure, you can put “\\wsl$” in windows file address.

### 1.1.2 code style

1. user@system:~$ ~ for home, \ for root.

$ tells you that you are not the root user, $variable indicates the value of the variable

斜杠(slash): “ / ”是斜杠，因为它的顶端是向前面倾斜的，有时候也叫做“forward slash”；

反斜杠(backslash)：“\”是反斜杠，注意，它的顶端是向后倾斜的。

1. Space key indicates the behind word is argument, so quote the argument with ' or " ( "My Photos" ), or escape just the relevant characters with \ ( My\ Photos ).
2. $PATH shell command searches the executed program, PATH use : to separate list of directory.
3. A path that starts with / is called an absolute path. Any other path is a relative path. Relative paths are relative to the current working directory, which we can see with the pwd command and change with the cd command. In a path, . refers to the current directory, and .. to its parent directory
4. ls, ll, ll -R

[ls命令结果解析\_sinat\_16046537的博客-CSDN博客\_ls命令识别设备文件的结果解析](https://blog.csdn.net/sinat_16046537/article/details/51803708)

1. use “which” + [command] to show the path of the command
2. -h or --help flag will print some help text echo --help
3. echo -e “1\n2” 输出转义符模式
4. 想在sh文件中执行cd 就得用source而不是bash

### 1.1.3 basic commands

* touch
* cat
* vim i insert mode esc command mode :q! ...
* cp cp -r to copy a folder -u 有同名文件较新的文件才会被更新 -p 默认保持源文件的时间戳，所有权
* rm rm -r
* mv

|  |  |
| --- | --- |
| **命令格式** | **运行结果** |
| mv source\_file(文件) dest\_file(文件) | 将源文件名 source\_file 改为目标文件名 dest\_file (文件改名) rename |
| mv source\_file(文件) dest\_directory(目录) | 将文件 source\_file 移动到目标目录 dest\_directory 中 |
| mv source\_directory(目录) dest\_directory(目录) | 目录名 dest\_directory 已存在，将 source\_directory 移动到目录名 dest\_directory 中；目录名 dest\_directory 不存在则 source\_directory 改名为目录名 dest\_directory |
| mv source\_directory(目录) dest\_file(文件) | 出错 |
| mv /usr/runoob/\* . | 将 **/usr/runoob** 下的所有文件和目录移到当前目录下 |

* rm -r flag to delete a directory and its content
* mkdir -p - p flag recursively creates all directories e.g. system wiil automatically create /home, /home/test and /home/test/demo
* man ls manual page for various argument of program

## 1.2 Shell Techniques & Tools

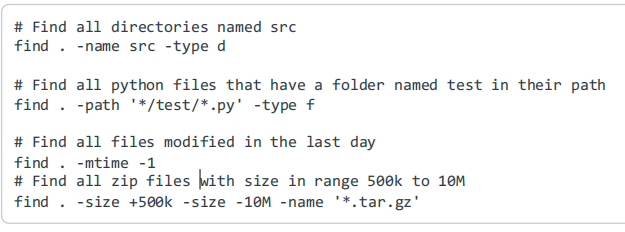
### 1.2.1 Speeding things up

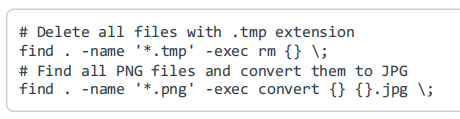
* Ctrl + C stop running a program or fresh a new command line. ^C indicating when Ctrl+C was pressed.
* In Ubuntu, ctrl+shift+v may be not available, right-click will be ok.

Ctrl+shift+c can be used to copy

* Ctrl+r the search will suggest the most recent matching command.
* Tab
* find syntax : find path -option [ -print ] [ -exec - command ] {} \;

find . -name ‘\*test.py’ -type f 记住带星号时要用’’括起来





* grep [Linux grep 命令 | 菜鸟教程 (runoob.com)](https://www.runoob.com/linux/linux-comm-grep.html)

grep -r - i firsT /home/mark/ 在mark文件夹下所有子目录寻找带有不区分大小写firsT字 符的文件及语句

* sort command arranges the lines of a file in alphabetical order
* head and tail commands can be used to output only the start/end of a file respectively

head -n 2 test.txt

* diff -i -B -w
* Redirecting Input & Output

echo < > cover and << >> append

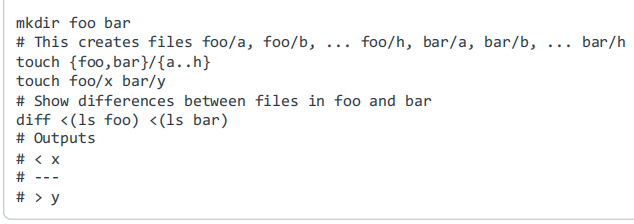
cat test.txt and cat < test.txt the result is same, but the former input stream is keyboard, the later one is the file of test.txt.

* Pipes | the output of one command is passed to the next

ls -l / | tail -n1

* globbing

wildcards ? (match one character) \*(match multiple) Curly braces {}



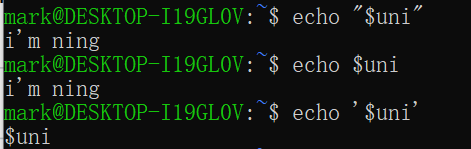
## 1.3 Basic Shell Scripting

### 1.3.1 Variables, environment variables

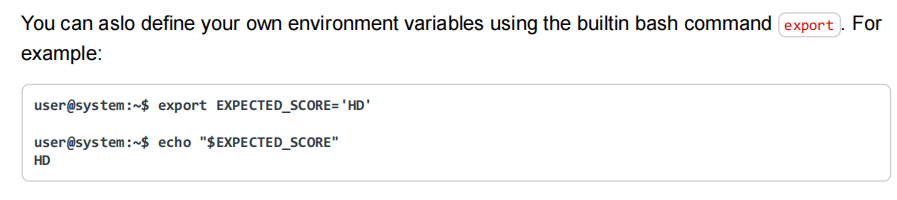
* ‘’ “” are not equivalent

ehco “$string” is different from echo $string

When you enclose a variable in double quotes, like "$string", it preserves any spaces or special characters that are in the value of the variable. So if you have a string with spaces or special characters, like "hello world", then using echo "$string" will print the string with the spaces and special characters intact.



* Self-define environment variables



* [程序员常用的符号英文 - justin.wang - 博客园 (cnblogs.com)](https://www.cnblogs.com/justin-wang/p/4238891.html)

Run a script : bash hello.bash

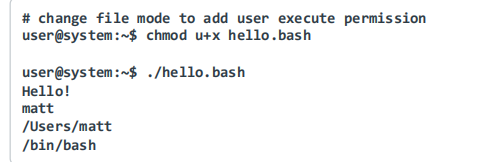
#! hash-bang shebang

The first line specifies the path to the shell script interpreter, and this specified path can only be placed on the first line of the file. If the first line is written incorrectly or not, the system will have a default interpreter to interpret it.

#!/bin/bash

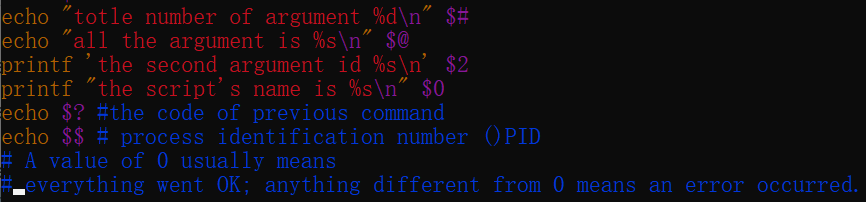
Input the path of the bash file, then the script will be executed automatically

#!/usr/local/bin/python (python interpreter)



#!/usr/bin/env python ??

* Bash parameter setting and application





## 2.1 git and github

### 2.1.1 git



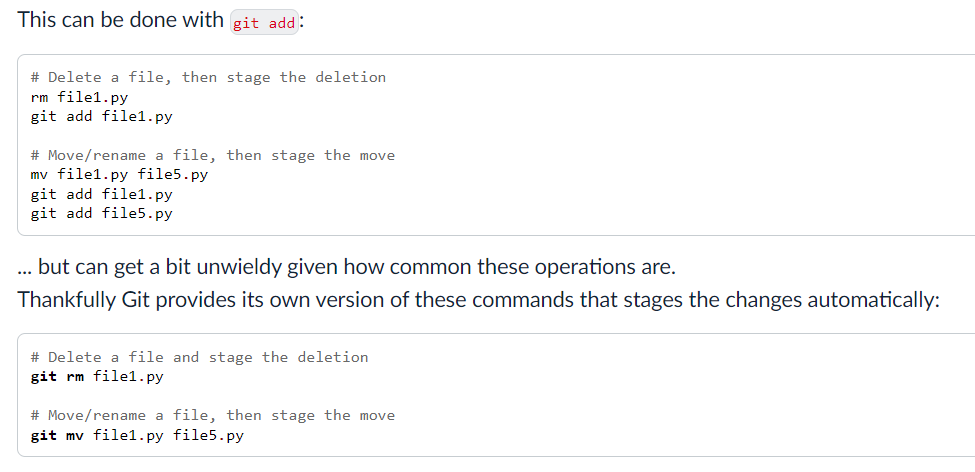
git add [file dictionary] empty folders cannot be added until they contain at least 1 file.

git commit -am "although the modified files don't be git add"

一个文件被modified但是没有add。可以直接用commit -am 提交

通过使用 commit -am 命令可以跳过 "add" 步骤，将修改的文件添加到暂存区并提交，但这仅适用于已经被Git跟踪过的文件。

如果文件没有被Git跟踪，即文件在之前没有被添加到版本库中，那么无论是使用 commit 还是 commit -am 命令都无法将其提交到版本库中。此时，你需要使用 git add 命令将文件添加到暂存区，然后使用 git commit 命令进行提交

git commit -am is used to stage and commit changes to the repository in one command

git log full edition log

git diff

git show

git reset HEAD + 文件名: 撤销add添加的文件。

the modified history of files can be reviewed easily in the github.



when we encounter the conflicts of merge operation, vim relevant file and manually

fix the conflict. Then git add [file] -> git commit -m [file], that means you must operate add can commit again to finish the merge operation.



git checkout -b new\_feature

 creating a new branch and switching to it is such a common task

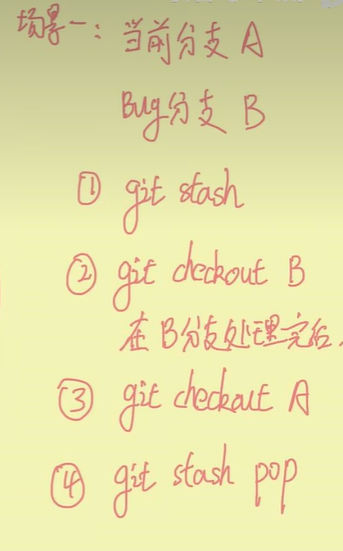
git branch -d new\_feature delete a branch

git branch -D new\_feature delete a branch compulsively (forcibly)

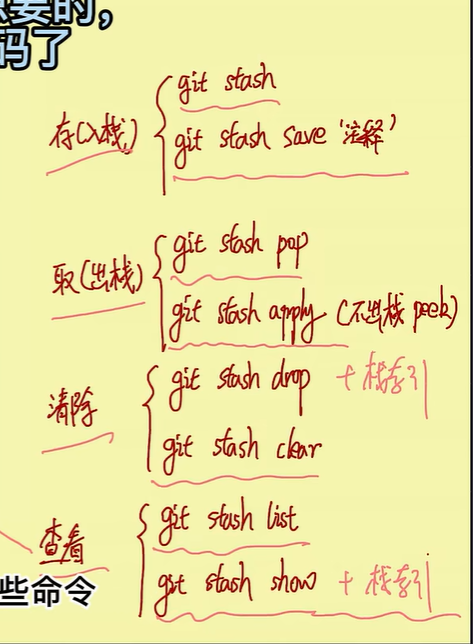
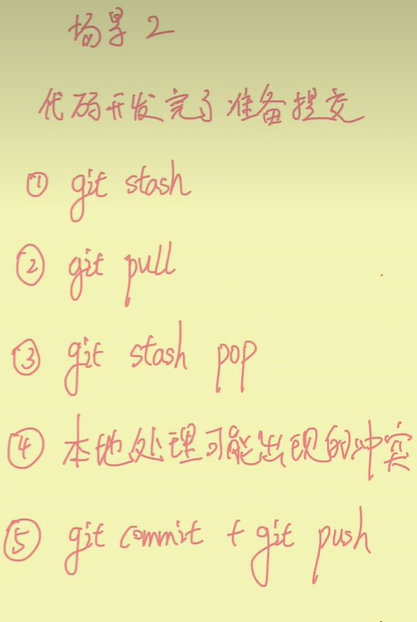
git stash

<https://www.bilibili.com/video/BV1nv4y1R78d/?spm_id_from=333.337.search-card.all.click&vd_source=126af14c03e31e589235aa086eb88819>

使用场景（1）当前branch A 上有修改过但没有commit过的文件，这是转换到branch B是不允许的，此时输入git stash，然后转换到branch B。当branch B 的文件修改完成并commit后，再切换回branch A，输入git stash pop，继续修改branch A 的文件。



使用场景（2）在local repository修改完成时，先不要commit，先从remote repository pull一下。这样的目的时为了避免在remote repository发生merge冲突，而把冲突引入到local repository发生，这样的修改conflict的代价较小，尽量不要在remote repository发生conflict，that will be terrible。但是，在你没有没有commit之前，不允许你pull，所以先用git stash再用git pull，再用git stash pop，发生冲突在本地解决。



<https://www.bilibili.com/video/BV1VG411F7rB/?spm_id_from=333.788&vd_source=126af14c03e31e589235aa086eb88819>

上面链接是讲的git merge 和git rebase的区别。

### 2.1.2 github

git classical token:

ghp\_0Xt7bEPDhKXOhHQGYjsL1qQmySXCJm02P1rT

Personal Access Token：

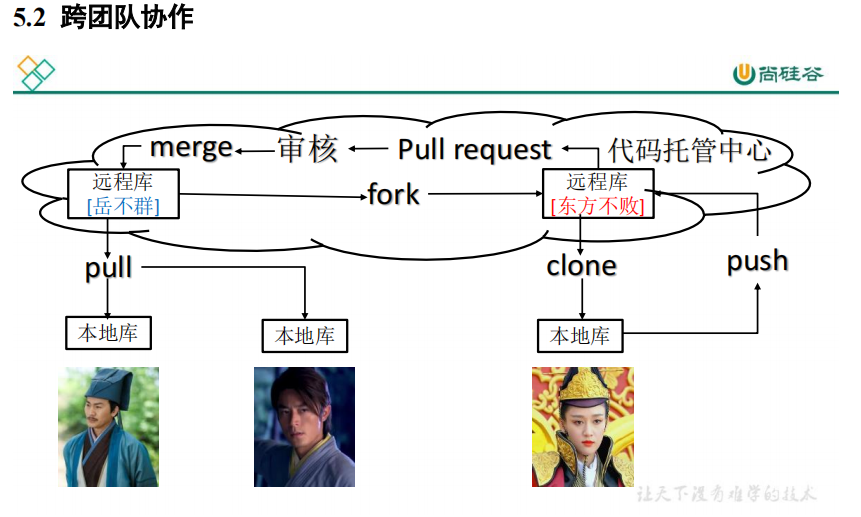
hahawang1986：

github\_pat\_11A4B7AKQ0kPsQaPEf2G2q\_jInJxZE9xqfmEJKJFPRls6nel4F0OAy9jVLPJYEMqZ1PCFJUDNTCjsM7Idm

youhistory：

github\_pat\_11AVBX3UQ0XctHvW9wAkg6\_PY6w7z3UCxDjUH0wJGxkgXhVUw0E139uKp4VQcxkIqSQOXER4Q5oUzkTEnG







fork是在github上对于别人的仓库创建了一个仓库副本到自己的github云仓库中，发生在github上

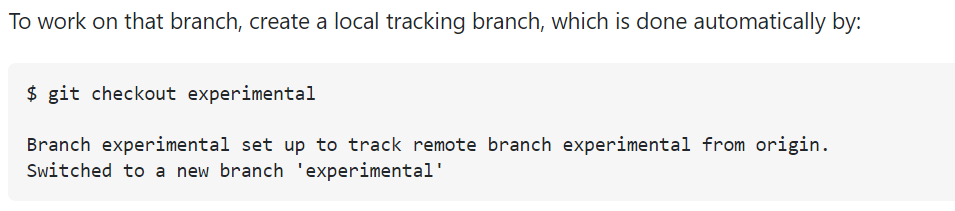
clone是将github上的项目（可以是自己的项目，可以是别人的项目，可以是自己fork别人的项目副本）在自己本地目录下创建一个本地仓库，发生于github->本地

简单来讲, fork是线上服务器的一份拷贝, clone是拷贝到本地了. 在代码合作的时候, fork部分进行修改了, 可以通过pull request提交请求, 进行合并操作. clone到本地的代码, 没有这个机制,如果是repository的创建者或者cooperator in the group, 他们能通过pull，push 去更新github或者local，其他人clone后只能在自己的local修改代码，没法跟repository实现branch操作。

clone 会做如下操作。1、拉取代码。2、初始化本地仓库。（也就是说不用自己在local输入git init 的命令）3、创建别名 default Alias： origin

clone the repository will retrieve the main branch only; you will need to take additional steps to retrieve other branches.

git branch -a or git branch -r can check all the branches in the remote repository, if we want to pull a remote branch to local, we can git checkout the branch which will track remote branch automatically.

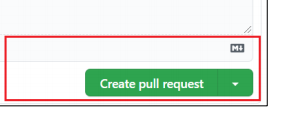


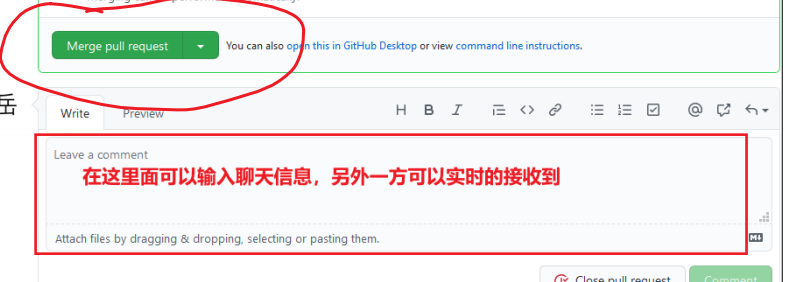
git checkout -b mybranch origin/mybranch

the git checkout command creates a new local branch named mybranch that tracks the remote mybranch branch.



fork过来后相当于自己的repository



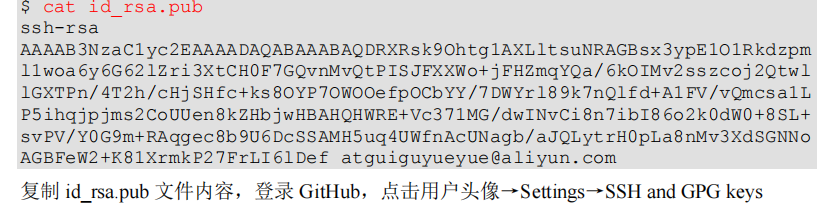




SSH免密登录







接下来再往远程仓库 push 东西的时候使用 SSH 连接就不需要登录了

## 2.2 quize

question 6

question 8

question 9

question 10

question 14

question 15

question 17

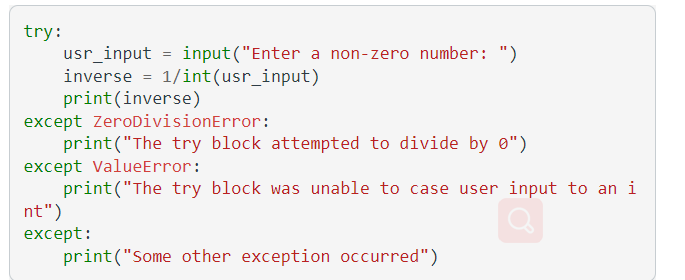
question 19

# 3 Module 3

## 3.1 debugger

* two debugger: pbd,ipbd
* simply add the lines import pdb and pdb.set\_trace() to your code. Any code above the line pdb.set\_trace() will run as normal, but all code after it will run through the debugger.
* terminal: python3 -m pdb [file name]
* from IPython.core.debugger import set\_trace

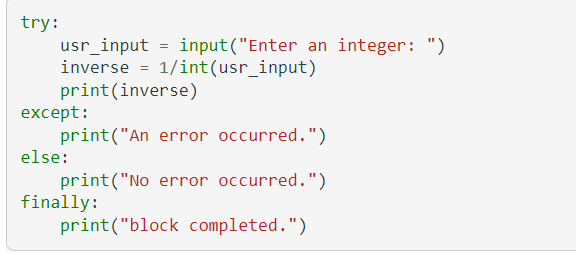
## 3.2 try-exception



a try-except-else-finally

The else block is executed only if the code in the try block completes without raising an exception

The finally block is always executed, regardless of whether an exception was raised or not.



Static analysis means the code is analysed without being run, while dynamic analysis involves analysing code while the code is running

## 3.3 pyflakes pylint

in the terminal, typing pyflakes [filename]

.ipynb file cannot be accepted by pylakes but the .py is ok

jupyter nbconvert --to scrip [filename.ipynb] download as not work

build a new ipynb and type %load filename.py can convert py to ipynb

## 3.4 try except else finally

try：

。。。。

except (ValueError, TypeError) as e:

print(“the error is [%s]”%str(e))

except:

....

else；

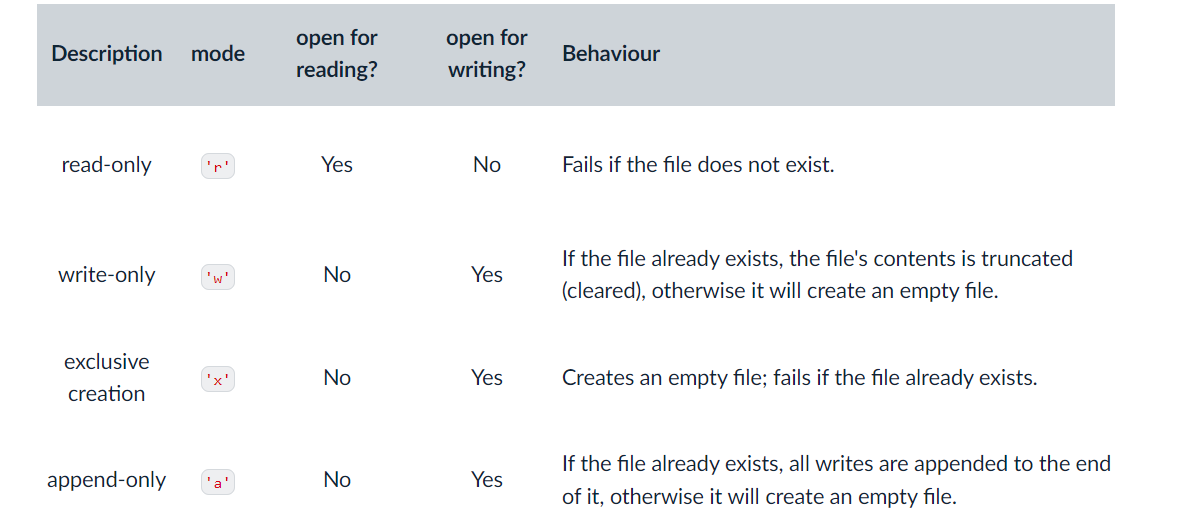
。。。

finally：

。。。

# 4 Module 4

## 4.1 file



* open('file0.txt', 'r+') 带有写功能 其他加+ 带有读功能

使用 **open('file0.txt', 'r+')** 打开一个文件时，文件指针会被放置在文件的开头，可以进行读写操作，如果进行写操作，会将新写入的内容覆盖原有内容，但不会清空整个文件。

当打开一个文件时，如果使用 'r+' 模式，文件指针会被放置在文件的开头。这样在读取文件时，从文件开头开始读取，而在写入文件时，从文件开头开始覆盖写入。

因此，如果在 'r+' 模式下写入一个较短的字符串，那么新写入的内容会覆盖原有的内容，而文件的其余部分不会被修改。如果新写入的内容比原有内容短，则新写入的内容后面部分的数据可能会保留下来，而文件的其余部分不会被修改。

需要注意的是，如果新写入的内容比原有内容长，那么原有内容后面的部分将被新内容覆盖，而原来文件尾部的数据将会丢失。

f = open('file0.txt', 'w')*# do something*

f.close() # manually close

with open('file0.txt', 'r+') as f:

*# do something*

when the block completes or an exception is thrown, automatically close

with open("carroll1.txt",'r+',encoding='utf-8') as f: 避免读取错误

file.write('This is a new line.')

* file.read(5) 前5个字符不足5个，那么将读取整个文件，并将其作为一个字符串返回,注意返回的是字符串

read() 方法是一次性将文件内容读入内存的，因此如果文件非常大，可能会导致内存不足的问题。对于大型文件，应该逐行读取或使用迭代器等方法来分批读取文件内容。read读取完返回的一个整个的string，包含newline。

file.readlines()用于将所有行读取到一个列表中。。返回一个包含文件所有行的列表对象['Hello\n', 'World\n', 'Python\n']

这些方法也可以接受参数，用于指定读取的字符数或字节数，readlines() 方法会将每一行的末尾的换行符（\n）也一并读取出来。如果不需要换行符，可以使用 strip() 方法将其从字符串中去除

carroll21 = f.readlines()[21] 读取第22行

~~carroll21 = f.readlines(21) 读取21个字节这个基本不用~~

difference between readlines() and readline()

readline() 方法并不会读取并删除文件中的内容。当你使用 readline() 方法从文件中读取一行内容时，Python 会将光标移动到下一行的开头，并返回刚刚读取的那一行内容。也就是说，文件中的内容并没有被删除，只是光标移动到了下一行的开头

* seek() 方法是用于在文件中移动光标位置的方法。它有一个参数，用于指定要移动的字节数。seek(10) 表示将光标移动到文件中的第 10 个字节处（假设文件是二进制文件，一个字节是一个字符）。如果文件是文本文件，一个字节可能不等于一个字符，因为不同的字符可能使用不同数量的字节表示。在这种情况下，可以使用 tell() 方法查看当前光标位置的字节数。

UnsupportedOperation: can't do nonzero cur-relative seeks 错误通常发生在尝试从文件的当前位置进行相对定位（即指定的偏移量不是相对于文件的开头或结尾）时，但是文件对象没有进行二进制模式打开。这是因为在文本模式下，Python 文件对象的光标只能向前移动，而不能向后移动或从当前位置开始移动。

如果您要在文本模式下使用 seek() 方法，只能将第二个参数设为 0（即从文件开头开始计算偏移量），而不能设为 1 或 2。

* **0**：从文件的开头开始计算偏移量，默认值为 **0**。
* **1**：从当前位置开始计算偏移量。
* **2**：从文件的结尾开始计算偏移量。
* f.tell()will tell you where is the pointer

在 Python 中，enumerate() 是一个内置函数，它接受一个可迭代对象（如列表、元组、字符串或文件对象）作为输入，并返回一个产生 (index, element) 对的迭代器，其中 index 是元素在可迭代对象中的索引，从 0 开始（除非指定了开始索引），element 是可迭代对象中的相应元素。

fruits = ['apple', 'banana', 'orange']

for index, fruit in enumerate(fruits, 1):

print(f'{index}: {fruit}')

output：

1: apple

2: banana

3: orange

可用来文件循环是作为条件使用。

f.readline() 在文件末尾返回空字符串时“’’”,可作为while循环的终止条件

while True:

line\_text = input\_file.readline()

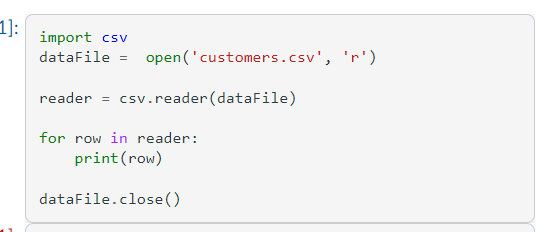
if not line\_text:

break

## 4.2 CSV

* import csv

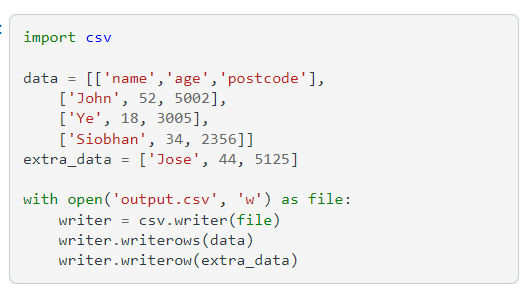
csv.reader()

* reads a row and returns the values as a list of strings
* 
* csv.DictReader(f) 读取出来时字典。key为csv的表头

['\ufeffname', 'age', 'postcode']

* Here, \ufeff has appeared before the first header field. it’s the encoding information

,to remove it open('customers.csv', 'r', encoding='utf-8-sig')

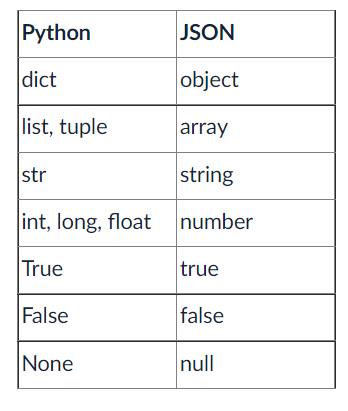
* 

## 4.3 JSON

* A JSON object is an unordered set

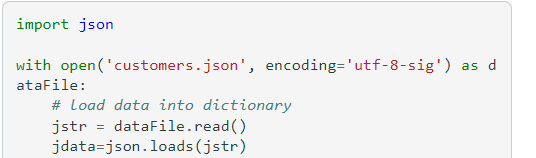
The difference of JSON and dictionary:

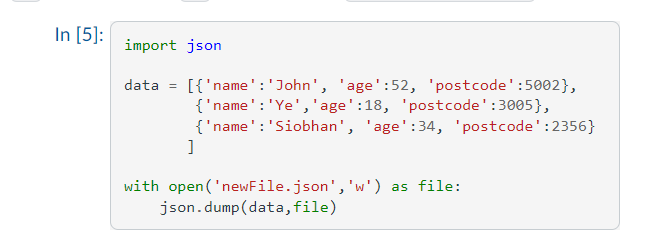
1. JSON是数据格式，字典是python的数据类型
2. JSON的键必须用“”括起来，而字典的键是不可变类型。
3. JSON的值如果是字符串必须用“”括起来



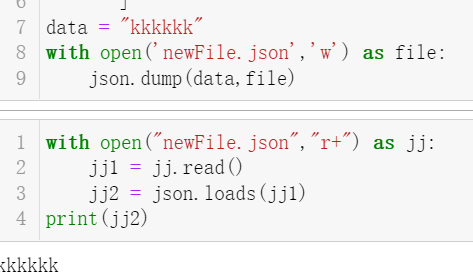
* import json

json.loads(json)





dump()的第一个值可以是list tuple dictionary string number 等等，但只有dictionary转换为JSON时能够生成key，因为dict有key值，其他类型数据用dump，会被转换成JSON的value，key为null。再使用json.loads()转换回python还是原来类型。



* binary file

with open('Tiny.class', 'rb') as f:

data = f.read() # 一次性读取整个文件

# 或者

#data = f.readline(4) # 读取文件的一行数据

size = len(data)

import os

os.path.getsize("Tiny.class") os的方法读取size

* encoding

import chardet

#先以2进制文件读进来，chardet方法检测文件编码方式，然后再以该编码方式#打开文件。

with open('utf-16.txt', 'rb') as f:

data2 = f.read()

encoding = chardet.detect(data2)['encoding']

print(encoding)

#decoded\_data = data.decode(encoding)

with open('utf-16.txt', 'r',encoding = "UTF-16") as f:

data3 = f.read()

f.seek(0)

data4 = f.readlines()[2]

print(data4)

with open("utf-16-mod.txt",'w',encoding = "UTF-16") as ww1:

ww1.write(data4)

print(data3)

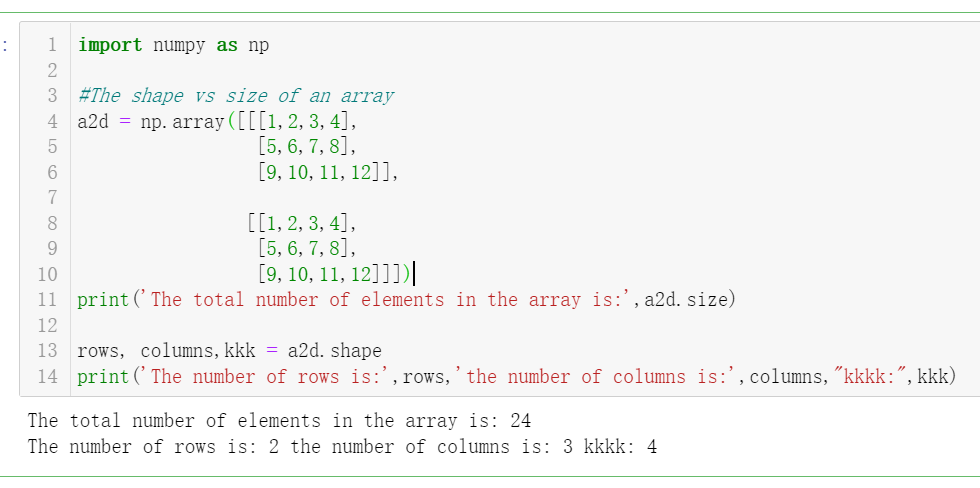
# 5 Modual Numpy and Pandas

## 5.1 Numpy

narry.len() only show the amount of rows

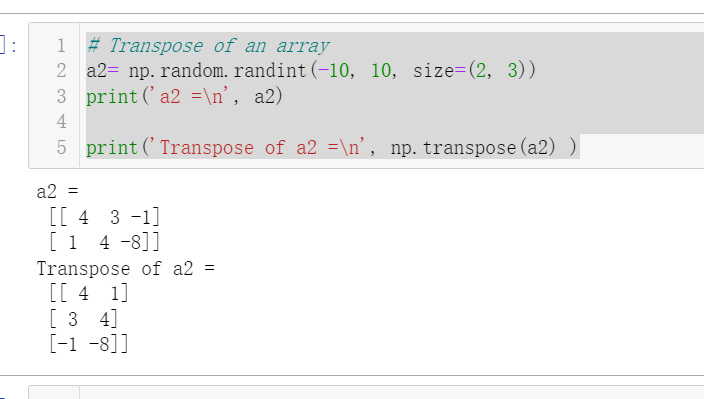
narray.size total numbers

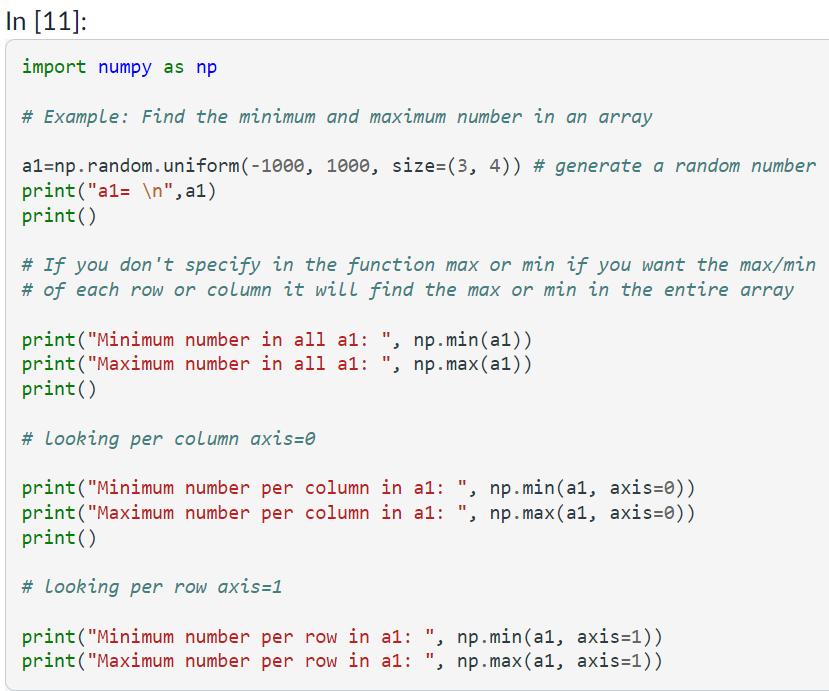
narray.shape 结果是tuple可以用拆包方式赋给其他变量值



Operator是一类特殊的函数，用于执行基本数学操作，而ufunc是一类通用函数，用于对数组执行逐元素操作

axis只是一个方向 axis =0 是垂直方向，一般为行方向，axis =1是水平方向





arange([start,] stop[, step,], dtype=None)

zeros(shape, dtype , order)

zeros dtype的default value is float

a3 = np.zeros((3), dtype=int)

a3 = np.arange(5,19,2,float)

想用arange生成2维array。先用arange生成1维array，再用reshape转化。

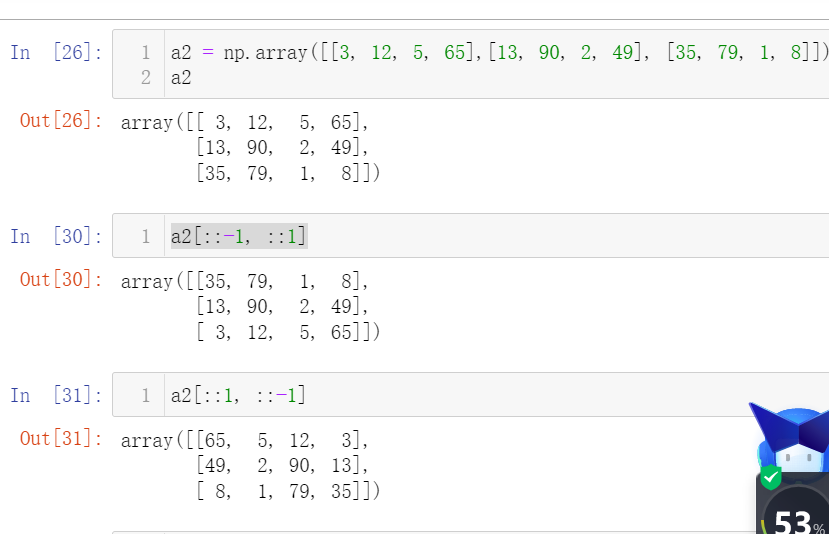
import numpy as np # 生成一维数组

a = np.arange(12) # 将一维数组转换为二维数组

b = a.reshape((3, 4))

print(b)

reshape(1,-1)将多元array 转化成一维array



np.concatenate([a1,a2],axis = 1)

按行join，必须有相同的列，按列join，必须有相同的行axis

np.vstack([a1,a2])

np.vstack([1,2,3])

np.hstack([1,2,3])

np.split(matrix,3,axis = 1)

返回的是list，里面的元素的分开的array。

np.split(matrix,[3],axis = 1)

3是指平均分3份，【3】是指split的第二个部分的开头索引是3

vec = np.arange(8)

vec = [0 1 2 3 4 5 6 7] 这只是python的显示，

实际上是产生了 8行,就像下面所示。vec.shape 结果为（8，）can prove。

vec = [0

1

2

.

.

]

v1，v2，v3=np.split(vec,[3,5])

# note [3,5] 3 indicates that the first split ends at index 3. So this first split includes indexes 0-2,

# second split starts at next index, 3.

# 5 indicates second split ends at index 5. So the second split will include indexes 3-4

v1 = [0 1 2]

v2 = [3 4]

v3 = [5 6 7]

#Split the first two rows from the rest

m1 = np.arange(24).reshape((6, 4))

mS1,mS2 = np.split(m1,[2]);

#split the array, stop at the second row from the end

mS1,mS2 = np.split(m1,[-2]);

np.vsplit(m1,2)

np.vsplit(m1,[2])

np.vsplit(m1,[-2])

np.vsplit(m1,[30]) m1一共24个值 30 的话就返回空array

NumPy arrays are mutable objects.  This can be avoided by using the copy() method or np.array(), which will give you a new array that has the same values.

a2Slice= a2[:2,:2].copy()

## 5.2 Pandas

NumPy arrays can't store mixed data in a single array or collection and can't index labels. Pandas comes in handy in such cases. pandas is built on top of NumPy and provides the performance of NumPy arrays.

### 5.2.1 pd.Series()

A pandas Series is a one-dimensional data structure that comprises of a key-value

pair.

pd.Series(data, index= index)

* data can be an array-like, iterable, dict, or scalar (ie: a single value, like int, string, etc.) value.
* index is an optional parameter, by default it is an integer sequence starting from zero.

*Note: When you assign a mutable object to a series changes to the series or mutable object will change both! so you should use*

*copy()*

data = pd.Series(np.copy(a1))

or

data = pd.Series(a1.copy())

if so ,changing either data or a1 will never affect the other

import pandas as pd

创建

# 通过单一值，list,tuple,array创建 Series

s = pd.Series(22,['a','b','c','d','e'])

s = pd.Series([1, 2, 3, 4, 5])

s = pd.Series([1, 2, 3, 4, 5],index=[‘1’,’2’,3,4,5])

s = pd.Series((1,2,3,4,5),['1','2',3,4,5])

import string

s = pd.Series(np.arange(5),list(string.ascii\_lowercase)[:5])

# 通过字典创建 Series

d = {'a': 1, 'b': 2, 'c': 3}

s = pd.Series(d)

d = {'a': 1, 'b': 2, 'c': 3}

s = pd.Series(d,index[‘a’,’c’])

如果想改变index，得先创建好series后，利用set\_index函数设置。

s.set\_index(‘A’,’c’)

查看数据

# 查看 Series 中的数据 s.values 返回array类型

# 查看 Series 中的索引 s.index 返回array类型

# 查看 Series 中的数据类型 s.dtype 返回字母o，代表object，也就是混合型

# 查看 Series 的形状 s.shape

# 查看 Series 中的描述性统计信息 s.describe()

访问

#切片和index

d = {'a': 1, 'b': 2, 'c': 3, '3': np.nan}

s = pd.Series(d)

s[['a','c','3']]选多个需要加【】

s['a':'3']

s[0:3]

When you use label-based indexing with a slice on a Pandas Series object, the end label is included in the slice. Therefore, a1 will contain all the elements between (and including) labels 'a' and 'c'.

When you use positional-based indexing with a slice on a Pandas Series object, the end index is excluded from the slice. Therefore, a2 will contain the first two elements of the Series, which are 'a' and 'b'.

# 根据条件修改元素 s[condition] = new\_value

s[[type(i) == str for i in s.index]]

# 根据位置修改元素 s.iloc[position] = new\_value

# 根据索引修改元素 s.loc[index] = new\_value

排序

# 根据索引排序

s.sort\_index()

# 根据值排序

s.sort\_values()

ascending: a boolean value, if set to True (default), the Series will be sorted in ascending order. If set to False, the Series will be sorted in descending order.

axis: a numeric value indicating the axis along which to sort. By default, axis is set to 0, which means sorting along the index. If axis is set to 1, the Series will be sorted along the columns.

na\_position: a string value indicating where the missing values should be placed. Possible values are 'last' (default) and 'first'.

ignore\_index: a boolean value, if set to True, the function will reset the index after sorting.

Series是可以运算的.

### 5.2.2 pd.DataFrame()

pd.DataFrame()基础是2D np.array。

在 DataFrame 中，每个 Series 都代表着一列数据，每个 Series 的名称都是该列的标签，而 DataFrame 的行索引用来标识每一行的数据。因此，DataFrame 可以看做是一组有序的 Series 对象。

创建

list, set,np.array创建

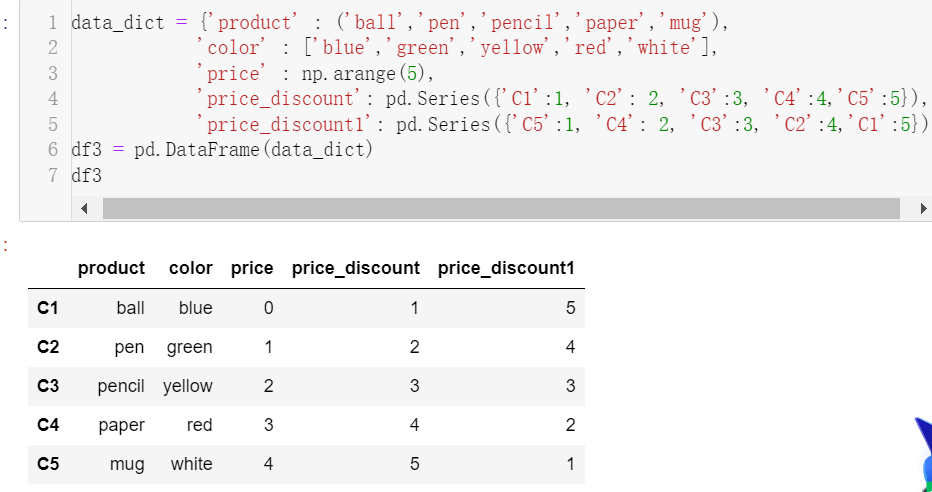
data = [['Alice', 25], ['Bob', 30], ['Charlie', 35]]

data = ({'Alice', 25}, {'Bob', 30}, ['Charlie', 35])

data = np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9]])

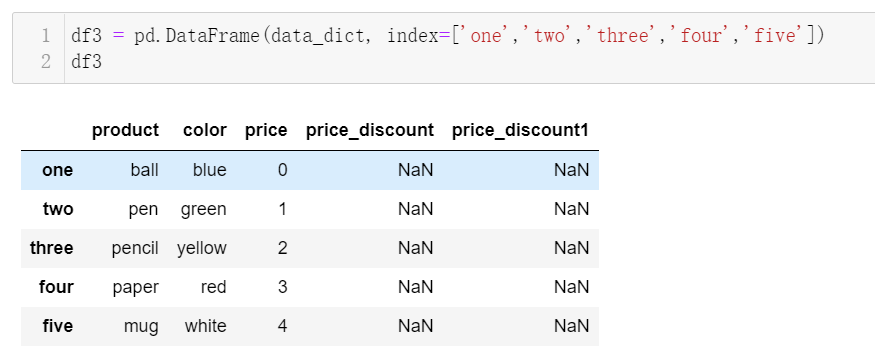
df = pd.DataFrame(data,index=['one','two','three'], columns=['Name', 'Age'])

字典创建



df3 = pd.DataFrame(data\_dict, index=['one','two','three','four','five'])

df3



Pandas allows you to explicitly define Not a Number (NaN) values and add them to a Series or a DataFrame.

if the number of index is more than the number of value, the index will be assigned by NaN

series1.isnull() （isna()同样作用）

series2.notnull()

<seriesName>.*isnull().sum()*

<seriesName>[<seriesName>.*notnull()*] (you'll get the notnull (i.e. valid) data).

这也适用于DataFrame，结果就是各个元素位置True还是False

unique()，value\_counts() isin()这些方法通常用于Series，如果向应用到Dataframe，需要从Dataframe选取相应的Series，也就是列，在进行这些函数的操作。

行与列的操作

df1 = pd.DataFrame(np.array([[6.5, 90.3], [3.6, 3.2]]), columns = ['col1','col2'])

df1 = pd.DataFrame(np.array([[6.5, 90.3], [3.6, 3.2]]), columns = ['c1','c2'], index=['row1','row2'])

查

df1.index and df1.columns #用于查询index and columns

df1.index[0] and df1.columns[1] #具体的某行某列

dataframe.dtypes #查询每列的数据类型，返回的是以列为index的series

df['prices'] = df['prices'].astype(int) #某列改变数据类型

series\_5 = series\_5.astype("float").rename("Float data") #改变series\_5的名字。

df1.values # 其用途是将 DataFrame 转换为 NumPy 数组，以便于对数据进行科学计算和数值计算

以下选取操作，如果是单列返回都是Series，多列都是dataframe，想要获得纯数据就将拣选出来的结果加上.values

直接选取某行

<DataFrame\_name>[1:2] #对行操作

直接选取某列

<DataFrame\_name>.<name\_column> e.g df.`Name of Column`

<DataFrame\_name>['<name\_column>']

直接选取某行某列

<DataFrame\_name>[1:2].’col1’

<DataFrame\_name>[1:2].[[col1,col2]]

<DataFrame\_name>[1:2][0:2]

loc和iloc选取

loc是label索引，iloc是位置索引。loc是闭区间，iloc是左闭右开

df.iloc[0, 0] # 选取第一行第一列的单元格

df.iloc[0:5, 0:5] # 选取前五行前五列的区域

df.iloc[[0, 2, 4], [0, 2, 4]] # 选取第 135 行和第 135 列的区域

互相可以混用。

df.loc[:, 'column\_name'] # 选取标签为 column\_name 的列数据

df.loc['row1':'row2', ['column\_name1', 'column\_name2']]

# 选取标签为 column\_name1 和 column\_name2 的列数据

df.loc[row1:row2, column\_label1:column\_label2]

# 选取标签在 row\_label1 和 row\_label2 之间，并且标签在 column\_label1 和 column\_label2 之间的区域数据 包含row2和column2

<DataFrame>.loc[<condition>, [<col1,..., coln>]]

loc[<condition>] 是一种基于布尔型条件进行筛选的选取方式，可以用于选取 DataFrame 中满足特定条件的行和列。具体来说，<condition> 是一个布尔型的 Series 或数组，其长度应该和 DataFrame 的行数相等，用来指示哪些行满足条件。

people["Survived"].replace(1,"True").replace(0,"False") 替换某列里的值。

改

df1.index = ['r1','r2']

df1.columns = ['c1','c2']

#直接改，但是需要保持和原来行列数一致.

df1.index[0] = ‘r1’ or df1.columns[0] = ‘c1’ are not supported

df.rename(index={"c1": "a1", "c2": "a2"}, inplace=True)

#利用字典和rename，单纯改变某些index的名字

df.rename(columns={"old\_col\_name": "new\_col\_name"}, inplace=True) #利用字典和rename，单纯改变某些columns的名字

或者用axis =0 或者1来代表index= 或者columns = ，默认是index

同理：

df.index= ["new\_index\_name" if x=="old\_index\_name" else x for x in df.index]

df.columns = ["new\_col\_name" if x=="old\_col\_name" else x for x in df.columns]

df.set\_index("c1", inplace=True, name="student\_name")

df.index.name 就能查询到为student\_name

#设置c1列为index，inplace代表替代原有index成为新的Dataframe，

name表示更改列名字为student\_name.

df.set\_index(['A', 'B'], inplace=True)设置2列为索引，多层次索引，超纲，先了解

删

#按照index-label和列名字删除

df.drop([1, 3, 5], inplace=True) 这里1，3，5是label，默认axis = 0，删除行

df.drop(['B', 'C'], inplace=True, axis=1)

#用position索引并删除

df1.drop(df1.index[0])

df1.drop(df1.columns[1], axis=1)

df1.drop(index=df1.index[0], columns=df1.columns[1])

排序

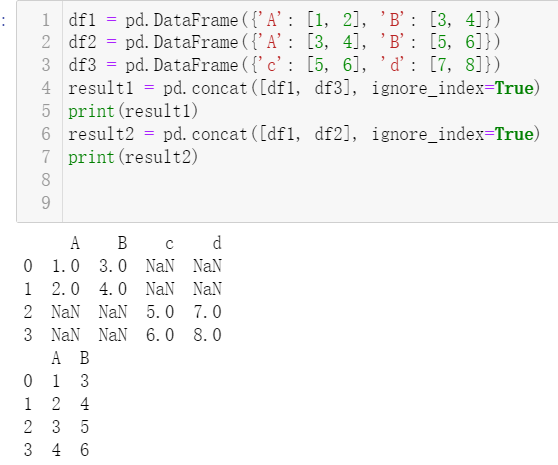
df = df.sort\_values('Salary') #按一列排序

df = df.sort\_values(['Salary', 'Age'], ascending=[True, False])

#先按Salary排，再按Age排

df = df.sort\_index()

增



# 创建一个 Pandas DataFrame

df = pd.DataFrame({'A': [1, 2, 3], 'B': [4, 5, 6]})

# 创建一个 Pandas Series

s = pd.Series([7, 8, 9])

#将 Pandas Series 添加到 Pandas DataFrame 列中

df = pd.concat([df, s], axis=1)

df.rename({0:'C'},axis =1)

data = np.genfromtxt('workshop-randoms.csv', delimiter=',', skip\_header=1)

从一个csv文件读取成np格式

df = pd.read\_csv('data.csv')

从一个csv文件读取成DataFrame格式

DataFrame 中的 groupby() 方法可以根据一个或多个列对数据进行分组，然后对每个分组应用一个函数进行聚合。下面是一个简单的示例：

import pandas as pd

# 创建一个 DataFrame

data = {'Name': ['Alice', 'Bob', 'Charlie', 'Alice', 'Bob'],

'Age': [25, 30, 35, 40, 45],

'Salary': [50000, 60000, 70000, 80000, 90000]}

df = pd.DataFrame(data)

# 按照 Name 列进行分组，并计算每个分组的平均年龄和薪水

grouped = df.groupby('Name')

result = grouped.mean()

# 按照 Name 列进行分组，并计算每个分组的年龄范围和薪水总和

# 对不同的列应用不同的聚合函数。

def age\_range(series):

return series.max() - series.min()

grouped = df.groupby('Name')

result = grouped.agg({'Age': age\_range, 'Salary': 'sum'})

# 6 Module 6 Refactoring Techniques

## 6.1 Refactoring Techniques

* Use boolean values directly

b = not a

* Use list comprehensions
* Use any() instead of a loop

has\_positives = any(n > 0 for n in numbers)

numbers = [3, 7, 8, 5, 2, 9]

if any(num % 2 == 0 for num in numbers):

print("At least one number is even")

else:

print("No numbers are even")

any() return True or False and its meaning is at least one element in a group is true or none in a group

* enumerate()

for i, player in enumerate(players):

print(i, player)

"enumerate(players)" is a built-in Python function that returns an iterator object that generates a tuple for each element in "players". Each tuple contains two values: the first value is the index of the element (starting from 0) and the second value is the element itself

* Use default values on functions where appropriate