字符串

字符串定义使用单引号或双引号均可

.title() 首字母大写

.upper() 全部大写

.lower() 全部小写

.strip() 去掉两端空格

.lstrip()去掉左边空格

.rstrip()去掉右边空格

制表符 \t , 换行符 \n

f 字符串 f”xxx{param\_a}xx” : 通过把花括号内的变量替换为其值来设置字符串的格式。

name="becky"  
miss=f"{name} Miss"  
  
print(miss)  
print(f"Hello {name.title()}")

f 字符串是python 3.6之后才有的，更早之前的版本的format 写法是

fname="becky"  
miss="{} Miss".format(name)  
print(miss)  
print("Hello {}".format(name.title()))

整数/浮点数

+-\*/ ： 加减乘除运算符

\*\* ： 乘方运算，例如3\*\*4 ，就是3的4次方运算

ps，

1. 在字符串中使用整数/浮点数时，需要显式地调用str() 方法将整数/浮点数用作字符串，如

age = 5

print("I am "+str(age) +" years old.")

2. 在python2中，整数除法的结果只包含整数部分，小数部分会被删除。如要保留小数部分，务必确保至少有一个操作数为浮点数，这样结果也会是浮点数

如，3/2 结果是1 ， 3.0/2 或 3/2.0 结果都是1.5

3. 在python 3.6 后，书写很大的数时，可使用下划线讲其中的数字分组。但打印这种下划线定义的数组时，python不会打印其中的下划线

>>> big\_num = 1\_0000\_0000\_999

>>> print(big\_num)

100000000999

赋值

可以在一行代码中给多个变量赋值，需用用逗号将变量名分开，赋给变量的值也需要用逗号分隔开

>>> x,y,z=0,1,2

>>> print(f"{x},{y},{z}")

0,1,2

注释

在python中，使用井号#标识，井号后面的内容会被python解释器忽略。

列表

定义：使用中括号[]表示列表，并使用逗号来分隔其中的元素

长度：len(xxx), 返回列表长度

索引：索引从0开始，例如xxx[2]就是访问第三个元素

索引如果为负数，就是指倒数第几个元素，例如xxx[-2]就是倒数第二个元素

bicycles=['trek','cannondale','redline','specialized']

bicycles[0] 就是第一个元素trek

bicycles[-2] 就是倒数第二个元素redline

修改元素：通过索引去指定并修改元素, eg, array[2]='abc'

添加元素：

.append(value): 在列表末尾添加元素

.insert(index,value): 在指定索引出添加元素

删除元素：

del array[index] : 删除指定索引位置的元素,eg, del array[1], 删除array中的第二个元素

.pop(): 删除末尾元素，并返回该元素的值

.pop(index): 删除指定索引位置的元素，并返回该元素的值

.remove(value): 删除第一个指定值的元素, eg array.remove('abc') , 在数组array中删除第一个值为abc的元素。

排序：xxx.sort() :对列表进行永久性排序（默认是升序排序），会改变原来列表里面元素顺序

xxx.sort(reverse=True) : 对列表进行永久性“倒序”排序

sorted(xxx): 对列表进行临时性排序，不会改变原来列表里面元素排序

sorted(xxx,reverse=True):对列表进行临时性倒序排序

xxx.reverse(): 对列表进行反转排序，会改变原来列表里面的元素排序

计算：

min(xxx): 返回最小值

max(xxx)：返回最大值

sum(xxx)：返回所有元素的总和

列表切片：使用中括号[x:y]方法截取，包含x元素，但不包含y元素

array[x:y], 从x索引元素开始，到y-1索引元素结束

array[:y], 从第一个元素开始，到y-1索引元素结束

array[x:], 从x索引元素开始，到结尾最后一个元素结束

array[:], 返回整个列表，这也是复制列表副本的方法

array[-x:-y], 从倒数第x个元素开始，到倒数第(y+1)个元素结束

array[:-y], 从第一个元素开始，到倒数第(y+1)个元素结束

array[-x:], 从倒数第x个元素开始，到结尾最后一个元素结束

复制列表: array[:], 返回一个新的与原来列表一样的列表，新旧列表不相互影响，独立的两个列表

注意：

newArray=array[:] , newArray 与array是两个独立的，互不影响的两个列表

array2=array , array2 与array 是指着同一个列表

元组 ： 就是不可改变的列表

元组使用圆括号()去定义，列表使用中括号[]去定义，

dimension=("abc","cde")

list=["abc","cde"]

元组不能通过索引去改变某个元素赋值，但可以重新给整个元组赋值

dimension[2]=xxx --> 报错，不允许

dimension=("zzz","yyy") --> 允许

## for循环

格式：for 循环第一行以for 开头，in一个list，冒号结果，循环体为for之后每行行头缩进了的内容， 行头没有缩进代表循环已借宿。

for xxx in xxx\_list :

....

....

## while 循环

while conditional\_test:

xxxxx

break 关键字退出循环

continue 关键字返回循环开头，并根据条件判断结果决定是否继续执行循环

for循环中不应修改列表，但可以使用while循环在遍历列表的同时进行修改

## 数值列表

range(a,b) : 生成一系列数字，从a开始，到b-1结束，不包含b，默认步长为1

range(a,b,c) : 生成一系列数字，从a开始，以c为步长，到b-1结束，不包含b

list(range(xxx)) : 将一系列数字转换为列表，如list(range(1,20,2))

例子

squars=[]

for number in range(1,11):

squars.append(number \*\* 2)

print(squars)

//上面例子为计算1～10的平方数，并放到squars列表中，头三行代码可以用一行“列表解析”代替

squars=[number\*\*2 for number in range(1,11)]

列表解析的格式是

列表名=[expression for iter\_val in iterable]

## if 条件判断

可使用 == != < <= > >= 等作判断符号，也可以使用布尔表达式True/False

python会将非空字符串解读为True

多个条件判断时可以使用and 或 or 去连接

1. if 语句

if conditional\_test:

do something

2. if-else 语句

if conditional\_test:

do something

else:

do others

3. if-elif 语句 (可以有多个elif判断语句,通过了条件测试后，python会跳过余下的条件测试)

if conditional\_test\_1:

do something

elif conditional\_test\_2:

do something\_2

4. if-elif-else 语句(中间可以有多个elif判断语句)

if conditional\_test\_1:

do something

elif conditional\_test\_2:

do something\_2

else:

do others

## 字典 Dictionary

定义：字典用放在大括号｛｝中的一系列键值对表示，键和值之间使用冒号分隔，每个键值对之间使用逗号分隔

Eg: alien={‘color’:’green’, ‘point’: 5}

获取与键相关的值，可以指定字典名和放在方括号内的键

alien[‘color’]

添加键值对，字典名[‘新的键‘’] = 新的值

alien[‘speed’]=’fast’

删除键值对， del 字典名[‘键名’]

del alien[‘speed’]

遍历字典中所有键值对(遍历字典是，键值对的返回顺序与存储顺序可能不一样，如有需要可以使用sorted() 函数)

for key, value in 字典名.items() :

xxxxx

遍历字典中所有的键( 遍历字典时，会默认遍历所有的键)

for key in 字典名.keys():

xxxxx

相当于

for key in 字典名

xxxxx

遍历字典中的所有值（对于包含重复元素的列表可以通过set() 函数，返回一个元素不重复的列表）

for value in 字典名.value():

xxxx

## 用户输入

函数input(xxxx) 让程序暂停运行，等待用户输入一些文本，xxxx就是要向用户显示的提示或说明

message=input(‘Please input your message’)

print(message)

函数int(xxx) : 可以让xxx从字符串转换成数字

number = input(‘please input a number’)

number = int(number)

求模运算符 % , 返回两数相除后的余数

## 函数

1. 函数定义, 关键字def去定义函数

def function\_name(parameters):

function\_body

1. 参数

形参：函数定义中的参数成为形参, 每个形参都可以指定默认值

def function\_name(param\_0=’default\_value’,param\_1,xxxx):

function\_body

实参：函数调用中传递给函数的值成为实参

1. 传递实参

位置实参：要求实参的顺序矛形参的顺序相同

关键字实参：传递给函数的实参是名称-值对(形参名=value)，与顺序无关

传递任意数量的实参

\*形参名：让python创建一个名为xxx(形参名)的空元组，并将收到的所有值都放入到这个空元组中

\*\*形参名：让python创建一个名为xxx(形参名)的空字典，并将收到的所有名称-值对都放入到这个空字典中

ps,

1. 将列表传送给函数后，函数可以对其进行修改，这种修改是永久性的。如不想修改是永久性的，可以传递列表副本(xxx[:])
2. 要让函数接受不同类型的实参，必须在函数定义中将接纳任意数量实参的形参放到最后
3. 返回值

可使用return 语句将值返回到调用函数的代码行

def function\_name(params…):

function\_body

return some\_value

python 可以返回多个值，如

函数的return 语句只能返回一个值，可以是任何类型。

因此，我们可以“**返回一个 tuple类型，来间接达到返回多个值** ”。

例:   x 除以 y 的余数与商的函数

**def    F1 ( x, y ):**

**a = x % y**

**b = (x-a) / y**

**return ( a,b )     # 也可以写作 return a, b**

**(c, d )= F1( 9, 4)        # 也可以写作 c , d = F1 ( 9, 4 )**

**print c ,d**

结果显示: **1, 2**

1. 模块

模块是扩展名为.py的文件，包含要导入到程序中的代码，.py的文件名就是模块名

导入的几种方法

import module\_name

调用方式就是 module\_name.function\_0(xxxxx) / module\_name.function\_1(xxx)

from module\_name import function\_0, function\_1, function\_2

调用方式就是 fuction\_0(xxxx) / function\_1(xxxx)

import module\_name as mn

调用方式就是 mn.function\_0(xxxxx) / mn.function\_1(xxx)

from module\_name import function\_0 as fn

调用方式就是 fn(xxxx)

## 类

在python中，首字母大写的名称指的是类。(parent\_ClassName) 是可选的，没有就不需要写明

class ClassName(parent\_ClassName):

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

……

others class body

每个于类相关联的方法定义中第一个形参必定是self，代表实例本身，在调用时，self实参自动传入，无需额外传递。

\_\_init\_\_(self,xxx) 是特殊方法，用于创建实例是传递类的属性值。 形参self必不可少，而且还必须位于其它形参的前面。 在调用\_\_init\_\_()创建实例时，将自动传入实参self。

访问属性：可使用句点表示法

my\_car.name

调用方法：可使用句点表示法去调用

my\_dog.sit()

my\_staff.say\_hello(my\_name)

给属性指定默认值：可在方法\_\_init\_\_() 内指定初始值，如果定义了初始值，在调用时就无需传递实参

导入类

导入单个类

from module import ClassName

从一个模块中导入多个类

from module import ClassNameA, ClassNameB

导入整个模块（在调用时，须通过module.ClassName 去调用）

import module

导入模块中的所有类

from module import \*

### 继承

一个类继承另一个类时，它将自动或者另一个类的所有属性跟方法；原有的类称为父类，而新类称为子类，子类继承其父类的所有属性与方法，并且可以定义属于自己的属性和方法。

创建子类时，父类必须包含在当前文件中，并位于子类的前面。

定义子类是，必须在括号内定义父类的名称。

super() 是特殊函数，将函数的父类和子类关联起来。在Python2.7中函数super()需要两个参数，就是子类名跟self

class ClassName(ParentClassName):

def \_\_init\_\_(self,a,b,c):

super().\_\_init\_\_(a,b,c)

xxx

Python2.7 的定义就是

class ClassName(ParentClassName):

def \_\_init\_\_(self,a,b,c):

super(ClassName,self).\_\_init\_\_(a,b,c)

xxx

重写父类方法：在子类中定义一个与父类方法名相同的方法即可。

## 文件处理

1. 读取文件

函数open() 接受一个打开文件名称的参数，并返回该文件对象

with open(‘filePath+fileName’) as file\_object:

file.object.read()

xxxxx

（1）使用关键字with就不需要在访问文件后将其关闭，open() 返回的文件对象仅在with代码块中可用，如要在外部使用，可在with代码块内将文件各行储存在一个列表中

（2）也可以调用open()/close() 去打开和关闭文件

（3）read() 到达文件末尾时会返回一个空字符串，可使用rstrip() 去删除。

（4）readlines() 读取文件中的每一行，并返回一个列表

（5）要以每一个行的方式检查文件时，可对文件对象使用for 循环

for line in file\_object:

xxxxx

（6）反斜杠在python中被视为转义字符，应以原始字符串(即在开头的单引号前加上r )的方式去指定路径

(7) 读取文本文件时，python都将所有文本解读为字符串。如果作为数值使用，可以使用函数int() / float() 将其转换。

2. 写入文件

要写入文本，可以在调用open()方法时传入文件打开模式实参’w’ (第二个参数), 告诉python要以写入模式打开文件

‘r’ 只读模式, 默认为只读模式

‘w’ 写入模式(先清空文件原有内容，再写入)

‘a’ 附加模式(在文件末尾追加)

‘r+’ 读取和写入模式

（1）如果要写入的文件不存在，open() 函数将自动创建

（2）python 只能将字符串写入文本文件中，如要写入数值，需先通过str()转换后再写入

（3）write(xxxx) 不会在文本文件末尾添加换行符，需额外写入换行符\n

## 异常Error

异常是使用try-except 代码块处理的,能让python执行指定的操作，同时告诉python发生异常时怎么办。

try:

xxx

except xxxError:

xxxx /pass

else:

xxxx (else部分可选)

try-except-else代码块：只有可能引发异常的代码块才放到try语句中，如果引发了异常，就会执行except代码块，没有引发异常就执行else代码块。

pass 关键字，可以用python什么都不要做。

ZeroDivisionError : 除0异常

FileNotFoundError : 文件没找到异常

## 存储数据

json.dump(data, fileObject) : 用于将data数据保存到fileObject对象的文件中

import json

data=xxxx

with open(‘fileName’,’w’) as file\_object:

json.dump(data,file\_object)

json.load(fileObject) : 以json格式读取fileObject对象的文件，并返回json对象

import json

with open(‘fileName’) as file\_object:

data = json.load(file\_object)

## 测试用例

Python标准库中的unittest 模块提供了代码测试工具。单元测试用于核实函数的某个方面没有问题，测试用例就是一组单元测试。

1. 单元测试

先导入模块unittest以及要测试的函数/模块，在创建一个继承unittest.TestCase的类，并编写一系列方法对函数不同行为的不同测试方法

import unittest

from module\_name import xxx\_class\_name

class ClassNameTestcase(unittest.Testcase):

def setUp(self,xxxx…):

xxxx

def test\_xxxx\_1(self):

xxxx

def test\_xxx\_2(self):

xxxx

if \_\_name\_\_ == ‘\_\_main\_\_’:

unittest.main()

(1) 模块unittest以及要测试的函数/模块

(2) 创建一个继承unittest.TestCase的类

(3) 测试方法都以test\_ 开头，并且test\_ 开头的方法都会自动运行

(4) 使用断言方法去合适得到的结果是不是与期望的一致

(5) 如果含有setUp() 方法，python会先运行setUp() 方法在运行各个以test\_开头的方法，并且在每个测试方法中都可以使用在setUp()方法中创建的对象

2. 断言方法

assertEqual(a,b) 🡪核实a == b

assertNotEqual(a,b) 🡪核实a !=b

assertTrue(x) 🡪核实x为True

assertFalse(x) 🡪核实x为False

assertIn(item,list) 🡪核实item在list当中

assertNotIn(item,list) 🡪 核实item不在list当中

## python编写指南

1. 函数编写指南
2. 给函数指定描述性名称，而且只使用小写字母跟下划线
3. 给形参指定默认值跟函数调用中使用关键实参时，等号两边不要有空格
4. 如果形参很多，在函数定中左括号后按下回车键，并在下一行按两次Tab键，将形参列表跟函数体分开

def function\_name(

param\_0,param\_1,param\_2,

param\_3,param\_4):

function\_body

1. 使用两个空行将相邻的函数分隔开
2. 所有的import语句都应该放开文件开头
3. 类编码风格
4. 雷鸣都采用驼峰命名法，即类名中每个单词的首字母都大写，而且不使用下划线
5. 实例名跟模块名都采用全小写格式，单词之间用下划线链接
6. 每个类定义后面殴斗包含一个文档字符串，用于描述类的功能

class ClassName():

‘’’this class is for xxx’’’

class\_body

(4) 在类中，可使用一个空行来分隔方法

（5）在模块中，可使用两个空行分隔类

（6）导入模块时，先导入标准模块库的，后导入自己编写的模块

random 模块 (https://www.runoob.com/python/func-number-random.html)

Python 标准库

python 标准库是一组模块，安装的python都包含它。可以参考http://pymotw.com/

install pip

python get-pip.py

install matplotlib

python -m pip install -U pip setuptools

python -m pip install matplotlib

python -m pip install --user matplotlib

install plotly

python -m pip install --user plotly

绘制折线图

使用matplotlib包可制作简单的图表 (https://matplotlib.org/)

matplotlib.pyplot 模块，可以制作折线图

函数subplots() 可以在一张图中绘制一个或多个图表，返回fix,ax 两个参数

ax.plot() : 尝试根据给定的数据以有意义的方式绘制图表

ax.set\_title() : 设置标题

ax.set\_xlabel() / ax\_set\_ylabel：设置x/y轴的标题

ax.tick\_params() : 设置刻度标记的大小

ax.scatter(x,y) : 传递一对xy坐标，并在对应位置绘制一个点

ax.scatter(x\_values,y\_values): 依次从每个列表中读取一个值来绘制一个点

ax.scatter(x,y,c=xxx, cmap=xxx) : 通过参数c指定要使用的颜色，如c=’red’通过cmap 指定从开始颜色渐变到结束颜色，如 cmap=pyplot.cm.Blues

函数show() 打开Mapplotlib 查看器并显示绘制的图表

函数 savefig() : 保存图表

pyplot.style.use(xxx) : 设置内置样式

使用Plotly 包来生成交互式图表，它生成的图表将自动缩放以适合屏幕大小，并且用户将鼠标指向特定元素时，将突出显示该元素的信息(<https://plotly.com/python/>)

Bar()类表示用于绘制条形图的数据集，需要一个存储x值的列表和一个存储y值的列表。

data=[Bar(x=x\_value, y=y\_values)]

Layout() 类返回一个指定图表布局和配置的对象

my\_layout = Layout(title='Becky Testing', xaxis=x\_axis\_config, yaxis=y\_axis\_config)

函数offline.plot() ,传递一个包含数据和布局的字典，还有一个文件名，指定要将图表保存位置。

offline.plot({'data': data, 'layout': my\_layout}, filename='bar.html')