Python基础

list(列表)

运算符	作用	举例	结果
+	拼接组合	[1, 2, 3] + [4, 5, 6]	[1, 2, 3, 4, 5, 6]
*	重复	['Hi!'] * 4	['Hi!', 'Hi!', 'Hi!', 'Hi!']
in	判断元素是否在列表中	3 in [1, 2, 3]	True
for in :	迭代	for x in [1, 2, 3]: print(x, end=" ")	1 2 3

函数	作用
len(list)	列表元素的个数
max(list)	返回列表元素的最大值
min(list)	返回列表元素的最小值
list(seq)	将元组转化成为列表
del list[index]	删除列表中的元素
list[a:b]	截取列表中下标$a \leq index < b$

方法	作用
list.append(obj)	在列表末尾追加新的对象
list.count(obj)	统计某个元素在列表中出现的次数
list.extend(seq)	在列表中一次性追加另一个序列中的多个值（用新的列表扩展原来的列表）

方法	作用
list.index(obj)	从列表中找出某个值的第一个匹配项的索引位置
list.insert(index,obj)	将对象插入列表
list.pop([index=-1])	移除列表中的一个元素（默认为最后一个值），并返回他的值
list.remove(obj)	移除列表中某个值的第一个匹配项
list.reverse()	反向列表中的元素
list.sort(key=None,reverse=False)	对原列表进行排序
list.clear()	清空列表
list.copy()	复制列表

Tuple(元组)

- 与List的不同之处：元组的元素不能修改
- 元组使用 ( ) 不需要括号也可以
- 其方法及函数同list

String(字符串)

- 三引号，允许一个字符串跨多行

```
para_str = """这是一个多行字符串的实例
多行字符串可以使用制表符
TAB ( \t )
也可以使用换行符 [ \n ]
"""
```

方法	作用
find()	查找子串，查到则返回第一个字符的索引，否则返回
join()	合并字符串，其中合并的序列元素都必须是字符串[1]
split()	将字符串按照分隔符拆分为序列,默认在单个或多个连续的空白字符（空格、制表符、换行符等）处进行拆分[2]
strip()	将字符串开头和末尾的空白（但不包括中间的空白）删除[3]
lower()	返回字符串的小写部分

join()用法

```
number_list = ['1','2','3','4','5']
add = '+'
add.join(number_list)
```

输出：'1+2+3+4+5'

split()用法

```
'1+2+3+4+5'.split('+')
"I am a student from BUPT".split()
```

输出：['1', '2', '3', '4', '5'] ['I', 'am', 'a', 'student', 'from', 'BUPT']

strip()用法

```
"      I am a student from BUPT       ".strip()
'*****!!!!!Something important!!!!!*****'.strip('*!')
```

输出：'I am a student from BUPT' 'Something important'

Dict(字典)

- 字典的创建格式：dict={key1:value1, key2:value2}
- 键值必须唯一，若重复，后面的值会覆盖前面的值
- 字典无序

方法	作用
clear()	删除所有字典项
copy()	返回一个新字典，包含原有值
get()	使用get来访问不存在的键时，没有引发异常，而是返回None。而如果字典包含指定的键，get的作用将与普通字典查找相同。[4]
keys()	返回一个字典视图，其中包含指定字典中的键[5]
pop()	用于获取与指定键相关联的值，并将该键值对从字典中删除。
items()	方法items返回一个包含所有字典项的列表，其中每个元素都为(key, value)的形式。

get()用法

```
d = {}
print(d['name'])
KeyError: 'name'
print(d.get('name'))
None
d['name'] = 'Eric'
d.get('name')
'Eric'
```

keys()用法

```
x = {'username': '201820091', 'grades': [90,87,10]}
x.keys()
dict_keys(['username', 'grades'])
```

条件、循环语句

- if-elif-else
- if 语句的判断条件可以用 : >（大于）、<（小于）、== (等于)、>=（大于等于）、<=（小于等于）、!=（不等于）来表示其关系
- Python不支持switch语句

```
num = 9
if num < 0:
    print('Negative')
elif num > 0:
    if (num >= 0 and num <= 100) or (num >= 100 and num <=
150):
        print(num)
else:
    print('More than 150!')
```

- Python的基本循环语句是while循环和for循环

```
numbers = [12,23,34,45,56,67,78,89]
even = []
odd = []
while len(numbers) > 0:
    number = numbers.pop()
    if(number % 2) == 0:
        even.append(number)
    else:
        odd.append(number)
else:
```

```
    print(even)
    print(odd)
```

```
输出结果：
[78, 56, 34, 12]
[89, 67, 45, 23]
```

- for循环可以实现遍历任何序列的项目
- 还可以使用计数器来作为索引，此时往往会使用到内置len()获取列表的长度和range()返回元素的范围

```
fruits = ['bananan','apple','mango']
for index in range(len(fruits)):
    print("fruit "+str(index)+":", fruits[index])
else:
    print("Bye")
```

```
输出结果：
friut 0: bananan
friut 1: apple
friut 2: mango
Bye
```

- break语句用来终止循环语句
- continue 语句用来跳出本次循环
- 空语句，pass不做任何事情，占位语句

函数

- 函数代码块以 def 关键词开头，后接函数标识符名称和圆括号

> def 函数名（参数列表）：
> 函数执行语句段……
> return [表达式]

- Python 使用 lambda 来创建匿名函数,它的主体是一个表达式，而不是一个代码块

> lambda [arg1 [,arg2,.....argn]]:expression

```
sum = lambda arg1, arg2: arg1 + arg2

print("相加后的值为 : ", sum( 10, 20 ))
print("相加后的值为 : ", sum( 20, 20 ))
```

```
输出结果:
相加后的值为 :  30
相加后的值为 :  40
```

- 扩展

函数	作用
map()	对可迭代对象中的每一个元素应用一个函数，并返回一个新的可迭代对象[6]
filter()	过滤可迭代对象中的元素，返回一个新的可迭代对象，只包含使函数返回 `True` 的元素[7]
reduce()	对可迭代对象中的元素进行累计计算，返回一个单一的结果。需要导入 `functools` 模块[8]

map()用法

- 语法：map(function,iterable)
- 示例：

```
def square(x):
        return x*x

numbers = [1,2,3,4]
squares = map(square,numbers)
print(list(squares))
```

```
输出结果：
[1, 4, 9, 16]
```

filter()用法

- 语法：filter（function,iterable）
- 示例：

```
def is_even(x):
    return x % 2 == 0

numbers = [1, 2, 3, 4, 5, 6]
evens = filter(is_even, numbers)
print(list(evens))
```

```
输出结果：
[2, 4, 6]
```

reduce()用法

- 导包：from functools import reduce
- 语法：reduce(function,iterable[,initializer])

- 示例：

```
from functools import reduce

def add(x, y):
    return x + y

numbers = [1, 2, 3, 4]
total = reduce(add, numbers)
print(total)
```

输出结果：

```
10
```

文件

```
f = open('lines.txt','w')
string_list = ['Hello\n','World\n','Good\n','Morning\n ']
f.writelines(string_list)
f.close()
f = open('lines.txt','r')
lines = f.readlines()
lines
['Hello\n', 'World\n', 'Good\n', 'Morning\n']
f.close()
```

- 基本操作：

值	描 述
‘r’	读取模式（默认值）
‘w’	写入模式
‘x’	独占写入模式
‘a’	附加模式
‘b’	二进制模式（与其他模式结合使用）
‘t’	文本模式（默认值，与其他模式结合使用）
‘+’	读写模式（与其他模式结合使用）

综合实例

问题描述

- 能够制定商品条目
- 初始启动程序，让用户输入初始金额
- 用户可选择操作：
  0：退出 1：查看商品列表 2：加入购物车
  3：结算购物车 4：查看余额 5：清空购物车及购买历史d) 允许用户根据商品编号购买商品
- 用户选择结算购物车后检测余额是否足够，够就直接扣款，不够就提醒
- 用户可以一直购买商品，也可以直接退出
- 用文件保存购买历史、购物车历史以及商品列表

```
def initialize():
    try:
        f = open('shopping_cart.txt','r')
        a = f.read()
        global shopping_cart
        shopping_cart = eval(a)
        f.close()
        f = open('buy.txt','r')
        a = f.read()
        global buy
```

```
        buy = eval(a)
        f.close()
        f = open('products.txt','r')
        a = f.read()
        global products
        products = eval(a)
        f.close()
    except FileNotFoundError:
        pass
```

- 提示:
    - `shopping_cart`: 从 `shopping_cart.txt` 文件中读取购物车数据,并将其转换为 Python 对象(使用 `eval()` 函数)。
    - `buy`: 从 `buy.txt` 文件中读取购买记录数据,并将其转换为 Python 对象。
        - `products`: 从 `products.txt` 文件中读取产品数据,并将其转换为 Python 对象。

```
def show_item(content):
    print("########")
    if content == 1:
        print("序号{:<10s}商品名称{:<10s}价格{:<10s}".format("
","  ","  "))
        k = 0
        for i in products:
            print("{:<14d}{:<18s}{}".format(k,i,products[i]))
            k = k+1
    elif content == 3:
        print("购物车中有如下商品:")
        print("序号{:<10s}商品名称{:<10s}价格{:<10s}数量{:
<10s}".format("  ","  ","  ","  "))
        k = 0
        for i in shopping_cart:
 print("{:<14d}{:<18s}{:<14d}{}".format(k,i, products[i],
```

```
shopping_cart[i] ) )
            k = k+1
```

- 提示:
  - 如果 `content` 等于 1:
    - 打印一个分隔符 `#######`
    - 打印表头,包括"序号"、"商品名称"和"价格"
    - 遍历 `products` 字典,依次打印每个商品的序号、名称和价格
  - 如果 `content` 等于 3:
    - 打印一个分隔符 `#######`
    - 打印表头,包括"序号"、"商品名称"、"价格"和"数量"
    - 遍历 `shopping_cart` 字典,依次打印每个商品的序号、名称、价格和数量

```
def show_operation():
    print("########")
    print("您可进行如下操作（选择对应序号即可）")
    print("0 退出")
    print("1 查看商品列表")
    print("2 加入购物车")
    print("3 结算购物车")
    print("4 查看余额")
    print("5 清空购物车及购买历史")
    choice = input('您选择的操作是:')
    return choice
```

- 提示：暂无

```
def in_cart():
    show_item(1)
    print("您想加入购物车的是？")
    while True:
```

```
        choice = input('请输入所选择商品序号:')
        if choice.isdigit() :
            choice = int(choice)
            if 0<=choice<len(products) :
                break
            else:
                print("无该商品！ ")
        else:
            print("无该商品！ ")
    product = list(products)[choice]
    if product in shopping_cart:
        shopping_cart[product] +=1
    else:
        shopping_cart[product] = 1
    print("已帮您加入购物车")
```

- 提示:
  - 首先调用 `show_item(1)` 函数,显示所有可购买的商品信息。
  - 然后打印提示信息,让用户输入想要加入购物车的商品序号。
    - 进入一个 `while` 循环,直到用户输入一个有效的商品序号:
    - 检查用户输入是否是数字,如果不是,则打印错误信息并继续循环。
    - 检查用户输入的序号是否在 `products` 字典的索引范围内,如果不在,则打印错误信息并继续循环。
  - 一旦用户输入了一个有效的序号,就根据序号获取对应的商品名称。
  - 检查该商品是否已经在购物车中:
    - 如果在,则将该商品的数量加 1。
    - 如果不在,则将该商品添加到购物车,数量设为 1。
  - 最后打印提示信息,告诉用户该商品已经成功加入购物车。

```
def pay(money):
    show_item(3)
    list_pay = input("您想结算的商品是？ ")
```

```
    xlist = list_pay.split(",")
    xlist = [int(xlist[i]) for i in range(len(xlist)) if
0<=int(xlist[i])<len(shopping_cart)]
    c,s=np.unique(xlist,return_counts=True)
    total = 0
    pay_item = [list(shopping_cart)[c[i]] for i in
range(len(c))]
    for i in range(len(c)):
        total += products[pay_item[i]]*s[i]
    if total<money:
        for i in range(len(pay_item)):
            if pay_item[i] in buy:
                buy[pay_item[i]] +=s[i]
            else:
                buy[pay_item[i]] =1
            shopping_cart[pay_item[i]] -=1
            if shopping_cart[pay_item[i]] == 0:
                del shopping_cart[pay_item[i]]
        print("已经结算清！")
        return total
    else:
        print("余额不足！")
        return 0
```

- 提示：
  - 1首先调用 `show_item(3)` 函数,显示用户当前购物车中的商品信息。
  - 然后提示用户输入想要结算的商品序号,用户可以输入多个序号,用逗号分隔。
  - 将用户输入的序号转换为整数列表 `xlist`，并过滤掉无效的序号(超出购物车范围的序号)。
  - 使用 `np.unique()` 函数统计每个商品在 `xlist` 中出现的次数,得到商品序号 `c` 和对应的数量 `s`。

- 计算需要支付的总金额 `total`。遍历 `c` 和 `s`，根据商品价格和数量计算总金额。
- 如果总金额小于或等于用户输入的 `money` 金额:
  - 更新 `buy` 字典,记录用户的购买记录。
  - 更新 `shopping_cart` 字典,减少对应商品的数量,如果数量减为 0 则从购物车中删除该商品。
  - 打印结算成功的提示信息,并返回实际支付金额 `total`。
- 如果总金额大于用户输入的 `money` 金额:
  - 打印余额不足的提示信息,并返回 0。

```
def clean_history():
    global buy
    buy.clear()
    global shopping_cart
    shopping_cart.clear()
```

Numpy

方法	作用
a1.shape	一个表示各维度大小的元组
a2.dtype	一个用于说明数组数据类型的对象
a3.astype(np.type)	类型之间进行显式地转换
a4.T	转置

Pandas

方法	作用
Series()	索引在左边，值在右边[9]

方法	作用
DataFrame()	表格型的数据结构,可以被看作由Series组成的字典
obj.reindex()	创建一个适应新索引的新对象,索引值不存在，就引入确缺失值
obj.drop()	返回一个丢弃了指定行或列中的新对象

Series()用法

> obj = Series([2,5,7,-2])
> obj
> 0 2
> 1 5
> 2 7
> 3 -2
> dtype: int64

> obj2 = Series([2,7,-5,3], index = ['d','b','a','c'])
> obj2
> d 2
> b 7
> a -5
> c 3
> dtype: int64
> obj2.index
> Index(['d', 'b', 'a', 'c'], dtype='object')

+运算

- 在将对象相加时，如果存在不同的索引对，则结果的索引就是该索引对的并集

```
s1=Series([7.3,-2.5,3.4,1.5], index=['a','c','d','e'])
s2=Series([-2.1,3.6,-1.5,4,3.1], index=['a','c','e','f', 'g'])
s1+s2
a    5.2
c    1.1
d    NaN
e    0.0
f    NaN
g    NaN
dtype: float64
```

Scipy

- 峰度计算：峰度描述的是概率分布曲线的陡峭程度。可以利用stats.kurtosis()计算峰度。
- 正态分布程度检验：正态性检验同样返回两个值，第二个返回p-values。利用stats.normaltest()进行检验。
- 计算某一百分比处的数值：使用scoreatoercentile(数据集，百分比)计算在某一百分比位置的数值。

Matplotlib(绘图)

暂略