# 产生一个随机整数，范围自定义  
import random  
  
print(random.randint(1, 100))

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

# 计算直角三角形的第三边。  
# 提示:从键盘提示输入两条直角边长度  
a = int(input("输入第一条直角边长"))  
b = int(input("输入第二条直角边长"))  
print("直角三角形斜边长" + str((a \*\* 2 + b \*\* 2) \*\* 0.5))

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

# 1从键盘的输入包括自己的学号，姓名，年龄，班级，性别，邮箱信息，身高；  
# 最后将这些变量通过格式化的方式打印在屏幕上（选用两种格式化方式打印）；  
# 显示的格式例如:我的姓名是  
#             我的身高xxx米  
#             我的年龄xxx 等  
number = int(input("学号:"))  
name = input("姓名:")  
age = int(input("年龄:"))  
your\_class = input("班级:")  
sex = input("性别:")  
e\_mail = input("邮箱:")  
height = int(input("身高:"))  
  
print("输入格式1\n我的学号是{}\n我的姓名是{}\n我的年龄是{}岁\n我的班级是{}\n我的性别为{}\n我的邮箱地址是{}\n我的身高{}米\n".format(number, name, age, your\_class,  
                                                                                         sex, e\_mail, height))  
print(  
    f"输入格式2\n我的学号是{number}\n我的姓名是{name}\n我的年龄是{age}岁\n我的班级是{your\_class}\n我的性别为{sex}\n我的邮箱地址是{e\_mail}\n我的身高{height}米\n")

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

# 添加学生信息(例如添加5个)  
# 注：每个学生的信息包括：姓名、学号、班级即可  
# 2、删除学生（根据学号删除）  
# 3、查找学生（例如根据学号查询，显示该学生的所有信息）  
# 4、修改学生（例如根据学生姓名找到该学生，修改其学号或班级信息）  
# 5、显示所有学生信息  
# 6、退出  
  
print("=" \* 46)  
print("学生管理系统".center(40))  
print("输入1：表示添加学生".center(40))  
print("输入2：表示删除学生".center(40))  
print("输入3：表示查找学生".center(40))  
print("输入4：表示修改学生".center(40))  
print("输入5：表示查看所有学生".center(40))  
print("输入6：表示退出".center(40))  
print("=" \* 46)  
student = []  
  
while True:  
 id = input("请输入你想要的操作")  
 if id == "1":  
 stu = {}  
 name = input("请输入想姓名")  
 number = input("请输入学号")  
 classroom = input("请输入班级")  
 stu["name"] = name  
 stu["number"] = number  
 stu["classroom"] = classroom  
 student.append(stu)  
 print("添加成功")  
 elif id == "2":  
 del\_flag = 1  
 while del\_flag == 1:  
 number = input("请输入你要删除的学生的学号,如果不知道学号,请先查看全部人(序号-1）,退出删除操作(序号-2)")  
 if number == "-1":  
 print("序号\t姓名\t学号\t班级")  
 for count, i in enumerate(student, 1):  
 print("%s\t\t%s\t\t%s\t\t%s\t" % (count, i["name"], i["number"], i["classroom"]))  
 continue  
 elif number == "-2":  
 print("退出删除操作")  
 break  
 else:  
 for count, i in enumerate(student, 1):  
 if i["number"] == number:  
 student.remove(student[int(count) - 1])  
 print("删除成功")  
 del\_flag = 0  
 if del\_flag == 1:  
 print("删除失败,请输入正确学号")  
 elif id == "3":  
 number = input("请输入你要查找同学的学号")  
 for stud in student:  
 if stud["number"] == number:  
 print("姓名\t学号\t班级")  
 print("%s\t\t%s\t\t%s\t" % (stud["name"], stud["number"], stud["classroom"]))  
 break  
 else: # 注意 for else  
 print("你查找的%s不存在" % number)  
 elif id == "4":  
 name = input("请输入你要修改同学的姓名")  
 for stud in student:  
 if stud["name"] == name:  
 iaa = input("请输入修改的信息编号【1.姓名 2.学号 3.班级】")  
 if iaa == "1":  
 name = input("请输入姓名")  
 stud["name"] = name  
 elif iaa == "2":  
 number = input("请输入学号")  
 stud["number"] = number  
 elif iaa == "3":  
 classroom = input("请输入班级")  
 stud["classroom"] = classroom  
 print("修改成功")  
 break  
 elif id == "5":  
 print("序号\t姓名\t学号\t班级")  
 for count, i in enumerate(student, 1):  
 print("%s\t\t%s\t\t%s\t\t%s\t" % (count, i["name"], i["number"], i["classroom"]))  
 elif id == "6":  
 print("退出")  
 break

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

def Fb(n):

a, b = 1, 1

count = 0

while (count != n - 2):

a, b = b, a + b

count += 1

return b

n = eval(input('please input n:'))

print(Fb(n))

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

# 编写一函数Prime(n)，对于已知正整数n，判断该数是否为素数，如果是素数，返回True,否则返回False。

from math import sqrt

def Prime(n):

s = 0

tag = True

if n > 0:

for i in range(2, int(sqrt(n)) + 1):

if (n % i) == 0:

s += 1

if s >= 1:

tag = False

return tag

primes = []

for i in range(1, 101):

if Prime(i):

primes.append(i)

print(primes)

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

# 求润年Leapyear(n)，输入年份，统计该年是不是润年，如果是润年，返回True；否则返回False。

def Leapyear(n):

if n % 4 == 0 and n % 100 != 0 or n % 400 == 0:

return True

else:

return False

if Leapyear(eval(input('请输入年份：'))):

print('该年份是闰年')

else:

print('该年份不是闰年')

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

# 设计一个函数，要求对字典{5：“five”，2:”two”，3：“three”，1：“one”}中的键按照有小到大的顺序，一次输出每个键的值

def my\_sort(dic):

for key, value in sorted(dic.items(), key=lambda x: x[0]):

print(value)

dict1 = {5: "five", 2: "two", 3: "three", 1: "one"}

my\_sort(dict1)

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

# 1定义一个学生类：成员属性包括 姓名（私有）、年龄（私有）

# 成绩（语文，数学，英语)（私有）

# 1、成员方法：

# a、获取学生的姓名

# b、获取学生的年龄

# c、显示该学生的姓名 年龄，成绩

# d、返回3门科目中最高的分数

# 2、设置一个静态方法：

# e、返回当前产生的学生对象数量

# 学生类

class Student:

count = 0

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

self.\_\_name = name

self.\_\_age = age

self.\_\_course = cou

Student.count += 1

# 获取学生的姓名

def get\_name(self):

print('学生的姓名是', self.\_\_name)

# 获取学生的年龄

def get\_age(self):

print('学生的年龄是', self.\_\_age, '岁')

# 显示该学生的姓名 年龄，成绩

def show\_list(self):

self.get\_name()

self.get\_age()

for key, value in self.\_\_course.items():

print(key + "的成绩为", value, "分")

# 返回3门科目中最高的分数

def get\_course(self):

print('该学生分数最高的科目是', max(self.\_\_course, key=self.\_\_course.get), '\n分数为',

self.\_\_course[max(self.\_\_course, key=self.\_\_course.get)], '分')

'''

笔记：

python的max()函数解决

获取字典中value最大的值对应的key ：max(self.\_\_course, key=self.\_\_course.get)

获取字典中value最大值 : self.\_\_course[max(self.\_\_course, key=self.\_\_course.get)]

'''

# 返回当前产生的学生对象数量

@staticmethod

def students\_number():

print('现在总共有', Student.count, '位学生')

cjy = Student('Chen Jiongyu', 20, {'语文': 70, '数学': 100, '英语': 80})

cjy.get\_name()

cjy.get\_age()

cjy.show\_list()

cjy.get\_course()

cjy.students\_number()

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

# 基于学生类再派生一个 小学生类

# 成员属性：班级

# 成员方法：显示该学生的姓名 年龄，成绩,班级

# 学生类

class Student:

count = 0

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

self.\_\_name = name

self.\_\_age = age

self.\_\_course = cou

Student.count += 1

# 获取学生的姓名

def get\_name(self):

print('学生的姓名是', self.\_\_name)

# 获取学生的年龄

def get\_age(self):

print('学生的年龄是', self.\_\_age, '岁')

# 显示该学生的姓名 年龄，成绩

def show\_list(self):

self.get\_name()

self.get\_age()

for key, value in self.\_\_course.items():

print(key + "的成绩为", value, "分")

# 小学生类

class pupil(Student):

# 成员属性：班级

def \_\_init\_\_(self, name, age, cou, group):

Student.\_\_init\_\_(self, name, age, cou)

self.\_\_group = group

# 成员方法：显示该学生的姓名 年龄，成绩,班级

def show\_list(self):

Student.show\_list(self)

print('学生的班级为', self.\_\_group)

cjy = pupil('Chen Jiongyu', 20, {'语文': 70, '数学': 100, '英语': 80}, '物联网1901班')

cjy.show\_list()

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

# 实现一个学生管理系统（用函数实现以下每个功能，用类描述学生）：

# 要求有以下功能：

# 1、添加学生(添加5个先)

# 2、删除学生（根据学号删除）

# 3、查找学生（根据学号查询，显示该学生的所有信息）

# 4、修改学生（根据学生姓名找到该学生进行信息修改，学号和班级）

# 5、查看所有学生

# 6、退出

# 每个学生的信息包括：姓名 学号 班级

import os

# 学生类

class Student:

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

self.name = name

self.number = number

self.group = group

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

return str(self.name) + "\t" + str(self.number) + "\t" + str(self.group)

"""

笔记:

\_\_str\_\_（）这个特殊方法会在尝试将对象转换为字符串的时候调用

它的作用可以用来指定对象转换为字符串的结果 （print函数）

"""

# 学生管理系统类

class StudentManager(Student):

students = []

# 1、添加学生

@classmethod

def addStudent(cls):

name = input("please input name:")

number = input("please input number:")

group = input("please input group:")

student = Student(name, number, group)

cls.students.append(student)

cls.write()

# 2、删除学生（根据学号删除）

@classmethod

def delete(cls):

number = input("please input the number what you want delete:")

for student in cls.students:

if student.number == str(number):

cls.students.remove(student)

cls.write()

break

else:

print("there is not have the student")

cls.delete()

# 3、查找学生（根据学号查询，显示该学生的所有信息）

@classmethod

def search(cls):

number = input("please input the number what you want search:")

for student in cls.students:

if student.number == str(number):

print("Name\tNumber\tGroup")

print(student)

break

else:

print("there is not have the student")

# 4、修改学生（根据学生姓名找到该学生进行信息修改，学号和班级）

@classmethod

def update(cls):

name = input("please input the name what you want update")

for student in cls.students:

if student.name == name:

index = cls.students.index(student)

cls.students[index].number = str(input("Please input your new number:"))

cls.students[index].group = str(input("Please input your new group:"))

cls.write()

break # 考虑到学生有同名的情况

else:

print("there is not have the student")

"""

笔记:

python String index（）函数返回找到指定子字符串的最低索引。 如果未找到子字符串，则引发ValueError 。

"""

# 5、查看所有学生

@classmethod

def display(cls):

print("Name\tNumber\tGroup")

for student in cls.students:

print(student)

# 将学生数据存储到student.dat

@classmethod

def write(cls):

with open("student.dat", "w") as f:

for student in cls.students:

f.write(student.name + "," + str(student.number) + "," + str(student.group) + "\n")

# 初始化从student.dat读取已经有的学生数据

@classmethod

def read(cls):

a = os.path.exists('student.dat')

if a:

with open("student.dat", "r") as f:

line = f.readline()

while line != '':

lines = line.rstrip().split(',')

student = Student(lines[0], lines[1], lines[2])

cls.students.append(student)

line = f.readline()

else:

print("file open error")

# 学生管理系统菜单

@staticmethod

def menu():

print("\*" \* 30)

print("1:add")

print("2:delete")

print("3:search")

print("4:update")

print("5:display")

print("6:exit")

print("\*" \* 30)

# 学生管理系统运行

@staticmethod

def run():

StudentManager.read()

while True:

StudentManager.menu()

operator = input("please input your choice:")

if operator == "1":

StudentManager.addStudent()

if operator == "2":

StudentManager.delete()

if operator == "3":

StudentManager.search()

if operator == "4":

StudentManager.update()

if operator == "5":

StudentManager.display()

if operator == "6":

print("成功退出学生管理系统")

break

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

StudentManager.run()

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

# 输入一个正整数n，计算 1!+2!+3!+...+n! 的和并输出

n = int(input())

tmp = 1

count = 0

i = 1

while n >= i:

tmp = tmp \* i

count = count + tmp

i = i + 1

print(count)

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

# 判断一个列表是否存在重复元素，如果存在，则全部输出

# 同时，输出重复个数最多的元素（注：要考虑到存在数量相同的元素情况）

hash\_table = {}

nums = [0, 0, 1, 2, 3, 4, 4, 4, 5, 6, 7, 7, 7, 8, 8, 9]

for i in nums:

hash\_table[i] = 1 + hash\_table.get(i, 0)

print(hash\_table)

max\_count = 0

print("重复的元素有")

for key, value in hash\_table.items():

if value != 1:

print(key, end=" ")

if value > max\_count: max\_count = value

print("\n重复出现最多次数的元素是")

for key, value in hash\_table.items():

if value == max\_count:

print(key, end=" ")

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

# 某个公司采用公用电话传递数据，数据是四位的整数，在传递过程中是加密的

# 加密规则如下：每位数字都加上5,然后用和除以10的余数代替该数字

# 再将第一位和第四位交换，第二位和第三位交换。试输入一个数，并求出对应的数字

# 方法一/二选其一运行

# 方法一

a = int(input('输入四个数字:'))

b = [a % 10, a % 100 // 10, a % 1000 // 100, a // 1000] # 个十百千

for i in range(4):

b[i] += 5

b[i] %= 10

for i in range(2):

b[i], b[3 - i] = b[3 - i], b[i]

for i in range(3, -1, -1):

print(str(b[i]), end="")

# 方法二

# a = int(input('输入四个数字:'))

# b = [a % 10, a % 100 // 10, a % 1000 // 100, a // 1000] # 个十百千

#

# for i in range(4):

# b[i] += 5

# b[i] %= 10

# for i in range(4):

# print(str(b[i]), end="")

# 1、4交换，2、3交换其实就是倒序输出

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

# -- coding: utf-8 --

# 1、设计一个函数，生成指定长度为6位数的验证码，验证码由数字和大小写英文字母构成的随机字符串。

import random

def yanzhengma():

length = 6

x = 0

while x < length:

s = random.choice(range(1, 4))

if s == 1:

a = random.choice(range(1, 10))

print(a, end='')

x += 1

if s == 2:

b = random.choice(range(65, 90))

print(chr(b), end='')

x += 1

if s == 3:

c = random.choice(range(97, 122))

print(chr(c), end='')

x += 1

yanzhengma()

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

# -- coding: utf-8 --

# 2、数列求和

# 描述

# 用户输入一个小于10的正整数，求1 + 12 + 123 + 1234 + ……的前n项的和输入

# 一个正整数n（测试数据保证小于10）

# 输出 数列的和

#

# 输入输出示例

#

# 输入

# 5

# 输出

# 13715

def SUM(n):

s = 0

i = 1

s += i

while i % 10 < int(n):

i = i \* 10 + i % 10 + 1

s += i

return s

n = eval(input("请输入一个正整数n(n<10):"))

while n >= 10:

n = eval(input("n的范围错误,请重新输入:"))

print(SUM(n))

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

# -- coding: utf-8 --

# 用户登录

#

# 描述

# 有字典如下：

# dic = {'admin': '123456', 'administrator': '12345678', 'root': 'password'}

#

# 实现用户输入用户名和密码，当用户名与密码和字典中的键值对匹配时，显示“登录成功”，

# 否则显示“登录失败”，登录失败时允许重复输入三次。

#

# 输入格式:

# 在两行中分别输入用户名和密码

#

# 输出格式:

# "登录成功"或"登录失败"

dic = {'admin': '123456', 'administrator': '12345678', 'root': 'password'}

timer = 0 # 循环次数

while timer < 3:

name = input("用户名:")

password = input("密码:")

n = 0

for key, value in dic.items():

if key == name and value == password:

break

else:

n += 1

if n < 3:

print("登录成功")

break

else:

print("登录失败")

timer += 1

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

# 【实例】小学生计算器。利用函数实现能够求100以内的加减法、10以内的乘除法。

import random

def compute():

op = random.choice('+-\*/')

if op == '\*':

op1 = random.randint(0, 10)

op2 = random.randint(0, 10)

result = op1 \* op2

elif op == '/':

op2 = random.randint(0, 10)

m = random.randint(0, 10)

op1 = op2 \* m

result = op1 / op2

else:

op1 = random.randint(0, 100)

op2 = random.randint(0, 100)

result = op1 + op2

if op == '-':

if op1 < op2:

op1, op2 = op2, op1

result = op1 - op2

print(op1, op, op2, '=', end='')

return result

count = 0

for i in range(10):

answer = compute()

yAnswer = int(input())

if yAnswer == answer:

count = count + 10

print("your score is :", count)

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

# 【实例】利用函数实现将2~20之间的所有素数

import math

def prime(num): # 判断m是否是素数，是返回1，不是返回0

i = 2

while i <= math.sqrt(num):

if num % i == 0:

return 0

i = i + 1

return 1

print("2~20之间的素数有：")

for m in range(2, 20):

if prime(m) == 1:

print(m, " ", end='')

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

# 【实例】如果一个 3 位数等于其各位数字的立方和，则称这个数为水仙花数。例如：153 = 1\*\*3 + 5\*\*3 + 3\*\*3，因此 153 就是一个水仙花数。利用函数求1000以内的水仙花数（3位数）。

def Narcissistic(num):

a = num // 100

b = (num - a \* 100) // 10

c = (num - a \* 100 - b \* 10)

if num == pow(a, 3) + pow(b, 3) + pow(c, 3):

return 1

else:

return 0

print("三位数的水仙花数有：")

for i in range(100, 1000):

if Narcissistic(i) == 1:

print(i)

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

# 【实例】猜数字

# 本案例的任务：系统随机生成一个1.100之间的整数

# 然后让用户猜测该数字，如果用户猜的数据比答案大

# 提示太大了，如果小，则提示太小了， 正确则输出用户猜测正确。

# 案例分析：根据案例需要实现的功能

# 可以将任务分解为两个子任务：产生数字和用户猜数字，并用两个函数实现

import random

def main():

# 1.系统生成一个随机数并放到number中。

number = newNumber()

# 2.让用户猜测number的值到底是几。

guessNumber(number)

def newNumber():

number = random.randint(1, 101)

return number

def guessNumber(number):

yAnswer = int(input("enter a integer between 0-100:"))

if yAnswer > number:

print("too big")

elif yAnswer < number:

print("too small")

else:

print("right")

if \_\_name\_\_ == '\_\_main\_\_':

main()

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

# 【实例】猜数字

# 程序只给了用户一次机会，现在考虑给用户10次机会

# 因此程序中增加了一个函数guessTimes(number)

# 该函数用一个循环来调用guessNumber(number)函数，循环次数为10。

import random

def main():

# 1.系统生成一个随机数并放到number中。

number = newNumber()

# 2.让用户猜测number的值到底是几。

guessTime(number)

def newNumber():

number = random.randint(1, 101)

return number

def guessNumber(number):

yAnswer = int(input("enter a integer between 0-100:"))

if yAnswer > number:

print("too big")

elif yAnswer < number:

print("too small")

else:

print("right")

def guessTime(number):

for i in range(10):

guessNumber(number)

if \_\_name\_\_ == '\_\_main\_\_':

main()

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

# 【实例】猜数字

# 目前guessNumber函数中对于猜对猜错仅仅给出提示信息

# 没有返回任何值给调用它的函数guessTime，这样无法控制何时退出循环体

# 考虑修改guessNumber函数

# 如果猜对，除了输出提示信息外，还返回1

# 如果猜错，提示太大或者太小，并且返回0

# 这样可以在guessTimes中根据guessNumber的返回值来判断是否退出循环。

import random

def main():

number = newNumber()

times = 10

guessTime(number, times)

def newNumber():

number = random.randint(1, 101)

return number

def guessNumber(number):

yAnswer = int(input("enter a integer between 0-100:"))

if yAnswer > number:

print("too big")

return 0

elif yAnswer < number:

print("too small")

return 0

else:

print("right")

return 1

def guessTime(number, times):

for i in range(times):

if guessNumber(number) == 1:

print("你一共猜了", i + 1, "次")

break

if i > times:

print("sorry,你没有猜对，正确答案是", number)

if \_\_name\_\_ == '\_\_main\_\_':

main()

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

# 分析：我国目前将垃圾分为四种类型：

# 可回收垃圾、厨余垃圾、有害垃圾和其他垃

# 用列表变量将每种类型的常见垃圾罗列出来，

# 定义三个全局列表变量用来存储可回收垃圾、厨余垃圾、有害垃圾，其他的不在这三个列表中的就都属于其他垃圾。

# 系统首先根据用户输入的垃圾名称判断属于四种垃圾中的哪一类，然后执行对应的操作。

# 垃圾分类

waste\_recycle = ["paper", "glasses", "bottle", "plastic"] # 可回收垃圾：

waste\_kitchen = ["peel", "vegetable", "leftover", "flower"] # 厨余垃圾

waste\_harmful = ["battery", "light tube", "daily chemical"] # 有害垃圾

def handle\_harmful():

print("This is harmful waste and please put it into the red trash can ")

def handle\_kitchen():

print("This is kitchen waste and please put it into the green trash can ")

def handle\_recycle():

print("This is recycle waste and please put it into the blue trash can ")

def handle\_others():

print("This is other waste and please put it into the yellow trash can ")

def do\_with\_waste(waste):

if waste in waste\_harmful:

handle\_harmful()

elif waste in waste\_kitchen:

handle\_kitchen()

elif waste in waste\_recycle:

handle\_recycle()

else:

handle\_others()

waste = input("enter waste name:")

do\_with\_waste(waste)

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

# 【实例】编写程序，输入圆的半径，输出圆的面积和周长

import math

# Circle类的定义

class Circle:

# \_\_init\_\_方法是一个特殊的方法：构造方法

# 当创建对象时，系统自动调用\_\_init\_\_方法，它没有返回值

# 一般都在构造方法中初始化实例属性

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

self.\_\_radius = r

# area、perimeter是普通的实例方法,是圆类对外的接口

# 通过对象名.方法名(参数列表)来调用

def area(self):

return math.pi \* self.\_\_radius \*\* 2

def perimeter(self):

return 2 \* math.pi \* self.\_\_radius

r1 = float(input("请输入圆的半径："))

c1 = Circle(r1)

print(c1.area())

print(c1.perimeter())

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

# 【实例】现在养狗的人越来越多

# 狗的主人都会给自己的狗起一个名字，每一条狗都有年龄，每一条狗都会跑，会叫。

# 设计一个狗类，完成对狗的抽象和封装

# 类Dog

class Dog:

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

self.\_\_name = name # 私有成员name的初始化

self.\_\_age = age # 私有成员age的初始化

def run(self):

print(self.\_\_name + " is running")

print("It is " + str(self.\_\_age) + " old years")

def bark(self):

print(self.\_\_name + " is barking:汪汪，汪汪...")

# 测试代码

dog1 = Dog("黄豆", 4) # 创建一个对象dog1

dog1.run() # 在类外调用公有成员方法run

dog1.bark() # 在类外调用公有成员方法bark

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

from types import MethodType

class Student(object):

pass

def set\_name(self, name):

self.name = name

def set\_age(self, age):

self.age = age

def set\_sc(self, x, y):

self.x = x

self.y = y

x1 = Student() # 类的实例化

x2 = Student()

x1.set\_age = MethodType(set\_age, x1) # 实例绑定一个方法

x2.set\_age = MethodType(set\_age, x2)

Student.set\_sc = MethodType(set\_sc, Student) # 另一种绑定

Student.set\_name = MethodType(set\_name, Student) # 给此类绑定一个方法

x1.set\_age(12) # 初始化

x2.set\_age(13)

x1.set\_sc("xmc", "xu")

Student.set\_name("hhh")

print(x1.age) # 12

print(x2.age) # 13

print(x1.x, x1.y) # xmc xu

print(x1.name) # hhh

print(x2.name) # hhh

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

# python中子类继承父类成员变量之间的取值逻辑

class Person():

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

self.name = "jasn"

self.age = '18'

self.sex = sex

def talk(self):

print("i want to speak something to yo!!")

class Chinese(Person):

def \_\_init\_\_(self, name, age, sex, language):

Person.\_\_init\_\_(self, name, age, sex) # 用父类的name,age，sex 覆盖掉子类的属性

self.age = age # 覆盖掉了父类的age,取值为子类实例中传入age参数

self.language = "chinese"

def talk(self):

print("我说的是普通话！！")

Person.talk(self)

obj = Chinese("nancy",'18','male',"普通话")

print(obj.name) # 对应场景A

print(obj.age) # 对应场景B

print(obj.language) # 对应场景C

obj.talk() # 对应场景D

# 总结：

# A:若父类中初始化了成员变量，子类调用父类构造方法未覆盖属性（self.name），则调用子类属性时取值为父类中初始化的成员变量；

# B:若父类中初始化了成员变量，若子类调用父类构造方法覆盖属性（self.age）则取值为子类实例中传入参数

# C:若父类未初始化该成员变量，则无论子类中有无进行对父类构造方法进行属性的覆盖，均取子类实例中传入的参数

# D:对于方法,如果子类有这个方法则直接调用，如果子类没有则去父类查找。父类没有则报错

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

class animal: # 定义父类

country = 'china' # 这个叫类的变量

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

self.name = name # 这些又叫数据属性

self.age = age

def walk(self): # 类的函数，方法，动态属性

print('%s is walking' % self.name)

def say(self):

pass

class people(animal): # 子类继承父类

pass

class pig(animal): # 子类继承父类

pass

class dog(animal): # 子类继承父类

pass

haha = people('haha', 60) # 实例化一个对象

print(haha.name)

haha.walk()

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

# 五、派生类

# 1. 在父类的基础上产生子类，产生的子类就叫做派生类

# 2. 父类里没有的方法，在子类中有了，这样的方法就叫做派生方法。

# 3. 父类里有，子类也有的方法，就叫做方法的重写（就是把父类里的方法重写了)

class Hero:

def \_\_init\_\_(self, nickname, aggressivity, life\_value):

self.nickname = nickname

self.aggressivity = aggressivity

self.life\_value = life\_value

def attack(self, enemy):

print('Hero attack')

class Garen(Hero):

camp = 'Demacia'

def attack(self, enemy): # self=g1,enemy=r1

# self.attack(enemy) #g1.attack(r1)，这里相当于无限递归

Hero.attack(self, enemy) # 引用 父类的 attack，对象会去跑 父类的 attack

print('from garen attack') # 再回来这里

def fire(self):

print('%s is firing' % self.nickname)

class Riven(Hero):

camp = 'Noxus'

g1 = Garen('garen', 18, 200)

r1 = Riven('rivren', 18, 200)

g1.attack(r1)

print(g1.camp)

print(r1.camp)

g1.fire()

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

"""

当类是新式类，多继承的情况下，在要查找的属性不存在时，会按照广度优先来查找

默认从左侧分支查找，如 ：class G(D,E,F):会先从 D 的分支查找 找到 D-C 然后从E-B -F 顺序查找

最后再查找 几个类所共同继承的顶级父类，如果顶级没有，则报错

"""

class A:

def test(self):

print('A中的test')

pass

class B(A):

def test(self):

print('B中的test')

pass

class C(A):

def test(self):

print('C中的test')

pass

class D(C):

def test(self):

print('D中的test')

pass

class E(B):

def test(self):

print('E中的test')

pass

class F(A):

def test(self):

print('F中的test')

pass

class G(D, E, F):

def test(self):

print('G中的test')

pass

obj = G()

obj.test()

'''

第一次 E中的test

注释D，第二次：C中的test

注释C，第三次：E中的test

注释E，第四次：B中的test

注释B，第五次，F中的test

注释F，第六次，A中的test

所以新式类的继承关系（查找顺序）为:

obj-->G --> D-->C -->E -->B -->F -->A

也称广度优先

'''

print(F.\_\_mro\_\_) # 只有新式才有这个属性可以查看线性列表，经典类没有这个属性

# 新式类继承顺序:F->D->B->E->C->A

# 经典类继承顺序:F->D->B->A->E->C

# python3中统一都是新式类

# python2中才分新式类与经典类 class A(object) 为经典类

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

"""

类的组合，即在类实例化时，将另一个类的实例作为参数传入，这样可以将两个实例关联起来。

当类之间有显著不同，并且较小的类是较大的类所需要的组件时，用组合比较好。

比如学生类和课程类,学生需要选课,可以选一门或者多门,在没选课之前,学生类和课程类无关

当类之间有很多相同的属性，提取这些统统的属性做成基类，用继承比较好。

"""

# 学校类 是学生类和老师类的父类

class School:

schoolName = '学习训练营'

def \_\_init\_\_(self, name, sex, age, birth):

self.name = name

self.sex = sex

self.age = age

self.birth = birth # 类的组合 ，将生日在初始化的时候就被调用

# 学生类

class students(School):

def \_\_init\_\_(self, name, sex, age, id, birth):

School.\_\_init\_\_(self, name, sex, age, birth)

self.id = id

self.course = []

def Learn(self):

print('%s,正在学习' % self.name)

# 老师类

class teacher(School):

def \_\_init\_\_(self, name, sex, age, level, birth):

School.\_\_init\_\_(self, name, sex, age, birth)

self.id = id

self.level = level

self.course = []

def Teach(self):

print('%s,正在上课' % self.name)

# 课程类 是学生和老师的组合类

class Course:

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

self.name = name

self.price = price

self.cycle = cycle

def all\_info(self):

print('课程信息：%s---%s---%s' % (self.name, self.price, self.cycle))

# 生日类，是学生类和老师类的组合类

class date:

def \_\_init\_\_(self, year, month, day):

self.year = year

self.mouth = month

self.day = day

def birthday(self):

print('出生日期为：%s-%s-%s' % (self.year, self.mouth, self.day))

# 学生

birth = date('2000', 12, 19)

stu1 = students('jasn', 'male', 18, '002', birth)

stu1.Learn()

stu1.birth.birthday()

# 老师

birth = date('1993', 12, 23) # 定义生日对象

python = Course('python', 13000, '3mon') # 定义课程对象

linux = Course('linux', 13000, '3mon')

teach1 = teacher('nancy', 'male', 18, 10, birth)

teach1.Teach()

teach1.birth.birthday()

teach1.course.append(python) # 将课程添加到老师中，这也是一种组合关系

# 查看课程

for course in teach1.course:

course.all\_info()

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

# 九、抽象类和归一化设计

# 接口提取了一群类共同的函数，可以把接口当做一个函数的集合。然后让子类去实现接口中的函数。

# 这么做的意义在于归一化，什么叫归一化，就是只要是基于同一个接口实现的类，

# 那么所有的这些类产生的对象在使用时，从用法上来说都一样。

# 归一化的好处：

# 1、归一化让使用者无需关心对象的类是什么，只需要的知道这些对象都具备某些功能就可以了，这极大地降低了使用者的使用难度。

# 2、归一化使得高层的外部使用者可以不加区分的处理所有接口兼容的对象集合

class Interface: # 定义接口Interface类来模仿接口的概念，python中压根就没有interface关键字来定义一个接口。

def read(self): # 定接口函数read

pass

def write(self): # 定义接口函数write

pass

class Txt(Interface): # 文本，具体实现read和write

def read(self):

print('文本数据的读取方法')

def write(self):

print('文本数据的读取方法')

class Sata(Interface): # 磁盘，具体实现read和write

def read(self):

print('硬盘数据的读取方法')

def write(self):

print('硬盘数据的读取方法')

class Process(Interface):

def read(self):

print('进程数据的读取方法')

def write(self):

print('进程数据的读取方法')

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

'''

我们有一个汽车参数接口，里面定义了汽车所有的必须需要的参数，然后由本田汽车的类，奥迪汽车的类，大众汽车的类，

他们都实现了汽车接口，这样就好办了，大家选择汽车的时候，只需要将不同汽车的参数来进行对比，就能评判出一个车的好坏。

'''

import abc

# 抽象类：本质还是类，与普通类额外的特点的是：加了装饰器的函数，子类必须实现他们

class Car(metaclass=abc.ABCMeta):

@abc.abstractmethod

def name(self): # 如果子类没有name 函数，主动抛出异常

'必须定义车名字'

pass

@abc.abstractmethod

def Model(self): # 如果子类没有Model 函数，主动抛出异常

'必须定义车型号'

pass

@abc.abstractmethod

def drive(self):

' 必须定义驱动方式 '

pass

@abc.abstractmethod

def Fuel(self):

'必须定义油耗'

pass

@abc.abstractmethod

def engine(self):

'必须定义发动机'

pass

@abc.abstractmethod

def power(self):

'必须定义功率'

pass

class AudiCar(Car):

name = '一汽-大众奥迪-奥迪Q5L'

def Model(self):

print('中型SUV')

def drive(self):

print('前置四驱7挡双离合')

def Fuel(self):

print('7.1L/100km(工信部)')

def engine(self):

print('2.0T 190马力 L4')

def power(self):

print('140kW/320N.m')

class BMWCar(Car):

name = '宝马'

def Model(self):

print('中型SUV')

def drive(self):

print('前置四驱7挡双离合')

def Fuel(self):

print('5.1L/100km(工信部)')

def engine(self):

print('3.0T 190马力 L4')

def power(self):

print('1990kW/320N.m')

Audi = AudiCar()

BMW = BMWCar()

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

with open('student.dat', 'r') as fp:

data = fp.readlines()

print(data)

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

# -- coding: utf-8 --

import xlrd

# 读取本地csv文件,打开excel文件（默认是rb方式打开）

# xlrd更新到了2.0版本，只支持.xls文件，不支持.xlsx文件

myWorkbook = xlrd.open\_workbook('xlrdtest.xls')

mySheet = myWorkbook.sheet\_by\_index(0) # 通过索引顺序获得

# 获取行数和列数

nrows = mySheet.nrows

ncols = mySheet.ncols

print("The rows: %d" % nrows)

print("The cols: %d" % ncols)

# 获取一行和一列

for i in range(nrows):

myRowValues = mySheet.row\_values(i)

print(myRowValues)

for j in range(ncols):

myColValues = mySheet.col\_values(j)

print(myColValues)

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

# -- coding: utf-8 --

import pickle

class Person:

def \_\_init\_\_(self, n, a):

self.name = n

self.age = a

def show(self):

print(self.name + "\_" + str(self.age))

aa = Person("张三", 20)

i = 13000000

list1 = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]

tu1 = (-5, 10, 8)

dic1 = {'a': 'apple', 'b': 'banana', 'g': 'grape', 'o': 'orange'}

with open('序列化操作pickle.dat', 'wb') as f:

pickle.dump(5, f, 0)

pickle.dump(i, f, 0)

pickle.dump(list1, f, 0)

pickle.dump(tu1, f, 0)

pickle.dump(dic1, f, 0)

pickle.dump(aa, f, 0)

with open('序列化操作pickle.dat', 'rb') as f:

n = pickle.load(f)

while n > 0:

print(pickle.load(f))

n -= 1

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

import requests

from lxml import etree

import time

# 获取网页源代码

def get\_page(url):

'''

查找headers:

1、打开F12后点击Network后随意登录一个网站

2、点击官网标志

3、点击Headers往下便能找到User-Agent

'''

headers = {

'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/87.0.4280.88 Safari/537.36'}

response = requests.get(url=url, headers=headers)

html = response.text

# print(type(html))

return html

# 解析网页源代码

def parse\_page1(html):

html\_elem = etree.HTML(html)

'''

etree.HTML()：

调用HTML类对HTML文本进行初始化，成功构造XPath解析对象，同时可以自动修正HTML文本（标签缺少闭合自动添加上）

'''

pages = html\_elem.xpath('//\*[@id="app"]/div/div/div[1]/dl/dd/div/div')

'''

快速获取xpath：

先F12,然后用Ctrl+Shift+C快速选中Web界面的元素，然后在元素上右键-->复制-->复制XPath

'''

for i in pages:

titles = i.xpath('./div[1]/p[1]/a/text()')

infos = i.xpath('./div[1]/p[2]/text()')

times = i.xpath('./div[1]/p[3]/text()')

scores = i.xpath('./div[2]/p/i/text()')

'''

text()函数

获取元素文本内容

'''

print("----------------------------------------")

print("电影名：", titles[0], '\n', "", infos[0].strip(), '\n', "", times[0].strip(), '\n', "",

"".join(scores) + '分')

time.sleep(1)

# 开始爬取网页

def crawl():

url = 'https://maoyan.com/board/4?offset={0}'

'''

用Request时对于大部分的页面来说只需要两个参数，一个是url也就是请求的网址，另外一个就是headers，

而headers中最重要的就是user-agent，其他的参数都没有这两个参数重要。

所以可以只留下这两个参数再去请求页面，如果请求不下来再去尝试添加其他参数。

'''

print('开始爬取')

for page in range(0, 110, 10): # 0 25 50 75

# print('正在爬取第 ' + str(page+1) + ' 页至第 ' + str(page+25) + ' 页......')

html = get\_page(url.format(page))

data = parse\_page1(html)

time.sleep(1) # 延迟1秒

print('结束爬取')

crawl()

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

import requests

from lxml import etree

import time

import re

import os

# 获取网页源代码

def get\_page(url):

'''

查找headers:

1、打开F12后点击Network后随意登录一个网站

2、点击官网标志

3、点击Headers往下便能找到User-Agent

'''

headers = {

'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/87.0.4280.88 Safari/537.36'}

response = requests.get(url=url, headers=headers)

html = response.text

# print(type(html))

return html

# 解析网页源代码

def parse\_page1(html):

html\_elem = etree.HTML(html)

'''

etree.HTML()：

调用HTML类对HTML文本进行初始化，成功构造XPath解析对象，同时可以自动修正HTML文本（标签缺少闭合自动添加上）

'''

'''

快速获取xpath：

先F12,然后用Ctrl+Shift+C快速选中Web界面的元素，然后在元素上右键-->复制-->复制XPath

'''

movies\_urls = html\_elem.xpath('//h4[@class="video-name one-line"]/a[@href]/@href')

movies\_names = html\_elem.xpath('//h4[@class="video-name one-line"]/a[@href]/text()')

'''

text()函数

获取元素文本内容

'''

print("----------------------------------------")

for movies\_url, movies\_name in zip(movies\_urls, movies\_names):

print(movies\_url, movies\_name)

# 获取文本

movie\_id\_string = requests.get(movies\_url).text

print(movie\_id\_string)

# . 表示所有的,但是只匹配一个,除了/n

# \* 表示按照前面的方式进行匹配

movie\_mp4\_url = re.search('source src="(.\*)" type', movie\_id\_string).group(1)

print(movie\_mp4\_url)

# 因为视频是二进制文件,所以需要获取content

mp4 = requests.get(movie\_mp4\_url).content

# 写入文件夹

with open('./movie/%s.mp4' % movies\_name, 'wb') as data\_file:

data\_file.write(mp4)

time.sleep(1)

# 开始爬取网页

def crawl():

url = 'https://www.maoyan.com/news?showTab=3&showTab=3&offset={0}'

'''

用Request时对于大部分的页面来说只需要两个参数，一个是url也就是请求的网址，另外一个就是headers，

而headers中最重要的就是user-agent，其他的参数都没有这两个参数重要。

所以可以只留下这两个参数再去请求页面，如果请求不下来再去尝试添加其他参数。

'''

print('开始爬取')

os.makedirs("movie")

for page in range(0, 160, 16): # 0 25 50 75

# print('正在爬取第 ' + str(page+1) + ' 页至第 ' + str(page+25) + ' 页......')

html = get\_page(url.format(page))

data = parse\_page1(html)

time.sleep(1) # 延迟1秒

print('结束爬取')

crawl()

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

import requests

from lxml import etree

import time

import os

# 获取网页源代码

def get\_page(url):

'''

查找headers:

1、打开F12后点击Network后随意登录一个网站

2、点击官网标志

3、点击Headers往下便能找到User-Agent

'''

headers = {

'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/87.0.4280.88 Safari/537.36'}

response = requests.get(url=url, headers=headers)

html = response.text

# print(type(html))

return html

# 解析网页源代码

def parse\_page1(html, page):

html\_elem = etree.HTML(html)

result = etree.tostring(html\_elem) # 输出修正后的html

'''

etree.HTML()：

调用HTML类对HTML文本进行初始化，成功构造XPath解析对象，同时可以自动修正HTML文本（标签缺少闭合自动添加上）

'''

# pages = html\_elem.xpath('//\*[@id="app"]/div/div/div[1]/dl/dd/div/div')

'''

快速获取xpath：

先F12,然后用Ctrl+Shift+C快速选中Web界面的元素，然后在元素上右键-->复制-->复制XPath

'''

address = html\_elem.xpath('//div[@class="pl2"]/a/@href')

title = html\_elem.xpath('//div[@class="pl2"]/a/@title')

info = html\_elem.xpath('//p[@class="pl"]/text()')

num = html\_elem.xpath('//div[@class="star clearfix"]/span[@class="rating\_nums"]/text()')

pic = html\_elem.xpath('//a[@class="nbg"]/img/@src')

'''

text()函数

获取元素文本内容

'''

os.makedirs("books")

for j in range(0, 25):

print(pic[j])

with open('books\\' + str((page \* 25) + j + 1) + '.jpg', 'wb') as fp:

pi = requests.get(pic[j])

fp.write(pi.content)

print("保存第%d本书成功" % int((page \* 25) + j + 1))

time.sleep(1) # 延迟1秒

# 开始爬取网页

def crawl():

url = 'https://book.douban.com/top250?start={0}'

'''

用Request时对于大部分的页面来说只需要两个参数，一个是url也就是请求的网址，另外一个就是headers，

而headers中最重要的就是user-agent，其他的参数都没有这两个参数重要。

所以可以只留下这两个参数再去请求页面，如果请求不下来再去尝试添加其他参数。

'''

print('开始爬取')

for page in range(0, 110, 10): # 0 25 50 7 5

# print('正在爬取第 ' + str(page+1) + ' 页至第 ' + str(page+25) + ' 页......')

html = get\_page(url.format(page))

data = parse\_page1(html, page)

time.sleep(1) # 延迟1秒

print('结束爬取')

crawl()

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

import requests

from lxml import etree

import time

# 获取网页源代码

def get\_page(url):

'''

查找headers:

1、打开F12后点击Network后随意登录一个网站

2、点击官网标志

3、点击Headers往下便能找到User-Agent

'''

headers = {

'USER-AGENT': 'Mozilla/5.0 (Windows NT 10.0; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/67.0.3396.99 Safari/537.36'}

response = requests.get(url=url, headers=headers)

html = response.text

print(type(html))

return html

# 解析网页源代码

def parse\_page1(html):

html\_elem = etree.HTML(html)

'''

etree.HTML()：

调用HTML类对HTML文本进行初始化，成功构造XPath解析对象，同时可以自动修正HTML文本（标签缺少闭合自动添加上）

'''

pages = html\_elem.xpath('//\*[@id="content"]/div/div[1]/ol/li')

'''

快速获取xpath：

先F12,然后用Ctrl+Shift+C快速选中Web界面的元素，然后在元素上右键-->复制-->复制XPath

'''

for i in pages:

titles = i.xpath('./div/div[2]/div[1]/a/span[1]/text()')

infos = i.xpath('./div/div[2]/div[2]/p[1]/text()')

scores = i.xpath('./div/div[2]/div[2]/div/span[2]/text()')

'''

text()函数

获取元素文本内容

'''

print("----------------------------------------")

print("电影名：", titles[0], '\n', "详情：", infos[1].strip(), '\n', "评分：", scores[0] + '分', '\n')

time.sleep(1)

# 开始爬取网页

def crawl():

url = 'https://movie.douban.com/top250?start={0}&filter='

'''

用Request时对于大部分的页面来说只需要两个参数，一个是url也就是请求的网址，另外一个就是headers，

而headers中最重要的就是user-agent，其他的参数都没有这两个参数重要。

所以可以只留下这两个参数再去请求页面，如果请求不下来再去尝试添加其他参数。

'''

print('开始爬取')

# for page in range(0,250,25): # 0 25 50 75

for page in range(0, 25): # 0 25 50 75

# print('正在爬取第 ' + str(page+1) + ' 页至第 ' + str(page+25) + ' 页......')

html = get\_page(url.format(page))

data = parse\_page1(html)

time.sleep(1) # 延迟1秒

print('结束爬取')

crawl()

# -- coding: utf-8 --

# 传感器采集数据

# 传感器采集数据文件 sensor-data.txt 的一部分：

# 2018-02-28 01:03:16 19.3024 38.4629 45.08 2.68742

# 2018-02-28 01:06:16 19.1652 38.8039 46.08 2.68742

# 2018-02-28 01:06:46 19.175 38.8379 47.08 2.69964

# 其中，每行是一个读数，空格分隔多个含义，分别包括日期、时间、温度、湿度、光照和电压。其中，光照处于第 5 列。 请编写程序，统计并输出传感器采集数据中光照部分的最大值、最小值和平均值，所有值保留小数点后 2 位。‪‬‪‬‪‬‪‬‪‬‮‬‭‬‪‬‪‬‪‬‪‬‪‬‪‬‮‬‫‬‭‬‪‬‪‬‪‬‪‬‪‬‮‬‪‬‮‬‪‬‪‬‪‬‪‬‪‬‮‬‭‬‪‬‪‬‪‬‪‬‪‬‪‬‮‬‭‬‪‬

# 示例1：

# 输入：无

# 输出："最大值、最小值、平均值分别是：49.08,40.08,44.37"

data = []

class chuanganqi:

def \_\_init\_\_(self, date, time, shidu, wendu, guangzhao, dianya):

self.date = date

self.time = time

self.shidu = shidu

self.wendu = wendu

self.guangzhao = guangzhao

self.dianya = dianya

with open("sensor-data.txt", "r") as f:

line = f.readline()

while line != '':

lines = line.rstrip().split(' ')

values = chuanganqi(lines[0], lines[1], lines[2], lines[3], lines[4], lines[5])

data.append(values)

line = f.readline()

print(max(i.guangzhao for i in data))

print(min(i.guangzhao for i in data))

average = sum(float(i.guangzhao) for i in data) / len(data)

print("%.2f" % average)

<https://gitee.com/F4-8C-50-08-81-12/python.git>

<https://github.com/GitHub-Arthur/pythonProject.git>