# python学习

**Python**

**1.Python的简介：**

1. python由荷兰Guido（龟叔）吉多.范罗苏母于1989年圣诞节设计出来。
2. pytho的特点：**1>简单 2>开源 3>面向对象 4>可扩展 5>可移植**

**是一门动态语言、纯面向对象的语言，与C、C++等静态语言不同，再创建变量时不需要指定变量类型**

1. python的应用：（胶水语言）
2. web开发（豆瓣、知乎、web框架Django）
3. 科学计算（1998^1998=）
4. 网络爬虫
5. 大数据
6. 云计算
7. **人工智能**
8. 自动化维护

4.python3.x与python2.x有什么区别（不能简单的认为3.x就是2.x的升级）

1>性能：总体来说3.x的性能低于2.x

2>编码：python3.x默认支持UTF-8(万国码)

3>语法：去除了<>全部换成 !=

5.python之禅：(python的哲学)

import this

执行代码的快捷方式：ctrl+shift+F10（可以改快捷键）

**IDE：集成开发环境**

**IDLE：python自带的集成开发环境**

**2.python的原理以及与C语言的区别：**

C语言、java、python：他们三者的历史地位如何？**C语言是鼻祖**

C语言的执行原理：编辑(.c)------->编译(.obj)------>链接(.exe)------>执行可执行文件

**python的执行原理：源代码(xxx.py)------>字节码(xxx.pyc)---------->PVM**

把python的源代码编译成字节码，然后把编译好的字节码发送到python虚拟机中执行。

python与C语言的区别：

1. python是面向对象的而**C语言是面向过程的。**
2. python的生产效率高而**C语言的执行效率高。**

**3.Python语法：注释+缩进+变量+数据类型+标识符和关键字+数据类型的转换+运算符**

1.注释：注释率：

是评价代码好坏的一个标准（20%的注释率）

C语言：// /\* ...\*/

#:单行注释：快捷方式ctrl+/

**"""XXXXX""":多行注释 ’’’ ’’’**

eg:  
print(**"hello world1"**)  
**"""print("hello world2")  
print("hello world3")#这段代码很重要"""**print(**"hello world4"**) # 打印出来的是字符串

**2**.缩进**：**

python最具特色的地方就是使用缩进来表示代码块。

C语言使用大括号来表示代码块，使用大括号表示代码块的化允许代码写的很烂，这一点与python的哲学不符合

缩进不光让python代码显得精简有层次，缩进还具有实际的意义，如果错误的使用缩进不光不美观还会让代码的执行结果与我们期望相差甚远。缩进是python的灵魂。

eg：

x=-8

if x>=0:

print("+++")

else:

print("---")

缩进在python中表示一种层级关系。

**3.变量：**

常量：在程序的执行过程中其值不变的量。例如：s=PI\*r\*r

变量：在程序的执行过程中其值改变的量。variable

在python我们使用**‘=’给变量赋值，等号左边为变量名，右边为变量的值。变量名=变量值**

eg：

num\_1=1008

num\_2=2009

num\_1=999

num\_3=num\_1+num\_2

print(num\_3)

在python中当我们把一个实际值赋给一个名字时，它存存储在内存中，这块内存就是变量。

python中可以不声明变量但是每个变量在使用之前必须赋值，python的变量是在被赋值的那一刻创建的。

Python的赋值与其他语言的不同：

**a.可以给多个变量赋同一个值。(x=y=z=9)**

**b.可以给多个变量同时赋值。（a,b,c）=（9,5,2）**------->通过元组来实现

**总结：**

不管是给python的变量还是python的函数起名最好做到四个字：见名识意，并且不建议使用驼峰式的命名规则，建议使用下划线。UserName----->user\_name

eg：用户输入两个数然后交换输出

a=input("请输入第一个数:")————————a=input("请输入第一个数:")

b=input("请输入第二个数:")————————b=input("请输入第二个数:")

print(a,b)——————————print（a,b）

temp=a————————————a,b=b,a

a=b——————————————

b=temp————————————

print(a,b)————————————print（a,b）

**学习语言的方法：读代码------->抄代码-------->写代码。**

**程序中最重要的就是算法和数据结构。**

**4.python的数据类型：（数字类型、字符串、列表、元组、字典、集合）**

**不可变数据类型：number、string、tuple**

**可变数据类型：list、dictionary、set**

**数字类型（number）：**

python3.x支持四种数字类型：

整型/int、浮点型/float、布尔型/bool、复数型/complex

**int：整型：**

python3.x没有长整型只有整型。

**float：浮点型：**

用于表示实数，可以是十进制也可以是科学计数。1.9\*10^7

**bool:布尔型对于数字只有0和1,0表示假，1表示真。**

对于逻辑来说就是：**真(True)、假(False)**。 （在语言中：0表示假，除0之外都是真）

**complex：用于表示数学中的复数。**

数学中的复数：我们把形如z=a+bi称为复数，其中a为实部，b为虚部。

Python中负数有以下特点：

1. 虚数不能独立存在；
2. 虚数部分需要有后缀j/J;
3. 实数和虚数部分都要是浮点数；

python中：我们把形如real+imagej或者real+imageJ称为复数。

共轭负数：实属部分相同，虚数部分相反；

a = 50 + 2j  
print(a.real)  
print(a.imag)  
b = a.conjugate()  
print(b)

50.0

2.0

(50-2j)

如何判断数据类型？我们可以使用python的**内置函数:type()来判断**

例如：

a, b, c, d = 36, 9.8, True, 3 + 9j  
print(a, b, c, d)  
print(type(a), type(b), type(c), type(d))

36 9.8 True (3+9j)

<class 'int'> <class 'float'> <class 'bool'> <class 'complex'>

**为什么上面的True首字母要大写？因为True是关键字。**

if 0:

print("+++")

else:

print("---")

**字符串（string）：简写‘str’**

字符串是一种表示**文本的数据类型**

**a.字符串在python中字符串有三种表现形式：**

使用单引号表示字符串

使用双引号也表示字符串

使用三引号表示字符串 ''' ''' """ """（这三个引号有什么区别？）

1. **从显示的结果来说没有区别；**
2. **引号中不能包含自己；**
3. **单引号和双引号只能有一行而三引号可以包含多行。**

**b.转义字符：**

\ （转义） \b(退格) \n(换行) \r(回车) \f(换页) \a(响铃) \t(横向制表)

举例：

s = 'let\'s go home'  
print(s)

let's go home

s = 'let\b's go home'  
print(s)

le’s go home

string\_1 = "dasdsadsa89\ndasdasdas"  
print(string\_1)

dasdsadsa89

dasdasdas

string\_1 = "dasdsadsa89\rdasdasdas"  
print(string\_1)

dasdasdas # 只取了第二行，第一行不要了

**c.字符串的输入\输出（input\print）：**

input(“”)

user\_name = input("请输入用户名:")  
user\_password = input("请输入密码：")

print("鲁迅是一位作家" + ",乔丹是一个运动员")

鲁迅是一位作家,乔丹是一个运动员

**e.使用百分号格式化字符串输出**

格式：%[name][flag][width][.][precision][type]

name:名称，可以为空。

flag：标记限定符号：+-#0

+：表示右对齐

-：表示左对齐

#：表示在不同的进制前加符号

0：表示填充0

width：宽度

precision：小数点后的位数

type：类型

%d：整型

%s：字符串

%o：八进制 0开头

%x：十六进制 0x十六进制

%X：十六进制（字母大写）

%f：浮点型

%e：浮点型(科学计数)

%g：浮点型（根据实际大小选择。6位）

eg：如果想要输出：已经ID=23，该学生的学号为：00023怎么办？

ID = 23

print("该学生的学号为:%.5d" % (ID))

该学生的学号为:00023

ID=23

sting\_1="该学生的学号为:"

print("%s %.5%d"%(sting\_1,ID)) 00023==>%.5%d(双引号中表示格式，括号中表示要输出的变量，变量要与格式呈现一一对应关系。)

疑问：默认是左对齐还是右对齐？**（右对齐）**

print("%5s" % "abc")  
print("%+5s" % "abc")

如何输出%？ **“%%”表示百分号**

s1 = "我的数学成绩为:"  
s2 = "在班级属于前"  
s3 = "的水平"  
print("%s%.1f,%s%d%%%s" % (s1, 98.5, s2, 5, s3))

我的数学成绩为:98.5,在班级属于前5%的水平  
print("%s,%.1f,%s,%d%%,%s" % ("jjj", 98.5, "hhh", 5, "ooo"))

jjj,98.5,hhh,5%,ooo  
print("%.2f%%" % 5)

5.00%

**f.使用format()格式化字符串输出：**

format()函数功能非常强大，该函数可以把字符串当成一个模板，通过传入参数进行格式化。

format()有三种：

第一种：不带编号的

format()=>print(“{}{}”.format(“a”,”b”))=>ab

第二种：带数字编号

format()=>print(“{2}{1}{0}{1}”.format(“a”,”b”,”c”))=>cbab

第三种：带关键字

format()=>print(“{x}{x}{y}{y}”.format(y=“a”,x=”b”))=>aabb

print(3.1415926)  
print("{0:.3f}".format(3.1415926))

3.142

aa = 234567.567890  
print('%f' % aa)  
print('%.3f' % aa)  
print('%.1f' % aa)

234567.567890

234567.568

234567.6

**g:访问字符串中的值：**

**1>使用下标（索引）来截取字符串中的值：**

在python中不支持单个字符，也就是说没有C语言中的char类型，只有字符串。

如果想要访问或者使用字符串中的值可以使用下标(索引)来实现。

例如:string\_1="zhourunfa" print（string\_1（[2]））==>o

012345678

Guido说python使用0-base原因是python切片的语法。

**2>使用切片来截取**字**符串中的值：**

使用索引只能访问字符串中的某一个元素，切片指的是可以对字符串截取一部分的操作。

切片不仅可以对字符串操作，也可以对列表等其他数据类型操作。

**切片的语法：[起始：结束：步长] [::-1]这是倒序**

注意事项：切片也属于前闭后开。（range()）

string\_1 = "zhourunfa"  
print(string\_1[2:7:2])

orn  
print(string\_1[::-1])

afnuruohz

**h：字符串的运算符：**

+ ：连接字符串

\* ：重复输出字符串（乘号）

in ：成员运算符，如果字符串中包含子字符串则返回True，否则返回False。

not in ：成员运算符，如果字符串中不包含子字符串则返回True，否则返回False。

str\_1 = "I love you rose!"

if "maliya" not in str\_1:————if "maliya" in str\_1:

print("111")—————— print("111")

else:

print("222")

str\_1 = "I love"

str\_2 = " The world"

print(str\_1 + str\_2)

print(str\_1 + str\_2 \* 3)

print((str\_1 + str\_2) \* 3)

I love The world

I love The world The world The world

I love The worldI love The worldI love The world

**i:字符串的内建函数：**

字符串作为一种最常见的数据类型，python为他提供了很多内建函数。这些函数有些也能直接用于其他数据类型。

**find():该函数可以检查字符串中包含的指定的子字符串，如果包含就返回子字符串的索引值（所在位置值），否则返回-1。**

格式：文件名.find(查找内容，start,end)

第一个str:指定检索的字符串(在哪里找)

第二个str:表示要找的子字符串(找什么)

start：开始的索引值

end：结束的索引值

str\_1 = "one world one dream"

index = str\_1.find("ld", 0, 10)  
print(index)

7

print(str\_1.find('y'))

-1

**count():该函数用于统计字符串中某个子字符串出现的次数。**

格式：str\_a.count(‘chars’, start, end)

第一个str:指定检索的字符串(在哪统计)

第二个str:表示要找的子字符串(统计什么)

start：开始的索引值

end：结束的索引值

str\_1="one world one dream"

print(str\_1.count(“o”,0,20))

3

index=str\_1.count("666",0,20)

print(index)

0

**replace():该函数可以将就字符串替换为新的字符串。**

格式：str\_a.replace(old, new, num)

str\_a:指替换的字符串

old: 被替换的子字符串

new:新的子字符串

**num:替换的次数,默认替换所有**

str\_001="I love you rose,rose I　promise!"

print(str\_001)======>I love you rose,rose I　promise!

str\_new=str\_001.replace("rose","maliya",1)

print(str\_new)====>I love you maliya,maliya I　promise!

**split():该函数可以通过指定的分隔符对字符串进行分割。**

格式：str\_.split(str,num)

第一个str:要分割的字符串

第二个str:分隔符（默认为空格）

num：分割次数

print('%d' % (time.time())) # 时间戳取整数  
time\_aa = time.time()  
print((str(time\_aa).split('.'))[0])

1654247432

1654247432

**current\_time = '2022/04/04 23:18:55'**

**# 获取年月日 2022/04/04**

**print(current\_time.split(' ')[0]) # split(chars) 表示以字符串chars为分隔符进行分割目标字符串**

**# 获取时分秒 23:18:55**

**print(current\_time.split(' ')[-1])**

str\_001="I love you rose,rose I　promise!"

index=str\_001.split("o")

print(index)=======>['I l', 've y', 'u r', 'se,r', 'se I\u3000pr', 'mise!']

\u3000：是干什么的？

三种空格：不间断空格：\u00A0 英文半角空格：\u0020 中文全角空格： \u3000:

**疑问：使用split函数分割与字符串的切片有什么区别？**

**1>split分割是使用指定的分隔符来分割，而切片使用的是索引。**

**2>split分割后返回的是是列表，而切片后返回的是子字符串。**

**strip(chars):若方法里面的chars不指定，则默认去掉字符串的首、尾空格或者换行符，但是如果指定了chars，那么会删除首尾的chars（字符串）**

str\_a = 'lady gaga '  
print(str\_a.strip('l')) # ady gaga   
print(str\_a.replace('a', 'an', 5)) # landy gangan

str\_a = 'lady gaga '  
print(str\_a.strip('l')) # ady gaga # 删掉首尾的’l’

str\_001=" I love you rose,rose I　promise! "

str\_002=str\_001.strip()

print(str\_001)====> I love you rose,rose I　promise! （首尾有空格符）

print(str\_002)====>I love you rose,rose I　promise!（首尾无空格符）

print(str\_001.strip())====>I love you rose,rose I　promise!

str\_001="I love you rose,rose I　promise! "

str\_002=str\_001.strip("I")

print(str\_002)====> love you rose,rose I　promise!

**capitalize():将首字母大写其余小写。**

**title():首字母全部大写==>index=str.title()**

**upper():全部大写==>index=str.upper()**

**l just():左对齐==>index=str.ljust(50)**

**r just():右对齐==>index=str.rjust(50)**

**center():居中==>index=str.center(50)**

**index():该函数与find函数一样，但是如果在某一个范围内没有找到该字符串的时候，不再返回-1而是直接报错。**

str\_1="one world one dream"

index=str\_1.index("666")

print(index)=======> 报错

**isalpha():字符串是否全是由字母组成的，是返回true，否则返回false**

str\_1="oneworldonedream"

index=str\_1.isalpha()

print(index)======>True

**isalnum():字符串是由字母或数字组成则返回True否则返回False。**

str\_1="oneworldonedream888"

index=str\_1.isalnum()

print(index)====>True

str\_1="oneworldonedr eam888"

index=str\_1.isalnum()

print(index)=====>False

**isdigit(): 字符串是否全是由数字组成，是则返回True，否则返回False**

str\_1="543543535453543"

index=str\_1.isdigit()

print(index)=======>True

**lower():将字符串中的所有大写字母转换为小写字母的字符串**

str = "AbcdEfg"

print( str.lower() ) =====> abcdefg

**列表（list） [ ]: list**

属于Python中的序列类型，它是任意对象的有序集合，通过“位置”或者“索引”访问其中的元素，它具有可变对象、**可变**长度、异构和任意嵌套的特点。

列表里第一个元素的“位置”或者“索引”是从**“0”**开始，第二个元素的则是**“1”**，以此类推。

在创建列表时，列表元素放置在方括号[ ] 中，以逗号来分隔各元素，格式如下：

**listname = [“元素1”, “元素2”, “元素3”, “……”, “元素n”]**

举例如下：

sample\_list1 = [0, 1, 2, 3, 4]  
sample\_list2 = ["P", "y", "t", "h", "o", "n"]  
sample\_list3 = ['Python', 'sample', 'list', 'for', 'your', 'reference']

1. **访问列表中的元素===>print(sample\_list1[2])**

print(sample\_list1[2])————>2

**2.使用for循环遍历列表===>**

for i in sample\_list1:  
 print(i)

———> 0 1 2 3 4

**3.向列表中增加元素===>append、insert、extend**

**方法一：append向列表中增加元素(末端)**

A.append(“xxxx”)====list\_1.append("甜蜜蜜")

**方法二：通过insert增加(在指定的索引位置增加)**

1. insert（n,”xxxx”）====list\_1.insert(2,"甜蜜蜜")

**方法三：通过extend把一个列表添加到另一个列表中（无缝添加）**

A.extend(B)——>print(A)=======list\_1.extend(list\_2)

**4.删除列表元素===>del、remove、pop、clear**

**del：通过索引删除列表的元素。是一个方法不是函数不能采用调用函数的方法进行。**

**——— del 文件名 [ 数字 ]**

del sample\_list3[0]  
print(sample\_list3)

==> ['sample', 'list', 'for', 'your', 'reference']

**remove：删除列表中给定的对象（值）。**

**———— A.remove(“xxxx”)**

sample\_list3.remove('for')

print(sample\_list3)

==> ['sample', 'list', 'your', 'reference']

**pop：用于移除列表中的元素，默认是最后一个元素，并且返回该元素的值。**

**———— A.pop( 数字 )**

格式：list.pop(index=-1)

sample\_list3.pop()  
print(sample\_list3)

==> ['sample', 'list', 'your']

**del与pop区别：**

**1.del是方法pop是函数。**

**2.pop是list类的成员函数有返回值，del没有返回值（被删除的值不返回）。**

排除上面两点，del与pop没有区别。

**clear：清空列表（生成空列表）**

**——A.clear( )**

list\_1.clear()

====> []

**5.列表是python中非常重要的内置数据结构，列表可以说是一个由有序的可以索引的元素组成的可变的线性数据结构 list.copy**

|  |  |
| --- | --- |
| list\_1=["教父","美丽人生","当幸福来敲门"]  list\_2=list\_1  list\_2[1]="999"  print(list\_1)  print(list\_2)  print(id(list\_1),id(list\_2))  ['教父', '999', '当幸福来敲门']  ['教父', '999', '当幸福来敲门']  2068739639752 2068739639752 | list\_1=["教父","美丽人生","当幸福来敲门"]  list\_2=list\_1.copy()  list\_2[1]=888  print(list\_1)  print(list\_2)  ['教父', '美丽人生', '当幸福来敲门']  ['教父', 888, '当幸福来敲门']  print(id(list\_1),id(list\_2))  2068739639560 2068739640520  List\_3 = list\_1[:]  Print(list\_3)  ['教父', '美丽人生', '当幸福来敲门']  print(copy\_list([0, 31, 59, 90, 120, 151, 181, 212, 243, 273, 304, 334])) list\_aa = [0, 31, 59, 90, 120, 151, 181, 212, 243, 273, 304, 334] list\_bb = list\_aa.copy() print(list\_bb) |

**6.修改列表元素：**

列表[index] = 新元素 list\_2[1] = 888

对列表的元素进行排序：

**7.将列表转换为字典：**

i = ['a', 'b']  
e = [1, 2]  
cc = dict([i, e])  
print(cc)

{'a': 'b', 1: 2}

**8.列表排序有两种方式：**

**方式1：sort排序：默认从小到大（永久排序）**

list\_1 = [9, 5, 2, 7]  
list\_1.sort()  
print(list\_1)

[2, 5, 7, 9]

**方式2：reverse排序：将列表逆置**

list\_1 = [9, 5, 2, 7]  
list\_1.reverse()  
print(list\_1)

[7, 2, 5, 9]

**方式3：reverse排序：将列表从大到小排列（先从小到大排序，然后再倒序）**

list\_1 = [9, 5, 2, 7]  
list\_1.sort(reverse=True)  
print(list\_1)

[9, 7, 5, 2]

**方式4：sorted排序：临时将列表进行从小到大排序**

list\_1 = [9, 5, 2, 7]  
a = sorted(list\_1)  
print(a) # [2, 5, 7, 9]

print(list\_1) # [9, 5, 2, 7]

**sort与sorted排序的区别：**

sort只适用于列表，是在原位重新排序列表

sorted运用范围广，可运用与任意的可迭代的对象（字典、元组、字符串、列表）

sort属于永久排序list\_a.sort()，sorted属于临时排序list\_b = sorted(list\_a)

**元组（Tuples）（）**

与列表一样，属于Python中的序列类型，它是任意对象的有序集合，通过“位置”或者“索引”访问其中的元素，它具有可变长度、异构和任意嵌套的特点，与列表不同的是：元组中的元素是不可修改的。

元组与列表非常类似只是：元组用圆括号表示并且元组**不能修改**。

eg:：tuple\_1=("教父","美丽人生","当幸福来敲门")

**1.访问元组的元素：print(tuple\_1[1]）——＞print(A[ ]) # 美丽人生**

**2.使用for循环遍历元组：for i in tuple\_1:——for i in A:**

**3.元组的内置函数：——＞len、max、min.**

len():计算元组的个数：print(len(A))

max():找出最大元素：print(max(A))

min():找出最小元素:print(min(A))

|  |  |  |
| --- | --- | --- |
| list\_1 = ["a", "c", "X"] print(len(list\_1))  3 | tuple\_1 = ("a", "c", "X") print(max(tuple\_1))  c | tuple\_1 = ("a", "c", "X") print(min(tuple\_1))  X |

**4.元组也支持切片等操作：当然列表也能使用上面讲的这些函数。**

list\_1=["a","c","X"]

print(len(list\_1))====>3

**5.字符串、列表、元组相互转换：（固体、液体、气体）**

list\_1=["a","c","X"]

a=tuple(list\_1)

print(a)====>('a', 'c', 'X')

tuple\_1=("a","c","X")

a=list(tuple\_1)

print(a)====>['a', 'c', 'X']

str\_1="Iloveyourose"

a=list(str\_1)

print(a)==>['I', 'l', 'o', 'v', 'e', 'y', 'o', 'u', 'r', 'o', 's', 'e']

str\_1="Iloveyourose"

a=tuple(str\_1)

print(a)====>('I', 'l', 'o', 'v', 'e', 'y', 'o', 'u', 'r', 'o', 's', 'e')

list\_1=["a","c","X"]

a="".join(list\_1)

print(a)===>acX

tuple\_1=("a","c","X")

a="".join(tuple\_1)

print(a)===>acX

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

**其他方法：eval()**

s="['a','c','X']"

a=eval(s)

print(a)===>['a', 'c', 'X']

s="'a','c','X'"

a=eval(s)

print(a)===>('a', 'c', 'X')

s="{'a','c','X'}"

a=eval(s)

print(a)===>{'a', 'X', 'c'}

**字典(dictionary)：{ }**

字典属于映射类型，它是通过键实现元素存取，具有无序、可变长度、异构、嵌套和可变类型容器等特点。

**dictname = {键1: 值1, 键2: 值2, 键3: 值3, ……, 键n: 值n}**

name、ID就是键 周润发、007就是值

当数据量特别大，或者元素的顺序发生了改变此时适合使用字典。

字典没有索引，既然字典没有索引凭什么能够快速准确定位到某个元素？

**靠的就是：“键”——————需要注意的是键必须唯一，值可以不唯一。**

**字典中有：键（key）、值（value）**

**dict\_1 =** {'name': '周润发', 'ID': '007', 'sex': '男', 'address': 'HK', 'tall': '183'}

1. **访问字典：通过键来访问：**

print(dict\_1["sex"])===>"男"

1. **修改字典：**

dict\_1["sex"]=”女”

print(dict\_1)===>{'name': '周润发', 'ID': '007', 'sex': '女', 'address': 'HK', 'tall': '183'}

1. **添加字典的元素（修改时的键不存在则添加）**

dict\_1["tall"]="183"  
print(dict\_1)===>{'name': '周润发', 'ID': '007', 'sex': '男', 'address': 'HK', 'tall': '183'}

1. **删除字典的元素（del、clear）**

**del dict\_1["name"] 根据键来删除值（键值相对应）  
print(dict\_1)=====>{ 'ID': '007', 'sex': '男', 'address': 'HK', 'tall': '183'}**

**dict\_1.clear() 清空字典**

print(dict\_1)===>{}

1. **计算字典中键的个数：**

print(len(dict\_1))===>4

1. **遍历字典的元素：(键、值、元素)**

dict\_1={"name":"周润发","ID":"007","sex":"男","address":"HK"}  
for i in dict\_1: # 默认循环遍历‘键’  
 print(i)===>name ID sex address  
  
dict\_1={"name":"周润发","ID":"007","sex":"男","address":"HK"}  
**for i in dict\_1.keys():** # 指定了键（keys）  
 print(i)===>name ID sex address  
dict\_1={"name":"周润发","ID":"007","sex":"男","address":"HK"}  
**for i in dict\_1.values():** # 指定了值（values）  
 print(i)===>周润发007男HK  
dict\_1={"name":"周润发","ID":"007","sex":"男","address":"HK"}  
**for i in dict\_1.items(): # 输出为元组  
 print(i)===>('name', '周润发')('ID', '007')('sex', '男')('address', 'HK')**  
dict\_1={"name":"周润发","ID":"007","sex":"男","address":"HK"}  
**for key,value in dict\_1.items():  
 print(key,value)===>name 周润发 ID 007 sex 男 address HK**

**集合(set)：**

集合是一种集合类型，可以理解为就是数学课里学习的集合。它是一个可以表示任意元素的集合，它的索引可以通过另一个任意键值的集合进行，它可以无序排列、哈希。

集合不同于字符串、列表、元组、字典，集合的元素无序并且不能重复。

同数学中的集合一样，python的集合也能进行合并、差、交等运算。

**因为集合的元素无序且唯一，所以可以作为字典的键，其他的类型不行。**

集合分为两类：

可变集合（set）：可以添加、删除元素。

不可变集合（frozenset）：不可以添加、删除元素。

不可变集合不能添加删除但是可以合并、差。

**1.声明和访问集合：**

**不能像列表和元组一样使用索引来访问，也不能像字典一样使用键来访问，因为它没有。**

**对于集合的元素可以使用成员运算符in和not in 来判断。**

set\_1=("hello","world","老王") 这个不是集合。

**集合必须使用函数来声明，可变集合使用set函数，不可变集合使用frozenset函数**

set\_3=set("laowang")

set\_4=frozenset("hello")

1. **集合的使用方法：(xxx&ppp表示文件名；yyy：表示文件内容)**

***输出的结果是无序且会变化的***

set\_1 = set("hello")  
set\_2 = set("python")  
set\_3 = set("laowang")

1. **向集合中添加元素：add()——>xxx.add(“yyyy”)**

set\_1.add("k")

print(set\_1)===> {'e', 'o', 'k', 'l', 'h'}

1. **删除集合的元素：remove()——>xxx.remove(“yyy”)**

set\_1.remove("l")

print(set\_1)===>{'h', 'e', 'o'}

1. **清空集合的元素：clear()——>xxx.clear()**

set\_1.clear()

print(set\_1)===>set()

1. **将两个集合合并：union()——>print(ppp.union(xxx))**

x = set\_2.union(set\_1)

print(x)===>{'e', 'h', 'y', 't', 'p', 'n', 'l', 'o'}

a = [1, 2, 3]  
b = [1, 2, 3, 4, 5, 6]  
c = list(set(a).union(set(b)))  
print(c)

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

1. **将两个集合的差作为一个集合输出：difference->print(ppp.difference(xxx))**

**前者减去后者ppp - xxx**

x = set\_2.difference(set\_1)

print(x)===>{'p', 't', 'n', 'y'}

x = set\_1.difference(set\_2)

print(x)===>{'l', 'e'}

set\_1=set("abcd")

set\_2=set("cdef")

x=set\_2.difference(set\_1)

print(x)==>{'f', 'e'}

**3.集合的运算：子、并、交、差(xxx&ppp表示文件名；yyy：表示文件内容)**

1. **子：< >——>print(ppp<xxx)**

|  |  |
| --- | --- |
| set\_1=set("abcd")  set\_2=set("cdef")  x=set\_1<set\_2  print(x)===>False | set\_1 = set("abcd")  set\_2 = set("cd")  x = set\_2 < set\_1  print(x)===>True |

1. **并：union——>print(ppp.union(xxx))**

set\_1=set("abcd")

set\_2=set("cd")

print(set\_2.union(set\_1))===>{'b', 'd', 'a', 'c'}

1. **交：intersection——>print(ppp.intersection(xxx))**

set\_1=set("abcd")

set\_2=set("cdef")

print(set\_1.intersection(set\_2))====>{'d', 'c'}

1. **差：difference——>print(ppp.difference(xxx))**

set\_1=set("abcd")

set\_2=set("cdef")

x=set\_1.difference(set\_2)

y=set\_2.difference(set\_1)

print(x,y)===>{'b', 'a'} {'f', 'e'}

**5.Python标识符和关键字：**

**标识符：**在现实生活中我们要表示一些事物可以给它起名字：太阳、花。。。

在程序中要表示一些事物，我们需要定义一些符号和名称，这些符号和名称就是标识符。

eg：变量和函数名都属于标识符。

**标识符的规则：**

1. **首字符必须是字符或下划线；**
2. **非首字符可以是字母、数字、下划线，可任意组合；**

**Python的标识符不能以数字开头，切除了下划线以外的符号都不可以使用。**

**一般情况下，\_a和\_\_a都被视为私有的；\_\_a\_\_风格变量在系统中有特殊含义，如\_\_init\_\_被视为构造函数。**

Name\_1=66；不能3a=6。

**关键字：keyword。**关键字是一种特殊的标识符，关键字是python自己定义的不允许开发者自定义。

import keyword

print(keyword.kwlist)

python提供了很多关键字：33个。C语言的关键字是32个。

如何查看python的关键字：help() ————> keywords

False class from or

None continue global pass

True def if raise

and del import return

as elif in try

assert else is while

async except lambda with

await finally nonlocal yield

break for not

**注意：变量名和函数名都属于标识符，要符合标识符的规范，并且不能起关键字。**

**疑问：在python中插入一个内容时发现会出现替换情况，怎么办？**

**【按一下insert就好。】**

**6.数字类型的转换：**

不同的数字类型之间是可以转换的只是需要用到一些函数。

常见：int() float()

a=6.8--------- -----------a=6.9999999999999999999999

b=25---------- ---------b=25

print(a,b)-------- ------print(a,b)

print(int(a),float(b))====>6 25.0======>7 25.0

注意：取整只是取整数部分，不会四舍五入。

**7.Python的运算符**

**1)算数运算符：+、-、\*、/、%（取余/求模）、//（取整）、\*\*（乘方）**

print('5 + 6')； 5 + 6 print(5 + 6)； 11

print(10 % 3)； 1 print(102 % 2) 0

print(5 - 6)； -1 print(5 \* 6)； 30

print(5 / 6)； 0.8333333333333334

print(10 // 3)； 3 print(5 \*\* 2) 25

print(4.3 - 2)； 2.3 print(4.1 - 2) 2.0999999999999996

为什么4.1-2的结果差一点点？因为计算机只能识别二进制，4.1变为二进制使用(乘数法)会有误差。

1. 符合运算符： **i = i + 1 == i += 1**

c=c+a------------>c+=a；

c=c-a------------>c-=a；

c=c\*a------------>c\*=a；

c=c/a------------>c/=a；

c=c\*\*a----------->c\*\*=a；

c=c%a------------>c%=a;

c=c//a----------->c//=a

1. **比较运算符：<、 > 、<=（小于等于）、 >=（大于等于） 、!=（不等于）**
2. **逻辑运算符：and or not（&& || not）【与、或、非】**

and：逻辑与——x and y：——如果x为真，则继续判断并输出y。如果为假，直接输出x。

or：逻辑或——x or y：——如果x为假，则继续判断并输出y。如果x为真，直接输出x。

not：逻辑非——not：——不是！

1. **成员运算符：in 和 not in**

in：如果在指定的序列中找到了成员则人会True，否则返回False。

not in：如果在指定的序列中没有找到指定的成员则返回True，否则返回False。

eg：

a = 889  
list = [9, 5, 66, 88, 33, 57]  
if a in list:  
 print("哈哈")  
else:  
 print("呵呵")  
  
a = 889  
list = [9, 5, 66, 88, 33, 57]  
if a not in list:  
 print("哈哈")  
else:  
 print("呵呵")

1. **运算符优先级：**

逻辑运算符的优先级( )>not>and>or

算术运算符\*\*--------------------------->逻辑运算符 or

算术运算符>关系运算符>赋值运算符>成员运算符>逻辑运算符

**挑战：“==”在python是什么意思它与is有什么区别？**

**is:相等性的比较，比较两个对象的值是否相等**

**==：一致性的比较，比较两个对象的内存空间地址是否相同**

**=：表示变量与值得关系； eg：a = 007**

**4.Python常用的语句:if; elif; else ; while; for**

**任何程度都可以仅用三种结构来实现：**

顺序结构：**就是按照程序书写的顺序执行，除非特别声明，程序总是按照顺序执行。**

选择结构：就是程序有条件的执行。（if）

循环结构：就是程序有条件的重复执行。（while/for）

**选择结构：中使用最多的是if判断：**

**1.if-------------------------单分支**

age=int(input("请输入年龄:"))

if age>=35:

print("你是一个三旬老汉！")

**2.if else--------------------双分支**

age=int(input("请输入年龄:"))

if age>=35:

print("你是一个三旬老汉！")

else:

print("你是一个小奶狗！")

**3.if 嵌套：**

嵌套可以在if中嵌套，也可以在else中嵌套。

在if中可以嵌套一个单分支，也可以嵌套双分支，else一样。

m=int(input("请输入一个月份:"))  
if m<=6:  
 if 0<m<=3:  
 print("第一季度")  
 else:  
 print("第二季度")  
 print("上半年")  
else:  
 if 6<m<=9:  
 print("第三季度")  
 else:  
 print("第三季度")  
 print("下半年")

**4.if elif——————————多分支**

语法：

if 表达式1：

语句块1

elif表达式2：

语句块2

elif表达式3：

语句块3

。。。

else：

语句块n

多分支选择结构由if、一个或多个elif和一个else子块组成，else子块可省略。一个if语句可以包含多个elif语句，但结尾最多只能有一个else。

h=float(input("请输入身高:"))

m=float(input("请输入体重:"))

BIM=m/h\*\*2

print(BIM)

if BIM<18.5:

print("太瘦")

elif 18.5<=BIM<25:

print("正常")

elif 25<=BIM<28:

print("微胖")

elif 28<=BIM<32:

print("肥胖")

else:

print("严重肥胖")

**总结：**

**1.else、elif为if语句的子语句块，不能独立使用。**

**2、每个条件后面要使用冒号“:”，表示满足条件后需要执行的语句块，选择结构和循环结构中是冒号也是必须要有的。**

**3、使用缩进来划分语句块，相同缩进数的语句组成一个语句块。**

**循环结构：**

1. **while 循环**

语法：while 条件表达式

满足条件表达式时执行的语句

eg：使用while循环计算1--100之间的整数和。

i=0

s=0

while i<=100:

s=s+i#s+=i

i=i+1#i+=1

print(s)====>5050

1. **while嵌套：应该也是先内后外**

指的是在while逊汗中又包含while循环。

语法：while条件1：

满足条件1时做的事情1

。。。

while 条件2：

满足条件2时做的事情2

。。。。。。。

使用while嵌套实现：

\*\*\*\*

\*\*\*\*

\*\*\*\*

\*\*\*\*

\*\*\*\*

\*\*\*\*

思路：首先输出一个\*，循环四遍变为\*\*\*\* 再把\*\*\*\*循环六遍。

print(‘\*’)

python默认是换行的，想要不换行使用**end=""**

j = 0  
while j < 6:  
 print('')  
 j += 1  
 i = 1  
 while i <= 4:  
 print('\*', end='')  
 i += 1

1. **for循环：python的for循环可以遍历任何有序的项目。**

注意：

for和else后面冒号不能丢，循环体、语句块缩进严格对齐。

list\_1 = [9,5,2,7,9]

print(list\_1)

for i in list\_1:

print(list\_1)

上述的for循环可以将列表的元素循环一遍。要深刻理解for还需要结合其他内容来讲解：

1>结合range()函数讲解for循环

**range()函数是python提供的一个内置函数，它可以生成一个数字序列。**

range()格式：

range(start,end,step) start:开始 end：结束 step:步长

例如：

for i in range(1,10):

print(i)

**range()函数属于前闭后开，步长为1可以省略。**

print([[0 for i in range(5)] for j in range(5)])  
[[0, 0, 0, 0, 0], [0, 0, 0, 0, 0], [0, 0, 0, 0, 0], [0, 0, 0, 0, 0], [0, 0, 0, 0, 0]]  
print([0 for i in range(5)])

[0, 0, 0, 0, 0]

for i in range(1,10,2):

print(i)

2>结合break和continue讲解for循环

python的break和continue与C语言的一样。

**break：结束整个循环。**

**continue:结束本次循环，继续下一次循环。**

list\_c = []  
list\_d = []  
for i in range(1, 10):  
 list\_c.append(i)  
 if i == 3:  
 list\_d.append(i)  
 break  
 print(i)  
print(list\_c) # [1, 2, 3]  
print(list\_d) # [3]

为什么没有结果？

因为print()在break后，而break是结束的意思。

list\_c = []  
list\_d = []  
for i in range(1, 10):  
 list\_c.append(i)  
 if i == 3:  
 list\_d.append(i)  
 continue  
 print(i)  
print(list\_c) # [1, 2, 3, 4, 5, 6, 7, 8, 9]  
print(list\_d) # [3]

对之前使用while的循环做的案例采用for循环：

i = 0  
s = 0  
while i <= 100:  
 s = s + i # s+=i  
 i = i + 1 # i+=1  
print(s) # 5050

s = 0  
for i in range(1, 101):  
 s += i  
print(s) # 5050

1. **for嵌套：先内而外**
2. # 9\*9乘法口诀  
   for i in range(1, 10):  
    for j in range(1, i + 1):  
    print("%d \* %d = %d" % (i, j, i \* j), end="\t")  
    print()

使用for嵌套实现：

\*\*\*\*

\*\*\*\*

\*\*\*\*

\*\*\*\*

\*\*\*\*

\*\*\*\*

1. for i in range(1, 7):  
    print("")  
    for j in range(1, 5):  
    print("\*", end="")
2. 一般程序中不建议超过三层嵌套。

**本章总结：**

本章介绍了python常用的结构：顺序结构、选择机构循环结构。其中选择结构有：if单分支、if else双分支、if嵌套、if elif 多分支。循环有两种：while循环和for循环。

**5.函数**

软件——>程序——>子系统——>模块——>函数

**1.函数的概念：**

函数是软件的最基本的组成单位。函数是组织好的可以重复使用的，用来实现单一或者相关功能的代码段。

python提供了很多内置函数：print() input() len() range() max()...

**2.定义函数：**

定义函数就是把具有独立功能的代码段组织成为一个代码块以便在需要时调用。

专业来讲定义函数就是封装独立的功能。

调用函数：专业来讲调用函数就是享受这个封装成果。

python使用def(define)关键字来定义函数：格式：

def 函数名(形参列表):————def sum()：

函数体【由零条到多条可执行语句组成的代码块】——— print(r)

return 表达式————return(r)

def greet\_user():  
 print('what\'s your name')  
 return 'what\'s your name'

基于以上格式说明：

1. 定义函数都是以def开头的后面紧跟**函数名和圆括号。**
2. 函数名和变量名一样都属于标识符，命名需要符合标识符并且不能是关键字（函数名应该由一个或多个有意义的单词连缀而成，每个单词的字母全部小写，单词与单词之间使用下画线分隔）。
3. 函数的内容以**冒号起始并且有缩进；**
4. 形参列表：用于定义该函数可以接收的参数。形参列表由多个形参名组成，**多个形参名之间以英文逗号（,）**隔开。一旦在定义函数时指定了形参列表，调用该函数时就必须传入相应的参数值，也就是说，谁调用函数谁负责为形参赋值。
5. **return表达式结束函数，选择性的返回一个值给调用方，不带表达式的return相当于返回了None。**

注意，在创建函数时，即使函数不需要参数，也必须保留一空的“()”，否则 Python 解释器将提示“invaild syntax”(无效语法)错误。另外，如果想定义一个没有任何功能的空函数，可以使用 **pass 语句作为占位符**。

1. **调用函数：**

让定义的函数运行非常简单，只需要调用函数即可。

调用函数也非常简单，通过函数名()的方式就能完成调用。

——————函数名() 即为调用函数

def get\_name(user\_name):  
 return 'please input text:' + user\_name.title() + '!'  
name = get\_name(‘刘家华’)  
print(get\_name('刘家华'))

print(name) # please input text:刘家华!

1. **函数的参数：**

**上述的函数没有参数**，只能计算6+7，不能计算任意两个数，这样使得该函数的局限性非常大。为了让我们定义的函数容易的计算任意两个数，我们可以定义参数，让这两个参数来接收用户传递的值。这里面就有形参和实参的概念。

**形参**：本质上就是一个**变量**。 ——add(a, b)

**实参**：本质上就是一个**实际值**。——add(3, 5)

# 形参和实参：形参（user\_h, user\_sex），实参（'185', 'boy'）  
def get\_user\_info(user\_h, user\_sex):  
 return 'this\'s a ' + user\_sex.title() + ',' + 'high:' + user\_h.title()

定义函数时写的参数称为形式参数(形参)，调用函数时的赋值称为实际参数（实参）。形参与实参要类型一致并且呈现一一对应关系。

A函数调用B函数：其实就是A函数把实参传递给B函数的形参，B函数经过计算会给A函数的一个返回值。

|  |  |
| --- | --- |
| a = int(input("请输入a:")) 3 b = int(input("请输出b:")) 5 def add(a, b):  c = a + b  print(c)  return c aa = add(8, 6) print(aa)  为什么结果是14不是8呢？  因为按照顺序，8和6是后面赋值会覆盖前面的赋值。 | def add(a, b):  a = int(input("请输入a:")) 3  b = int(input("请输出b:")) 5  c = a + b  print(c)  return c aa = add(8, 6) print(aa)  结果为什么是8不是14？ |

**默认参数**：定义函数时，我们可以使用函数的参数设置默认值，这个参数就是默认参数，当调用函数时由于默认参数在定义时已经被赋值，所以可以忽略，而其他参数必须传入。（默认参数本质上其实就是为了省事。）

**如果有默认参数但是又传入了新值怎么办？**

**新传入的值会覆盖默认参数。**

sum(a=7,b=8,c)错、 sum(a,b=8,c)错、 sum(a=6,b,c)错

sum(a=2,b=7,c=5)对、 sum(a,b=7,c=8)对

**注意!!!默认参数不能仅在参数列表的前面。**

|  |  |
| --- | --- |
| def add\_sum(a,b,c=8):  d=a+b+c  print(d)  add\_sum(1,2)  11 | def add\_sum(a,b,c=8):  d=a+b+c  print(d)  add\_sum(1,2,3)  6 |

**不定长参数**:之前说过：实参要与形参呈现一一对应关系，但是有时候我们希望一个函数能够处理的参数比当初定义的参数多，这个时候我们可以使用不定长参数。

（默认参数只能解决实际值比参数多的情况）

**\*args**：不定长参数，用来将参数打包成tuple给函数体调用。

\*\*args:键值对应的数值

user\_info = {}  
def add(a, b, \*\*info\_data):  
 user\_info['name'] = a  
 user\_info['age'] = b  
 for keys, values in info\_data.items():  
 user\_info[keys] = values  
 return user\_info  
print(add('kobe', '42', high='196', weight='150kg'))

{'name': 'kobe', 'age': '42', 'high': '196', 'weight': '150kg'}

|  |  |
| --- | --- |
| def test\_one(a,b,c):  print(a,b,c)  test\_one(1,3,5,7,9) | def test\_one(a, b, **\*args**):  print(a, b, **\*args**)  test\_one(1,3,**5,7,9**) |

1. **函数的返回值：return**

返回值：所谓的返回值就是程序中的函数在完成一件事情后，**最终给调用方的结果**。

|  |  |  |
| --- | --- | --- |
| def yuanmianji():  r=6  PI=3.14  s=PI\*r\*r  print(s)  return s  yuanmianji()  return表示结束，结束后再输出没有意义 | def yuanmianji():  r=6  PI=3.14  s=PI\*r\*r  print(s)  return s  print(yuanmianji())  函数本身有输出 最终又输出了返回值，所以有两个结果。 | def yuanmianji():  r=6  PI=3.14  s=PI\*r\*r  print(s)  print(yuanmianji())  为什么第二个值为none？因为没有renturn。 |

**print()只是在屏幕输出，而return不在屏幕输出但是会给调用者一个值。**

**如果你只想在屏幕输出，只需要print()即可，如果你想让调用函数获得返回的结果就需要return。**

A函数调用B函数本质上是A函数把实参传递给B函数的形参，此时B函数可以输出，也可以给A函数一个返回值。

好比是相亲：我(调用者A函数) 相亲对象(被调用者B函数) 媒人(return)

**总结：**

如果你只想输出可以只写print(),如果只想给调用者一个返回值可以只写return，如果既想输出还想给调用者一个返回值，可以先print()输出再return。

警告：不要参与任何形式的赌博，包括网络赌博。假如函数写的非常公平，但是到了你永远返回谢谢惠顾。

1. **函数的类型：根据函数的参数和返回值，我们可以将函数分为四类：**

|  |  |
| --- | --- |
| 无参数，无返回值类型。  def demo():  print("\*"\*25)  print("人生苦短,python是岸!")  print("\*"\*25) | 无参数，无返回值类型。  def yuanmianji():  r=6  PI=3.14  s=PI\*r\*r  return s  print(yuanmianji()) |
| 有参数，无返回值类型。  def yuanmianji(r):  PI=3.14  s=PI\*r\*r  print(s)  yuanmianji(6) | 有参数，有返回值类型。  def yuanmianji(r):  PI=3.14  s=PI\*r\*r  print(s)  return s  yuanmianji(6) |

**7.函数的嵌套和递归：**

1. **嵌套**

指的是一个函数中又包含另一个函数。

|  |  |
| --- | --- |
| def x1():  print("搞笑")  print("你？")  def x2():  print("真你")  x1()  print("你我呢吗")  x2() | list\_1=[]  for i in range(0,20,1):  if i==0 or i==1:  list\_1.append(1)  else: list\_1.append(list\_1[i-2]+list\_1[i-1])  print(list\_1) |

如何评价一个开发人员：看键盘！C程序:F9 F10 python:F7 F8

断点：可以让程序停在想要的地方。

F8：单步调试：step over ：如果是函数则直接执行。

F7：单步调试：step into：如果是函数则进入函数内部执行。

1. **递归**

指的是一个函数调用的不是另一个函数而是这个函数本身，这种调用称为递归。递函数必须要有结束条件否则会无限循环。递归是一种特殊的嵌套。

|  |  |
| --- | --- |
| def jiechen(n):  if n==0 or n==1:  return 1  else:  return n\*jiechen(n-1)  print(jiechen(5)) | def fb(n):  if n==1 or n==2:  return 1  else:  return fb(n-1)+fb(n-2)  print(fb(8)) |

**8.变量的作用域：**

python的变量并不是什么位置都能访问的，访问权决定于这个变量是在哪里赋值。

局部变量 vs 全局变量：

|  |  |
| --- | --- |
| a=10  def test\_1():  a=100  print(a)  test\_1()==100 | a=10  def test\_1():  b=100  print(a)  test\_1()====10 |

**局部变量：指的是定义在函数内的变量，即def函数内的变量，只能在def函数内使用。**

**全局变量：指的是定义在函数外的变量，全局变量整个程序都可以使用。**

全局变量在函数外，局部变量在函数内。程序执行是由内向外查找变量的。同时定义全局变量和局部变量，局部变量会覆盖全局变量。全局变量哪儿都可以用，局部变量只能在函数内用。全局变量变量名不能一样，不然会后者会覆盖前者；局部变量变量名可以一样，互不影响。

a = 80  
a = 10000  
print(a)  
10000  
  
def test\_2():  
 a = 15  
 print(a)  
  
  
def test\_3():  
 a = 30  
 print(a)  
  
  
test\_2() # 15  
test\_3() # 30

变量本质是一块内存空间，一旦函数结束，变量随之会消失。

python的变量作用域还可以细分为：L-E-G-B

L：local 函数内区域

E:enclosing:嵌套函数区域

G：global：全局作用域

B：build-in：内建作用域

python的变量采用：L--E--G--B：首先在函数内查找，一次向外。

**9.随机函数：**

**python中random模块用于生成随机函数，**

**random模块提供了很多函数。**

**import random**

**random.random() 该函数用于生成0--1之间的浮点数。**

**random.uniform(a, b) 该函数用于生成a,b之间的浮点数。（a不一定比b小）**

**random.randint(a, b) 该函数用于生成a,b之间的整数，a一定要小于b。范围[a,b]**

**random.randrange(start,end,step)该函数用于返回start和end 之间的整数，可以设置step的范围()**

**random.shuffle() 该函数用于将列表的元素打乱，就是俗称的洗牌函数。（只能输出一个元素）**

|  |  |  |
| --- | --- | --- |
| p = ["红K", "大王", "梅Q", "方5"] print(p) random.shuffle(p) for i in p:  print(i) | import random  print(random.random())  print(random.random())  print(random.random())  print(random.uniform(1,10))  print(random.uniform(1,10))  print(random.uniform(1,10))  print(random.uniform(100,10)) | print(random.randint(1,5))  print(random.randint(1,5))  print(random.randint(10,5))  print(random.randint(1,5))  print(random.randint(1,5))  print(random.randint(10,5)) |

**10.时间和日期函数：**python中与时间相关的模块有三个：time、datetime、calendar

**time与datetime的区别：datetime是一个使用面向对象思想设计的模块，他比time模块增加了很多类函数。**

1. **时间戳：timestamp：**

时间戳能唯一的表示某一刻的时间。时间戳主要用在数字签名、知识产权保护、金融转账、股票交易等方面。时间戳是从1970年1月1日00:00:00秒计算偏移量。

|  |  |
| --- | --- |
| import time  print(time.time())  # 输出的就是时间戳  1593329294.4934788  1593755838.0141323 | import time  print(time.time())  print(time.time())  time.sleep(5)  print(time.time())  print(time.time()) |

**%Y：四位数年份**

**%y：两位数年份**

**%m：月份 %d：天数**

**%H：24小时制**

**%I：12小时制**

**%M：分钟**

**%S：秒钟**

**%p：上下午**

**%A：星期**

1. **格式化时间字符串：strftime()—————>格式：**

print(time.strftime("%Y-%m-%d %H\_%M\_%S"))  
print('%d' % (time.time())) # 时间戳取整数  
time\_aa = time.time()  
print((str(time\_aa).split('.'))[0])  
print(time.localtime()) # 当前的具体时间  
print(time.strftime("%Y:%m:%d %I:%M:%S", time.localtime()))

import time

print(time.localtime()) # 当前的具体时间

time.struct\_time(tm\_year=2022, tm\_mon=6, tm\_mday=3, tm\_hour=17, tm\_min=12, tm\_sec=34, tm\_wday=4, tm\_yday=154, tm\_isdst=0)

暂停5s：time.sleep(5)

print(time.strftime("%Y-%m-%d %H\_%M\_%S"))

2022-06-03 17\_10\_32

import time

print(time.strftime("%Y:%m:%d %I:%M:%S", time.localtime()))

2020:07:03 02:04:17

1. **时间元组：time.struct\_time**

import time

print(time.localtime())

==>time.struct\_time(tm\_year=2020, tm\_mon=7, tm\_mday=3, tm\_hour=14, tm\_min=4, tm\_sec=17, tm\_wday=4, tm\_yday=185, tm\_isdst=0)

1. **datatime:**

print(datetime.date.today().strftime('%d/%m/%Y'))

02/06/2022

print(datetime.date(1941, 1, 5).strftime('%d/%m/%Y'))

05/01/1941

print((datetime.date(1941, 1, 5) + datetime.timedelta(days=1)).strftime('%Y/%m/%d'))

1941/01/06

print((datetime.date(1941, 1, 5).replace(year=datetime.date(1941, 1, 5).year + 1)).strftime('%Y/%m/%d'))

1942/01/06

wday:一周的第几天(0-6)

yday:一年的第几天

isdst：夏令时(1为夏令时，0不是夏令时)

calendar：该模块用于处理年历和月历:calendar.calender (9999)（年份） calendar.month(2020,6) （月份）

**import calendar**

a = calendar.month(2020, 6)  
print(a)

输出：2020年6月份的日历

June 2020

Mo Tu We Th Fr Sa Su

1 2 3 4 5 6 7

8 9 10 11 12 13 14

15 16 17 18 19 20 21

22 23 24 25 26 27 28

29 30

b = calendar.calendar(2022)  
print(b)

输出：9999年的全年日历

**6.异常：**

1. **介绍异常：异常会让程序无法继续进行下去**

python在执行的过程中产生的错误称为异常。例如：打开不存在的文件、索引越界、语法错误等。

1. **常见的异常类：**
2. **NameError: 访问一个未声明的变量。**
3. **IndexError： 索引越界。**
4. **FileNotFound：访问一个不存在的文件。**

**4>SyntaxError: 语法错误**

**5>ZeroDivesionError:除数为0**

**6>TypeError: 类型错误（不同数据类型之间的无效操作）**

**7>KeyError: 字典中的键不存在**

**8>AttributeError: 试图访问的对象属性不存在**

**9>AssertionError: 当assert关键字后的条件为假时程序回你停止并抛出的异常。**

1. **异常处理：**

python异常处理的能力非常强大，它能准确反馈异常的信息并且让程序继续执行下去。也能帮助研发人员定位到问题的位置和原因。

总结：异常处理的作用：1>能够友好的呈现异常信息方便识别。

2>让程序继续执行下去。

与异常处理相关的主要有三个：try、except、raise

python使用try-except语句来处理异常，其中try用于检测异常，except用于捕获异常。

语法： **try:**

**被检测**

**except：**

**异常处理的语句**

try:  
 num\_1 = int(input("请输入第一个数:"))  
 num\_2 = int(input("请输入第二个数:"))  
 s = num\_1 / num\_2  
 print(s)  
except:  
 print("这段代码有问题,除数不能为0!")  
finally:  
 print('it\'s ok')

请输入第一个数:8

请输入第二个数:0

这段代码有问题,除数不能为0!

it's ok

**7.文件操作和模块**

1. python的文件操作：文件就是可以用来保存数据和提供数据的。文件的目的就是为了让程序在下一次执行的时候直接使用之前的数据而不需要重新计算。
2. 文件的打开与关闭
3. 打开文件：在python中使用open打开文件：格式：**open (文件名, 访问模式)**

访问模式：python提供了很多访问模式最主要的有：r、w、a：

r：只读，默认模式。

w：编辑，如果打开的文件是存在的则修改，如果不存在则新建。

a: 追加，如果文件存在则会把内容放在文件的末尾。

1. 文件关闭：凡是打开的文件切记使用close方法关闭。即使文件在程序退出后会自动关闭，但是考虑到数据的安全性，在每次使用完后都要用close关闭，否则程序一旦崩溃可能会导致数据丢失。

**操作小技巧：**

**1>快速注释:ctrl + /**

**2>批量缩进:选中tab**

**3>批量去掉缩进:选中shift\_tab**

**获取当前路径的方法：**

**1>右键--copy path**

**2>查看控制台**

**3>import os**

**print(os.getcwd())**

1. 文件的读写：文件最主要的功能就是提供数据和写入数据。
2. **向文件写入数据：**

向文件写入数据使用write方法：

**file=open("test01.txt","w",encoding='utf-8')**

**file.write("生命的意义在于解放！")**

**file.close()**

1. **读文件：从文件中读取数据有很多种方法，不能直接print()。**

方法一：read方法 格式：read(size) # 返回的字符串

方法二：readline方法 格式：readline 只读取一行

方法三：readlines方法 格式：readlines 一次性读取：与read很相似，但是输出的形式不一样，

**readlines返回一个列表，列表的每一个元素为之前每一行的内容。**

a=open("test01.txt","r",encoding="utf-8")

print(a.read()) **# 默认读取全部内容（str）**

a=open("test01.txt","r",encoding="utf-8")

print(a.read(5))  **# 读取前5个字符串**

a=open("test01.txt","r",encoding="utf-8")

print(a.readline()) **# 默认读取一行（str）**

a=open("test01.txt","r",encoding="utf-8")

print(a.readline(5)) **# 读取前5个字符串**

a=open("test01.txt","r",encoding="utf-8")

print(a.readlines()) **# 默认读取全部内容（list）**

a=open("test01.txt","r",encoding="utf-8")

print(a.readlines(30))**# 读取前三十个字符，如果第三十个字符在列表中的第二个元素那就输出到第二个列表元素（list）**

**总结：**

**1.不加参数：read用于读取全部文件内容，readline读取一行，readlines也是全部读取但是以列表显示。**

**2.加参数：read表示读取内容长度，readline表示读取一一行的长度，如果长度超过第一行只读取第一行，readlines如果不超过第n行长度则只显示前n行。**

**补充：with with是从python2.5引入的，它是一种上下文管理协议。**

list\_1=["hello world","hello linux"]

with pen("test01.txt","a") as ff:

s=''

for i in list\_1:

s+=i

s+="\n"

ff.write(s)

eg：

**test\_01.txt文件内容如下：**

jhkhkhklh  
asdgsdadsafdsgdsgdasdgsdadsafdsgdsgd  
asdgsdadsafdsgdsgdfadg  
asdgsdadsafdsgdsgdfadads

file\_path = os.path.abspath(os.path.join(os.path.dirname(\_\_file\_\_), '../')) + '\\' + 'data' + '\\' + 'test\_01.txt'  
print(file\_path)  
print(os.path.exists(file\_path))  
file\_text = open(file\_path, 'r', encoding='utf-8')  
# bb = file\_text.read(2)  
# print(bb, type(bb))  
# # jh <class 'str'>  
  
# aa = file\_text.read()  
# print(aa, type(aa))  
# # jhkhkhklh  
# # asdgsdadsafdsgdsgdasdgsdadsafdsgdsgd  
# # asdgsdadsafdsgdsgdfadg  
# # asdgsdadsafdsgdsgdfadads <class 'str'>  
  
# cc = file\_text.readline()  
# print(cc, type(cc))  
# # jhkhkhklh  
# # <class 'str'>  
# cc = file\_text.readline(4)  
# print(cc, type(cc))  
# # jhkh <class 'str'>  
  
# dd = file\_text.readlines()  
# print(dd, type(  
# dd)) # ['jhkhkhklh\n', 'asdgsdadsafdsgdsgdasdgsdadsafdsgdsgd\n', 'asdgsdadsafdsgdsgdfadg\n', 'asdgsdadsafdsgdsgdfadads'] <class 'list'>  
# dd = file\_text.readlines(9)  
# print(dd, type(dd))  
# # ['jhkhkhklh\n'] <class 'list'>  
  
list\_1 = ["hello world", "hello linux"]  
with open(file\_path, 'r', encoding='utf-8') as fd:  
 # file\_tt = fd.read()  
 # fd.close()  
 # print(file\_tt, type(file\_tt))  
 # # jhkhkhklh  
 # # asdgsdadsafdsgdsgdasdgsdadsafdsgdsgd  
 # # asdgsdadsafdsgdsgdfadg  
 # # asdgsdadsafdsgdsgdfadads <class 'str'>  
 # file\_txt = fd.readline()  
 # fd.close()  
 # print(file\_txt, type(file\_txt))  
 # # jhkhkhklh  
 # # <class 'str'>  
  
 f\_text = fd.readlines()  
 fd.close()  
print(f\_text, type(f\_text))  
# ['jhkhkhklh\n', 'asdgsdadsafdsgdsgdasdgsdadsafdsgdsgd\n', 'asdgsdadsafdsgdsgdfadg\n', 'asdgsdadsafdsgdsgdfadads'] <class 'list'>

**读取配置文件new\_1.xml**

import html  
import json  
import os  
  
file\_path = os.path.abspath(os.path.join(os.path.abspath(\_\_file\_\_), '../..', 'data'))  
print(file\_path)  
curr\_path = file\_path + '\\' + 'new\_1.xml'  
print(os.path.exists(curr\_path))  
print(curr\_path)  
'''  
读取配置文件  
'''  
# file\_text = open(curr\_path, 'r', encoding='utf-8', errors='ignore')  
# file\_text.read()  
# file\_text.close()  
# print(file\_text)  
  
#  
# with open(curr\_path, 'r', encoding='utf-8', errors='ignore') as fd:  
# file\_txt = fd.read()  
# print(file\_txt, type(file\_txt))  
# '''  
# <map>  
# <string name="last">{"hbld"w:"354jk545324624hklhklh62424vd34","updateTag":536148524987}<string>  
# <boolean name="contain\_disable\_tag"value="true"/><long name="disableTag" value="jfhdkghi378546"/>  
# </map> <class 'str'>  
# '''  
with open(curr\_path, 'r', encoding='utf-8', errors='ignore') as fd:  
 file\_txt = fd.read()  
 file\_text = html.escape(file\_txt)  
# 学会使用unescape与escape，如果有转义字符则用escape函数  
print(file\_txt, type(file\_txt))  
print(file\_text, type(file\_text))  
'''  
<map>  
 <string name="last">{"hbld":"354jk545324624hklhklh62424vd34","updateTag":536148524987}</string>  
 <boolean name="contain\_disable\_tag" value="true"/>  
 <long name="disableTag" value="jfhdkghi378546"/>  
</map> <class 'str'>  
&lt;map&gt;  
 &lt;string name=&quot;last&quot;&gt;{&quot;hbld&quot;:&quot;354jk545324624hklhklh62424vd34&quot;,&quot;updateTag&quot;:536148524987}&lt;/string&gt;  
 &lt;boolean name=&quot;contain\_disable\_tag&quot; value=&quot;true&quot;/&gt;  
 &lt;long name=&quot;disableTag&quot; value=&quot;jfhdkghi378546&quot;/&gt;  
&lt;/map&gt; <class 'str'>  
'''  
# 让文件以列表的形式读取  
with open(curr\_path, 'r', encoding='utf-8', errors='ignore') as fa:  
 file\_a = fa.readlines()  
  
aa = ''  
bb = ''  
for file\_tt in file\_a:  
 if 'updateTag' in file\_tt:  
 aa = file\_tt  
 print(aa, type(aa), len(aa))  
 # <string name="last">{"hbld":"354jk545324624hklhklh62424vd34","updateTag":536148524987}</string>  
 # <class 'str'> 100  
 aa = aa.strip().partition('>')  
 print(aa, type(aa), len(aa))  
 # ('<string name="last"', '>', '{"hbld":"354jk545324624hklhklh62424vd34","updateTag":536148524987}</string>') <class 'tuple'> 3  
 for i in aa:  
 if 'updateTag' in i:  
 bb = i  
 break  
 print(bb, type(bb), len(bb))  
 # {"hbld":"354jk545324624hklhklh62424vd34","updateTag":536148524987}</string> <class 'str'> 75  
 aa = aa[2]  
 print(aa, type(aa), len(aa))  
 # {"hbld":"354jk545324624hklhklh62424vd34","updateTag":536148524987}</string> <class 'str'> 75  
 aa = aa.strip().partition('<')  
 print(aa, type(aa), len(aa))  
 # ('{"hbld":"354jk545324624hklhklh62424vd34","updateTag":536148524987}', '<', '/string>') <class 'tuple'> 3  
 aa = aa[0]  
 print(aa, type(aa), len(aa))  
# {"hbld":"354jk545324624hklhklh62424vd34","updateTag":536148524987} <class 'str'> 66  
'''  
将json格式的字符串数据转换成字典  
'''  
dic\_aa = json.loads(aa)  
print(dic\_aa, type(dic\_aa), len(dic\_aa))  
# {'hbld': '354jk545324624hklhklh62424vd34', 'updateTag': 536148524987} <class 'dict'> 2  
  
assert dic\_aa['updateTag'] != None  
print(dic\_aa['updateTag'] != None) # True

1. 文件的重命名与删除：
2. **重命名：os模块提供了rename可以完成对文件的重命名操作**

格式：**os.rename(old, new)**

old:需要重命名的文件

new：新的文件

**import os**

**os.rename("test0001.py","test0001.doc")**

1. **删除：os模块提供remove用于对文件删除**

**import os**

**os.remove("abc")**

1. 文件夹的操作：os模块也可以对文件夹操作
2. 创建文件夹：os模块提供了mkdir用于创建文件夹
3. 删除文件夹：os模块提供了rmdir用于删除**空**文件夹
4. 删除非空文件夹：shutil模块中的rmtree用于删除非空文件夹

**import os**

**os.mkdir("xyz")**

**os.rmdir("xyz")**

**import shutil**

**shutil.rmtree("abc")**

**8.模块（module）**

1. 模块的基本使用：引入模块的格式1：import 模块1，模块2，模块3。。。

引入模块的格式2：from 模块名 import 函数1，函数2，函数3。。。

1. 模块的制作：在python中每一个python文件都可以作为一个模块，模块名就是python的文件名。

补充：如何把函数写入到python文件？(推荐使用三引号)

1. 包：python中的包(package)

为了组织好模块我们可以将多个模块放在一个包中。

包是模块所在的文件夹，包不是一个普通的文件夹，包中一定要有一个文件：\_\_init\_\_.py 该文件可为空。

引入包的两种方式:方式1：import 包.模块

方式2：from 包 import 模块

import html  
import json  
import os

from src.common.TestData\_ConfigInfo import TestData\_ConfigInfo

from src.common import \*

**注意：给模块和包起名时要符合标识符规范，不能以数字开头。**

**函数、模块、包、库、框架之间的区别：**

**1.函数:封装好的一段代码实现特定的功能。def**

**2.模块：将若干个有相关性的函数写入.py文件，供别人使用。本质上模块就是一个python文件。**

**3.包：包是模块所在的文件夹，包中包含模块，包中还可能包含子包。包中一定要有\_\_init\_\_.py**

**4.库：是完成一定功能的代码的集合，库可以是一个模块，也可以是包。**

python库分为:标准库、第三方库、自定义库

标准库：就是下载安装的python自带的模块。

人话：模块、包、库实际上都是模块，只不过是个体和集合的区别。框架>库>包>模块>函数

**5.框架：是一个架构层面的概念，是一个为了解决一个开放性问题而设计的具有一定约束性的支撑结构。**

通过框架可以快速实现一个问题的解决骨架，按照框架中的角色去填充、交互就能完成一个质量好的可维护的项目。Diango是一个开放源代码的web应用框架，由python写成，采用MTV的架构模式，即模型(M)、模板（T）、视图（V）

注意：abc命名有风险。

1. 模块的发布：(包的发布：)自己建的模块和包只能自己使用要想让其他人也能使用就需要发布。

将模块写好后我们可以将这些模块或包放在setup.py所在的文件夹下进行发布。

**注意：不建立包也能发布模块，包只是让模块的层次结构更清晰！**

详细步骤：

1. 准备好要发布的模块或包
2. 在需要发布的模块所在的文件夹下新建一个配置文件:setup.py
3. 编辑setup.py文件，指明要发布哪些包和模块：

from distutils.core import setup

setup(

name="test123练习",#安装包名称

version="V1.0.1.8", #版本号

description="hello 123",

author="XBW", #作者

py\_modules=["aa.test123","aa.test124","bb.test125"]

)

distutils是标准库的一部分，该库的目的是为开发者提供一个方便打包的方式，同时提供一个方便的安装方式。当我们有了自己的模块就可以使用distutils库的setup.py包进行打包。

1. 构建模块：在windows中开始--cmd-进入setup.py所在的文件夹。(如果不是C盘需要先切换盘符)

输入命令:python setup.py build

1. 发布生成压缩包：在刚才的位置输入命令:python setup.py sdist
2. 模块的安装：安装有两种方式：

**1>pip安装: pip install 包 +**

**2>源码包: python setup.py install**

步骤：

1>解压安装包

2>进入包装宝setup.py所在的文件夹

3>python setup.py install

1)获得并解压安装包

2)找到解压文件中setup.py路径

3)在windows中开始输入cmd 并cd 到刚才setup.py路径(如果切换其他盘符不能直接cd)

4)输入安装命令:python setup.py install

5)查看是否安装成功。(import 导入 注意别安装到其他python版本)

**python常见的库：**

|  |  |  |  |
| --- | --- | --- | --- |
| os  sys  time  datetime  calendar  hashlib：加密  logging：运行输出日志  subprocess：进程 | configparser：处理配置文件  random  shutil  paramiko：远程控制  re：正则表达式  requests、BeautifulSoup、LXMI： | 爬虫常用  matplotlib：可视化  configparser：处理配置文件 | json、pickle  elementTree:处理XML文件 |

**9.面向对象（oo:object oriented）**

1. **面向对象的概念：**

面向对象**技术**是软件工程领域的**重要技术**，这种编程思想属于上帝思维(万物皆为对象)，

面向对象**思想**是软件工程领域的**主流思想，**面向对象中有**类和对象**的概念。

1. **面向对象vs面向过程：**

**面向对象思想的三大核心是：封装、继承、多态。**

面向过程：就是分析出解决问题的步骤，然后用函数把这些步骤一一实现，使用时再一一调用。

面向对象：世间万物都存在着各种联系。在程序中使用对象来映射现实中的事物，使用对象之间的关系来面熟事物之间的联系，这种思想就是面向对象的思想。

**10.类与对象：**

1. **类与对象概念：**

面向对象中有两个非常重要的概念：类、对象。

**类是对象的集合，对象是类的实例化。**

类：我们把具有相似特征和行为的事物的集合称为类，对象是根据类创建的，一个类可以有多个对象。

1. **创建类：**

在Python中使用class来创建一个类：格式如下：

class xxx：=== class xxx（object）：

**class 类名：（首字母必须大写） class Car(object):**

类的属性：（一组特征数据：身高、肤色）

类的方法：（进行的动作和操作：说话、吃饭、开车）

class Cat:  
 # 属性  
 def catch(self):  
 return '我会喵喵叫'  
  
 def sing(self):  
 return '我会唱歌'  
  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 Tom = Cat()  
 print(Tom.catch())  
 print(Tom.sing())

1. **创建对象：**

对象不能凭空产生，需要根据类来创建，如果根据某个类创建了对象，那么这个对象就拥有这个类的所有的属性和方法。创建对象非常简单：

**对象名 = 类名()————> jiafeimao = Cat() BMW = Car()**

1. **给类添加属性：**

**方法一：给对象赋值属性**

（其实不是给类添加属性只是给对象添加属性）

class Cat:  
 # 属性  
 def catch(self):  
 return '我会喵喵叫'  
  
 def sing(self):  
 return '我会唱歌'  
  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 Tom = Cat()  
 print(Tom.catch())  
 print(Tom.sing())

# 给对象直接添加属性

Tom.color = 'black'  
print(Tom.color)

**方法二：构造方法**

刚才讲的方法一种其实并没有给类添加属性，只是给对象添加了属性。

通过方法一只能让某一个对象拥有属性，如果通过该类创建的新的对象还得继续赋值，比较麻烦，能不能直接给类添加属性？

能！构造方法就可以。

class Cat:  
 # 属性  
 def \_\_init\_\_(self):  
 self.color = 'black'  
 self.size = 'big'  
  
 def catch(self):  
 return '我会喵喵叫'  
  
 def sing(self):  
 return '我会唱歌'  
  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 Tom = Cat()  
 print(Tom.catch())  
 print(Tom.sing())  
 # Tom.color = 'black'  
 # print(Tom.color)  
 print(Tom.color)  
 print(Tom.size)

我会喵喵叫

我会唱歌

black

big

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

class Cat:  
 # 属性  
 def \_\_init\_\_(self, color, size='small'):  
 self.color = color  
 self.size = size  
  
 def catch(self):  
 return '我会喵喵叫'  
  
 def sing(self):  
 return '我会唱歌'

if \_\_name\_\_ == '\_\_main\_\_':  
 Tom = Cat('black', 'small')  
 luce = Cat('blue', 'big')

print(luce.sing())  
print(luce.size)

我会唱歌

big

luce = Cat('blue')

print(luce.size)

我会唱歌

small

构造方法：\_\_init\_\_():就是直接给类添加属性。

# 属性  
def \_\_init\_\_(self, color, size):  
 self.color = color  
 self.size = size

析构方法：\_\_del\_\_():用于释放类所占的资源，

释放类所占的内存空间。

**方法三：带参数的构造方法**

class Cat:  
 # 属性  
 def \_\_init\_\_(self, color, size):  
 self.color = color  
 self.size = size  
  
 def catch(self):  
 return '我会喵喵叫'  
  
 def sing(self):  
 return '我会唱歌'  
  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 Tom = Cat('black', 'small')  
 luce = Cat('blue', 'big')  
 print(Tom.catch())  
 print(Tom.sing())  
 # Tom.color = 'black'  
 # print(Tom.color)  
 print(Tom.color)  
 print(Tom.size)  
 print(luce.sing())

我会喵喵叫

我会唱歌

black

small

我会唱歌

1. 析构方法：

**\_\_del\_\_() 用于释放类所占的资源。**

**\_\_del\_\_(): 放在程序的前与后都可以，因为整个程序执行完才执行析构方法释放资源。**

**类似于：time.time random.random**

import time

class Person:  
 def \_\_del\_\_(self): # 析构方法  
 print("---使用析构方法释放了资源---")  
  
 def \_\_init\_\_(self, name, age):  
 self.name = name  
 self.age = age  
  
 def study(self):  
 return ("学习")

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

body = Person('李雪健', '男')  
print(body.study())

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

class Person:  
 def \_\_init\_\_(self, name, age):  
 self.name = name  
 self.age = age  
  
 def study(self):  
 return ("学习")

def \_\_del\_\_(self): # 析构方法  
 print("---使用析构方法释放了资源---")

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

body = Person('李雪健', '男')  
print(body.study())

1. **self：在属性和方法中的第一个参数永远是self。**

（self就是自己的意思，我们可以把self当成是C++语言的this指针，表示本身。）

当某个对象访问属性和方法时，python解释器就会把这个对象作为第一个参数传递给self，开发者只需要传递后面的参数即可。self其实就是为了省事，不需要再单独给属性和方法添加参数。

类似于：

**time.time**

**random.random**

eg:

class Cat:  
 def \_\_init\_\_(ccc, aaa, bbb="公"):  
 ccc.color = aaa  
 ccc.sex = bbb  
  
 def catch(self):  
 return ("---抓老鼠--")  
  
 def sing(self):  
 return ("--喵喵喵----")  
  
  
jiafeimaoA = Cat("黄色", "公")  
tom = Cat("黑色", "母")  
print("一只%s的%s加菲猫正在%s" % (jiafeimaoA.color, jiafeimaoA.sex, jiafeimaoA.catch()))  
print("一只{}的{}加菲猫正在{}".format(jiafeimaoA.color, jiafeimaoA.sex, jiafeimaoA.catch()))  
print("一只%s的%stom正在%s" % (tom.color, tom.sex, tom.catch()))

一只黄色的公加菲猫正在---抓老鼠--

一只黄色的公加菲猫正在---抓老鼠--

一只黑色的母tom正在---抓老鼠—

if \_\_name\_\_ == '\_\_main\_\_':  
 guys = Cat('嘻嘻')  
 print('tom正在' + guys.catch())  
 print('tom正在%s的笑' % guys.color)  
 print('tom的性别是{}的'.format(guys.sex))

tom正在---抓老鼠--

tom正在嘻嘻的笑

tom的性别是公的

作业2：编写一个student类，包括姓名、性别、年龄、家庭住址，并在info()方法中显示这些信息。 根据类生成两个对象s1、s2分别调用对象的info()方法输出学院信息。

class Student:  
 def \_\_del\_\_(self):  
 return '释放资源'  
  
 def \_\_init\_\_(self, name, sex, age, address):  
 self.name = name  
 self.sex = sex  
 self.age = age  
 self.address = address  
  
def info(self):  
 print(self.name + '是一名 ' + self.sex + '孩子，' + '今年' + str(self.age) + '岁' + '来自' + self.address)

s1 = Student("小明", "男", 18, "陕西")  
s2 = Student("小丽", "女", 17, "山西")  
s1.info()  
s2.info()

小明是一名男孩子，今年18岁来自陕西

小丽是一名女孩子，今年17岁来自山西

**11.封装：**

封装：在python中，类的封装是指将类的某些部分(属性、方法)隐藏起来，封装也称为私有化属性和方法。通常我们把隐藏属性、方法和实现细节的过程称为封装。

封装的目的：

第一：保护隐私，明确区分内外。

第二：隔离复杂度。

封装具体可分为：封装数据和封装方法

封装数据：类似于日常生活中把物品包装起来，目的是起到隐私保护的作用。

**封装方法：类似于日常生活中把复杂的步骤封装成简单的动作。快门就是给摄影小白提供的一个方法，这个方法将内部复杂的照相功能封装隐藏起来，只对外提供一个按钮，你只需要轻轻一按就可完成拍照，目的是为了隔离复杂度。**

注意：封装时的隐藏并不是最终的目的，隐藏只是一个手段，隐藏起来后对外提供操作该数据的接口，然后我们可以在接口上附上对该数据操作的限制，以完成对数据的控制。

**封装的格式：属性或方法名称前加上\_\_（双下划线）**

**私有属性不能随意访问，要封装，需要先set再get；**

**set是封装，get是获得封装内容。set好比是给家中的门上一把锁，而get是拿钥匙打开**

class Teacher:  
 def \_\_init\_\_(self, name, age):  
 self.\_\_name = name  
 self.\_\_age = age  
  
 def set\_info(self, name, age):  
 if not isinstance(name, str):  
 raise TypeError('姓名必须是字符串类型')  
 if not isinstance(age, int):  
 raise TypeError('年龄必须是整型')  
 self.\_\_name = name  
 self.\_\_age = age  
  
 def get\_info(self):  
 print('姓名:%s,年龄:%s' % (self.\_\_name, self.\_\_age))  
  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 t = Teacher("zhuorunfa", 18)  
 t.set\_info("xbw", 25)  
 t.get\_info()

class Cat:  
 def \_\_init\_\_(self, color="白色"):  
 self.color = color  
  
 def sing(self):  
 return "--喵喵喵--"  
  
  
class Jiafeimao(Cat):  
 pass  
  
  
TOM = Jiafeimao()  
print("%s的猫正在%s" % (TOM.color, TOM.sing()))

class Person: # —（类名首字母必须大写）  
 def \_\_init\_\_(self, name, age): # （\_\_init\_\_ ——构造方法）  
 self.name = name  
 self.\_\_age = age # ——（年龄需要封装，则需要在）  
  
 def sing(self):  
 return "唱歌"

if \_\_name\_\_ == '\_\_main\_\_':  
 laowang = Person("王思聪", 33)  
 laowang.age = 300  
 print(laowang.age)

300

**输出：300 因为访问出错，应该访问封装的年龄**

**应该这样访问：print（laowang.\_\_age）**

**12.继承：**

白种人继承了父辈的白皮肤、蓝眼睛，而黑人继承了父辈的黑皮肤。这种继承与我们python的继承类似。

1.面向对象中的继承是描述事物之间的从属关系的。例如：猫和狗都继承自动物，波斯猫，加菲猫都继承自猫。

**2.类的继承指的是在现有类的基础上构建一个新的类，构建的新的类称为子类，现有的类称为父类，子类会自动拥有父类所有的属性和方法。**  **继承是创建新类的方法**

class Cat: # 父类  
 def \_\_init\_\_(self, color="白色"):  
 self.color = color  
  
 def sing(self):  
 return "--喵喵喵--"  
  
  
class Jiafeimao(Cat): # 子类  
 def sing(self):  
 return "---咪咪咪---"  
  
  
TOM = Jiafeimao()  
print("%s的猫正在%s" % (TOM.color, TOM.sing()))

白色的猫正在---咪咪咪---

3.在python如何继承：

class 子类名(父类名)

假设有一个类A， B是A的子类则：

class A(object):

class B(A)

**注意：如果这个类定义中没有标出父类**

**则这个类默认继承自父类（object）。**

4.重写父类方法：在继承中子类会自动拥有父类所有的属性和方法，但是有时候子类想要按照自己的方式来实现，也即是不想完全继承

父类的属性和方法，这个时候就要对父类的方法进行重写。

**注意：重写父类方法和属性时要和父类的名称和参数一致。**

pass：主要起到占位作用，让代码整体完整，如果定义一个方法或者函数，里面的内容为空就会报错。当我们不知道或者还不确定里面写什么内容时可以使用pass先占位后面再补充。

5.多继承：刚才讲的都是单继承，如果一个子类拥有多个父类，

这种继承就是多继承。例如：水鸟、沙发床。

class Cat: # 父类  
 def \_\_init\_\_(self, color="白色"):  
 self.color = color  
  
 def sing(self):  
 return "--喵喵喵--"  
  
  
class Jiafeimao(Cat): # 子类  
 def sing(self):  
 return "---咪咪咪---"  
  
  
class Jiafeimaoa(Cat): # 子类  
 def song(self):  
 return "---哈哈哈---"  
  
  
TOMa = Jiafeimaoa()  
print("%s的猫正在%s" % (TOMa.color, TOMa.song()))  
白色的猫正在---哈哈哈---

TOM = Jiafeimao()  
print("%s的猫正在%s" % (TOM.color, TOM.sing()))

白色的猫正在---咪咪咪---

class Car:  
 def \_\_init\_\_(self, name):  
 self.name = name  
  
 def car\_length(self):  
 return 'This is long long car'  
  
  
class Train(Car):  
 def media(self):  
 return 'This is a kind of good music'  
  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 hot\_car = Train('aaa')  
 print(hot\_car.car\_length()) # This is long long car  
 print(hot\_car.media()) # This is a kind of good music  
 print(hot\_car.name) # aaa

**13.多态：**

**多态就是一个事物拥有多种形态**

例如：动物：人、马、猴子、猪···

多态就是同一个手段处理不同的情况，也就是说一个接口多种实现。

多态的作用：应用程序不必为每一个程序编写功能调用，只需要对父类进行处理即

可，这样就大大提高的程序的可复用性。

多态的体现：不需要再修改函数

run\_twice(animal)，只需要增加类或者子类即可。就是一个接口多种实现。

a = list() # a是list类型

b = Animal() # b是Animal类型

c = Dog() # c是Dog类型

isinstance(a, list)

True

isinstance(b, Animal)

True

isinstance(c, Dog)

True

isinstance(c, Animal)

True

看来b不仅仅是Dog，c还是Animal！

不过仔细想想，这是有道理的，因为Dog是从Animal继承下来的，当我们创建了一个Dog的实例c时，我们认为c的数据类型是Dog没错，但c同时也是Animal也没错，Dog本来就是Animal的一种！

所以，在继承关系中，如果一个实例的数据类型是某个子类，那它的数据类型也可以被看做是父类。但是，反过来就不行：

>>> b = Animal()

>>> isinstance(b, Dog)

False

Dog可以看成Animal，但Animal不可以看成Dog。

class Animal(object):  
 def run(self):  
 print("Animal is running...")  
  
  
class Dog(Animal):  
 print('dog is animal')  
  
  
class Cat(Animal):  
 print('Cat is animal')  
  
  
class Tortoise(Animal):  
 def run(self):  
 print("--乌龟跑的很慢--")  
  
  
def run\_twice(animal):  
 animal.run()  
  
  
run\_twice(Animal())  
run\_twice(Dog())  
run\_twice(Cat())  
run\_twice(Tortoise())

====> 输出：

dog is animal

Cat is animal

Animal is running...

Animal is running...

Animal is running...

--乌龟跑的很慢—

class Animal(object):  
 def run(self):  
 print("Animal is running...")  
  
  
class Dog(Animal):  
 pass  
  
  
class Jingba(Dog):  
 pass  
  
  
class Cat(Animal):  
 pass  
  
  
class Tortoise(Animal):  
 def run(self):  
 print("--乌龟跑的很慢--")  
  
  
def run\_twice(animal):  
 animal.run()  
  
  
TOM = Cat()  
  
run\_twice(Animal())  
run\_twice(Dog())  
run\_twice(Cat())  
run\_twice(Tortoise())  
run\_twice(Jingba())  
run\_twice(TOM)

* 输出：

Animal is running...

Animal is running...

Animal is running...

Animal is running...

--乌龟跑的很慢--

--乌龟跑的很慢--

Animal is running...

Animal is running...

Animal is running...

Animal is running...

class Animal(object): # 父类  
 def run(self):  
 print("Animal is running...")  
  
  
class Dog(Animal): # 子类  
 pass  
  
  
a = list()  
b = Animal()  
print(isinstance(a, list)) # a是list的类型  
print(isinstance(b, Animal)) # b是Animal的类型  
print(isinstance(b, list)) # b应该是animal类型而不是list类型

输出：

True

True

False