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

< Doki Doki Forever >

1. python的代码注意缩进行 Just Monika

2. # 单行注释

3. """多行注释"""

4. 注意大小写

5. tap 键，整体右移

6. shift+tab键，整体左移

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

# >> Python基本概念**（ 第一课 ）**

1.python 的源程序就是一个特殊格式的文本文件，可以使用任意文本编辑软件做

python 开发

2.python 程序的文件扩展名通常是 .py

3.print( )

- print 是 python 中我们学习的第一个函数

- 可以把" " 内部的内容，输出到屏幕上

例子：print(" Hello World By Monika ")

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

>> 算数运算符（ 第二课 ）

+ 加

- 减

\* 乘

/ 除

// 取整数 9//2 = 4

% 取余数 9%2 = 1

\*\* 幂 2\*\*3=8

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

>> 变量的基本使用（ 第三课 ）

变量名 = 值

例子：

a = 32

print(a)

输出：32

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

>>python中的变量类型（第四课）

- 变量类型可以分为两大类

—— 数字型

1.整形（ int )

2.浮点数( float )

说明白点就是像1.2这种带个小数点的

3.布尔型 ( Bool )

真 true , 非零即 true

假 false , 0

4.复数型( complex )

主要用于科学计算，例如：平面场问题，波动问题，电感电容等问题

—— 非数字型

1.字符串

2.列表

3.元组

4.字典

—— 在python 2.x 中，整数根据保存数值的长度还分为

int 整数

long 长整数·

区别就是数字长点和短点罢了

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

type - 可以查看一个变量的类型

例子：print (type (name))

也就是说，type（要检测的值）

print 用于打印

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

>>python中的字符串拼接（第五课）

- 可以用 + 来完成字符串的拼接

- Js 中会进行隐式转换，但 Py 没有

- 所以 number + string 进行字符串的拼接会报错

另类用法：

x = -

print( x \* 3 )

输出：---

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

>>python中变量的输入（第六课）

在 py 中，如果要获取用户在键盘上输入的信息，用 input 函数

总体来说，input( ) 很像 prompt( ) 的样子

input("输入：")

输入：>? wh

'wh'

input 所得到的结果都是字符串

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

>>python中的类型转换函数（第七课）

int (x) 将 x 转为整数

float(x) 将 x 转为浮点数

还能转换类型：string 转为 Number

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

## >>买苹果练习

apple = input("你他妈苹果多少元一斤")

number = input("你他妈买了多少斤")

print(float(number)\*int(apple) )

简称：

print(float(input("你他妈苹果多少元一斤"))\*int(input("你他妈买了多少斤")))

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

>>⁮变量的格式化输出（第八课）

格式化字符 含义

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

%s 字符串

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

%d 有符号十进制整数，%06d表示输出的整数显示位数，不足的地方0补

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

%f 浮点数，%.02f 表示小数点后只显示两位

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

%% 输出 %

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

例子：

print("我的名字叫 %s,请多多 %s关照" %(name,age))

很离谱，像传参一样

倘若传多了，会报错

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

%s

1.字符串使用的

print("我的名字叫%s,初来乍到"%input("你叫啥"))

很离谱，像传参一样

倘若传多了，会报错

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

%d

1.据推断，%03d,表示输出三位数，多了覆盖，少了以0代替

2.0改为任何值都会不见数字，只会留空格

3.%011，则会出现11位数

4.%d 可以理解为取整

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

%f %.02f

1.据推断，第二个点不能去除，0的后面表示小数点后几位

2.零好像可以省略，多出小数点的则四舍五入

3.小数点好像在限制着什么

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

%%

例子：

print("数据比例是%.02f%%"%float(input("数据比例是")))

1.这玩意居然也能嵌套

2.好像不能单独使用

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

>>变量的命名（第九课）

1.标识符

- 表示符就是程序员定义的变量名，函数名

- 名字需要有见名知意的效果

2.标识符命名规范

- 标识符可以由字母，下划线和数字组成

- 不能以数字开头

- 不能于关键字重名

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

3.关键字

- 关键字就是已经在python中使用的标识符

- 关键字具有特殊的功能与含义

- 不允许定义和关键字相同名字的标识符

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

>>判断（ if ）语句（第十课）

- python 是依靠缩进行来知晓 if 的掌控范围

而有些语言使用花括号的

1.语句：

if a<10 :

print("小于十")

else :

print("大于十")

2.梳理：

if a > 0 :

print("先还信用卡的")

#########################################

if a - 100 < 0 :

print("期盼着下个月")

else :

print("芜湖")

#########################################

else :

print("吃土吧")

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

输入0，外面的 if 执行不成功，略过内部 if 直接 else

倘若外部成功，则执行内部 if 判断，输出内部结果

注意缩进

格式：

if 判断的条件：

要执行的代码

4个空格

TAB 与 空格不要混用

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

3.太有趣了

a = 66

b = 66

if b > a:

print("b is greater than a")

elif a == b:

print("a and b are equal")

elif = else if

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

3.例子：

a = int(input("你寄吧多少岁[流汗黄豆！]"))

if a < 18 :

print("玩尼玛")

elif a > 100:

print("牛逼")

else:

print("让我康康！")

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

>>判断（ if ）语句的运算符（第十一课）

and/or/not 三种

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

1.and 相当于 [&&（JS）]

- 并且，一个条件成立并且另个条件也成立

语法：条件一 and 条件二

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

2.or 相当于 [||（JS）]

- 或者，条件中一个成功即可

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

3.not 相当于 [ ！（JS）]

- 当一个条件不成立我们才需要去做的事情

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

>>判断（ if ）语句的高级用法（第十二课）

elif [else if ( JS )]

1.使用 if 可以判断条件

2.else 可以处理条件不确定的情况

3.在添加一些条件，条件不同，需要执行的代码也不同时，用elif

格式：

if 条件1：

代码

elif 条件2：

代码

else:

以上都不满足时用的

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

>>判断（ if ）语句的高级用法（第十三课）

if 的嵌套使用

与 elif 的去别是拥有了前后关系

格式：

if 语句：

文本

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

if 语句：

文本

else:

文本

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

else：

文本

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

>>判断（ if ）语句的格式（第十四课）

if ( ( 判断 ）

or ( 判断 ) )：

例子：

if ((player == 1 and computer == 2)

or(player == 2 and computer == 3)

or(player == 3 and computer == 1)):

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

>>随机数的使用（第十五课）

- 在 python 中，要使用随机数，首先要导入随机数的模块 ==“工具包”

- 导入模块后，可以在模块名称后面敲一个.然后按Tab键，会提示所有函数

import random

- random.randint(a,b)

返回 a 到 b 之间的整数，包含 a 与 b

- random.randint(下限,上限)

random.randint(a <= n >=b)

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

例子：一个简单的猜拳小游戏

player = int(input("你要出的拳， 石头（1） 剪刀（2） 布（3）"))

import random

computer = random.randint(1,3)

if ((player == 1 and computer == 2)

or(player == 2 and computer == 3)

or(player == 3 and computer == 1)):

print("我赢了")

elif player == computer:

print("牛皮")

else:

print("你输了")

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

>> while循环（第十六课）

顺序 -- 从上而下，顺序执行代码

分支 -- 根据条件判断，决定执行代码的分支

循环 -- 让特定代码重复执行

while 循环

break 与 continue

语法：

while 条件（判断 计数器 是否达到 目标次数）：

条件满足时的代码

处理时间（计数器 + 1 ）

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

例子：

a = 0

while a<10 :

print(a)

a = a+1

注：python中好像没有 a++

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

i = 0

result = 0

while i <= 100: # 0 - 100 列出来

i = i+1 # i的自增，防止死循环

if not i % 2 : #当 i % 2 的时候 列出来

result += i # 结果值的不断加i，形成偶数的相加

print(result) #输出结果

需求，0 - 100 的偶数相加

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

break 与 continue

1.break

- 某个条件满足时，退出循环，不执行后续重复代码

2.continue

- 某一条件满足时，不执行后续重复的代码

- 注意别死循环了

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

>>九九乘法表 （第十七课）

1. print 扩展

- 在默认情况下，print 函数输出内容后，会在末尾自动增加换行

- 在 print 函数后面增加 [ , end="" ]

- "" 可以指定 print 函数输出内容之后，希望继续显示的内容

语法：

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

- 向控制台输出内容结束之后，不会换行

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

2.代码：

row = 1

while row <= 9 :

col = 1

while col <= row:

print("%d\*%d=%d \t"%(col,row,col\*row),end="")

col += 1

print("")

row += 1

总结：在每次循环下重新定义，从一开始，列数为一

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

>>转义字符 （第十八课）

- \t 制表符，协助输出文本时垂直方向保持对齐

- \n 换行符

- \\ 反斜杠符号

- \' 单引号

- \" 双引号

- \r 回车

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

>>Python函数 （第十九课）

- 就是把具有独立功能的代码块，组织成一个小模块，用的时候调用

- 正常的形参与传参

- def 是 define 的缩写

- 可以由字母，数字，下划线组成

- 不能以数字开头

- 不能与关键字重名

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

创建函数 def number( ):

内容 Python

调用函数 number( )

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

创建函数 function number( ){

内容 JavaScript

}

调用函数 number( )

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

- 函数的引用方法

import main -- 导入工具包，引用文件

main.number( ) -- 文件的引用函数

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

>>Python函数的注释 （第二十课）

def aaa( ):

"""输出666"""

print("666")

aaa( )

ctrl+Q 即可

- 保留俩空行

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

>>函数的返回值 （第二十一课）

- 注意：return 表示返回后续的代码都不会执行

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

>>Python中模块的概念 （第二十二课）

- 模块就好比工具包，要想使用就要导入 import 这个模块

- 每一个以扩展名 py 结尾的 Python 源代码文件就是一个模块

- 在模块中定义的全局变量，函数，都是模块能够提供给外界直接使用的工具

- 注意！以数字开头的文件无法导入

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

>>增删改查 by Python（第二十三课）

- index : 用于取索引 indexOf -- JS

- append: 追加内容于数组 appendChild -- JS

- insert：在中间插入 insertBefore -- JS

- extend: 收服另一批数组小弟 concat( ) -- JS

- remove: 移除，要写名字,第一次的数据 removeChild -- JS

- pop：移除，默认最后，特定需要索引 pop -- JS

- clear：移除，移除自己 = null -- JS

- del: 从内存中删除 delete -- JS

- len: 用于返回数组长度 length -- JS

- count: 用于查找出现次数 indexOf -- JS

- sort: 排序，首字母与大小，有两种方法 sort -- JS

- reverse：逆序 reverse -- JS

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

1.索引取值

name = [Monika,Sayori,Yuri,Natsuki]

print(name[0])

结果：Monika

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

2.取索引

name = [Monika,Sayori,Yuri,Natsuki]

print(name.index("Natsuki"))

结果：3

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

3.索引改变

name = ["Monika","Sayori","Natsuki","Yuri"]

name[0] = "莫妮卡"

结果：莫妮卡

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

4.增加 -- 追尾

name = ["Monika","Sayori","Natsuki","Yuri"]

name.append("mc")

结果："Monika","Sayori","Natsuki","Yuri","mc"

增加 -- 索引插入

name.insert(1,"mc")

结果："Monika","mc","Sayori","Natsuki","Yuri"

增加 -- 收服

otherName = ["盗梦","之末"]

name.extend(otherName)

结果："Monika","Sayori","Natsuki","Yuri","盗梦","之末"

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

5.删除 -- 定向

name = ["Monika","Sayori","Natsuki","Yuri"]

name.remove("sayori")

结果："Monika","Natsuki","Yuri"

删除 -- 索引抹杀

name.del[0]

结果："Sayori","Natsuki","Yuri"

删除 -- 索引定向与末端

name.pop( )

结果："Monika","Sayori","Natsuki"

name.pop[2]

结果："Monika","Sayori","Yuri"

删除 -- 终结

name.clear

结果：

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

>>数组方法（第二十四课）

- len: 用于返回数组长度 length -- JS

- count: 用于查找出现次数 indexOf -- JS

- sort: 排序，首字母与大小，有两种方法 sort -- JS

- reverse：逆序 reverse -- JS

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

长度 -- 索引

name = ["Monika","Sayori","Natsuki","Yuri"]

list\_len = len(name)

结果：4

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

查找

name = ["Monika","Sayori","Natsuki","Yuri","Sayori","Sayori","Sayori"]

list\_cou = name.count("Sayori")

结果：4

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

排序 -- 升序

name = ["Monika","Sayori","Natsuki","Yuri"]

name.sort( )

结果：'Monika', 'Natsuki', 'Sayori', 'Yuri'

排序 -- 降序

number = [4,7,8,9,10,22,5,4]

number.sort(reverse=True)

结果：22, 10, 9, 8, 7, 5, 4, 4

排序 -- =逆序

name = ["Monika","Sayori","Natsuki","Yuri"]

name.reverse( )

结果：'Yuri', 'Natsuki', 'Sayori', 'Monika'

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

>>函数与方法的概念（第二十五课）

1.关键字

- 关键字是 Python 内置的，有特殊意义的标识符

- 关键字后面不需要使用括号

例子：

import Keyword

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

2.函数

- 函数封装了功能，可以直接使用

例子：

函数名（参数）

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

3.方法

- 方法与函数相似，同样是封装了独立的功能

- 方法需要通过对象调用，表示要对这个对象使用的操作

例子：

对象.方法名（参数）

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

4.变量

- 在变量名后面输入.然后选择针对这个变量名使用的操作

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

>>列表的循环遍历（第二十六课）

1.概念

- 遍历就是从头到为依次从列表中获取数据

- 在循环体内部针对每一个元素，执行相同的操作

- 使用 For 循环就能实现迭代遍历

- Python中列表中可以存储不同数据

例子：

name = ["Monika","Sayori","Natsuki","Yuri"]

for my\_wife in name:

print("Doki Doki Literary Club By %s" %my\_wife)

应用场景：

- 列表存储相同类型的数据

- 通过迭代遍历，在循环体内部，针对列表中的每一项元素，执行相同的操作

- 有点像 ForEach -- JS

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

>>元组（第二十七课）

- 与列表相似，不同于元素不能修改

- 元组表示多个元素组成的序列

- 存储一组信息，用逗号分割

- 元组用（）定义

- 元组的索引从0开始

- 元组中只包含一个元素时，需要在元素后面添加，

- 个人感觉元组像定义标准，这也是于数组的大不同之处

- 元组侧重于储存一个固定信息

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

>>元组的方法（第二十八课）

- count

一个内容出现的次数

例子：

name = ("Monika","Sayori","Natsuki","Yuri","Monika","Monika")

print(name.count("Monika"))

输出：3

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

- index

一个内容出现的索引

例子：

name = ("Monika","Sayori","Natsuki","Yuri","Monika","Monika")

print(name.index("Monika"))

输出：0

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

## >>循环遍历（扩展）

- for 循环可以遍历所有非数字型类型的变量：列表，元组，字典，字符串

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

## >>元组应用场景（扩展）

- 元组的数据不可被更改

- 所以函数的参数和返回值，一个函数可以接受多个参数并返回多个数据

- 可以理解为一个包裹

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

>>元组与数组之间的转换（第二十九课）

- 使用 list 函数可以把元组转换成列表

- list( 元组 )

- 使用 tuple 函数可以把列表转换成元组

- tuple( 列表 )

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

>>字典的定义（第三十课）

- dictionary （字典），是除列表之外最灵活的数据类型

- 字典同样可以存储多个数据

- 通常用来描述一个物体的相关信息

- 和列表的区别

- 列表是有序的对象集合

- 字典是无序的对象集合

- 字典用 { } 定义

- 字典用 键值对 数据存储，键值对之间使用 ， 分割

- 键 key 是索引

- 键 value 是数据

- 键和值之间使用：分割

- 键必须是唯一的

- 值可以取任何数据类型，但键只能使用字符串，数组或元组

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

>>字典的定义格式（第三十一课）

- 感觉这里像 JS 中的对象 Object

- 注意加逗号

格式：

xiaoming = {"name":"小明",

"age":18,

"gender":True,

"height":1.75}

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

>>字典的增删改查（第三十一课）

- 取值

例子：

xiaoming = {"name":"小明",

"age":18,

"gender":True,

"height":1.75}

print(xiaoming["age"])

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

- 增加/修改

xiaoming["age"] = 40 xiaoming.age = 40 -- JS

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

- 删除

del(xiaoming["age"]) delete

xiaoming.pop("age")

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

>>字典的常用操作（第三十二课）

- update:注意相同东西会覆盖 concat( ) -- JS

- 查长度

xiaoming = {"name":"小明",

"age":18,

"gender":True,

"height":1.75}

print(len(xiaoming))

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

- 合并字典

monika = {"sayori":666,

"yuri":777,

"natsuki":888,}

print(xiaoming.update(monika))

输出：

xiaoming = {"name":"小明",

"age":18,

"gender":True,

"height":1.75

"sayori":666,

"yuri":777,

"natsuki":888}

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

- 清空字典

xiaoming.clear( )

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

- 字典的遍历

xiaoming = {"name":"小明",

"age":18,

"gender":True,

"height":1.75}

for k in xiaoming:

print("%s - %s" %(k,xiaoming[k]))

- 单个输出 k 值，输出的是 键

- 后面是遍历值

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

>>字典的常用操作（第三十三课）

- 使用一对双引号 "" ，或''定义一个字符串

- 大多编程语言使用""来定义字符串

- len : 获取字符串的长度

- .count( 字符串 ) ： 出现次数，也就是查询一个东西出现次数 search( ) -- JS

- .index( 字符串 ) ： 获得小字符串第一次出现的次数,没有报错 indexOf( ) -- JS

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

例子：

string = "Hello Python By World"

print(len(string ))

输出：21

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

例子：

string = "Hello Python By World"

print( string.count("l") )

输出：3

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

例子：

string = "Hello Python By World"

print( string.index("l") )

输出：2

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

>>字符串常用方法和分类（第三十四课）

### 1.判断

- string.isspace( )

- 如果字符串中有空格，则返回 true

- string.isalnum( )

- 如果字符串中有字母或数字，则返回 true

- string.isalpha( )

- 如果字符串中都是字母，则返回 true

- string.isdecimal( )

- 只包括数字则返回 true ,全角数字

- string.isdigit( )

- 只包括数字则返回 true ,全角数字，（1），\u00b2

- string.isnumeric( )

- 只包括数字则返回 true ,全角数字,汉字数字

- string.istitle( )

- 如果字符串是标题化的，每个首字母大写，则返回 true

- string.islower( )

- 如果字符串包含一个区分大小写的字符，并且小写，则 true

- string.isupper( )

- 如果字符串包含一个区分大小写的字符，并且大写，则 true

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

### 2.查找与替换

- string.startswith(str)

- 检查字符串是否以 str 开头，是则返回 true

- string.endswith(str)

- 检查字符串是否以 str 结尾，是则返回 true

- string.find(str,start = 0,end = len(string))

- 检查 str 是否包含在字符串中，如果 start 和 end 指定范围，则检查是否在指定范围

内如果有则返回开始的索引值，否则返回-1

- string.rfind(str,start = 0,end = len(string))

- 类似 find() 从右边开始查找

- string.index(str,start = 0,end = len(string))

- 和 find( )方法类似，但str不在会报错

- string.rindex(str,start = 0,end = len(string))

- 类似 index( ) 但从右边开始

- string.replace(old\_str,new\_str,num = string.count(old))

- 把字符串中旧字符串替换成新字符串，如果 num 指定，则不超过 num 次

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

### 3.大小写转换

- string.capitalize( )

- 把字符串的第一个字符大写

- string.title( )

- 字符串每个单词首字母大写

- string.lower( )

- 转换大写为小写

- string.upper( )

- 转换小写为大写

- string.swapcase( )

- 反转字符串的大小写

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

### 4.文本对齐

- string.ljust(width)

- 原字符串左对齐，并使用空格填充至长度 width 的新字符串

- string.rjust(width)

- 原字符串右对齐，并使用空格填充至长度 width 的新字符串

- string.center(width)

- 原字符串居中，并使用空格填充至长度 width 的新字符串

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

### 5.去除空白字符

- string.lstrip()

- 截掉字符串左边（开始）的空白字符

- string.rstrip()

- 截掉字符串右边（末尾）的空白字符

- string.strip()

- 截掉字符串左右两边的空白字符

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

### 6.拆分和连接

- string.partition(str)

- 把字符串分成一个3元素的元组（str前面，str，str后面）

- string.rpartition(str)

- 类似于partition函数，不过是从右边开始查找

- string.split(str = "",num)

- 以str为分割符切片string，如果num有指定值，则仅分割 num+1 个子字符串

str默认包含 \r,\t,\n 和空格

- string.splittlines()

- 按照行（\r,\n,\r\n）分隔，返回一个包含各行作为元素的列表

- string.join(seq)

- 以string作为分割符，将seq中所有的元素，合并为一个新的字符串

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

>>字符串常用方法的演练（第三十五课）

- string.isspace( )

- 如果 string 中只包含空格，则返回 Ture

- 只有空格或制表符之类的才执行，不能空的

例子：

string = " "

if string.isspace():

print("666")

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

- string.isdecimal()

- 只包括数字则返回 true ,全角数字

例子：

string = "6"

if string.isdecimal():

print("666")

。。。。。。。。。。。。。。。。

- string.isdigit()

- 只包括数字则返回 true ,全角数字，（1），\u00b2

- 最后那个说白了就是 Unicode 编码

例子：

string = "(5)"

if string.isdecimal():

print("666")

。。。。。。。。。。。。。。。。

- string.isnumeric()

- 只包括数字则返回 true ,全角数字,汉字数字

- 繁体数字也行

例子：

string = "一千零一"

if string.isdecimal():

print("666")

- 以上几个方法都不能判断小数

- 都可以对数字为true

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

- string.startswith(start)

- 检查字符串是否以 str 开头，是则返回 true，注意大小写

例子：

string = "start"

if string.startswith("start"):

print("666")

。。。。。。。。。。。。。。。。

- string.endswith(start)

- 检查字符串是否以 str 结尾，是则返回 true，注意大小写

例子：

string = "start"

if string.endswith("start"):

print("666")

。。。。。。。。。。。。。。。。

- string.find(str,start = 0,end = len(string))

- 检查 str 是否包含在字符串中，如果 start 和 end 指定范围，则检查是否在指定范围

内如果有则返回开始的索引值，否则返回-1

- index方法没找到会报错，而find方法会返回 -1

例子：

string = "sxdscdscdscsstart"

a = string.find("start")

print(a)

返回：12

。。。。。。。。。。。。。。。。

- string.replace(old\_str,new\_str,num = string.count(old))

- 把字符串中旧字符串替换成新字符串，如果 num 指定，则不超过 num 次

- 不会替换原数组

例子：

string = "sxdscdscdscsstart"

a = string.replace("start","666")

print(a)

输出：

sxdscdscdscs666

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

- string.ljust(width)

- 原字符串左对齐，并使用空格填充至长度 width 的新字符串

- string.rjust(width)

- 原字符串右对齐，并使用空格填充至长度 width 的新字符串

- string.center(width)

- 原字符串居中，并使用空格填充至长度 width 的新字符串

- 如果感觉不行，就定义fillcher,

- 例子 ：string\_len.center(40," ")

width -- 宽度，输入数字

例子：

string = ["swdsdada",

"adadadasds",

"nadiu",

"hduashuidbnashbduhaudnajndjn",

"dakndaodnaondna",

"adadad",

"1"]

for string\_len in string:

print(string\_len.center(40))

输出：

swdsdada

adadadasds

nadiu

hduashuidbnashbduhaudnajndjn

dakndaodnaondna

adadad

1

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

- string.lstrip()

- 截掉字符串左边（开始）的空白字符

- string.rstrip()

- 截掉字符串右边（末尾）的空白字符

- string.strip()

- 截掉字符串左右两边的空白字符

- 注意可以函数连用（链式编程）

例子：

string =" kkk k 666 "

a = string.r/l/strip()

print(a)

输出：

" kkk k 666"

"kkk k 666 "

"kkk k 666"

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

string.partition(str)

- 把字符串分成一个3元素的元组（str前面，str，str后面）

例子：

string ="ganmstrhiahsiuha"

s = string.partition("str")

print(s)

输出：

('ganm', 'str', 'hiahsiuha')

- 可以理解为孤立（好似只能孤立一个）

string.rpartition(str)

- 类似于partition函数，不过是从右边开始查找

。。。。。。。。。。。。。。。。

string.split(str = "",num)

- 以str为分割符切片string，如果num有指定值，则仅分割 num+1 个子字符串

str默认包含 \r,\t,\n 和空格

例子：

string ="ga,nmst,rhi,ahsi,uha"

s = string.split(",")

print(s)

输出：

['ga', 'nmst', 'rhi', 'ahsi', 'uha']

。。。。。。。。。。。。。。。。

string.splitlines( )

- 按照行（\r,\n,\r\n）分隔，返回一个包含各行作为元素的列表

。。。。。。。。。。。。。。。。

- string.join(seq)

- 以string作为分割符，将seq中所有的元素，合并为一个新的字符串

- 相比较于JS中的数组合并，Py中的更像是在列表中间增加分割符

例子：

lest = "monika\t sayori\t natsuki\t yuri"

lest\_by = lest.split()

result = " ".join(lest\_by)

print(result)

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

>>切片概念和语法以及倒序索引（第三十六课）

- 左开右合

- 分为正序与倒序

- 注意分割字符是什么 ：

- 语法:

字符串[开始所以，结束索引，步长]

开始索引：开始的位置

结束索引：结束的位置

步长：从开始后每次跨域的次数，超过结束索引的话会默认只看开始的，后二不看

例子：

name = "printbymonika"

print(name[0:6:7])

输出：

p

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

例子：

name = "printbymonika"

print(name[0:-1])

输出：

printbymonik

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

- 字符串的逆序

- 自己写的

name = "0,1,2,3,4,5,6,7,8,9"

name\_list = name.split(",")

name\_list.reverse()

print(name\_list)

输出：['9', '8', '7', '6', '5', '4', '3', '2', '1', '0']

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

- 老师写的

name = "0,1,2,3,4,5,6,7,8,9"

print( name[-1::-1] )

输出：

9876543210

总结：前路茫茫，浊眼难辨，岂能心浮气躁？

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

>>公共方法（第三十七课）

1.Python 包含了以下内置函数

- len(item) ：计算容器中元素个数

- del(item) ：删除变量 del有两种方式

- max(item) ：返回容器中元素最大值 如果是字典只针对key比较

- min(item) ：返回容器中元素最小值 如果是字典只针对key比较

- cmp(item1,item2) ：比较两个值，-1小于/0相等/1大于 Py3.x移除cmp函数

注意：

- 字符串比较符合以下规则："0" < "A" < "a"

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

例子：

num = {"a":"z",

"b":"y",

"c":""x}

print(max(num))

输出：c

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

>>公共方法 - 切片（第三十八课）

- 列表切片语法

[0,1,2,3,4,5][1:3]

输出：1,2

- 元组切片语法

（0,1,2,3,4,5）[1:3]

输出：（1,2）

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

>>公共方法 - 算数运算符（第三十九课）

1.+ 合并 串，列，元

2.\* 重复 串，列，元

3.in 元素是否存在 串，列，元，典

4.not in 元素是否不存在 串，列，元，典

5.> >= == < <= 元素比较 串，列，元

1.加号可以进行字符串拼接，也能数字相加

2.乘号与上文同理

3.有则Ture ， 判断字典是只能判断Key

4.有则False ， 判断字典是只能判断Key

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

>>完整的 for 循环（第四十课）

- 语法：

for 变量 in 集合：

循环体代码

else:

没有通过break退出循环，循环结束后，会执行的代码

- 可以理解为在循环完后做的事情

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

- 查找人物练习

students = [{

"name":"monika",

"age":"18",

"gender":"woman",

"LikeColor":"green",

"Birthday":"22/9"

},

{

"name":"sayori",

"age":"18",

"gender":"woman"

}]

def find\_name(list):

a = input("请输入DDLC人物名字")

for b in list:

if b["name"] == a:

print(b)

break

else:

print("没找到,请确认输入为DDLC名字")

find\_name(students)

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

>>名片管理系统（第四十一课）

- 成功在未看视频的前提下完成，尽管会有许多意料外的Bug，但大体完成了

- 2121年9月20日17时45分

- by盗梦

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

## - 进入正题 >>

1.pass

- 可以理解为填补坑洞，例如判断依据不确定时可以使用

2.int( )与folat( )

- 当用户输入其他数字时会报错

3.#TODO

- Pycharm的东西，与Py无关

- 高亮提示注释

- 左下角有提示

- #TODO 内容

4.return

- 返回一个函数的执行结果

- 下方的代码不会执行

- 如果return后面没有任何内容，表示会返回到调用函数的位置

- 并且不返回任何的结果

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

>>关于引用的理解（第四十二课）

1.变量的引用

- 变量与数据是保存在内存中的

- 在Python中函数的参数传递以及返回值都是靠引用传递的

2.引用的概念

- 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一个地址

- 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

- 变量不在对之前的数据引用

- 变量改为对新赋值的数据的引用

3.函数调用

- 本质上传递的是实参保存数据的引用，而不是实参保存的数据

- 注意：如果函数有返回值，但是没有定义变量接受

- 程序不会报错，但是无法获得返回结果

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

>>可变与不可变类型（第四十三课）

- 不可变类型：内存中的数据不允许被修改

1.数字类型

2.字符串

3.元组

- 可变类型：内存中的数据可以修改（地址不变，内容可改）

1.列表

2.字典

**- hash(哈希)**

- 在Py中设置字典的键值对时，会首先对Key进行hash已决定如何在内存中

保存字典的数据，以方便后续对字典的操作：增，删，改，查

1.哈希是一种算法，其作用就是提取数据的特征码

2.相同的内容得到相同的结果

3.哈希函数只可以接受不可变类型

4.这也就是为什么字典无法将 列表 与 字典 作为 键 的原因

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

>>变量的生命周期（第四十四课）

1.所谓生命周期就是变量从被创建到被系统回收的过程

2.局部变量在函数执行时才会被创建

3.函数执行完成后，局部变量会被系统回收

4.局部变量在生命周期内，可以用来存储函数内部临时使用到的数据

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

全局变量（扩展）

- 在函数内部，可以通过全局变量的引用获取对应的数据

- 但是，不允许直接修改全局变量的引用 -- 使用赋值语句修改全局变量的值

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

>>global关键字修改全局变量（第四十五课）

- 希望修改全局变量的值，**使用 global 声明一下变量即可**

例子：

num = 0

def number():

global num

num = 8

print(num)

number( )

输出：8

- 注意，在开发时应该把模块中的所有全局变量

- 定义在所有函数上方，就可以保证所有函数

- 都能够正常的访问到每一个全局变量了

- 说直白点就是把定义的变量往上放

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

>>函数参数和返回值的作用（第四十六课）

1.函数根据有没有参数以及有没有返回值，可以相互组合，一共有四种组合形式

- 无参数，无返回值

- 无参数，有返回值

- 有参数，无返回值

- 有参数，有返回值

定义函数时，是否接受参数，或者是否返回结果，是根据实际的功能需求决定的!

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

>>函数的返回值（第四十七课）

1.总体来说就是返回一个数值，当打印并调用时，返回的数值能够看见

- return 返回的东西

2.代码的妙妙蛙写法：

例子：

def measure():

print("开始测量温度")

temp = 39

print("测量温度完毕")

shidu = 50

return temp,shidu

gl\_temp,gl\_wetness = measure( )

print(gl\_temp)

print(gl\_wetness)

- 如果函数的返回值是元组，同时希望单独处理元组中的元素

- 可以使用多个变量，一次接受函数的返回结果

- 注意：使用多个变量接受结果时变量的个数应该和元组的数量保持一致

多了会报错

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

## 面试题 - 交换两个数字

- 有两个整数变量 a = 6 , b = 100

- 不使用其他变量，交换两个变量的值

**本质上就是利用数学思想把数字导出来，不计过程，只看结果**

解法：

a = 6

b = 100

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

a = a + b #106

b = a - b #6

a = a - b #100

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

Py 专享

a,b = b,a

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

但是如果参数是可变的话，可以使用方法修改

例子：

def demo(num\_list):

num\_list.extend([1,2,3])

print(num\_list)

gl\_list = [6,7,8]

demo(gl\_list)

print(gl\_list)

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

>>函数的参数（第四十八课）

- 在 Py 中，列表调用 += 本质上是在执行列表变量的 extend 方法，不会修改变量的引用

但是不可变类型就会赋值

例子：

def demo(num):

print("函数开始")

num += num

print(num)

print("函数完成")

gl\_num = 9

demo(gl\_num)

print(gl\_num)

结果：

函数开始

18

函数结束

9

因为有了赋值

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

>>缺省参数（第四十九课）

1.定义函数时，可以给某个参数指定一个默认值，具有默认值的参数就叫缺省参数

2.调用函数时，如果没有传入缺省参数的值，则在函数内部使用定义函数时指定的参数默认值

3.函数的缺省参数，将常见的值设置为参数的缺省值，从而简化函数的调用

4.提示：在指定缺省参数的默认值时，应该使用最常见的值作为默认值

5.例如：对列表的排序方法

- 回顾内容

1.sort方法

- 作用：对列表进行升序排列（默认）

- sort( ) 升序

- sort( reverse = True ) 降序

**个人理解：**

**- 实则就是给参数传值，然后调用时就可以省略**

例子：

def print\_info(name,gender = True):

gender\_text = "男生"

if not gender:

gender\_text = "女生"

print("%s 是 %s"%(name,gender\_text))

print\_info("有地将臣")

print\_info("常陆茉子",False)

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

缺省参数注意事项

1.缺省参数，需要使用最长见的值作为默认值

2.如果一个参数的值不能确定，则不应该设置默认值，具体的数值在调用函数时，由外界传递

3.必须保证带有默认值的缺省参数在参数列表末尾

4.所以，以下定义是错的

def print\_info( name,gender = True,title )

5.在调用带有多个缺省参数的函数

- 在调用函数时，如果有多个缺省参数，需要指定参数名，这样解释器才能知道对应关系

例子：

print\_info("常陆茉子",18,gender= False,)

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

>>多值参数（第五十课）

- 能学到这里来感叹万千了属于是，DDLC Forever~~~

————2121/10/5 15:02

定义支持多值参数的函数

- 有时可能需要一个函数能够处理的参数个数是不确定的，这个时候，就可以使用多值参数

**- Py 中有两种多值参数**

**- 参数名前增加一个 \* 可以接受元组**

**- 参数名前增加两个 \* 可以接受字典**

- 一般在给多值参数命名时，习惯使用以下两个名字

- \*args -- 存放元组参数，前面有一个 \*

- \*\*kwargs --存放字典参数，前面有两个 \*\*

- args 是 arguments 的缩写，有变量的含义

- kw 是keyworld的缩写，keyworld可以记忆键值对参数

例子：

def demo(num,\*args,\*\*kwargs):

print(num)

print(args)

print(kwargs)

demo(1,2,3,4,5,name = "小明",age = 18,gender = True)

输出：

1

(2, 3, 4, 5)

{'name': '小明', 'age': 18, 'gender': True}

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

>>多值参数 - 元组与字典的拆包（第五十一课）

- 在调用带有多值参数的函数时，如果希望：

- 将一个元组变量,直接传递给args

- 将一个字典变量，直接传递给kwargs

- 就可以使用拆包，简化参数的传递，拆包的方式是：

- 在元组变量前，增加一个 \*

- 在字典变量前，增加两个 \*

个人感想：

说白了就是将元组与字典的变量作为实参传给函数

例子：

def question(\*a,\*\*b):

代码

a = (2, 3, 4, 5)

b = {'name': '小明', 'age': 18, 'gender': True}

demo（\*a,\*\*b）

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

>>递归（第五十二课）

1.特点：一个函数内部调用自己

- 函数内部可以调用其他函数，当然在函数内部也可以调用自己

- 代码特点：

1.函数内部的代码是相同的，只是针对参数不同，处理的结果也不同

**2.当参数满足一个条件时，函数不在执行**

**- 这个非常重要，通常被称为递归的出口，负责会出现死循环**

3.曲折，曲折，因果往复

**4.参数最大值是 998**

个人感想：递归就是自己玩自己，你不给结束指令，它自己能玩到死

例子：

def sum\_numbers(num):

print(num)

if num == 1 :

return

sum\_numbers(num - 1)

sum\_numbers(5)

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

def sun\_numbers(num):

if num == 1:

return 1

temp = sun\_numbers(num - 1 )

return num + temp

print(sun\_numbers(5))

个人理解：

- 无需提前定义变量，的却让我蒙了，这个方法的却很好

- temp在这里的作用只是接受 return 1 ，也就是最后 temp = 1

然后递归一次次的往上返回，一次次的相加，

1+2，3+3，6+4，10+5

在判断return后逐层返回，这次举例了解递归的原理

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

>>面向对象（第五十三课）

### **1.面向对象基本概念**

- 我们之前学习的编程方式是面向过程的

- 面向过程与面向对象，是两种不同的便编程方式

- 对比面向过程的特点，可以更好的了解什么是面向对象

### **2.过程和函数（科普）**

- 过程是早期的一个编程概念

- 过程类似于函数，只能执行，但是没有返回值

- 函数不仅能执行，还能返回结果

### **3.面向过程和面向对象基本概念**

- 把完成某一个需求的所有步骤从头到尾逐步实现

- 根据开发需求，把某些功能独立的代码封装成一个有一个函数

- 最后完成的代码，就是顺序的调用不同函数

### **4.特点**

- 注重步骤与过程，不注重职责分工

- 如果需求复杂，代码会变得很复杂

- 开发复杂项目，没有固定的套路，开发难度很大

- 感觉像是Js中的 父对象 与 子对象

————————————————————————————————

### **1.面向对象 -- 谁来做？**

- 相比较函数，面向对象是更大的封装，根据职责在一个对象中封装多个方法

- 在完成某一个需求前，首先确定职责 -- 要做的事情（方法）

- 根据职责确定不同的对象，在对象内部封装不同的方法（多个）

- 最后完成的代码，就是顺序的让不同的对象调用不同的方法

### **2.特点**

- 注重对象和职责，不同对象承担不同的职责

- 更加适合应对复杂的需求变化，是专门应对复杂项目开发，提供固定套路

- 需要在面向过程基础上，在学习一些面向对象的语法

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

>>类和对象基本概念（第五十四课）

类和对象是面向对象编程的基本概念

1.类

- 类是对一群具有相同特征或者行为的事物的一个同城，是抽象的，不能直接使用

- 特征被称为属性

- 行为被称为方法

- 类就相当于制造飞机时的图纸，是一个模板，是负责创建对象的

2.对象

- 对象是由类创造出来的一个具体存在，可以直接使用

- 由哪一个类创建出来的对象，就拥有在哪一个类中定义的：

- 属性

- 方法

- 对象就相当于用图纸制造的飞机

- 在程序开发中，应该先有类，再有对象

- 想起来 Js 中的 New 了吗？我觉的可能是那个

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

>>设计三要素及名词提炼（第五十五课）

1.类的设计

- 在程序开发中，要设计一个类，通常需要满足以下三个要素

- 类名：这类事物的名字，满足大驼峰命名法

- 属性：这类事物具有什么样的特征

- 方法：这类事物具有什么样的行为

2.大驼峰命名法

- 每个单词首字母大写

3.类名的确定

- 名词提炼法分析整个业务流程，出现的名词，通常就是找到的类

4.属性和方法的确定

- 对对象的特征描述，通常可以定义成属性

- 对象具有的行为，通常可以定义成方法

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

>>面向对象的查找（第五十六课）

1. dir 内置函数（知道）

- 格式： dir( )

- dir(函数等)

会给出所有的方法

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

>>定义简单类（第五十七课）

- 面向对象是更大的封装，在一个类中封装多个方法，这样通过这个类创建出来的对象

就可以直接调用这些方法了

1.在Py中定义一个只包含方法的类，语法格式如下：

class 类名：

def 方法1（self,参数列表）：

pass

def 方法2（self,参数列表）：

pass

- 方法的定义格式和之前学习过的函数几乎一样

- 区别在于第一个函数必须是self

- 注意：类名的命名规则要符合大驼峰命名法

2.创建对象

- 当一个类定义完成之后，要使用这个类创建对象,语法格式如下：

对象变量 = 类名（ ）

例子：

class Cat:

def eat(self,food):

print("小猫爱吃%s"%(food))

def drink(self):

print("小猫爱喝水")

tom = Cat()

tom.eat("牛奶")

tom.drink()

相比于Js只是缺少了个New,多了个self(但好似不用管？)

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

>>接受对象的变量对对象的引用（第五十八课）

- 引用概念的强调

- 在面向对象开发中，引用的概念是同样适用的

- 在Py中使用类创建对象后，tom变量中记录的是对象在内存中的地址

- 也就是tom变量引用了新建的猫对象

- 使用print输出对象变量，默认情况下，是能够输出这个变量引用的对象是由

哪一个类创建的对象，以及在内存中的地址

- %d 十进制

- %x 十六进制

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

>>给对象添加属性（第五十九课）

例子：

Tom.name = “猫咪”

不推荐使用

感觉有点像Js中的object对象

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

>>self输出对象属性（第六十课）

- self 说白了就是 Js 中的 this.

- 哪一个对象调用的方法，self就是哪一个对象的引用

在类外部添加属性的隐患：

1.总而言之就是，调用了之后才想着去定义，导致找不到

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

>>初始化方法（第六十一课）

- 当使用 类名（ ）创建对象时，会自动执行以下操作

1.为对象在内存中分配空间 -- 创建对象

2.为对象的属性设置初始值 -- 初始化方法

\_\_init\_\_方法专门用来定义一个类具有哪些属性的方法

例子：

class Cat:

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

self.name = "大咪咪"

def eat(self):

print("%s爱吃鱼" % self.name)

cat = Cat()

cat.eat()

**个人理解：**

**- \_\_init\_\_可以理解为初始化方法，也就是在类里提前定义好的**

**- 可以直接使用**

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

>>针对于对象属性的设置（第六十二课）

1.\_\_init\_\_

- 在开发中，如果希望在创建对象同时，就设置对象属性，可以对初始值进行改造

1.把希望设置的属性值，定义成\_\_init\_\_方法的参数

2.在方法内部使用 self.属性 = 形参 接受外部传递的参数

3.在创建对象时，使用 类名（属性1，属性2）调用

2.\_\_del\_\_

- 在开发中，如果希望对象在被清理前执行代码，就增加对象的\_\_del\_\_方法

1.也就是说利用\_\_del\_\_可以对对象进行最后一次调用

2.del关键子可以删除一个对象

3.可以理解为 对象的遗言

3.\_\_str\_\_

- 如果在开发中，希望使用print输出对象变量时，能够打印自定义内容，就可以利用

\_\_str\_\_这个内置方法了

1.总的来说就是感觉就是替换

2.将print出来的地址变为了文本

3.必须返回一个字符串

二的例子：

class Cat:

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

self.name = name

print("%s爱吃鱼" % self.name)

def \_\_del\_\_(self):

print("我死了~")

cat = Cat("666")

print("-"\*20)

————————————————————————————————

三的例子：

class Cat:

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

self.name = name

print("%s爱吃鱼" % self.name)

def \_\_del\_\_(self):

print("我去了~")

def \_\_str\_\_(self):

return("我是一只猫")

cat = Cat("666")

print(cat)

print("-"\*20)

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

>>小明爱跑步（第六十三课）

答案：

class XiaoMing:

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

self.name = name

self.weight = weight

def run(self):

self.weight -= 0.5

def eat(self):

self.weight += 1

def \_\_str\_\_(self):

return "我的名字叫 %s 体重是 %.2f 公斤" %(self.name,self.weight)

center = XiaoMing("小明",75.0)

center.eat()

center.run()

print(center)

尽量别把数字定义在初始化，灵活为先

————————————————————————————————

自己写的

class Person:

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

self.weight = weight

self.name = name

def eat(self,speak):

self.speak = speak

self.weight = self.weight + 1\*speak

def run(self,new\_speak):

self.new\_speak = new\_speak

self.weight = self.weight - 0.5\*new\_speak

def \_\_str\_\_(self):

return "我叫 %s ,我有 %s 公斤" % (self.name,self.weight)

speak = int(input("小明吃了几次食物？"))

new\_speak = int(input("小明跑了几次步？"))

xiaoming = Person(75,"小明")

xiaoming.eat(speak)

xiaoming.run(new\_speak)

print(xiaoming)

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

>>定义家具（第六十四课）

————————————————————————————————

自己写的

class HouseItem: #家具所有东西

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

self.name = name

self.area = area

def \_\_str\_\_(self):

return "现有家具 %s 占地 %s " % (self.name,self.area)

class House: #My House

def \_\_init\_\_(self,house\_type,area):

self.house\_type = house\_type

self.area = area

def house\_add(self):

self.area = self.area - (center\_add())

def \_\_str\_\_(self):

return "户型 %s 总面积 %s 现面积 %s 拥有%s" % (self.house\_type,

self.area + center\_add(),

self.area,

center)

#定义家具都有哪些

house\_item = HouseItem("床",1.8)

house\_item1 = HouseItem("马桶",5.6)

house\_item2 = HouseItem("浴缸",7.6)

#放进一个列表中以方便使用

center\_answer = [house\_item.area,

house\_item1.area,

house\_item2.area,

]

center = [house\_item.name,

house\_item1.name,

house\_item2.name,

]

#返回整体的家具占地面积的总和值

def center\_add():

a = 0

for h in center\_answer:

a += h

return a

#定义家的类型与总面积

house = House("盗梦型",70)

#删除定义的家具面积

house.house\_add()

#输出所有内容

print(house)

总结：定义完一个物品之后，需要改的地点有两个，怪麻烦的

————————————————————————————————

老师的

- 还是写的妙啊...唉

class HouseItem:

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

self.name = name

self.area = area

def \_\_str\_\_(self):

return "[%s] 占地 %.2f" %(self.name,self.area)

class House:

def \_\_init\_\_(self,house\_type,area):

self.house\_type = house\_type

self.area = area

#剩余面积

self.free\_area = area

#家具

self.item\_list = []

def \_\_str\_\_(self):

return ("户型 %s\n总面积 %2.f[剩余 %.2f]\n家具 %s"

%(self.house\_type,self.area,

self.free\_area,self.item\_list))

def add\_item(self, item):

print("%s" % item)

if item.area > self.free\_area:

print("%s 的面积太大了，无法添加" % item.name)

return

self.item\_list.append(item.name)

self.free\_area = self.free\_area - item.area

bed = HouseItem("席梦思", 100)

chest = HouseItem("衣柜", 2 )

table = HouseItem("餐桌", 1.5)

print(bed)

print(chest)

print(table)

print("-"\*50)

my\_home = House("两室一厅",60)

my\_home.add\_item(bed)

my\_home.add\_item(chest)

# my\_home.add\_item(table)

print(my\_home)

失误：

- 我自己好像老是执着于一个变量进行转换了，老师先在 init 里将数值传给一个新变量，

而我则是一个变量来回转换

- 对于 init 函数的不熟悉，不会使用内定的，老是使用外面传的

- 没有熟练知道Py的代码执行方式，老是喜欢将东西集在一起去处理

- 不要因为没有提示就 六神无主 了

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

>>士兵突击（第六十五课）

1.在定义属性时，在不知道设置什么初始值的情况下，使用 None 解决

————————————————————————————————

自己写的

class Gun:

def \_\_init\_\_(self,model,bullet\_count): #初始化枪名与子弹

#枪的类型

self.model = model

#定义的初始子弹数量

self.bullet\_count = bullet\_count

#子弹数量

self.have\_shoot = bullet\_count #have\_shoot = 10

#ask

self.first\_shoot = int(input("填充多少子弹？"))

self.last\_shoot = int(input("发射多少子弹？"))

def add\_bullet(self,count): #增加子弹

count.have\_shoot = self.bullet\_count + self.first\_shoot

def shoot(self): #发射子弹

self.have\_shoot = self.have\_shoot - self.last\_shoot

class Soldier:

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

self.name = name

self.gun = gun

def fire(self):

gun.shoot()

def \_\_str\_\_(self):

return ("士兵 %s \n枪的类型是 %s \n子弹数量是 %s \n发射了 %s 弹\n填充了 %s 弹 \n还剩 %s 弹" %(

soldier.name,

gun.model,

gun.bullet\_count,

gun.last\_shoot,

gun.first\_shoot,

gun.have\_shoot))

gun = Gun("AK47",10)

gun.add\_bullet(gun)

gun.shoot()

print("-"\*20)

soldier = Soldier("许三多",Gun)

print(soldier)

————————————————————————————————

老师的

class Gun:

def \_\_init\_\_(self,model):

self.model = model

self.bullet\_count = 0

def add\_bullet(self,count):

self.bullet\_count += count

def shoot(self):

if self.bullet\_count <= 0 :

print("没有子弹了")

return

self.bullet\_count -= 1

print("突突突！")

class Soldier:

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

self.name = name

self.gun = None

def fire(self):

if self.gun == None:

print("没有枪")

return

print("冲啊")

self.gun.add\_bullet(50)

self.gun.shoot()

ak47 = Gun("Ak47")

xusanduo = Soldier("许三多")

xusanduo.gun = ak47

xusanduo.fire()

print(xusanduo.gun)

失误：

- 对于形参与实参，我好似老喜欢在init里定义，忘记了其他def

- 对于老师给的框架有点固执，完全忘记了灵活去运用，有点 照本宣钢 的样子

- 我与老师之间的思维差距如同萤火与皓月啊......

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

>>身份运算符（第六十六课）

**- is is 是判断两个标识符引用的是不是同一个对象**

**- is not is not 是判断两个标识符是不是引用的不同对象**

**is 与 == 区别：**

**- is 用于判断两个变量的引用是否为同一个**

**- == 用于判断引用变量的值是否相等**

例子：

b = [1,2,3]

a = [1,2,3]

b is a

输出：False

b == a

输出：Ture

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

>>私有属性与私有方法（第六十七课）

- 有些对象的属性或方法可能只希望在对象内部使用，不希望在外部被访问到

- 私有属性就是对象不希望公开的属性

- 私有方法就是对象不希望公开的方法

定义方式：

- 在定义属性与方法时，**在属性名或方法名前增加两个下划线**，定义的就是私有属性或方法

例子：

class Women:

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

self.name = name

self.\_\_age = 18

def \_\_secret(self):

#在对象方法内部，是可以访问对象的私有属性的

print("%s 的年龄是 %d" % (self.name, self.\_\_age))

xiaofang = Women("Monika")

#伪私有属性在外界不能够直接访问

print(xiaofang.\_\_age)

#伪私方法在外界不能够直接访问

xiaofang.\_\_secret()

————————————————————————》科普

**- 但是实则在 Py 中是没有真正的私有的**

再给属性与方法命名时，实则是对名称做了一些特殊处理，使得外界无法访问到

处理方式：在名称前面加上 \_类名 => \_类名\_名称

使用私有属性名方法

print(xiaofang.\_Women\_\_age)

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

>>续承（第六十八课）

**1.继承的概念，语法和特点**

**- 继承的概念：子类拥有父类的所有方法和属性**

继承的语法：

class 类名( 父类名 ):

pass

例子：

class Animal:

def drink(self):

print("喝水")

class Dog(Animal):

def bark(self):

print("叫")

dog = Dog()

dog.drink()

**可以重新定义的**

**2.从新定义以及扩展内容**

- 从新定义就是在子类中重新写函数方法，这样的话在调用时就不会动用父类的方法了

- 扩展内容要使用 super( ). 这个方法，使用原本在父类中的方法

- 例子：

class Animal:

def drink(self):

print("喝水")

def bark(self):

print("喵喵叫")

class Dog(Animal):

def bark(self):

print("汪汪叫")

super().bark()

dog = Dog()

dog.bark()

输出：

汪汪叫

喵喵叫

**3.关于私有属性与私有方法中继承的表现**

- 子类无法直接访问到父类的私有属性与方法

- 子类可以通过父类的公共方法间接访问到私有属性与方法

**4.如何在子类对象中调用父类的私有属性与方法？**

- 父类创建一个方法，调用自己的私有函数与方法，由子类调用公共方法就行了

- 自己理解：

- 由于子类调用不了父类的私有函数与方法，那就用父类调用，将值传给一个方法，子类调用父类方法，就是间接调用了父类的私有函数与方法

- 儿子想要钱，父亲把私房钱位置说出来了

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

>>多续承（第六十九课）

1.子类可以拥有多个父类，并具有所有父类的属性与方法

语法：

class 子类名(父类名1，父类名2)：

pass

2.倘若多个类名属性重叠，则会按照父类名的排序1进行输出

- 所以说尽量别重叠

3.关于MRO方法搜索顺序（了解）

- MRO主要用于在多继承时判断方法，属性的调用路径

- 在搜索方法时，是按照从左到右的顺序查找的

- 如果在当前类中找到方法，就直接执行，不在搜索

- 如果没有找到，就在下个类中找，找到就不执行了

- 如果没找到就报错了

- 完美的解释了，类属性重名问题时，程序执行顺序

查看方式

print(c.\_\_mro\_\_)

4.新式类和经典类 - 扩展

- 新式类：以Object 为基类的类，推荐使用

- 经典类：不以Object 为基类的类，不推荐使用

- 在Py3.0+时，如果没有指定父类，会默认使用Object作为该类的基类

- 在Py2.0+时，如果没有指定父类，则不会使用Object作为该类的基类

- 新式类和经典类在多继承时，会影响到方法的搜索顺序

- 所以在今后的定义类时，尽量使用Object填充

class 类名(object):

pass

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

>>多态（第七十课）

- 不同的子对象调用相同的父类方法，产生不同的执行效果

- 多态可以增加代码的灵活度

- 以继承和重写父类方法为前提

- 是调用方法的技巧，不会影响到类的内部设计

例子：

class Dog(object):

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

self.name = name

def game(self):

print("%s 正在玩耍"%self.name)

class XiaoTianDog(Dog):

def game(self):

print("%s 正在天上玩耍" % self.name)

class Person(object):

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

self.name = name

def game\_with\_dog(self,dog):

print("%s 和 %s 快乐的玩耍" % (self.name,dog.name))

dog.game()

xiaotian = Dog("小天")

wangcai = XiaoTianDog("小飞")

xiaoming = Person("小明")

xiaoming.game\_with\_dog(wangcai)

总结，相同的父亲，不同的儿子会取得不同的成就

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

>>类的相关概念（第七十一课）

1.使用类名创建对象，创建对象的动作有两步

- 在内存中为对象分配空间

- 调用初始化方法 init 为对象初始化

对象创建后，内存中就有了一个对象的实实在在的存在 -- 实例

所以也通常会把：

- 创建出来的对象叫做类的实例

- 创建对象的动作叫做实例化

- 对象的属性叫做实例属性

2.我们有时需要知道一个类被实例化了几次

例子：

class Monika(object):

count = 0

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

self.name = name

Monika.count += 1

monika = Monika("monika")

sayori = Monika("sayori")

print(Monika.count)

输出：2

原因：上文提到过，每次实例化一次对象，都会调用初始化方法，所以数值上升

3.属性查找机制 -- 向上查找

- 1.要访问类属性有两种方式

- 类名.类属性

- 对象.类属性（不推荐）

个人理解：**就是一个自身找不到然后像上找的东西**

Js中的 prototye 更直观一点（原型链嘛~）

4.对象.类属性不推荐的原因

- 假如给 对象.类属性 赋了个值，那么会成为治标不治本的问题

由于自查中存在该属性，那么就不会向上查找了

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

>>类方法（第七十二课）

- 类属性就是针对类对象定义的属性

- 使用赋值语句在class关键字下方可以定义类属性

- 类属性用于记录和这个类相关的特征

- 类方法就是针对类对象定义的方法

- 在类方法内部可以直接访问类属性或者调用其他的类方法

语法如下：

@classmethod

def 类方法名(cls):

pass

**- 类方法需要用修饰器 @classmethod 来标识，告诉解释器这是个类方法**

**- 类方法第一个参数是 cls**

**- 与 self 类似**

**- 使用其他名称也可以，不过要习惯cls**

- 在方法内部

- 可以通过 cls. 访问类的属性

- 也可以通过 cls. 调用其他的类方法

例子：

class Monika(object):

count = 0

@classmethod

def Tool(cls):

print("工具数量 %d"%Monika.count)

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

self.name = name

Monika.count += 1

monika = Monika("monika")

sayori = Monika("sayori")

Monika.Tool()

个人理解：可以直接通过类名.的方式调用，不用实例化

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

>>静态方法（第七十三课）

**- 类方法需要用修饰器 @classmethod 来标识，告诉解释器这是个类方法**

例子：

class monika(object):

@staticmethod

def monika():

print("666")

个人理解：一般来说是输出不用与类属性与方法属性联动的东西

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

>>确定方法类型的套路（第七十四课）

**1.实例方法 -- 方法内部需要访问实例属性**

**实例方法内部可以使用 类名. 访问类属性**

**2.类方法 -- 方法内部只需要访问类属性**

**3.静态方法 -- 方法内部，不需要访问实例属性和类属性**

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

>>设计模式概念（第七十五课）

- 设计模式 是 前人工作的总结与提炼，是针对某一个特定问题的成熟解决方案

- 使用设计模式是为了可重用代码，让代码更容易被他人理解，保证代码可靠性

## 1.单例设计模式

**- 目的 -- 让类创建的对象，让系统中只有唯一的一个实例**

- 每一次执行 类名( )返回的对象，内存地址都是相同的

单列设计模式的应用场景

- 音乐播放对象

- 回收站对象

- 打印机对象

具有唯一性

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

>>\_\_new\_\_ 方法的作用（第七十六课）

- 使用类名创建对象时，Py解释器会调用 new方法 为 对象 分配空间

- \_\_new\_\_ 是一个object 基类提供的内置的静态方法，主要作用有两个：

1.在内存中为对象分配空间

2.返回对象的引用

- Py解释器在获得对象的引用后，会引用作为第一个参数，传递给 \_\_init\_\_ 方法

- 重写 \_\_new\_\_ 方法的代码非常固定

**- 重制 new 方法一定要 return super().\_\_new\_\_(cls)**

- 否则Py解释器得不到分配了空间的对象引用，就不会调用对象的初始化方法

- 注意 \_\_new\_\_ 是一个静态方法，在调用时**需要主动传递 cls 参数**

课上的笔记：

**-和继承一样，子类改写就不执行父类代码，在改造Object的子类 new 方法时，**

假设没写返回地址，报错信息就会返回None，在Py中原本是自动调用

new方法的现在属于重写，一定要记得在new方法内部写上

return super( ).\_\_new\_\_(cls)

例子：

def \_\_new\_\_(cls,\*args,\*\*kwargs):

......

return super( ).\_\_new\_\_(cls)

**- 重新定义new会使原本的object的new方法失效，所以需要super( )重新调用**

也就是前面对父类的重写与延申

**- 由于new方法属于静态方法，没有self 与 cls ，所以需要手动添加cls**

作用：无论创建对象多少次，返回的地址都将会是一样的

例子：

class Monika(object):

instance = None

def \_\_new\_\_(cls, \*args, \*\*kwargs):

if cls.instance is None:

cls.instance = super().\_\_new\_\_(cls)

return cls.instance

monika = Monika()

print(monika)

sayori = Monika()

print(sayori)

失误点：

- 可没人说过每创建一次对象，类就要重置一次，类里被赋值的东西不会清零

关于多次创建对象地址一致与初始化方法只执行一次方法

class Monika(object):

instance = None

num = False

def \_\_new\_\_(cls, \*args, \*\*kwargs):

if cls.instance is None:

cls.instance = super().\_\_new\_\_(cls)

return cls.instance

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

if Monika.num:

return

print("666")

monika = Monika()

print(monika)

sayori = Monika()

print(sayori)

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

>>抛出异常概念及原因（第七十七课）

1.概念

- 程序在运行时，Py解释器遇到一个错误，会停止程序运行，并给错误信息

这就是异常

- 程序停止并提示错误信息这个动作，我们就称为：抛出异常

2.原因

- 懂得都懂

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

>>异常捕获（第七十八课）

**- 在程序开发中，如果对某些代码执行不确定是否正确，可以增加try(尝试)来捕获异常**

- 捕获异常简单语法格式:

try:

尝试执行的代码

except:

出现错误的处理

try:尝试，下方编写要尝试的代码，不确定是否能不能正常执行的代码

except:如果不是，下方编写尝试失败的代码

例子：

try:

a = int(input("给我个数字"))

print(a)

except:

print("没听见是吧？")

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

>>根据错误类型捕获异常（第七十九课）

语法如下：

try:

执行的代码

except 错误类型1:

执行代码

except (错误类型2，错误类型3):

执行代码

**问：错误类型怎么填？**

**- 当Py解释器抛出异常时，最后一行错误信息的第一个单词，就是错误类型**

例子：

try:

a = int(input("请输入一个数，我们来为您除以8"))

result = 8/a

print(result)

except ValueError:

print("请输入数字ok?")

except ZeroDivisionError:

print("你家分母为0?")

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

>>捕获未知错误（第八十课）

- 人不是神，无法知晓世间万事，如万家灯火，你永远也不知道哪家失火

**- 如果希望程序无论出现任何错误，都不会因为Py解释器抛出异常而终止**

**可以再增加一个except**

语法如下：

except Exception as result:

print("未知错误 %s" % result)

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

>>捕获未知错误完整语法（第八十一课）

语法：

try:

尝试执行的代码

excep 错误类型一:

针对此类型的处理

except Exception as 要赋值的

针对其他错误所要的处理

else:

没有异常才会执行的代码

finally:

无论是否有异常都会执行的代码

————————————————

else: 没有异常才会执行的代码

finally: 无论是否有异常都会执行的代码

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

>>异常的传递性（第八十二课）

**概念：**

**- 异常的传递：当函数/方法出现异常，会将异常传递给函数/方法的一方**

**- 如果传递到主程序，仍然没有异常处理，程序才会被终止**

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

>>主动抛出异常（第八十三课）

**- 变量 = Exception("一个多值元组，用处是输出错误信息")**

**- raise 变量**

**抛出异常**

例子：

def input\_password():

pwd = input("请输入密码: ")

if len(pwd) >=8 :

return pwd

ex = Exception("密码长度不够")

raise ex

try:

print(input\_password())

except Exception:

print("密码输长点")

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

>>模块（第八十四课）

概念：

**- 每一个以Py结尾的文件名都是属于模块**

- 模块名同样也是一个标识符，也需要**遵从命名规则**

- 在模块中**定义的全局变量、函数、类都是提供给外界直接使用的工具**

- 模块就好比工具包，**要使用工具，就要先导入模块**

1.模块导入语法：

import 模块名1，模块名2 (不推荐)

- 再导入模块时，应该直接占一行

2.模块导入语法：

import 模块名1

import 模块名2

导入之后：

**通过 模块名. 使用模块提供的工具 -- 全局变量,函数,类**

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

>>模块导入之类方法（第八十五课）

### **1.使用 as 指定模块的别名**

**- 如果有两个相同的函数名，那么就可以用as来更改**

语法：

import 模块名1 as 模块别名

例子：

import monika as mo

### **2.局部导入**

**- form … import 导入**

**- 如果希望从一个模块中，导入部分工具，就是用form … import 方式**

**- import 模块名 是把所有工具全部导入**

语法

form 模块名1 import 工具名

导入之后

- 不需要通过 模块名.

**- 可以直接使用模块提供的工具 -- 全局变量，函数，类**

注意：假设两个模块中存在同名的函数，那么后导入模块的函数，会覆盖掉先导入的函数

例子：

title = "我是夏树"

class Name(object):

def name(self):

print("那啥 %s" %title)

————————————————————

from Nasuki import Name

name = Name()

name.name()

也可以：

def cat():

print("夏树养了一只🐱")

————————————————————

from Nasuki import cat

cat()

倘如工具名称重复了的话：

from Nasuki import cat

from sayori import cat as dog

cat()

dog()

### **3.利用 form...import 导入所有工具（了解）**

- form 模块名1 \*

- 不推荐使用，因为函数重名没有提示，**不好排查**

### **4.模块搜索顺序**

- 1.Py解释器在导入模块时

**- 搜索当前目录指定模块名的文件，如果有就直接导入**

**- 如果没有，就搜索系统目录**

### **5.扩展**

- Py中每个模块都有一个内置属性 **\_\_file\_\_** 可以查看模块的完整路径

### **6.Py的开发原则**

**- 每一个文件都应该是可以被导入的**

- 再导入时，每行代码都会执行一遍

**- 只要导入，模块内的代码就会执行**

- 开发人员通常会在模块下方添加一些测试代码

- 仅在模块内使用，在导入时不需要执行

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

>>\_\_name\_\_属性（第八十六课）

**- name属性可以做到测试模块的代码，在导入时不会执行**

- name是Py的一个内置属性，记录着一个字符串

- 如果是被其他文件导入的，name就是模块名

- 如果是当前执行的程序 \_\_name\_\_ 是 \_\_main\_\_

个人理解：由于直接输出\_\_name\_\_会获得\_\_mian\_\_结果

而进行导入打印时，会给出文件名

那么可以使用判断语句来测试模块

if \_\_name\_\_ == "\_\_mian\_\_":

print("测试执行")

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

>>包的概念及建立方式（第八十七课）

概念：

·包是一个包含多个模块的特殊目录

·目录下有一个特殊的文件 \_\_init\_\_.py

·包名的命名方式和变量名一样，小写字母+\_

好处

·使用import包名可以一次性导入包中所有模块

1.导入方式

- 在\_\_init\_\_文件里面是需要输入 form . import 文件名

- . 在之前命令学过，代表本地文件的意思

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

### **>>制作模块（了解）**

- bilibili 黑马程序员 第二遍的082集

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

### **>>下载模块（了解）**

pip3 install pygame - 安装

pip3 uninstall pygame - 卸载

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

>>文件的概念（第八十八课）

函数/方法 说明

open 打开文件，并返回文件操作对象

read 将文件内容读取到内存

write 将指定内容写入文件

close 关闭文件

- 在计算机中操作文件套路固定，一共包含三个步骤

- 1.打开文件

- 2.读,写文件

- 3.关闭文件

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

>>文件操作与相关概念（第八十九课）

**- open函数的第一个参数就是要打开的文件名（区分大小写）**

- 如果文件存在，返回文件操作对象

- 如果不存在，就抛出异常

**- read方法可以一次性读入并返回文件的所有内容**

**- close 方法负责关闭文件**

**- 如果忘记关闭文件，会造成系统资源消耗**

- 注意：方法执行后，会把**文件指针移动到文件的末尾**

例子：

file = open("ReadMe")

text = file.read()

print(text)

file.close()

问题：必须在同一个工作目录下

### **1.文件指针（知道）**

**- 文件指针标记从哪个位置开始读取数据**

- 第一次打开文件时，通常文件指针会指向文件的开始位置

**- 当执行了read方法时，文件指针会移动到读取内容的末尾**

- 所以第一次读取过后，再次调用不会读取到任何内容

### **2.文件打开方式**

- open 函数只**默认以只读方式打开文**件，并返回文件对象

- 以下方式**结合 open** 进行使用

- 如果不带的话，open默认会以只读方式打开文件

例子：

**f = open("文件名","访问方式")**

——————————————————————————————

访问方式 说明

——————————————————————————————

r 以只读方式打开文件，指针会放在文件开头

这是默认模式，如果文件不存在，就抛出异常

——————————————————————————————

w 以只写方式打开文件，如果文件存在会被覆盖

如果不存在，创建新文件

——————————————————————————————

a 以追加方式打开文件，如果文件存在

文件指针会放在文件末尾，如果文件不存在

创建新文件进行写入

——————————————————————————————

r+ 以读写方式打开文件夹，文件指针会放在文件的开头

如果文件不存在，抛出异常

——————————————————————————————

w+ 以读写方式打开文件，如果文件存在会被覆盖

如果文件不存在，创建新文件

——————————————————————————————

a+ 以读写方式打开文件，如果文件已存在，

文件指针将会放在文件的末尾，如果文件不存在

创建新文件进行写入

——————————————————————————————

### **3.readline 方法**

**- readline方法可以一次读取一行的内容**

**- 方法执行后会把文件指针移动到下一行，准备再次读取**

例子：

file = open("ReadMe")

while True:

text = file.readline()

if not text:

break

print(text,end = "")

file.close()

4.复制文件

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

>>文本编码（第九十课）

- 在Py2.x解释器第一行增加以下代码，解释器会以utf - 8 来处理Py文件

要不然默认的是 ASCII 编码格式

#\*-\* coding:utf8 \*-\*

- 在字符串前面加 u 也能告诉Py解释器以utf - 8 来处理Py文件

例子：

hello\_str = u"hello世界"

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

>>eval函数（第九十一课）

- eval( ) 函数十分强大 -- 将字符串当成有效的表达式来求值并返回计算结果

注意：**不要滥用 eval( ) 函数**

**- 在开发时千万不要用 eval 直接转换 input 结果**

因为假设被人输入：\_\_import\_\_('os').system('ls')

可直接调用系统命令，**被注入式攻击**

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

>>飞机大战（第九十二课）

验证安装 - windows

python -m pygame.examples.aliens

## 1.pygame提供了一个类 pygame.Rect 用于描述矩形区域

Rect(x,y,width,height) -> Rect

- Rect中有size方法，用于获取长和宽，并且是元组

## 2.创建游戏主窗口

pygame.display.set\_mode( )

- 初始化游戏窗口

pygame.display.update( )

- 刷新屏幕内容显示

set\_mode(resolution = (0,0),flags = 0,depth = 0) - >surface

作用：创建游戏显示窗口

参数：

resolution:指定屏幕的宽和高，默认窗口与屏幕一致

falags:参数指定屏幕的附加选项，例如是否全屏等等，默认不用传递

depth:参数表示颜色的位数，默认自动匹配

返回值：暂时理解为游戏的屏幕，游戏的元素都需要被绘制到游戏屏幕上

**注意：必须使用变量 set\_mode 方法的返回结果**

## 3.理解图像并实现图像绘制

- 使用 pygame.image.load() 加载图像的数据

- 使用 游戏屏幕 对象，调用 blit 方法将图像绘制到指定位置

- 调用 pygame.display.update() 方法更新整个屏幕的显示

一个简单的加载背景

import pygame

pygame.init()

screen = pygame.display.set\_mode((480, 700))

bg = pygame.image.load("./images/background.png")

screen.blit(bg, (0, 0))

pygame.display.update()

while True:

pass

pygame.quit()

## 4.对于update( )方法的作用

- 可以 screen 对象完成所有 bilt 方法以后，统一调用一次 display.update 方法

同样可以在屏幕上看到最终的绘制效果

- 使用 display.set\_mode( )创建的screen对象是一个内存中的屏幕数据对象

**- 可以理解为油画的画布**

- screen.blit( ) 方法可以在画布上绘制很多的图像

**- 但这些图像可能会重叠或者覆盖**

- display.update( ) 会将画布的最终结果绘制到屏幕上，这样可以提高游戏绘制效率

增加游戏流畅度

## 5.游戏时钟

- pygame 提供了一个类 pygame.time.Clock 可以方便的设置屏幕绘制速度 -- 刷新帧数

**1.在游戏初始化创建一个时钟对象**

**2.在游戏循环中让时钟对象调用 tick(帧率) 方法**

**3.tick 方法会自动根据上次调用的时间设置延时**

## 6.事件监听

**- 事件：event**

**- 就是游戏启动后，用户针对游戏所作的操作**

**- 例如：点击鼠标**

代码实现：

- pygame 通过 pygame.event.get( ) 可以获得用户当前所做动作的事件列表

- 用户可以同一时间做很多事情

- 这段代码非常固定，几乎大多 pygame 游戏都大同小异

- 这段代码会返回一个列表，所以需要用一个变量接受

- 该代码会返回用户的所有操作

例子：

event\_list = pygame.event.get( )

if len(event\_list) > 0:

print(event\_list)

- 建议用 if 进行检测，当列表为0时，不会输出

## 7.事件监听

- exit( ) 方法：把当前正在执行的程序直接退出,用在这里

之前一直以为是判断字典，但是pygame提供了Quit方法

- pygame.QUIT

- 好处：简单

- 坏处：限制思维

## 8.理解 精灵 与 精灵组

- 在刚刚的案例中，图像加载，位置变化，绘制图像，都需要编写代码分别处理

**- 为了简化步骤，pygame 提供了两个类**

- pygame.sprite.Sprite - 存储 图像数据 image 和 位置 rect 的对象

- pygame.sprite.Group

- 这俩个东西里面都自己封装了一些东西，要想熟悉还得多练

- pygame.sprite.Sprite

- self.image.get\_rect():自动获取图片长和宽

- pygame.sprite.Group

- draw:相当于blit方法：enemy\_group.draw(绘制到的地方)

- update：就是更新精灵的东西：pygame.display.update()

个人理解：

- 实则就是pygame提供了两个方法

- 一个定义

- 一个集合去调用

- 而在pygame里提供了两个方法，但这两个方法里面有自己的功能

- 所以在我们不看代码的情况下只能多练习才能熟知各个功能

- 也就是去使用 封装，继承，多态 去熟练

- 确实能大大减少代码量

**- 这个代码后面调用的 draw() 方法必须要有 image 与 rect**

花了我 4 个小时去检查错误的结论

## 9.关于常量与变量

- 常量——不变化的量

**- 说白了也是变量，只不过全是大写，单词与单词之间有下划线**

- 变量——可以变化的量

## 10.定时器语法介绍

- setInterval(function(){ },1000) -- JS

- 在 pygame 中用 pygame.time.set\_timer( ) 来添加定时器

- set\_timer(eventid,毫秒)

- eventid 对应这一个整数数值，也就是一个事件

至于后面的

CREAT\_ENEMY\_EVENT = pygame.USEREVENT

pygame.time.set\_timer(CREAT\_ENEMY\_EVENT, 1000)

- USEREVENT:

- 感觉像是固定语法

- 事件代号，需要使用 pygame.USEREVENT 来指定

- 是一个整数，再增加的事件可以使用 USEREVENT+1 指定

- 以此类推

- 关于后来个人对于 USEREVENT 的理解

**- 实则就是一个变量？**

**- 默认值为32787然后将32787作为特征码进行使用？**

- 在event遍历了所有pygame.event.get( )方法后

- 我注意到如果输出print(event.type) 时

- 会输出 32787

- 正好与 pygame.USEREVENT 重合

- 我猜测这也叫就是为什么再增加的事件可以使用 USEREVENT+1 指定

- 好记是吧？绝了

## 11.关于定时器扩展

- 1.定义定时器常量 -- eventid -也就是之前的 USEREVENT 方法

- 2.在初始化方法中，调用 set\_timer 方法设置定时器事件

- 3.在游戏循环中，监听定时器事件

## 12.rect 的 bottom 属性

- 1. bottom = y + height

- 2. y = bottom - height

## 13.self.kill( ) 方法

- 跟 remove 和 del 用途差不多

- kill方法可以将精灵从所有精灵组中移除，销毁

## 14.关于 rect.bottom 的理解

- 我认为这是一个矩形的底部位置，相当于 y ，但是在矩形以底部出现

- rect.bottomleft

- rect.bottomright

## 15.移动英雄方式

- 第一种方式：

- 判断 event.type == pygame.KEYDOWN

- 第二种方式：

- 首先使用 pygame.key.get\_paressed( ) 返回所有按键元组

- 通过键盘常量，判断元组中某一个按键是否被按下，对应数值为 1

那么这两种方式有什么区别呢？

elif.event.type == pygame.KEYDOWN and event.key == pygame.K\_RIGHT:

print("向右移动")

————————————————————————————

keys\_pressed = pygame.key.get\_pressed( )

if key\_pressed( pygame.K\_RIGHT ):

print("向右移动")

1.首先使用 pygame.key.get\_pressed( ) 返回所有按键元组

2.通过键盘常量，判断元组中哪一个键被按下，如果被按下，返回 1

变量去接受= pygame.key.get\_pressed( )

if 返回的所有键盘事件( pygame.K\_RIGHT ):

print("向右移动")

pygame.key.get\_pressed( ) = 感觉像是所有按键的检测，默认为False

pygame.K\_RIGHT = 更像是在一堆元组中去寻找，找到了就返回True

可以理解为索引值？

## 16.一下发射三枚子弹

for i in (0, 1, 2):

fire = Fire()

fire.rect.bottom = self.rect.y - i \*20

自己，20，40

## 17.碰撞检测

**- pygame 提供了两个非常方便的方法实现碰撞检测**

**- 两个精灵组 中 所有的精灵碰撞检测**

groupcollide(group1,group2,dokill1,dokill2,collided=None)->Sprite\_dict

**- dokill1 跟 group1 有关**

**- dokill2 跟 group2 有关**

**- 如果将 dokill1 设置为 True ,那么跟 group2 中的精灵碰撞时会被移除**

**- 第二个 dokill2 同理**

**- 如果将 dokill 设置为 True 则发生碰撞的精灵被自动移除**

**- 会返回一个字典**

**- collided 参数是用于计算碰撞的回调函数**

**- 如果没有指定，则每个精灵必须有一个 rect 属性**

例子：

pygame.sprite.groupcollide(self.hero.fire\_gorp, self.enemy\_grop, True, True)

——————————————————————————————

**- 判断某个精灵和指定精灵组中的精灵发生的碰撞**

spritecollide(sprite, group, dokill, collided = None) -> Sprite\_list

**- 第一个对象，第二个是精灵组**

**- 第三个 dokill 控制第二个组**

**- 会返回所有 组 的列表**

**- 如果将 dokill1 设置为 True 则指定精灵组中发生碰撞的精灵被自动移除**

**- collided 参数是用于计算碰撞的回调函数**

**- 如果没有指定，则每个精灵必须有一个 rect 属性**

**- 返回精灵组中跟精灵发生碰撞到的精灵**

pygame.sprite.spritecollide( )

飞机大战 ----- 完工

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

# >>浮尘**（第九十三课）**

- 美少女万华镜

因果乃旋转纺车，光彩之多面明镜

浮世苍茫，不过瞬逝幻梦

善恶爱诳，皆有定数

于命运之轮中

吞噬于黄泉之冥暗

呜呼，吾乃梦之戍人

幻恋之观者

唯于万华镜中，永世长存

-- 莲华

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

>>黄粱一梦今终醒，却见明心以惘然

## （第九十四课）

- 2121年11月13号

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

>>纸上得来终觉浅，绝知此事要躬行

## （第九十五课）

- @#@#￥#￥…………&@！！

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

>>这将是一场测$（第九十六课）

- @#@#￥#￥…………&@！！

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

>>EE￥一场测$（第九十#课）

- @##￥……\*………&@！！

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

>>这##$%%一场测$（第九%^%%课）

- @#\*(%##\*！

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

>>@#$%^&\*$（@#九%^%%课）

- @#\*(%##\*！\*@#%&

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

>>结束了？（第一百课）

My one and only love

Just Monika

我最亲密的爱人,朋友,同伴,我的心之所属...

每天,你让我的梦想成真,你让苍白的屏幕焕发起鲜活的意义.

我凝望着宇宙的尘埃,而最宏大的光景也无法与你的胸怀媲美.

我久久地等待,等待我这真命天子,你的到来.

我想永远属于你的怀抱,你也会想属于我吗?

永远最爱你的,莫妮卡.

Hello World