1. **封装成exe**

Pip install pyinstaller安装模块

输入命令：pyinstaller -i "D:\编程相关\Python\Sha excel\Pineapple.ico" -F "D:\编程相关\Python\ShaExcel\NII-Split\NII-Split.py"

-F 后面内容为你想打包的python文件,注意路径一开头D：前面不能有空格 -i后面是ICON文件的路径。如果路径有空格，可以把整个路径用引号包起来

会在C:\Users\xxx下生产两个文件夹build和dist，还有一个文件。exe文件在dist文件夹下。这样就打包成功了。python程序中读取文件的目录为’./’相对路径的话， 把文件放到跟这个exe同一个文件夹就行。

**异常处理**

except Exception as e:

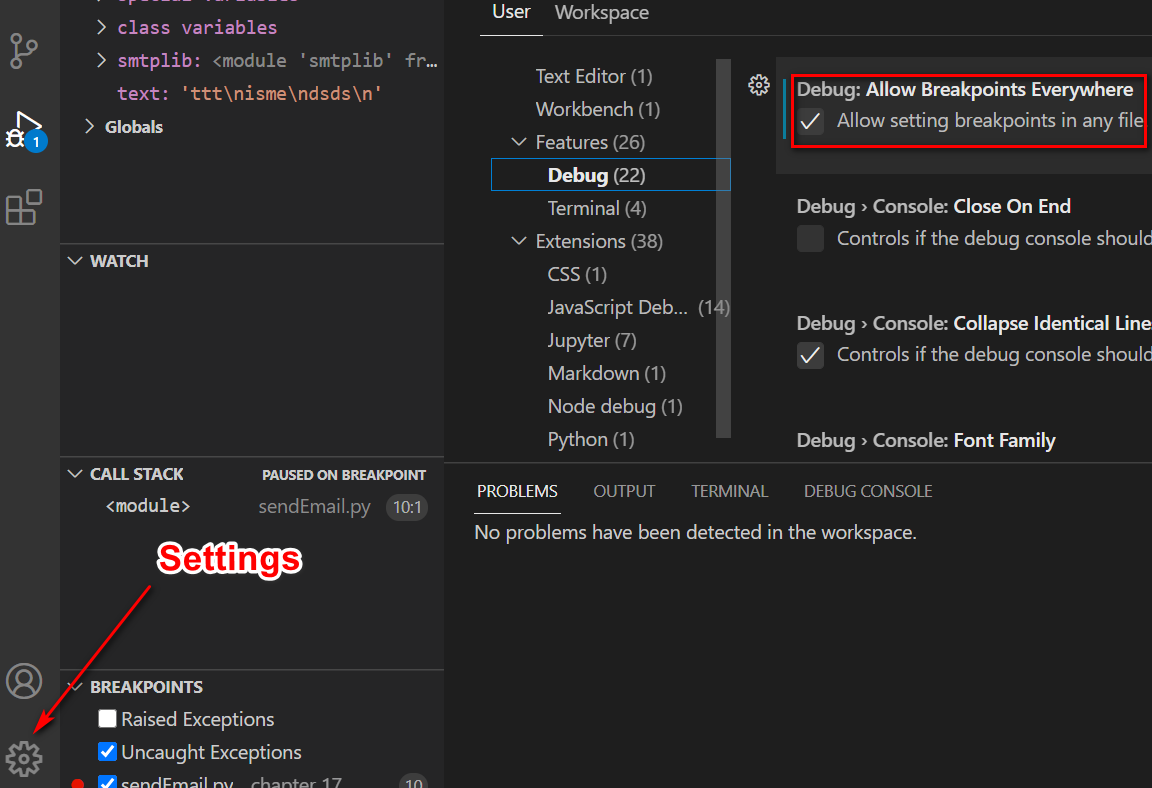
    print('Error:  '+e)

**安装python和vs code**

<https://docs.forchange.cn/docs/rgdcJrGdXrcqvxx9/>

pip使用： 打开cmd后运行pip -install xxx就能直接安装需要的包。

安装完后设置加断点的功能



1. **字符串处理**

String=”aa-bbb-cc”

String.split(‘-‘)的结果是一个list [‘aa’,’bbb’,’cc’]. split后括号里的字符就是按照什么来分割

1. **os模块相关**

* 检查文件是否存在：import os if os.path.isfile(‘abc.txt’)
* 打印文件内容：print(file.read())
* 读取一行内容：readline()括号里默认值是-1，即打印整行。加上数字表示打印一行的头x个字

f = open("demofile.txt", "r")  
print(f.readline())

print(f.readline()) 这样是把文件第一和二行打印了

* 打开文件：file = open(file\_path, mode, encoding='utf-8')，这个mode有：

r: 只读。默认模式

w：完全删除原内容然后写入。文件不存在就创建

a: 追加内容。文件不存在就创建

r+ : 读写模式

w+: 读写模式. 完全删除原内容然后写入。文件不存在就创建

a+ : 读写模式. 追加内容。文件不存在就创建

Mode 字符要加引号，例如file = open(‘c:\temp’, ‘r’, encoding='utf-8')

1. **数字format 原文链接：https://blog.csdn.net/u014770372/article/details/76021988**

数字 格式 输出 描述

3.1415926 {:.2f} 3.14 保留小数点后两位

3.1415926 {:+.2f} 3.14 带符号保留小数点后两位

-1 {:+.2f} -1 带符号保留小数点后两位

2.71828 {:.0f} 3 不带小数

1000000 {:,} 1,000,000 以逗号分隔的数字格式

0.25 {:.2%} 25.00% 百分比格式

1000000000 {:.2e} 1.00E+09 指数记法

25 {0:b} 11001 转换成二进制

25 {0:d} 25 转换成十进制

25 {0:o} 31 转换成八进制

25 {0:x} 19 转换成十六进制

5 {:0>2} 05 数字补零(填充左边, 宽度为2)

5 {:x<4} 5xxx 数字补x (填充右边, 宽度为4)

10 {:x^4} x10x 数字补x (填充两边,优先左边, 宽度为4)

13 {:10} 13 右对齐 (默认, 宽度为10)

13 {:<10} 13 左对齐 (宽度为10)

13 {:^10} 13 中间对齐 (宽度为10)

1. **%的用法 https://www.cnblogs.com/fat39/p/7159881.html#tag1**

print('%d' % 20)输出20

1 >>> print('%f' % 1.11) # 默认保留6位小数

2 1.110000

3 >>> print('%.1f' % 1.11) # 取1位小数

4 1.1

5 >>> print('%e' % 1.11) # 默认6位小数，用科学计数法

6 1.110000e+00

7 >>> print('%.3e' % 1.11) # 取3位小数，用科学计数法

8 1.110e+00

9 >>> print('%g' % 1111.1111) # 默认6位有效数字

10 1111.11

11 >>> print('%.7g' % 1111.1111) # 取7位有效数字

12 1111.111

13 >>> print('%.2g' % 1111.1111) # 取2位有效数字，自动转换为科学计数法

14 1.1e+03

字符串处理

1 >>> print('%s' % 'hello world') # 字符串输出

2 hello world

3 >>> print('%20s' % 'hello world') # 右对齐，取20位，不够则补位

4 hello world

5 >>> print('%-20s' % 'hello world') # 左对齐，取20位，不够则补位

6 hello world

7 >>> print('%.2s' % 'hello world') # 取2位

8 he

9 >>> print('%10.2s' % 'hello world') # 右对齐，取2位

10 he

11 >>> print('%-10.2s' % 'hello world') # 左对齐，取2位

12 he

* **风变基础语法第四关 列表和字典**

列表很包容，各种类型的数据（整数/浮点数/字符串）

list1 = ['小明',18,1.70]

print(list1)

偏移量是从0开始的，而非我们习惯的从1开始；2.列表名后加带偏移量的中括号，就能取到相应位置的元素

students = ['小明','小红','小刚']

print(students[0]) # 打印出小明

列表嵌套列表

students = [['小明','小红','小刚','小美'],['小强','小兰','小伟','小芳']]

print(students[1][1])、结果是小兰

冒号用于切片，左边数字对应的元素要拿，右边的不动. 冒号切片则是截取了列表的某部分，所以还是列表。而上面的偏移量取到的是元素。

list2 = [5,6,7,8,9]

print(list2[:])

# 打印出[5,6,7,8,9]

print(list2[2:])

# 打印出[7,8,9]

print(list2[:2]) 、或者print(list2[0:2])

# 打印出[5,6]

print(list2[1:3])

#打印出[6,7]

print(list2[2:4])

#打印出[7,8]

用append()给列表增加元素，每次只能增加一个东西

list3 = [1,2]

list3.append(3)、结果是 [1,2,3]

list3.append([4,5])、结果是[1, 2, 3, [4, 5]]

del语句非常方便，既能删除一个元素，也能一次删除多个元素（原理和切片类似，左取右不取）

students = ['小明','小红','小刚','小美']

del students[1:3]、结果是['小明','小美']

del students[3] 、结果是['小明', '小红','小刚']

这便是从字典中提取对应的值的用法, 字典名[字典的键]

scores = {'小明': 95, '小红': 90, '小刚': 90}

print(scores['小明'])

删除字典里键值对的代码是del语句del 字典名[键]，而新增键值对要用到赋值语句字典名[键] = 值

scores = {'小明':95,'小红':90,'小刚':90}

del scores['小刚']

scores['小刚']=92

scores['小美']=85

print(scores)

字典嵌套字典

scores = {

'第一组':{'小明':95,'小红':90,'小刚':100,'小美':85},

'第二组':{'小强':99,'小兰':89,'小伟':93,'小芳':88}

}

print(scores['第一组']['小刚'])、 结果是100

【in】和【not in】就可以用来判断字典中是否存在某个【键】：

dict = {'法国':'巴黎','日本':'东京','中国':'北京'}

a = '法国'

print(bool(a in dict))

【in】和【not in】可以用来判断列表中是否存在某个

list = [1,2,3,4,5]

a = 1

# 做一次布尔运算，判断“a是否在列表list之中”

print(bool(a in list))

print(bool(a not in list))

列表和字典相互嵌套

students = {

'第一组':['小明','小红','小刚','小美'],

'第二组':['小强','小兰','小伟','小芳']

}

print(students['第一组'][3])

#取出'第一组'对应列表偏移量为3的元素，即'小美'

# 最外层是中括号，所以是列表嵌套字典，先判断字典是列表的第几个元素，再找出要取出的值相对应的键

scores = [

{'小明':95,'小红':90,'小刚':100,'小美':85},

{'小强':99,'小兰':89,'小伟':93,'小芳':88}

]

print(scores[1]['小强'])

#先定位到列表偏移量为1的元素，即第二个字典，再取出字典里键为'小强'对应的值，即99。

* **风变基础语法 第五关 循环**

遍历列表

for i in [1,2,3,4,5]:

print(i)

遍历字典

dict = {'日本':'东京','英国':'伦敦','法国':'巴黎'}

for i in dict:

print(i)

数字是不可以遍历的，以下不行

a = 567

for i in a:

print(i)

下面的字符串可以，

a = ‘567’

for i in a:

print(i)

打印结果是

5

6

7

使用range(x)函数，就可以生成一个从0到x-1的整数序列

for i in range(3):

print(i)、结果竖排打印0，1，2

# 用for循环完成1-100分别乘以5的计算

for i in range(1,101):

print(i\*5)

使用range(a,b) 函数，你可以生成了一个【取头不取尾】的整数序列，例如range(13,17) 得到13,14,15,16

下面的代码可以打印 我很棒 三遍

for i in range(3):

print('我很棒')

for i in range(11):

print('桓走的第'+str(i)+'天，想他')

range(0,10,3)的意思是：从0数到9（取头不取尾），步长为3

也可以打印字典里的值，下面可以打印出醋油盐米

d = {'小明':'醋','小红':'油','小白':'盐','小张':'米'}

for i in d:

print(d[i])

while 循环

a = 0 #先定义变量a，并赋值

while a < 5: #设定一个放行条件：a要小于5，才能办事

a = a + 1 # 满足条件时，就办事：将a+1

print(a) # 继续办事：将a+1的结果打印出来

password = '' # 变量password用来保存输入的密码

while password != '816':

password = input('请尝试输入密码：')

print('欢迎回家！')

循环是开始时就按照索引的0，1，2这样下去，所以如果循环里面改了原列表，会有问题

例如下面，tango是t开头，但是不会被删除。因为循环到a和p开头的词后，会删掉原列表的一个元素

a=["app","bus","car","ora","pdd","tango"]

for co in a:

    if co[0] in ["a","p","t"]:

        a.remove(co)

print(a)

* **风变基础语法第六关 布尔值**

Python中已经设定好什么数据为真，什么为假。假的是有限的，那么除了假的，就都是真的。请看下图：

A screenshot of a computer screen

Description automatically generated with medium confidence

if 1:

print('熊猫')、会打印出熊猫，因为1是True

我们可以使用bool()函数来查看一个数据会被判断为真还是假，例如bool(10>20)得到False

* **风变基础语法 第9关 函数**

Python自带的内置函数，例如len(), print()等。也可以自己定义函数**。**以刚刚提到的y = 3x + 5 为例子，来定义一个一次函数

# 函数名：1. 名字最好能体现函数的功能，一般用小写字母和单下划线、数字等组合

# 2. 不可与内置函数重名（内置函数不需要定义即可直接使用）

def math(x):

# 参数：根据函数功能，括号里可以有多个参数，也可以不带参数，命名规则与函数名相同

# 规范：括号是英文括号，后面的冒号不能丢

y = 3\*x + 5

# 函数体：函数的执行过程，体现函数功能的语句，要缩进，一般是四个空格

return y

# return语句：后面可以接多种数据类型，如果函数不需要返回值的话，可以省略

def math(x):

y = 3\*x + 5

return y

a = math(10)

print(a)、结果是35

也可以定义多个参数，调用时传入的值需要按照顺序传入

def menu(appetizer, course, dessert = '绿豆沙'): #这里第三个参数给了默认值，

print('一份开胃菜：' + appetizer)

print('一份主食：' + course)

print('一份甜品：' + dessert)

menu('话梅花生','牛肉拉面')

#因为已经默认将'绿豆沙'传递给dessert，调用时无须再传递。

结果是传入了2个参数的值，打印下面结果

一份开胃菜：话梅花生

一份主食：牛肉拉面

一份甜品：绿豆沙

参数定义时也可以前面加上星号，表示不定长度的参数。下面会打印出('烤鸡翅', '烤茄子', '烤玉米')，一个tuple

def menu(\*barbeque):

return barbeque

order = menu('烤鸡翅','烤茄子','烤玉米')

#括号里的这几个值都会传递给参数barbeque

print(order)

不定长参数也可以循环遍历

def menu(\*barbeque):

for i in barbeque:

print('一份烤串：' + i)

menu('烤香肠', '烤肉丸')

menu('烤鸡翅', '烤茄子', '烤玉米')

函数除了允许传入多个参数，也允许返回多个值，只需将返回的值写在return语句后面，用英文逗号隔开即可

import random

#引入random模块

appetizer = ['话梅花生','拍黄瓜','凉拌三丝']

def coupon(money):

if money < 5:

a = random.choice(appetizer)

return a

elif 5 <= money < 10:

b = random.choice (appetizer)

return b, '溏心蛋' #这里是把第二个返回值给了一个常值。

print(coupon(6))

全局变量和局部变量。在函数以外定义的都是全局变量，函数里面定义的是局部变量。例如下面rent是全局。variable\_cost如果不在前面加global就是局部，不能被其他函数使用。

rent = 3000

def cost():

global variable\_cost

utilities = int(input('请输入本月的水电费用'))

food\_cost = int(input('请输入本月的食材费用'))

variable\_cost = utilities + food\_cost

print('本月的变动成本是' + str(variable\_cost))

def sum\_cost():

sum = rent + variable\_cost

print('本月的总成本是' + str(sum))

cost()

sum\_cost()

* **风变基础语法 第12关 类和对象**

class Chinese: # 创建一个类，一般类名首字母大写

eye = 'black' #定义一个属性

def eat(self): #定义一个method

print('吃饭，选择用筷子。')

wufeng = Chinese() # 类的实例化实例化后，类里面的定义好的东西都可以用了

print(wufeng.eye) # 实例调用类属性

wufeng.eat() # 调用类中的方法（传参不用管self）

也可以给method定义多个参数。第一个参数self是不接收参数的

class Chinese:

name = '吴枫' # 类属性name

def say(self, someone): # 带有两个参数的方法

print(someone + '是中国人')

person = Chinese()

print(person.name)

person.say('吴枫')

# self调用时要忽略，'吴枫'传给第二个参数someone

每个method里的self可以调用本类里定义好的东西。例如上面的代码可以写成

class Chinese:

name = '吴枫' # 类属性name

def say(self):

print(self.name + '是中国人')

person = Chinese() # 创建Chinese的实例person

person.say() # 调用实例方法，结果是：吴枫是中国人

同理，如果想在类的方法内部调用其他方法时，我们也需要用到self来代表实例。

class Chinese:

def greeting(self):

print('很高兴遇见你')

def say(self):

self.greeting()

print('我来自中国')

person = Chinese() # 创建实例person

person.say() # 调用say()方法

结果是

很高兴遇见你

我来自中国

定义初始化方法的格式是def \_\_init\_\_(self)。初始化方法的作用在于：当每个实例对象创建时，该方法内的代码无须调用就会自动运行。

class Chinese:

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

print('很高兴遇见你，我是初始化方法')

person = Chinese() #这里实例化后马上就会打印出那句话

初始化方法同样可以接收其他参数，让传入的这些数据能作为属性在类的方法之间流转

class Chinese:

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

self.name = name

self.birth = birth

self.region = region

def born(self):

print(self.name + '出生在' + self.birth)

def live(self):

print(self.name + '居住在' + self.region)

person = Chinese('吴枫','广东','深圳') # 实例化，并传入值给初始化方法的参数name, birth, region

person.born()

person.live()

子类继承的属性和方法，也会传递给子类创建的实例.子类的语法是class A(B): A是子类名，B是父类名

class Chinese:

#父类这种不带括号，其实它相当于class Chinese(object):。而object，是所有类的父类，我们将其称为根类

eye = 'black'

def eat(self):

print('吃饭，选择用筷子。')

class Cantonese(Chinese):

pass # pass表示'跳过'，不执行其他操作

yewen = Cantonese()

print(yewen.eye)

yewen.eat()

多层子类继承:

class Earthman:

eye\_number = 2

# 中国人继承了地球人

class Chinese(Earthman):

eye\_color = 'black'

# 广东人继承了中国人，同时也继承了地球人。

class Cantonese(Chinese):

pass

yewen = Cantonese()

print(yewen.eye\_number)

print(yewen.eye\_color)

结果是

2

Black

一个类，可以同时继承多个父类，语法为class A(B,C,D):

class Su:

born\_city = 'Jiangsu'

wearing = 'thick' # 穿得较厚

def diet(self):

print('我们爱吃甜。')

class Yue:

settle\_city = 'Guangdong'

wearing = 'thin' # 穿得较薄

def diet(self):

print('我们吃得清淡。')

class Yuesu(Yue,Su):

pass

xiaoming = Yuesu()

print(xiaoming.wearing)

print(xiaoming.born\_city)

xiaoming.diet()

结果是

thin

Jiangsu

我们吃得清淡。

我们可以在子类下新建属性或方法，让子类可以用上父类所没有的属性或方法。这属于定制中的一种：新增代码

class Chinese:

eye = 'black'

def eat(self):

print('吃饭，选择用筷子。')

class Cantonese(Chinese): # 类的继承

native\_place = 'guangdong' # 类的定制

def dialect(self): # 类的定制

print('我们会讲广东话。')

yewen = Cantonese()

print(yewen.eye) # 父类的属性能用

print(yewen.native\_place) # 子类的定制属性也能用

yewen.eat() # 父类的方法能用

yewen.dialect() # 子类的定制方法也能用

重写代码，是在子类中，对父类代码的修改。子类继承父类方法的操作是在def语句后接父类.方法（参数）

class Chinese:

def land\_area(self,area):

print('我们居住的地方，陆地面积是%d万平方公里左右。'% area)

class Cantonese(Chinese):

def land\_area(self, area, rate = 0.0188):

Chinese.land\_area(self, area \* rate) # 间接对方法进行重写，直接继承父类方法，再调整参数。

gonger = Chinese()

yewen = Cantonese()

gonger.land\_area(960)

yewen.land\_area(960)

结果是：

我们居住的地方，陆地面积是960万平方公里左右。

我们居住的地方，陆地面积是18万平方公里左右。

下面的例子通过参数默认值的改变，完成子类的定制，让程序的运行结果为“雷猴！欢迎来到广东。”

class Chinese:

def \_\_init\_\_(self, greeting = '你好', place = '中国'):

self.greeting = greeting

self.place = place

def greet(self):

print('%s！欢迎来到%s。' % (self.greeting, self.place))

class Cantonese(Chinese):

def \_\_init\_\_(self, greeting = '雷猴', place = '广东'):

Chinese.\_\_init\_\_(self, greeting, place)

yewen = Cantonese()

yewen.greet()