**Python基础目录**

day01回顾：

1.硬件和软件

2.软件分类： 1).系统软件 2).应用软件

3.人机交互(两种)： 1).命令行方式 2).图形化界面

4.启动dos窗口

快捷方式：wins键 + R键 回车(enter)--> 打开控制台

5.常用的dos指令

切换盘符：d: e: f:

dir： 查看当前路径下的所有子目录和文件(直接的那层)

help： 得到所有的dos命令和其中文解释

cd： 显示当前路径或将其更改

.和..： 一点(.)表示当前路径，两个点(..)表示上一层路径

/和\： 表示"根"路径

cls： 清除屏幕

start： 打开一个新的dos窗口

exit： 关闭dos窗口

6.计算机语言

7.path环境变量

作用：将一串路径(绝对)定义到path中，就可以使用dos窗口，在任何路径下都能执行其相应的命令(exe结尾)

8.python3和pycharm的环境搭建

9.常量和变量

变量的概念：它是一个"容器"，是内存中的一块存储区域，目的用来存储数据，并且可以随时改变其中的内容，第一次存入变量的数据/值，我们称为初始化值；

10.关键字和标识符

11.变量的定义格式：(三种)

格式一：变量名 = 值

格式二：变量名1 = 变量名2 = 值

格式三：变量名1,变量名2 = 值1，值2 **(python独有)**

12.输出语句：

函数：print(...,[end=])：

作用：将内容输出到控制台，如果定义end关键字，一般是用于不换行操作

print(**'为了部落！',end=''**)  
print(**'为了联盟！',end=""**)  
print(**'为了艾泽拉斯！'**)  
print(**'为了德玛西亚！'**)

13.内置函数：

print()、type()、id()、str()...

----------------------------------------------------------------------------------------

day02(回顾)：

1.输入语句：内置函数：**input(str)**

返回结果：字符串类型的数据

作用：从键盘读入字符串内容，返回给程序

2.**注释**：

作用：将代码程序翻译解释，给自己看，也给别人看

分类：单行：标记一行 符号：# 多行：标记多行 符号: ''' ''' """ """

3.进制：

理解：一种计数单位

二进制的由来：

第一台计算机 --> u.s.a在1946.02.14发明诞生的

进制之间的转换：

四种：2，8，10，16进制

4.**数据类型**：

分类学习：

1).**数值型**： --> 整数型(int)、浮点型(float) 布尔型：真/对(True) 假/错(False)

2).**容器类型**：**字符串(str)、**列表(list)、元祖(tuple)、字典(dict)、集合(set)

3).**自定义类型**：Person、Man、Women Animal、Dog、Cat

4).**NoneType**：None

5.**字符串类型：**

**特点：有序(有**索引、定义和显示的顺序是一致的)、**可重复**、**不可变的数据类型**

注意：字符串数据任何位置上的元素值不可单独改变(不接受重复赋值操作)，但是可以改变地址的引用

定义格式：

1).' ' 或者 ""

2).''' ''' 或者 """ """ 可以保留自定义的输出样式(接受换行)

转义字符：

有特殊含义的一些字符数据；常用的如下：\n、\t、\b、\"、\'....

强调：r的使用

如果字符串数据描述的是一串物理地址，我们都会显示的在第一个引号的前面添加r，避免转义；

字符串的+和\*操作：

对于+操作而言：拼接两个字符串数据，得到更长的一个新串返回

对于\*操作而言；字符串数据和整数数据运算，将字符串数据重复执行多次

如何访问字符串元素：

由于字符串属于容器类型的一种，所以它可以被内置函数len()执行得到其长度/容量；

字符串有下标(索引、角标、index)：

概念：起始索引(最小索引)：0 结束索引(最大索引)：-1 len(字符串变量) - 1

1).获取字符串元素：

通过字符串变量配合下标来实现

2).设置字符串元素

会报错，没有这项操作，因为字符串属于不可变的数据类型，一旦尝试设置和此特点冲突！！

字符串的**切片操作：**

目的：**从原串中获取部分的子串数据，注意：原串不会发生改变...**

格式：

str[[start]:[end]:[step]]

参数解释：

start：起始位置

end: 结束位置

strp： 步长

注意：**满足含头不含尾**

占位符：

分类：

1).**百分号原则：**

**%d： %f： %s：**

2)**.大括号原则：**

需要配合字符串(str)类中的format函数一起使用；

-------------------------------------------------------------------------------------------

day03回顾：

1.列表(list)：

特点：有序的、可重复的、可变的数据类型

1).定义格式：

**列表名 = [值1,值2,...,值n]**

列表名 = []

列表名 = [值]

2).可以被内置函数len()得到其长度

3).如何访问列表元素：

①.获取元素 ②.设置元素

4).和+、\*符号配合使用

5).列表的函数(少量、部分)：

append(obj)：将obj元素追加到列表的末尾

clear()：清空列表

6).del关键字配合列表使用

目的：回收列表元素，回收列表对象

7).列表的切片操作：

和str一样理解

2.元祖(tuple)：

特点：有序的、可重复的、不可变的数据类型

1).定义格式：

定义一个元素的元祖：元祖名 = (值,)

2).如何访问元祖元素：

获取元素

3).和+、\*符号配合使用

4).del关键字配合列表使用

目的：回收整个元祖对象

5).元祖的切片操作：

和list&str一样理解

3.字典(dict)：

特点：无序性、键不可重复、值可以重复、可变的数据类型(键不可变，值可变的)

键-值对：key-value对

键：满足唯一性、不可变的数据类型

值：没有要求、可以是任何的数据类型，甚至可以重复

【记住】：只能通过键找值，不能通过值找键，因为值可能重复，不能找到唯一的键

del关键字配合字典使用

目的：回收键值对，回收整个字典对象

4.集合(set)：

特点：无序性、不可重复性、可变的数据类型 【注意】存储的都是不可变的数据类型 例如：list、dict等类型对象不能存入

用途：对数据进行去重(相当于oracle中的distinct)

集合和集合对象之间可以有如下的一些操作：

**&：取交集**

**|：取并集**

**-：取差集(谁在左边保留谁独有的元素)**

**^：先取并集再舍弃交集部分**

【注意】：没有+和\*操作

5.运算符：

分类：

1).算数运算符：

符号：+ - \* / % // \*\* 二元运算

2).比较运算符：

符号：> >= < <= == !=

特点：对于>=和<=而言，只要满足其中一项操作，整个运算的结果就为True

3).赋值&混合赋值运算符：

赋值：=

混合赋值：+= -= \*= /= %=

4).逻辑运算符：

符号：and、or、not

对于and而言：满足一假即假，对False敏感

对于or而言：一真即真，对True敏感

对于not而言：取反操作

5).位运算符：(了解)

符号：&(按位与)、|(按位或)、^(按位异或)、>>(右移)、<<(左移)

6).成员运算符：

符号：in 、not in

作用：判断某个元素是否在容器对象中存在，结果一定是布尔值

**7).三元/三目运算符：**

**格式：表达式1 if 条件表达式 else 表达式2**

**6.流程控制语句之判断结构：**

**格式一：if ...**

**格式二：if ... else ...**

------------------------------------------------------------------------------------------

day04回顾：

1.判断结构之if的第三种格式：if elif ... else

2.循环结构

第一种格式：while

2.1.具备循环四要素：

1).初始化条件

2).循环条件

3).迭代条件

4).循环体

模板：

①

while ②:

④

③

2.2.break和continue关键字配合循环使用：

对于break关键字而言： 在执行循环的过程中，一旦遇到break关键字，立即结束当前循环

对于continue关键字而言： 在执行循环的过程中，一旦遇到了continue关键字，激励结束当次循环，开启下一次；

2.3.循环循环&死循环：

何时出现：

一旦循环条件永远为True，就是无限循环；此现象极度占用内存资源，需要避免；

但是一旦和break关键字配合使用，就会变的有意义；

2.4.else语句配合循环使用：

特点：

如果循环是正常执行完毕，就会执行else中的语句块

如果循环是由于break强行终止，那么就不会执行else中的语句块

2.5.random模块的初步了解：

random模块中的randint(a,b)函数的使用： 得到[a,b]之间的随机整数

步骤：

1).导入模块 代码体现：import random

2).通过模块名调用函数完成需求 代码体现：random.randint(1,5)

2.6.time模块的初步了解：

time模块中的sleep(sec)函数的使用： 程序一旦执行遇到sleep()函数，会立即休眠sec秒；时间到了，自动醒过来，继续往下执行

步骤：

1).导入模块 代码体现：import time

2).通过模块名调用函数完成需求 代码体现：time.sleep(3)

3.转换函数：

【强调】：对于bool()函数

如果传入的参数为：0、0.0、False、None、""、[]、()、{}、set() 得到的结果都是False

除了以上的取值，其余内容都是True；

理解：对于流程控制语句if、while后面的条件表达式都有一个隐式的bool()函数存在

-----------------------------------------------------------------------------------------------------

day05回顾：

1.forin循环

格式:

for 变量名 in 容器对象:

语句块

【注意】

1).forin循环不需要考虑下标/索引

2).需要和容器对象配合使用(range、str、list、dict...)

range对象：

类型：Range 容器对象的一种(容量/长度)

格式：

range([start],end,[step])

特点：

1).含头不含尾

2).默认从0开始

3).默认步长为1

对于dict类型：

items()函数：返回给程序一个dict\_items类型的对象，内部每个元素都是一个一个的元祖对象，元祖对象一共有两个元素，0索引位置存储的是键，1索引位置存储的是值

配合break&continue以及else语句一起使用：

和while循环一样理解

2.嵌套/多层循环：

存在的意义：

一个循环结构不了需求，我们可以定义多个循环，它们之间是嵌套的关系

实际开发中，两层足够了；

格式：4种

最常用的：for 中套 for

总结：

1).外层循环执行一次，内层循环从头到尾全部执行一遍

2).如果外层循环需要执行m次，内层循环需要执行n次，那么整个嵌套循环一共需要执行m \* n次

3.排序算法

特点：升序、降序

实现了计算机领域的选择排序(升序)

记住：不管是选择排序还是冒泡排序在之后的开发环境中，都不会使用，但是面试喜欢面

4.函数：

又称为：方法、method、function

概念：函数就是一种功能、行为；就是一种逻辑代码；需要被调用才能执行；

执行需要在栈内存的栈顶开辟空间(进栈)，一旦执行完毕了，需要被内存回收(弹栈)；

栈内存满足先进后出的原则；

定义格式满足考虑5要素：

1).函数修饰符： 有名函数(def) 匿名函数(lambda)

2).函数返回值类型： 由于python语言是弱类型语言，在定义函数上不需要显示的书写类型(考虑进去)

3).函数名： 也是标识符的一种，满足规则和规范，其规范和变量名保持一致(驼峰式)

4).形参列表： 定义在小括号内部，可能有参数，也可能没有参数

5).函数体： 功能代码

尝试去定义自定义函数，在此过程中，我们需要考虑两个原则：

1).有没有形参

2).有没有返回值

函数的基本概念：

1).形式参数：

2).实际参数：

3).返回值：

如果有返回值，我们需要在函数的最后位置使用return关键字将数据返回给程序的调用者

return关键字的作用：

1).返回数据给到程序

2).结束当前函数

注意事项：

1).return关键字的后面不能定义任何代码，因为永远不可能被执行到 --> 补充：在同一作用范围内

2).return的后面可以不定义任何内容，仅仅意味着函数要结束了

参数的多个使用场景：

1).默认参数的使用

2).一般参数的使用

3).关键字参数的使用

4).可变参数的使用

--------------------------------------------------------------------------------------------

day06回顾：

1.全局&局部变量：

区分：位置

2.外部&内部函数：

关于内部函数：

1).内部函数可以使用外部函数的内容

2).内部函数的执行需要依赖于它的外部函数

3).外部函数不能使用内部函数的任何内容

3.global&nonlocal关键字

作用：

global： 在局部位置修改全局变量的值 满足：先锁定，在修改

nonlocal： 在内部函数中修改外部函数的局部变量的值 满足：先锁定，在修改

3.ascii码

内置函数：

1).ord()：将字符转换为整数值(码值)

2).chr()：将整数数据转换为字符数据

4.列表生成式：

作用：以更加简洁、方便的代码来得到最终的一个列表对象，它也是语法糖之一

格式：

[expr for ver1,[ver2],... in 序列对象 if ...]

5.常用的内置函数：

round()、abs()、max()、min()、sum()、

hex()、oct()、bin()、

pow(x,y)、divmod(x,y)、

...

6.random、math模块中常用的函数：

random模块：

random()、uniform(a,b)、

randrange([start],end,[step])、randint()、

choice(seq)、sample(seq,num)、

shuffle(list)、

math模块：

属性：pi、e

函数：ceil()、floor()、

sqrt()、radians()、degrees()

补充：

sys模块中的argv属性 作用：将当前正在被执行的py文件的绝对路径以列表的形式返回

os模块中的system函数 作用：参数位置可以接受字符串类型的dos指令，解析执行 注意：可能乱码或者无效

7.str中常用的函数：

join()：ljust()：rjust()：center()：zfill()：

lstrip()：rstrip()：strip()：

split()：splitlines()：

index()：count()：

replace()：

find()：rfind()：

lower()：upper()：capitliazes()：title()：swapcase()：

startswith()：endswith()：

isalnum()：isalpha()：isdecimal()：isdigit()：islower()：isupper()：istitle()：

-----------------------------------------------------------------------------------------------------------

day07回顾：

1.list中常用的函数：

append()、extend(iterable)、

index()、count()、

pop()、remove()、

reverse()、sort([reverse=False],[key=...])、

论浅拷贝&深拷贝...

list中的copy()函数、copy模块中的copy()函数 都是属于浅拷贝

copy模块中的deepcopy()函数属于深拷贝(完全脱离)

2.dict中常用的函数：

update(dic1)、

popitem()、pop()、

items()、keys()、values()、

clear()、copy()...

3.set中常用的函数：

add()、pop()、remove()、clear()、copy()...

issubset()、issuperset()、isdisjoint()...

4.闭包

思想：函数式编程

满足3个条件：

1).外部函数和内部函数这样的结构

2).内部函数需要使用外部函数的变量

3).内部函数对象作为外部函数的返回值

5.装饰器

属于闭包的一种使用场景：

好处：在不改变功能的源代码的基础上，进行功能的扩展

常用的4种格式：

----------------------------------------------------------------------------------------

day08回顾：

1.os&os.path模块：

绝对&相对路径：

os模块：管理文件和目录

属性：

name、environ、

environ.get(环境变量名) --> 常用一点

函数：

rmdir(path)

removedirs(path)

remove(path)

mkdir(path)

makedirs(path)

rename(src,dest)

getcwd()

listdir(path) --> ['a','b','c','1.txt','2.avi']

path模块：处理物理地址(路径)

join(first,second)

getsize(path)

exists(path)

isfile(path)

isdir(path)

basename(path)

dirname(path)

split()

splitext()

2.计算机的基础常识：

2.1.内存和硬盘：

2.2.字节和字符：

记住：一切"皆"字节

2.3.如何区分字节和字符文件：

2.4.日常中常见的字符和字节文件：以后后缀名就能识别

2.5.字符编码(字符集)：

ascii、utf-8、gbk

2.6.编码和解码：

3.文件读写

底层：输入流(inputstream) 输出流(outputstream) 统称：io流/io体系

3个环节：

1).打开文件

2).读写数据

3).关闭文件

打开文件：

open(path、flag、encoding、errors)：

读写数据：

读： 一旦读到末尾就不能继续读取，需要将文件描述符移动到之前某位置...

read()： 弊端

read(num)： num记录了字符个数

readline()：

readlines：

tell()： 返回读取到的字节数(文件描述符在什么位置)

seek(num)： num记录了字节位置

写：

write(o)：

关闭文件：

close()

-----------------------------------------------------------------------------------------

day09回顾：

1.异常体系：

异常：不理想的状况

异常处理：

try：将可能出现异常的代码定义其中，它能第一时间发觉是否有异常对象出现，但是不能处理

except：将匹配成功的异常对象捕获并且处理

finally：将一定需要被执行的代码定义其中

else：如果没有异常出现，else一定会被执行；反之，不会被执行；出现在最后一个except的后面

raise：手动抛出一个异常对象

定义格式：

总结：异常处理主要学习5个关键字

补充语法糖：with open...as... 好处：可以不定义close()函数了

2.递归(思想)

递归函数：在一函数的内部继续调用自己，直到满足需求为止

执行原理：

在执行过程中，只有方法入栈，没有出栈；

此过程极度消耗内存资源，直到运行到了最后一次(临界点)，

开始逐个出栈(释放资源)；由于以上的特点：递归函数的执行需要保证不会出现内存溢出的情况(慎用...)

写一个简单递归的思考方式：

1).寻找一个临界点(何时终止)

2).寻找前一次和后一次函数调用的关系

-------------------------------------------------------------------------------------------------------------------------

day10回顾：

1.栈和队列：

栈结构的特点：先进后出

队列结构的特点：先出进出

对于栈而言：使用列表(list)进行模拟其存取的操作(append()和pop())

对于队列而言：使用collections模块下的deque()创建其对象，调用append()和popleft()实现存取操作

实现了深度遍历和广度遍历多层级目录

好处：除了使用递归也能进行相应的操作；比较节省内存资源

2.匿名函数：

格式：lambda 变量参数: 表达式

何时可以考虑使用匿名函数？

1).功能逻辑比较简单

2).调用的次数少

【注意事项】：

1).匿名函数也是函数的一种，同样需要调用才能执行

2).其也包含有参没参，有返回没有返回值的情况

3.生成器：(类型:generator)

两种定义模式：

1).(表达式 for 变量参数 in 容器对象 if ...)

2).和函数配合使用，函数体中会用到yield关键字，将关联到的公式/表达式定义在其之后；

返回的一定是一个generator对象

将生成器和列表生成式比较？

...

4.可迭代对象(iterable)和迭代器对象(iterator)

可迭代对象：

只要能被循环操作的对象就是可迭代对象

例如：list、str、tuple、set、range、generator、dict。。。

迭代器对象：

1).必须是可迭代对象

2).能够被next()函数直接操作的

例如：generator

学会了高效的去判断某个对象是可迭代还是迭代器

导入collections模块，调用Iterable、Iterator将其配合isinstance(obj,type)一起使用；

通过返回的布尔值，来确定是可迭代还是迭代器对象

将可迭代对象转换为迭代器对象...

使用内置函数：iter()即可

5.模块：

在python语言中只要是.py结尾的文件都是模块；

分类：

1).标准库模块：在安装完python环境已经存在；认为是实际开发中使用频率最多的那些

2).第三方模块：使用pip安装之后，可以直接使用

3).自定义模块：开发人员定义最多的，互相之间可以调用的...

导入方式：

1).精确导入：

from ... import ...

import ...

2).模糊导入：

from ... import \*

可以使用别名这种技术：

【注意】一旦使用了别名，原来的名字就不可以用了

if \_\_name\_\_ == '\_\_main\_\_'：

...

将导入时不想要被直接的代码定义其中；

-----------------------------------------------------------------------------------------------------

day11回顾：

面向对象：

面向过程和面向对象的区别

如果定义一个面向对象的程序，中间伴随这面向过程？

名词动词提炼法...

名词：设计为类

动词：定义为函数

关联步骤(初学者)：

1).设计类(难、花费大量时间)

2).实例化类对象(创建、new)

3).通过对象调用属性或者函数完成需求

将伪代码 --> 可以运行的代码

画了对象实例化的内存解析图

引入"堆"空间的概念：

只要是对象实例化，都需要在堆中开辟空间；此空间有对应的地址；

需要被栈中的某个变量所引用；如果没有任何引用指向它，

它就是一个"垃圾"空间，最终会被垃圾回收机制回收掉；

1.动态为对象添加属性(python有，很多语言不合法)

2.构造函数(重要)：

格式：\_\_init\_\_(self,...)

作用：

1).创建对象

2).为对象的属性赋值

执行时机：在创建对象时由系统自动调用执行

注意事项：

1).任何类都有构造函数

2).如果自己不显示的定义构造函数，系统会默认给程序一个空参数的构造

3.析构函数：

格式：\_\_del\_\_(self,...)

作用：

回收对象，清出内存

执行时机：在程序结束前由系统自动调用执行

4.\_\_str\_\_(self)函数

作用：将打印对象名(引用名)，显示具体的属性内容，而不是最初的类型地址；(函数重写/override)

执行时机：在打印对象名时由系统自行调用执行

5.魔术函数

\_\_开头\_\_结尾的函数，统称为魔术函数；

例如：\_\_init\_\_、\_\_del\_\_、\_\_str\_\_...

--------------------------------------------------------------------------------------------------

day12回顾：

1.面向对象特性一之封装性：(思想来源于生活)

理解：封装与隐藏

作用：确保属性数据的安全

操作：

将需要的属性私有化(\_\_属性名)，外界就不能直接访问了；

在类的内部提供给外界一套(2个)访问方式(函数，setter&getter)；

注意：

setter和getter函数需要正确的理解是否需要形参和返回值；

命名有规范的：set属性名(...)、get属性名(...) 属性名满足首字母大写其余字母小写的规范

2.面向对象特性之二继承性：(思想来源于生活)

作用：将类和类之间产生一种关系；一方面我们称为子类(派生类、subClass)；另一方面我们称为父类(超类、基类、superClass);

好处：

复用性、扩展性、维护性、阅读性都变好了；

弊端：

类和类之间有一种强耦合关系；

注意：

虽然有弊端，但是好处更多；所以我们在实际开发中，鼓励大家去使用这种特性的；

但是不能为了继承而继承；

子类和父类：

它们之间是一种相对的概念；

子类：偏向于底端的类，它更具体、更清晰；实例化的可能性越大

父类：偏向于顶层的类，它更模糊、更抽象；实例化的可能性小

在python中继承有3个方向：

1).单继承：

2).多重继承：

3).多继承： --》很多语言是没有的

函数重写：

满足的前提：需要有继承关系

何时使用？

父类中的某个函数子类需要去使用，但是其功能逻辑不符合当前需求；

我们可以重写其函数体，然后实例化子类对象，调用其此函数；

真正执行的就是子类重写父类以后的效果了；

类和对象的关系

核心：先有类、后有对象 --> 从内存的角度去划分

类属性和对象属性：

类函数和对象函数：

理解：关键字cls和self

3.pickle模块：

对象序列化和反序列化机制：

序列化(钝化)：将程序中的对象先转换为二进制数据，以序列化机制传输到硬盘文件中存储起来

反序列化(活化)：将硬盘文件中的数据(二进制)，以反序列化的方式将数据传输到内存程序中还原成为一个实实在在的对象

函数：

dump(obj,fw)： 没有返回值

load(fr)： 有返回值

-----------------------------------------------------------------------------------------

Day13

Day14

做项目 1银行管理系统 2员工管理系统

-----------------------------------------------------------------------------------------

day15回顾:

1.time模块：

clock()：打点操作，返回一个科学计数法的数据(很小)；作用：性能测试

time()：返回一个时间戳数据，数据类型：浮点(float)

sleep(s)：程序休眠s秒，然后继续往下执行

localtime(ts)：将时间戳数据转换为本地时间元祖对象(class为time.struct\_time)返回

gmtime(ts)：将时间戳数据转换为utc时间元祖对象(class为time.struct\_time)返回

【注意】：time.struct\_time类型的对象内部封装了9个元素

mktime(tp)：将本地时间元祖对象转换为时间戳对象

asctime(tp)：将本地元祖对象转换为字符串数据(显示的样式是默认的)

ctime(ts)：将时间戳对象转换为字符串数据

将本地元祖对象转换为字符串数据，我们想做到按照自己定义的格式显示

字符符号：

%Y：四位的年 相当于：yyyy

%y：两位的年 相当于：yy

%m：月份

%d：天数

%H：小时

%M：分钟

%S：秒钟

补充扩展：

%h：月份(英文简写)

%D：固定格式：月/日/年(两位)

%X：固定格式：小时:分钟:秒钟

strftime(fmt,tp)：将本地时间元祖对象以fmt进行格式化得到一个str类型的数据返回

strptime(str,fmt)：将str数据以fmt进行解析得到一个时间元祖对象返回

【注意】：这一组(2个)函数，在fmt这块必须完全保持一致

2.datetime模块：

理解：datetime可以认为是time补充/扩展

datetime模块中有一些常用的类：

datetime类：记录了时间和日期数据信息

date类： 记录了日期数据

time类： 记录了时间数据

补充：date类型和time类型的对象都可以由datetime类型的对象分离出来

datetime类型：

now()和today()：获取当前系统的日期时间对象(返回的数据类型为datetime)

utcnow()：获取当前utc日期时间对象(返回的数据类型为datetime)

构造函数：date(y,m,d,[H,M,S,MS]) --> 此构造有7个参数，y、m、d是不可省略的

从日期时间对象中分离出日期对象和时间对象

date() --> 得到date类型对象

time() --> 得到time类型对象

strftime(fmt)：将日期时间对象以fmt进行格式化得到字符串数据返回

timestamp()：将日期时间对象转换为时间戳对象

timetuple()：将日期时间对象转换为时间元祖对象

属性：

year、month、day、hour、minute、second、microsecond

fromtimestamp(ts)：将时间戳数据转换为日期对象 返回类型为date

补充：

timedelta类型：

timedelta(days=,hours=,minutes=,seconds=)：时间日期对象和其做算数运算(+、-)得到之前和以后的一个时间数据

3.calendar模块：

构造：calendar(year,[w=2,l=1,c=6])：返回year年的完成日历信息对象 类型：str

和闰年相关的一些函数：

isleap(year)：判断是否是闰年

leapdays(y1,y2)：得到[y1,y2)之间的闰年个数

和month相关的一些函数：

month(year,month)：返回year年month月的日历信息对象 类型：str

monthrange(year,month)：返回一个元祖独享，有两个元素；

第一个元素存储了当月1日在当前周的第几天，

第二个元素存储了month月一共有多少量

timegm(tp)：将时间元祖对象转换为时间戳对象返回

--------------------------------------------------------------------------------------

day16回顾：

正则表达式：regex

作用：过滤字符串数据的

python将正则纳入到re模块中(封装)，我们只需要导入模块，就可以使用其中函数...

1.正则(re模块)中最常用的4个函数：

match(regex,string,[flags=0])：从头开始匹配字符串数据，如果头就不匹配，直接返回None值；如果匹配了需要的数据内容，那么就返回一个match对象；

search(regex,string,[flags=0])：从头开始匹配字符串数据，头不匹配没有关系，继续往后尝试匹配，一旦匹配成功返回一个match对象，就算之后还有匹配的子数据(子串)，直接无视了

findall(regex,string,[flags=0])：从头开始匹配字符串数据，将所有匹配成功的数据存入到列表中返回； 如果没有匹配成功，那么就返回一个空列表

compile(regex,[flags=0])：返回Pattern对象，然后继续调用match或者search或者findall完成需求；注意：此时的match、search、findall只需要传递一个参数即可

2.正则中的元字符：

1).匹配单个字符(数字、英文、其它)：

符号：

[]、

[0123456789]、[0-9]、\d、\D

[a-z]、[A-Z]、[a-zA-Z]

[0-9a-zA-Z\_]、\w、\W、点(.)

2).匹配锚字符(边界字符)：

多行模式下有效：re.M

^：匹配头

$：匹配尾

多行模式下无效：re.M

\A：匹配头

\Z：匹配尾

\b和\B：

3).匹配多个字符：

分类：

①.模糊匹配：

符号：

?、+、\*

②.精确匹配：

符号：n

变量x,y,n

n{x} n{x,} n{x,y}

【注意】：

正则表达式在设计的时候，就满足贪婪匹配的特点(尽可能多个返回数据)，

如果有需要了，我们可以人为的去取消贪婪行为，只需再添加一个?即可

3.正则中的切割和替换

替换：

sub(regex,repl,string,[count],[flags])：...

subn(regex,repl,string,[count],[flags])：...

切割：

split(regex,string)：返回列表对象

4.扩展

函数：

finditer(regex,string,[flags=0])：封装一个个的match对象，以iterator的形式返回

复习了迭代器相关的注意点：

1).不能和len()使用

2).内部数据信息只能使用一次，不可逆；但是可以转换为其它的容器对象(譬如：list...)

就可以不断使用

------------------------------------------------------------------------------------------------------------

day17回顾：

1.高阶函数：

map(fn,lsd1,[lsd2])：返回一个map对象(**惰性、属于迭代器对象**)

reduce(fn,lsd)：

filter(fn,lsd)：根据返回的布尔值来决定是否需要保留到filter对象(**惰性、迭代器对象**)中

补充：

sorted(lsd,key,reverse)：

注意：key和内置函数或者匿名函数配合使用

2.单元测试：

分类:

1).模块中的函数

2).类中的函数

以上两类的相同点：

1).导入unittest模块，自定义类继承unittest.TestCase类

2).选择性的重写setUp()和tearDown()函数

3).修饰需要被测试的函数：命名规则满足 --> test\_函数名()

4).在函数内部通过self调用assertequal()，

形参有3个内容：

①.显示的调用需要被测试的函数(传入实际参数)

②.断言结果

③.错误提示信息

5).通过unittest启动内部的main()函数考到效果(红色、绿条)

其不同点：

对于类中的函数测试：

1).在test\_函数名()此函数的内部，需要先进行一个实例化操作

3).py2和py3的区别：

。。。

-------------------------------------------------------------------------------------------