Month01-python语言核心编程

# Day01

笔记网址：Code.tarena.com.cn

账号:tarenacode

密码:code\_2019

共享屏幕:176.234.6.5

账号:syud

密码:stud

投屏启动命令：

vncviewer

输入 ip 176.234.6.32

输入 密码 123456

1.stackoverflow.网站

2.pyc文件是python编译后的字节码文件。

ython会在执行.py文件的时候。将.py形式的程序编译成中间式文件（byte-compiled）的.pyc文件，这么做的目的就是为了加快下次执行文件的速度。

其实并不是所有的.py文件在与运行的时候都会产生.pyc文件，只有在import相应的.py文件的时候，才会生成相应的.pyc文件

字体 时间离职原因

# Day02

1.python 社区版本 （百度搜索）

2.编程思路：灵活 从左到右 、从右到左、先上下后中间。

3.linux快捷键：

Shift+Ctrl+T:新建标签页

Shift+Ctrl+W:关闭标签页

Ctrl+PageUp:前一标签页

Ctrl+PageDown:后一标签页

Shift+Ctrl+PageUp:标签页左移

Shift+Ctrl+PageDown:标签页右移

Alt+1:切换到标签页1

Alt+2:切换到标签页2

Alt+3:切换到标签页3

Shift+Ctrl+N:新建窗口

Shift+Ctrl+Q:关闭终端

终端中的复制／粘贴:

Shift+Ctrl+C:复制

Shift+Ctrl+V:粘贴

终端改变大小：

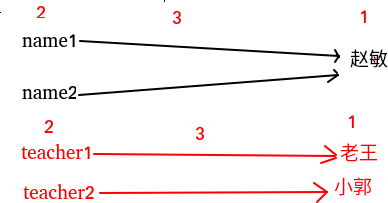
F11：全屏

Ctrl+plus:放大

Ctrl+minus:减小

Ctrl+0:原始大小

4.变量：



>>> name1 = name2 = '赵敏'

>>> id(name1)

140122147983184

>>> id(name2)

140122147983184

>>> id('赵敏')

140122148024456

>>> name1 = '小孩'

>>> id(name1)

140122148024456

>>>

>>> id('赵敏')

140122148024544

>>> id(name2)

140122147983184

5.**、*pcharm***

**查找/替换(Search/Replace)**

F3 下一个

Shift + F3 前一个

Ctrl + R 替换

Ctrl + Shift + R 全局替换

Ctrl + Shift + F 全局查找（可以在整个项目中查找某个字符串什么的，如查找某个函数名）

连续敲击两次Shift键 查找函数

# Day03

1.通过查阅网络和python的帮助文档，明确了几个函数的使用，记录如下：

ord(c):参数是长度为1的字符串，简称字符。当参数为统一对象时（unicode object），返回能代表该字符的统一编码，当参数为8比特的字符串时，返回该字节的值。例如，ord('a')返回整形数值97，ord(u'\u2020')返回8224。

chr(i)：返回一个字符，字符的ascii码等于参数中的整形数值。例如chr(97)返回字符'a'，该方法是ord()的反方法。参数必须是0-255的整形数值，否则会抛出valueError错误。

在应用函数时，观察到有个python函数比较高效，map(function,iterable,...)将函数应用于每一项iterable，并且返回一个结果的列表。map()是 Python 内置的高阶函数，它接收一个函数 f 和一个 list，并通过把函数 f 依次作用在 list 的每个元素上，得到一个新的 list 并返回。

注意：map()函数不改变原有的 list，而是返回一个新的 list。

利用map()函数，可以把一个 list 转换为另一个 list，只需要传入转换函数。

由于list包含的元素可以是任何类型，因此，map() 不仅仅可以处理只包含数值的 list，事实上它可以处理包含任意类型的 list，只要传入的函数f可以处理这种数据类型

2.等比数列

（1）定义式：



（2）[通项公式](https://baike.baidu.com/item/通项公式)（等比数列通项公式通过定义式叠乘而来）：

（3）求和公式：





（4）从等比数列的定义、通项公式、前nein项和公式可以推出：



3.

在Windows系统中，[正斜杠](https://baike.baidu.com/item/正斜杠/863308)/表示除法，用来进行整除运算；反斜杠\用来表示目录。

在Unix系统中，/表示目录；\表示跳脱字符将特殊字符变成一般字符（如enter,$,空格等）。

4.固定地址整数池（-5 - 256）

5.条件语句时，注意逻辑运算;

6.PEPB:python代码规范

7.Linux查看ascii码：man ascii

# 8月3日周六练习

1.from…import 语句

Python 的 from 语句让你从模块中导入一个指定的部分到当前命名空间中。语法如下：

from modname import name1[, name2[, ... nameN]]

例如，要导入模块 fib 的 fibonacci 函数，使用如下语句：

from fib import fibonacci

这个声明不会把整个 fib 模块导入到当前的命名空间中，它只会将 fib 里的 fibonacci 单个引入到执行这个声明的模块的全局符号表

2. 比较运算符返回值为bool类型

>>> s=6

>>> s==5 or 0 or 6

6

>>> s==(5 or 0 or 6)

False

>>> s==(5 or 0) or (6 or 9)

6

>>> s==5 or 6

6

>>> s==1

False

>>> s or 1

6

>>> s or 7

6

>>> s==9

False

>>>

3.打印方法内容

**import string**

*print(string.punctuation)*

*print(dir(string))*

**4.列表**

*# 方法1*

print (**'遍历列表方法1：'**)

**for** i **in** list:

print (**"序号：%s 值：%s"** % (list.index(i) + 1, i))

print (**'\n遍历列表方法2：'**)

*# 方法2*

**for** i **in** range(len(list)):

print (**"序号：%s 值：%s"** % (i + 1, list[i]))

5.**字典**

**import** time

dic = {**'男'**:**'张三 李四'**,**'女'**:**'貂蝉''西施'**}

*#调用字典*

print(dic[**'男'**])

*#通过get()调用字典*

print(dic.get(**'就'**,**'查无此项'**))

print(dic.get(**'男'**,**'查无此项'**))

*#显示字典所有 键 值 项*

print(dic.values())

print(dic.keys())

print(dic.items())

*#增加字典*

shiyi = input(**'请输入新加项：'**)

dic[**'不男不女'**] = shiyi

print(dic.items())

*#修改字典*

dic[**'不男不女'**] = **'小猫'**

print(dic.items())

*#删除字典*

**del** dic[**'不男不女'**]

print(dic.items())

**for** sex,name **in** dic.items():

print(sex,name)

time.sleep(1) #延迟一秒输出

6.菜鸟教程100练习到20

# Day04

1.变量定义：

如果你一开始不是赋值，而是直接用它的值就会报那个错

python 赋值的时候就是定义了， 没有赋值就不存在

2.逻辑简化：

**if** age < 0:

print(**'输入错误！'**)

**elif** age < 2:

print(**'婴儿'**)

**elif** 2 <= age < 13: #age < 13:

print(**'儿童'**)

**elif** 13 <= age < 20: #age < 20:

print(**'青年'**)

3.

### **UTF-8**

UTF-8以字节为单位对Unicode进行编码。从Unicode到UTF-8的编码方式如下：

|  |  |
| --- | --- |
| **Unicode编码(十六进制)** | **UTF-8 字节流(二进制)** |
| 000000-00007F | 0xxxxxxx |
| 000080-0007FF | 110xxxxx 10xxxxxx |
| 000800-00FFFF | 1110xxxx 10xxxxxx 10xxxxxx |
| 010000-10FFFF | 11110xxx10xxxxxx10xxxxxx10xxxxxx |

UTF-8的特点是对不同范围的字符使用不同长度的编码。对于0x00-0x7F之间的字符，UTF-8编码与[ASCII编码](https://baike.baidu.com/item/ASCII编码)完全相同。UTF-8编码的最大长度是4个字节。从上表可以看出，4字节模板有21个x，即可以容纳21位二进制数字。Unicode的最大码位0x10FFFF也只有21位。

例1：“汉”字的Unicode编码是0x6C49。0x6C49在0x0800-0xFFFF之间，使用3字节模板：1110xxxx 10xxxxxx 10xxxxxx。将0x6C49写成二进制是：0110 1100 0100 1001， 用这个比特流依次代替模板中的x，得到：11100110 10110001 10001001，即E6 B1 89。

4.几个常见字母的ASCII码大小： “A”为65；“a”为97；“0”为 48。

5.print(**"\0天大\0"**) 天大 （空字符）

# Day05

1.整除与除结果区别:

>>> 8//2

4

>>> 8/2

4.0

52 <class 'int'>

26.0 <class 'float'>

5.str1 = **' hell o '**

print(str1)

print(str1.lstrip(**' h'**))

print(str1.rstrip(**'o '**))

打印效果:

hell o

ell o

hell

创建有值的列表

list02 = list(range(4))

print(list02)

list02 = list(**"我是小妖怪"**)

print(list02)

打印效果:

[0, 1, 2, 3]

['我', '是', '小', '妖', '怪']

6.标准输入字符,输出列表流程

*#空列表 保存数据*

name\_list = []

*#循环添加数据到空列表*

**while True**:

name = input(**'请输入姓名：'**)

**if** name == **''**:

**break**

name\_list.append(name)

*#打印列表的每一个元素*

**for** name **in** name\_list:

print(name)

# Day06

1. 变量名1, 变量名2, = 数据1, 数据2 (#对应变量赋值,将数据1赋值给变量1,数据2赋值给变量2)

eg:a = 0

b = 1

a,b = b,a+b *#变量交换,对应赋值a=1,b=1*

print(a,b)

>>1 1

a = 0

b = 1

a = b *#a=1,b=1*

b = a+b *#b=2*

print(a,b)

>>1 2

变量 和 数据 都是保存在内存中的

* 在python函数 的 参数传递以及返回值都是靠引用传递的

在Python中

* 变量和数据是分开存储的
* 数据保存在内存中的一个位置
* 变量中保存着数据在内存中的地址
* 变量中记录数据的地址，就叫做 引用
* 使用id()函数可以查看变量中保存数据所在的 内存地址

注意：如果变量已经被定义，当给一个变量赋值的时候，本质上是 修改了数据的引用

* 变量不再对之前的数据引用
* 变量改为对新赋值的数据引用

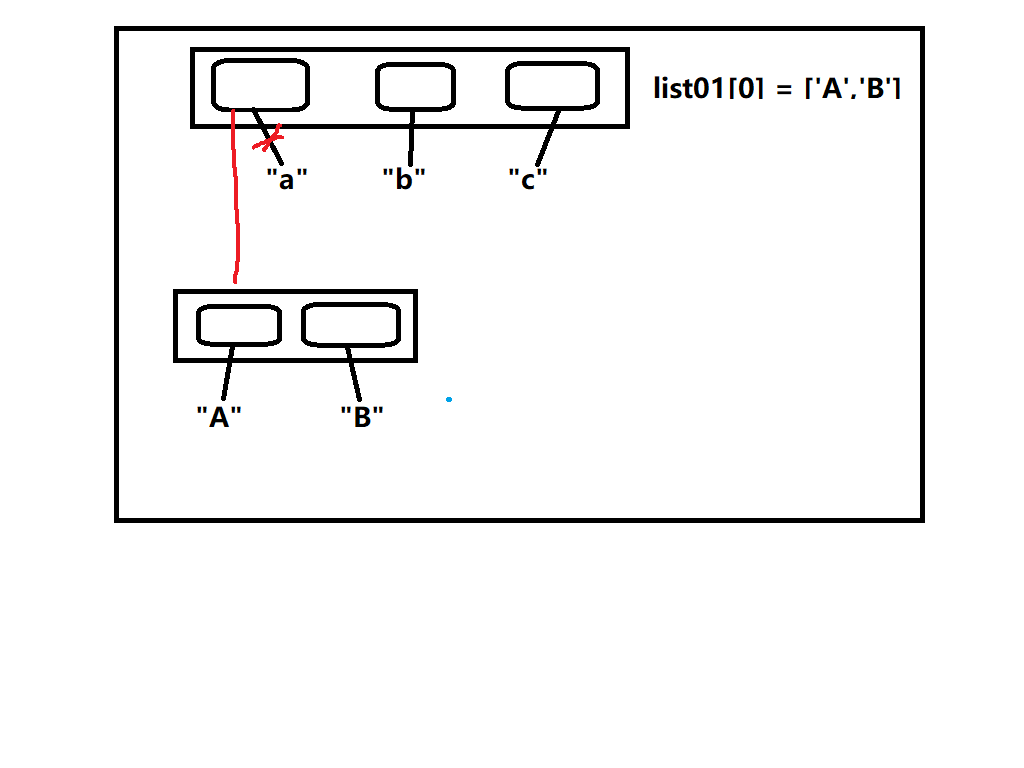
2.切片:步长是切片每次获取完当前元素后移动的偏移量

步长为:-1,表示从右向左取值.

#（print(str[:])*#复制字符串*

print(str[::-1])*#反转字符串）*

*for 索引名 in range(len(列表名)-1,-1,-1):*

***列表元素可变***

list01 = [**'a'**,**'b'**,**'c'**]

list01[0] = [**'A'**] *#索引,表示一个值*

*#将右边列表的值赋值给list01的第一个位置,*

print(**'索引赋值:'**,list01)*#索引赋值: [['A'], 'b', 'c']*

*#序列赋值*,*对应赋值*

*# name,age = '龙王',89*

list01[1:2] = [**'哪吒'**,**'敖丙'**] *#切片,表示一系列值*

print(**'序列赋值:'**,list01)*#序列赋值: [['A'], '哪吒', '敖丙', 'c']*

*#列表反向:*

list02 = list01[::-1]

print(**'反向列表:'**,list02)*#反向列表: ['c', '敖丙', '哪吒', ['A']]*

*#地址判断(变量赋值本质是引用)*

print(list01[0] **is** list02[-1])*#True*

list01[0] = **'李靖'**

print(**'切片后赋值:'**,list02)*#切片后赋值: ['c', '敖丙', '哪吒', ['A']]*

3.下午2点15左右 终端中查询tuple方法 2019.8.7周三

4.数据结构一:

dic\_1 = {**'name'**:name, **'age'**:age, **'sex'**:sex,

**'weight'**:weight}

list\_1.append(dic\_1)

print(list\_1)

**for** dict\_item **in** list\_1:

print(**'%s的年龄是%d,性别是%s,体重是%.1f'** % (dict\_item[**'name'**],

dict\_item[**'age'**],dict\_item[**'sex'**],dict\_item[**'weight'**]))

数据结构二:*{‘dsf':[18,'age',80.0...}*

dic\_1[name] = [age,sex,weight]

print(dic\_1)

**for** key,value **in** dic\_1.items():

print(**'%s的年龄是%d,性别是%s,体重是%.1f'** %

(key,value[0],value[1],value[2]))

5.tuple方法:

count(...)

| T.count(value) -> integer -- return number of occurrences of value

index(...)

| T.index(value, [start, [stop]]) -> integer -- return first index of value.

# Day07

1.*#获取字典所有元素*

*# for key in dict01:*

*# print(key)#获取的是字典中所有的键*

*# print(dict01[key])*

*#获取键值对(键，值)*

*# for item in dict01.items():*

*# print(item)*

*# for key,value in dict01.items():*

*# print(key)*

*# print(value)*

*# 获取字典的所有的值*

*# for value in dict01.values():*

*# print(value)*

2.*#外层循环走一次 内层循环走一遍*

*#外层循环控制行*

*#内层循环控制列*

**for** i **in** range(2):

**for** c **in** range(3):

*#end=' ' 不换行 每次打印结束用空格分隔*

print(**'\*'**,end=**' '**)

print()*#换行效果*

3.round*函数*

*（*1*）通过上面的函数，看着是不是很晕，感觉*round*（*x,n*）函数是否进位也没看出是啥规律*

*（*2*）*round*（*x,n*）函数中，是否进位或四舍五入，取决于*n*位以及*n+1*位小数的值*

*（*3*）只有当*n+1*位数字是*5*的时候，容易混淆，如果*n*为偶数，则*n+1*位数是*5*，则进位，例如*round*（*1.23456*，*3*）最终变为*1.235

*（*4*）如果*n*为奇数，则*n+1*位是数*5*，那不进位，例如*round*（*2.355*，*2*），最终为*2.35

*（*5*）如果*n*为*0*，即没有填写*n*的时候，最终结果与上面相反，即整数部分为偶数的时候，小数位*5*不进位，例如（*round*（*2.5*）变为*2*）。*

*（*6*）整数部分为奇数的时候，小数位*5*进位。（*round*（*3.5*）变为*4*）*

# Day08

1.append 与extend

list\_1 = [0,1,2,3,4]

list\_1.append([5,6,7])#添加元素 索引赋值

print(list\_1)

list\_1.extend([5,6,7])#扩展列表用于在列表末尾一次性追加另一个序列中的多个值 切片赋值

print(list\_1)

-->

[0, 1, 2, 3, 4, [5, 6, 7]]

[0, 1, 2, 3, 4, [5, 6, 7], 5, 6, 7]

2.在终端中查看对应容器的方法:

(1)help()函数eg:help(set)

(2)

>>> list1=[]

>>> list1. #写下list. 按两下 Tab键,自动补全方法.

list1.append( list1.count( list1.insert( list1.reverse(

list1.clear( list1.extend( list1.pop( list1.sort(

list1.copy( list1.index( list1.remove(

>>> list1.

3.*#定义一个函数 函数中包含位置形参、星号元组形参、默认形参、命名关键字形参和双星号字典形参*

*# def fun07(a,\*args,b='bbb',c,\*\*kwargs):*

*# print(a)*

*# print(args)*

*# print(b)*

*# print(c)*

*# print(kwargs)*

*#*

*# fun07(1,c='ccc')*

*# fun07(5,6,7,c='abc',d='789')*

*#定义一个函数 接受任意的参数 并输出结果*

*# def fun08(\*args,\*\*kwargs):*

*# for item in args:*

*# print(item)*

*# for k,v in kwargs.items():*

*# print(k,v)*

*#*

*# fun08()*

*# fun08(5,6,7,a=123,b='789'*

4.*定义列表升序排序函数(从小到大)*

*循环的次数 取值 第一个值 第二个值*

**def** mysort(list):

*#len(list)-1 最后一个数据 后面没有值了 不需要再比较*

**for** r **in** range(len(list)-1):

*#从当前数据的位置+1(下一个数据)开始 一直比到最后*

**for** c **in** range(r+1,len(list)):

*#如果当前数据比后面的数据大 交换位置*

**if** list[r] > list[c]:

*#a,b = b,a*

list[r],list[c] = list[c],list[r]

5.**语法**

以下是 print() 方法的语法:

print(\*objects, sep=' ', end='\n', file=sys.stdout)

### **参数**

* objects -- 复数，表示可以一次输出多个对象。输出多个对象时，需要用 , 分隔。
* sep -- 用来间隔多个对象，默认值是一个空格。
* end -- 用来设定以什么结尾。默认值是换行符 \n，我们可以换成其他字符串。
* file – 要写入的文件对象。

6.linux命令 查看代码行数

wc -l \*.py

8.10,周六练习:

1.num = input(**'请输入:'**)

print(type(num))*#input获得的数据是字符型*

2.*三个特性*

lambda*函数有如下特性：*

lambda*函数是匿名的：所谓匿名函数，通俗地说就是没有名字的函数。*lambda*函数没有名字。*

lambda*函数有输入和输出：输入是传入到参数列表*argument\_list*的值，输出是根据表达式*expression*计算得到的值。*

lambda*函数一般功能简单：单行*expression*决定了*lambda*函数不可能完成复杂的逻辑，只能完成非常简单的功能。由于其实现的功能一目了然，甚至不需要专门的名字来说明。*

*下面是一些*lambda*函数示例：*

lambda x, y: x\*y*；函数输入是*x*和*y*，输出是它们的积*x\*y

lambda:None*；函数没有输入参数，输出是*None

lambda \*args: sum(args); *输入是任意个数的参数，输出是它们的和*(*隐性要求是输入参数必须能够进行加法运算*)

lambda \*\*kwargs: 1*；输入是任意键值对参数，输出是*1

3.*# sort 与* *sorted 区别：*

*# sort 是应用在* *list 上的方法，sorted 可以对所有可迭代的对象进行排序操作。*

*# list 的* *sort 方法返回的是对已经存在的列表进行操作，无返回值，*

*# 而内建函数* *sorted 方法返回的是一个新的* *list，而不是在原来的基础上进行的操作。*

*# 语法*

*# sorted 语法：*

*# sorted(iterable, cmp=None, key=None, reverse=False)#默认由小到大,reverse = True时由大到小*

>>> students= [('john', 'A', 15), ('jane', 'B', 12), ('dave', 'B', 10)]

>>> sorted(students, **key=lambdas: s[2]**)# 按年龄排序***通过列表元素的[2]即是第三个进行比较排序***

[('dave', 'B', 10), ('jane', 'B', 12), ('john', 'A', 15)]

>>> sorted(students, key=lambdas: s[2], reverse=True)# 按降序

[('john', 'A', 15), ('jane', 'B', 12), ('dave', 'B', 10)]

# 获取列表的第二个元素

def takeSecond(elem):

return elem[1]

# 列表

random = [(2, 2), (3, 4), (4, 1), (1, 3)]

# 指定第二个元素排序

random.sort(key=takeSecond)

# 输出类别

print ('排序列表：', random)

>>排序列表：[(4, 1), (2, 2), (1, 3), (3, 4)]

*4.#定义一个函数 接受任意的参数 并输出结果*

*# def fun08(\*args,\*\*kwargs):*

# day09

1. %3d 表示输出3位整型数,不够3位[右对齐](http://www.baike.com/sowiki/右对齐?prd=content_doc_search)。  
　　%9.2f 表示输出场宽为9的浮点数,其中小数位为2,整数位为6,小数点占一位,不够9位右对齐。 （注意：小数点前的数字必须大于小数点后的数字。小数点前的数值规定了打印的数字的总宽度。如果忽略了（如：.2f），则意味着总宽度无限制。  
　　%8s 表示输出8个字符的字符串,不够8个字符右对齐。

2.**行列转换:**

*#方法一:*任意矩阵

game\_map = [

[1,2,3,4],

[5,6,7,8],

[9,10,11,12],

[13,14,15,16]]

list\_1 = []

**for** i **in** range(len(game\_map[0])):

list\_1.append([])

**for** j **in** range(len(game\_map)):

list\_1[i].append(game\_map[j][i])

print(list\_1)

#这个好

list\_1 = []

**for** i **in** range(len(game\_map[0])):

list\_1.append([])

**for** row **in** game\_map:

list\_1[i].append(row[i])

print(list\_1)

方法二:将上述转化为列表嵌

list\_1 = []

**for** i **in** range(4):

list\_1.append([row[i] **for** row **in** game\_map])

print(list\_1)

转换为列表嵌套式:print([[row[i] **for** row **in** game\_map]**for** i **in** range(4)])

*#方法三:*方阵

list\_2 = [

[1,5,9,13],

[2,6,10,14],

[3,7,11,15],

[4,8,12,16]]

**for** c **in** range(1,len(game\_map)):

**for** i **in** range(c,len(game\_map)):

game\_map[i][c-1],game\_map[c-1][i] = \

game\_map[c-1][i],game\_map[i][c-1]

print(game\_map)

3.*#最高得分表2.0 嵌套序列*

scores = []

choice = **None**

**while** choice != **'0'**:

print(**'''**

**最高得分表**

**0 - 退出**

**1 - 得分表**

**2 - 添加得分'''**)

choice = input(**'请选择：'**)

**if** choice == **'0'**:

print(**'再见'**)

**elif** choice == **'1'**:

print(**'最高得分表'**)

print(**'姓名\t得分'**)

**for** entry **in** scores:

score, name = entry

print(name, **'\t'**, score)

**elif** choice == **'2'**:

name = input(**'选手姓名：'**)

score = int(input(**'选手得分：'**))

entry = (score, name)

scores.append(entry)

scores.sort(reverse=**True**)

scores = scores[:5]

**else**:

print(**'输入非法'**)

input(**'\n按下回车键退出。'**)

**3.列表与字符串**

将多个字符串拼接为一个。result = "连接符".join(列表)

将一个字符串拆分为多个。列表 = “a-b-c-d”.split(“分隔符”)

str01 = **'welcome to beijng'**

str02 = str01[-11::-1] +str01[10:6:-1] +str01[16:10:-1]

print(str02)*#emoclew ot gnjieb*

str01 = str01[::-1]*#gnjieb ot emoclew*

str01 = str01.split()*#['gnjieb', 'ot', 'emoclew']*

str01 = str01[::-1]*#['emoclew', 'ot', 'gnjieb']*

str01 = **' '**.join(str01)*#emoclew ot gnjieb*

print(str01)

# Day10

*1.*

*#删除所有不及格学生,通过索引从后往前删,因为列表内预留内存空间*

*# 会自动进位,容易出错*

**def** delete\_nopass():

**for** i **in** range((len(list\_2)-1),-1,-1):

**if** list\_2[i].score < 60:

**del** list\_2[i]

**return** list\_2

2.灵活使用类

*#小明有10000块钱,小白有1000,然后小白问小明借5000,问小白小明各多钱*

**class** Person:

**def** \_\_init\_\_(self,name,money,obj=**None**):

self.name = name

self.money = money

self.obj = obj

**if** self.obj:

self.borrow()

**def** borrow(self):

self.obj.money -= 5000

self.money += 5000

xiaoming = Person(**'小明'**,10000)

xiaobai = Person(**'小白'**,1000,xiaoming)

print(**'小明有%s,小白有%s.'** % (xiaoming.money,xiaobai.money))

# Day11

1.在编写类实例变量时,alt+回车,会生成形参

2.*'''*

*面向对象*

*'''*

*#创建技能skill*

**class** Skill:

**def** \_\_init\_\_(self,name,atk\_ration,duration,mp\_cons):

self.name = name

self.atk\_ration = atk\_ration

self.duration = duration

self.mp\_cons = mp\_cons

@property

**def** atk\_ration(self):

**return** self.\_\_atk\_ratio

@atk\_ration.setter

**def** atk\_ration(self,value):

**if** 0.1 <= value <= 5:

self.\_\_atk\_ratio = value

**else**:

**raise** ValueError(**'攻击比例不在范围'**)

@property

**def** duration(self):

**return** self.\_\_duration

@duration.setter

**def** duration(self, value):

**if** 0.1 <= value <= 10:

self.\_\_duration = value

**else**:

**raise** ValueError(**'持续时间不在范围'**)

@property

**def** mp\_cons(self):

**return** self.\_\_mp\_cons

@mp\_cons.setter

**def** mp\_cons(self, value):

**if** 0 <= value <= 100:

self.\_\_mp\_cons = value

**else**:

**raise** ValueError(**'消耗法力不在范围'**)

skill\_1 = [

Skill(**'降龙十八掌'**,5,9,80),

Skill(**'九阴白骨爪'**,4,7,50),

Skill(**'一阳指'**,5,2,60),

Skill(**'打狗棒法'**,2,10,0)

]

*#1.查找降龙十八掌*

**def** find():

**for** item **in** skill\_1:

**if** item.name == **'降龙十八掌'**:

**return** item

*# item = find()*

*# if item: #先判断*

*# print('查找降龙十八掌:',item.name)*

*#2.查找不消耗法力的技能*

**def** find\_mp():

result = []

**for** item **in** skill\_1:

**if** item.mp\_cons == 0:

result.append(item)

**return** result

*# res = find\_mp()*

*# for item in res:*

*# print('不消耗法力的技能:',item.name)*

*#3.查找所有技能以及持续时间*

*# def find\_all():*

*# # list\_1 = []*

*# for item in skill\_1:*

*# print(item.name, item.duration)*

*# print('查找所有技能以及持续时间:')*

*# find\_all()*

**def** find\_3():

*# result = {}*

*# for item in skill\_1:*

*# result.setdefault(item.name,item.duration)*

**return** {item.name:item.duration **for** item **in** skill\_1}

*# res = find\_3()*

*# for k,v in res.items():*

*# print(k,v)*

*#4.删除所有不消耗法力的技能:*

**def** del\_mp():

count = 0

list\_1 = []

**for** i **in** range(len(skill\_1)-1,-1,-1):

**if** skill\_1[i].mp\_cons == 0:

*# print(skill\_1[i].name)*

list\_1.append(skill\_1[i].name)

**del** skill\_1[i]

count += 1

**return** list\_1,count

*# print('删除所有不消耗法力的技能:')*

*#*

*# print(del\_mp())*

*#5.查找攻击比例最大的技能*

**def** find\_atk():

max\_1 = skill\_1[0] *#赋值要灵活*

**for** i **in** range(1,len(skill\_1)):

**if** max\_1.atk\_ration < skill\_1[i].atk\_ration:

max\_1 = skill\_1[i]

**return** max\_1

*# print('\n查找攻击比例最大的技能:')*

*# res = find\_atk()*

*# print(res.name,res.atk\_ration)*

*#6.对技能列表按照持续时间升序排列*

**def** sort\_dur():

**for** r **in** range(len(skill\_1)-1):

**for** c **in** range(r+1,len(skill\_1)):

**if** skill\_1[r].duration > skill\_1[c].duration:

skill\_1[r],skill\_1[c] = skill\_1[c],skill\_1[r]

sort\_dur()

print(**'技能列表按照持续时间升序排列:'**)

**for** item **in** skill\_1:

print(item.name,item.duration)

# Day12

1.私有特性:

self.\_\_name = name(#对于crit = Critter(**''**) 在创建对象时,即使超范围值也可以赋值给name,不会报错,不会进行到写入方法,后续读写操作会通过方法,不影响操作)

*# self.name = name(#*在创建对象时*,name*赋值操作会进行到写入方法进行判断*,*若不符合会报错*.* 后续读写操作会通过方法*,*不影响操作*)*

*2.#老张开车去东北 有数据和功能才能定义为类*

*#老张 车 按需定义*

**class** Person:

**def** \_\_init\_\_(self,name):

self.name = name

*#面向对象 车自己的数据和行为 组合 将不同类合体 实现一个功能*

self.car = Car(self.name)

**def** go\_to(self,position,type):

*#1.需要车 耦合性过高*

*# Car().run()*

*#2.去耦合 定义变量 然后调用类方法*

print(self.name + **'去'** + position)

type.run()

*#3.通过组合思想调用类 低耦合*

*# print(self.name + '去' + position)*

*# self.car.run()*

**class** Car:

**def** \_\_init\_\_(self,owner):

self.owner = owner

**def** run(self):

print(**'这是%s的车'** % self.owner)

print(**'开车走'**)

*# c01 = Car()*

lz = Person(**'老张'**)

ll = Person(**'老李'**)

*#老张开老李的车*

lz.go\_to(**'东北'**,ll.car)

3.按需采用,老师推荐使用raise

*#程序报错停止***raise** ValueError(**'value不能小于0'**)

*#return 不影响用户继续使用程序* **return 'value*不能小于*0'**

**4.alt+ctrl+m** *自动将代码转换为函数*

# Day13

1.学生信息管理系统代码

*#student\_manager\_system*

*#第一步：*

*# 数据模型类：StudentModel*

*# 逻辑控制类：StudentManagerController*

*# 数据：学生列表* *\_\_stu\_list*

*# 行为：获取列表* *stu\_list,*

*# 添加学生* *add\_student*

*# 添加id 整数 递增 类变量*

*# 数据模型类*

**class** StudentModel:

*#id不需要传值 放在最后一位*

**def** \_\_init\_\_(self, name=**""**, age=0, score=0,id=0):

self.id = id

self.name = name

self.age = age

self.score = score

*# 逻辑控制类：*

**class** StudentManagerController:

\_\_stu\_id = 1000

**def** \_\_init\_\_(self):

self.\_\_stu\_list = []

@property

**def** stu\_list(self):

**return** self.\_\_stu\_list

*#1.添加学生*

**def** add\_student(self,stu):

*#为学生设置id 递增*

*# StudentManagerController.\_\_stu\_id += 1*

*#stu.id = StudentManagerController.\_\_stu\_id*

stu.id = self.\_\_generate\_id()

*#将学生添加到学生列表*

self.\_\_stu\_list.append(stu)

*#生成id*

**def** \_\_generate\_id(self):

StudentManagerController.\_\_stu\_id += 1

**return** StudentManagerController.\_\_stu\_id

*# 2.删除学生remove\_student*

*#根据id删除学生*

*#删除后返回结果 成功True/失败False*

**def** remove\_student(self,id):

**for** item **in** self.\_\_stu\_list:

**if** item.id == id:

self.\_\_stu\_list.remove(item)

**return True**

**return False**

*#3.修改学生update\_student*

*# s01 = StudentModel('zs',18,50)*

*# s01 = StudentModel('zs',19,40,1001)*

**def** update\_student(self,stu):

**for** item **in** self.\_\_stu\_list:

**if** item.id == stu.id:

item.name = stu.name

item.age = stu.age

item.score = stu.score

**return True**

**return False**

*# 4.根据成绩排序order\_by\_score。*

*# 你是谁 你要找谁*

**def** order\_by\_score(self):

**for** i **in** range(len(self.\_\_stu\_list)-1):

**for** j **in** range(i+1,len(self.\_\_stu\_list)):

**if** self.\_\_stu\_list[i].score > self.\_\_stu\_list[j].score:

self.\_\_stu\_list[i],self.\_\_stu\_list[j] = \

self.\_\_stu\_list[j],self.\_\_stu\_list[i]

*# 界面视图类：*

**class** StudentManagerView:

*#生成在此类下的逻辑管理器 组合*

**def** \_\_init\_\_(self):

self.\_\_manger = StudentManagerController()

*#1.显示菜单*

**def** \_\_display\_menu(self):

print(**'''**

**学生信息管理系统1.0**

**+-----------------------+**

**| 0)退出管理系统** **|**

**| 1)添加学生信息** **|**

**| 2)显示学生信息** **|**

**| 3)删除学生信息** **|**

**| 4)修改学生信息** **|**

**| 5)按照成绩排序** **|**

**+-----------------------+**

**'''**)

*#2.选择菜单*

**def** \_\_select\_menu(self):

*# import sys*

option = input(**'请选择:'**)

**if** option == **'1'**:

self.\_\_input\_students()

**elif** option == **'2'**:

self.\_\_output\_students()

**elif** option == **'3'**:

self.\_\_delete\_student()

**elif** option == **'4'**:

self.\_\_modify\_student()

**elif** option == **'5'**:

self.\_\_output\_student\_by\_score()

**elif** option == **'0'**:

*# print('谢谢使用,退出系统')*

*# sys.exit()*

exit(**'谢谢使用,退出系统'**)

**else**:

print(**'请正确输入选项!'**)

*#3.入口逻辑*

**def** main(self):

*'''*

*界面入口*

**:return***:*

*'''*

**while True**:

self.\_\_display\_menu()

self.\_\_select\_menu()

*# 4.输入学生*

**def** \_\_input\_students(self):

*#收集学生信息 要求用户输入 姓名年龄 成绩*

*#创建学生对象*

*#在控制器找add\_student方法*

name = input(**'请输入学生姓名:'**)

age = int(input(**'请输入学生年龄:'**))

score = int(input(**'请输入学生成绩:'**))

stu = StudentModel(name,age,score)

*#这样写会重复生成 管理器 列表,因此在视图界面类创造一个管理器*

*# manger = StudentManagerController()*

self.\_\_manger.add\_student(stu)

*#5.输出信息*

**def** \_\_output\_students(self):

**for** item **in** self.\_\_manger.stu\_list:

print(item.id, item.name, item.age, item.score)

*#6.删除学生*

**def** \_\_delete\_student(self):

*#需要用户输入id*

*#调用管理器对象的删除学生方法*

*#如果结果为True 显示删除成功*

*#否则显示删除失败*

id = int(input(**'请输入需要删除的学生ID:'**))

**if** self.\_\_manger.remove\_student(id):

print(**'删除成功!'**)

**else**:

print(**'删除失败!'**)

print(**'删除后学生信息:'**)

self.\_\_output\_students()

*#7.修改学生信息*

**def** \_\_modify\_student(self):

*#手机用户输入的信息保存到对象*

*#调用管理器的修改学生方法*

id = int(input(**'请输入需要修改的学生ID:'**))

name = input(**'请输入修改后学生姓名:'**)

age = int(input(**'请输入修改后学生年龄:'**))

score = int(input(**'请输入修改后学生成绩:'**))

stu = StudentModel(name, age, score,id)

**if** self.\_\_manger.update\_student(stu):

print(**'修改成功!'**)

**else**:

print(**'修改失败!'**)

print(**'修改后学生信息:'**)

self.\_\_output\_students()

*#8.按成绩升序排列*

**def** \_\_output\_student\_by\_score(self):

self.\_\_manger.order\_by\_score()

print(**'按照成绩排序学生:'**)

self.\_\_output\_students()

2. .程序退出的方法:

1**). sys.exit()** 执行该语句会直接退出程序，一般是退出Python程序的首选方法。

该方法中包含一个参数status，默认为0，表示正常退出，也可以为1，表示异常退出。

2**). os.\_exit()**效果也是直接退出，不会抛出异常，但是其使用会受到平台的限制，但我们常用的Win32平台和基于UNIX的平台不会有所影响。

**3). os.kill()**一般用于直接Kill掉进程，但是只能在**UNIX平台**上有效。

基本原理：该函数是**模拟传统的UNIX函数发信号给进程**，其中包含两个参数：一个是进程名，即所要接收信号的进程；一个是所要进行的操作。

4)exit退出整个运行。并且还可以在exit（）的括号里加入自己退出程序打印说明.

3.*# 老张去东北* *day11/demo06.py*

*# 需求变化 ： 坐飞机*

*# 坐火车*

*# 骑车*

*# Dependency Inversion Principle*

*# 客户端代码(调用的类)尽量依赖(使用)抽象。*

*# 抽象不应该依赖细节，细节应该依赖抽象*

**class** Vehicle:

*#交通工具 代表所有的具体的交通工具*

**def** transport(self,position):

*#因为父类太抽象，写不出具体的方法体*

**pass**

**class** Person:

**def** \_\_init\_\_(self, name):

self.name = name

**def** go\_to(self,position,vehicle):

print(**'去：'** + position)

vehicle.transport(position)

*#上边抽象代码不需要修改,改动只改变下边的具体代码*

*# ----------------------------------------*

**class** Car(Vehicle):

**def** transport(self,position):

print(**'开车到'**,position)

**class** Airplane(Vehicle):

**def** transport(self,position):

print(**'飞到'**,position)

c01 = Car()

a01 = Airplane()

lz = Person(**'老张'**)

lz.go\_to(**'东北'**, c01)

lz.go\_to(**'东北'**, a01)

# 8月17日周六练习

1.列表推导式:

1)list01 = [[1+i\*3,2+i\*3,3+i\*3] **for** i **in** range(2)] print(list01)

>>>[[1, 2, 3], [4, 5, 6]]

2)嵌套技巧:

>>>vec1=[2,4,6]  
>>>vec2=[4,3,-9]  
>>>[x\*y for x in vec1 for y in vec2]  
[8,6,-18,16,12,-36,24,18,-54]  
>>>[x+y for x in vec1 for y in vec2]  
[6,5,-7,8,7,-5,10,9,-3]  
>>>[vec1[i]\*vec2[i] for I in range(len(vec1))]  
[8,12,-54]

3)嵌套函数:

>>>[str(round(355/113,i)) for I in range(1,6)]  
['3.1','3.14',3.142','3.1416','3.14159']

eg:list1 =[]

**for** i **in** range(2):

list1.append([])

**for** j **in** range(1,4):

list1[i].append(j+i\*3)

print(list1)*#[[1, 2, 3], [4, 5, 6]]*

*#列表嵌套式:*

print([[j+i\*3 **for** j **in** range(1,4)] **for** i **in** range(2)])

#行列转换:list\_1 = []

**for** i **in** range(len(game\_map[0])):

list\_1.append([])

**for** row **in** game\_map:

list\_1[i].append(row[i])

print(list\_1)

方法二:将上述转化为列表嵌套式

list\_1 = []

**for** i **in** range(len(game\_map[0]):

list\_1.append([row[i] **for** row **in** game\_map])

print(list\_1)

转换为列表嵌套式:print([[row[i] **for** row **in** game\_map]**for** i **in** range(len(game\_map[0])])

# day14

1.count = **''**

**for** i **in** range(3):

count += str(i) + **' '**

print(count) *#0 1 2 先+= 再+*

# day15

1.高阶函数:

# DAY16

1.pip是Pip Installs Packages的缩写

2.用解释器运行代码; #!/usr/bin/env python3

终端:(直接运行.py文件)

chmod +x add\_student.py

./add\_student.py

3.**import** sys

*#在终端运行是,需要添加路径,为所在包的上一级目录*

sys.path.append(**'/home/tarena/pytnon-month01/month01-石博文笔记/day16-shibw/project'**)

print(sys.path)*#在文件名右键 点击mark再点root*

*#目录.包 导入模块*

**from** game **import** game\_2048

*#模块名.成员*

game\_2048.fun01()

4.sys.argv:argv」是「argument variable」参数变量的简写形式，一般在命令行调用的时候由系统传递给程序。这个变量其实是一个List列表，命令行参数List，第一个元素是程序本身路径:

sys.argv[0]表示代码本身文件路径，所以参数从1开始. sys.argv[1]表示第一个命令行参数 sys.argv[1:]表示从第一个命令行参数到输入的最后一个命令行参数

sys.argv[1][2:] 表示取第一个命令行参数，但是去掉前两个

**import** sys

*#python的位置参数列表*

print(sys.argv)

**def** add\_student(name,id):

print(**'添加学生成功，%s:%s'** % (id,name) )

**if** \_\_name\_\_ == **"\_\_main\_\_"**:

add\_student(sys.argv[1],sys.argv[2])

*#必须在终端中运行时 从外部获得参数*

print(sys.argv[0])

在终端中:

python3 add\_student.py '张无忌' 99

['add\_student.py', '张无忌', '99']

添加学生成功，99:张无忌

4.编码格式书写:

*#coding=utf-8*

*#coding:utf-8*

*#-\*- coding:utf-8 -\*-*

*5.***import** time

*#****获取当前时间戳***  *从1970年1月1日到现在的总秒数*

print(time.time())*#1566385830.1379533*

*#获取当前的时间元组*

*#(年 月 日 时 分 秒 一周的第几天(0 1 2...) 一年的第几天 夏令时)*

print(time.localtime())*#time.struct\_time(tm\_year=2019, tm\_mon=8,.....)*

*#获取指定时间戳的时间元组*

print(time.localtime(12345678))*#time.struct\_time(tm\_year=1970, ...)*

time\_tuple = time.localtime()

*#****将时间元组变成时间戳***

print(time.mktime(time\_tuple))*#1566386031.0*

*#将时间元组转换成时间字符串*

print(time.strftime(**'%y/%m/%d %H:%M:%S'**,time\_tuple))*#19/08/21 19:17:07*

print(time.strftime(**'%Y-%m-%d %H:%M:%S'**,time\_tuple))*#2019-08-21 19:17:07*

*#****将日期字符串转换为时间元组***

print(time.strptime(**"19/08/21 11:31:30"**, **'%y/%m/%d %H:%M:%S'**))*#time.struct\_time(tm\_year=2019, tm\_mon=8,.....)*

print(time.strptime(**"2019/08/21 11:31:30"**, **'%Y/%m/%d %H:%M:%S'**))

# day17

**super()** 函数是用于调用父类(超类)的一个方法。super(type[, object-or-type]) 一般为self

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。

MRO 就是类的方法解析顺序表, 其实也就是继承父类方法时的顺序表。

Python 3 和 Python 2 的另一个区别是: Python 3 可以使用直接使用**super().xxx**代替**super(**子类**Class, self).xxx**:

MRO指的是**type(obj)**的 MRO, MRO 中的那个类就是**a\_type**, 同时**isinstance(obj, a\_type) == True**。

**class** Aplication(Frame):

**def** \_\_init\_\_(self,master):

*# super(Aplication,self).\_\_init\_\_(master)*

super().\_\_init\_\_(master)#Aplication调用了父类Frame的构造函数

# day18

1.Ctlr +q 查找导入模块注释

ctlr+p 查看函数参数 #filter()#鼠标放在括号里，点击Ctrl+p

ctrl+鼠标左键 查看函数或变量的相互关系

# 8.24周六

1.今天总结第一阶段 需要重看

2.集成[开发环境](https://baike.baidu.com/item/开发环境)（[IDE](https://baike.baidu.com/item/IDE)，Integrated Development Environment ）是用于提供程序开发环境的应用程序，一般包括代码编辑器、[编译器](https://baike.baidu.com/item/编译器/8853067)、[调试](https://baike.baidu.com/item/调试)器和[图形用户界面](https://baike.baidu.com/item/图形用户界面/3352324)等工具。集成了代码编写功能、分析功能、[编译](https://baike.baidu.com/item/编译/1258343)功能、调试功能等一体化的开发软件服务套。所有具备这一特性的软件或者软件套（组）都可以叫集成开发环境。如微软的Visual Studio系列，Borland的[C++ Builder](https://baike.baidu.com/item/C%2B%2B Builder)、Delphi系列等。该程序可以独立运行，也可以和其它程序并用。IDE多被用于开发[HTML](https://baike.baidu.com/item/HTML)应用软件。例如，许多人在设计网站时使用IDE（如[HomeSite](https://baike.baidu.com/item/HomeSite)、[DreamWeaver](https://baike.baidu.com/item/DreamWeaver)等），因为很多项任务会自动生成。