10-23

1.模块

1.模块的导入

模块的导入使用import关键字，配合from和as有以下几种形式：

- import 模块
- import 模块 as 别名
- from 模块 import 子模块
- from 模块 import *
- from 模块 import 子模块 as 别名


```
import numpy as np  #是Python的一种开源的科学计算库
from matplotlib import pyplot as plt #Matplotlib 是Python编程语言的 2D 绘图库，
#matplotlib.pyplot是Matplotlib库的子库，主要用于提供类似MATLAB的绘图接口，支持通过Python语言创建2D图表实现数据可视化。
```

```
#定义函数
def add(a,b):
    return a+b


def  sub(a,b):
    return a-b

def mul(a,b):
    return a*b

def div(a,b):
    return a/b

__all__=['add','sub','mul']
#定义变量
a=1
b=2

#自定义的类
class Stu():
    Sno=0 #类变量
    def __init__(self,name,age):  #构造方法:用来初始化对象
        self.name=name
        self.age=age
        Stu.Sno+=1   #类变量 类名.变量
```

```
    def show(self):   #成员方法
        print("姓名:"+self.name, "年龄"+str(self.age)+"学号"+str(Stu.Sno))


print(__name__)
print(type(__name__))


#函数调用

'''__name__'''
if __name__=="__main__":
    ret=add(4,6)
    print(ret)
    #类对象的创建
    p1=Stu('zhangsan',18)
    p1.show()
```

```
#moduleB.py
#导入的第一种方法
'''
import  moduleA
#模块.函数名()
#模块.类名()
#模块.变量名
ret=moduleA.add(4,5)
print(ret)
#<1>.用类的变量 称为对象
s1=moduleA.Stu('zhangsan',18)
#<2>.对象.成员名
s1.show()
'''


'''

import  moduleA as  mA
ret=mA.add(4,5)
print(ret)

from moduleA import a  #只导入变量
print(a)

from moduleA import add  #只导入函数
'''
'''
import builtins
import moduleA
print(dir())


from moduleA import *
ret=add(1,2)
```

```
print(ret)

ret1=sub(1,2)
print(ret1)

ret2=mul(1,2)
print(ret2)
'''

#import moduleA
print(dir())
from moduleA import *
ret1=add(15,18)
print(ret1)
ret2=sub(15,18)
print(ret2)
ret3=mul(15,18)
print(ret3)

ret4=div(15,18)  #报错
print(ret4)
```

2.模块的分类

1.内置模块

2.第三方模块

3.自定义模块

3.模块的内置变量

模块内置变量，可通过dir()查看模块的内置变量：
- __name__：用于确定模块是被直接运行还是被导入到其他模块中。当一个模块被直接运行时，__name__的值是”__main__”，否则为模块的名称。
- __doc__：包含模块的说明性文档。
- __file__：包含模块的文件路径。
- __all__：定义一个模块中的哪些变量、函数或类可以通过from module import *导入时可以用。
- __package__：包含模块所在的包的名称。
- __dict__：包含模块的全局命名空间。

```
__main__:
__all__:
```

2.包

1.os

https://www.runoob.com/python/os-file-methods.html

2.random

https://blog.csdn.net/qq_41009483/article/details/142493119

3.math

https://blog.csdn.net/weixin_44440552/article/details/88431156

4.time (今天必须掌握)

https://www.runoob.com/python/python-date-time.html

```
import  time
start=time.time()
#time.sleep(5)
stop=time.time()
second=stop-start
print(second)
localtime=time.localtime(time.time())
print(localtime)
# 格式化成2016-03-20 11:45:39形式
print(time.strftime("%Y-%m-%d %H:%M:%S", time.localtime()))
```

```
python中时间日期格式化符号：

%y 两位数的年份表示（00-99）
%Y 四位数的年份表示（000-9999）
%m 月份（01-12）
%d 月内中的一天（0-31）
%H 24小时制小时数（0-23）
%I 12小时制小时数（01-12）
%M 分钟数（00-59）
%S 秒（00-59）
%a 本地简化星期名称
%A 本地完整星期名称
%b 本地简化的月份名称
%B 本地完整的月份名称
%c 本地相应的日期表示和时间表示
%j 年内的一天（001-366）
%p 本地A.M.或P.M.的等价符
%U 一年中的星期数（00-53）星期天为星期的开始
%w 星期（0-6），星期天为星期的开始
%W 一年中的星期数（00-53）星期一为星期的开始
%x 本地相应的日期表示
%X 本地相应的时间表示
%Z 当前时区的名称
%% %号本身
```

```
'''
```

```
import test.moduleA  as mA
s1=mA.Stu('zhangsan',19)
s1.show()
'''
#os模块
import  os # 是用来进行办公的，做一些表，一些文件夹之类的 (先将笔记上的写会--->自主学习其他函数)
#使用os模块创建一个文件夹

#定义一个变量,用来代表要创建的文件夹的名字
folder_name='new_folder'
#用来检查你要的文件夹是否存在
if  not os.path.exists(folder_name):
    #当文件夹不存在时,在if分支总创建文件夹
    os.mkdir(folder_name)
else:
    print('Folder already exists')
print("当前目录:",os.getcwd())

dirs = os.listdir( folder_name )
print("dirs:",dirs)


for dir in dirs:
    print(dir)



#sys模块  获取系统相关信息 (了解)
import sys
#使用sys模块打印python解释器版本
print(f'该文件使用的python解释器版本为:',sys.version)

#终止程序
#sys.exit(0)  #进程终止函数

#math模块
import math
print(math.pi)
a=math.pi
print(math.sin(a))

print("2的10次方",math.ceil(math.pow(2,10)))#2的10次方




#time模块(必须掌握)
import  time
start=time.time() #1970到现在的秒数
print("start:",start)
time.sleep(5)  #让程序暂停5秒
stop=time.time()
second=stop-start
print("second:",second)
localtime=time.localtime(time.time())  #获得当地时间
print("localtime:",localtime)
```

```
#time()
#localtime(time())
#strftime(localtime())

# 格式化成2016-03-20 11:45:39形式
print(time.strftime("%Y-%m-%d %H:%M:%S", time.localtime()))

# 格式化成Sat Mar 28 22:24:24 2016形式
print(time.strftime("%a %b %d %H:%M:%S %Y", time.localtime()))

# 将格式字符串转换为时间戳
a = "Sat Mar 28 22:24:24 2016"
print(time.mktime(time.strptime(a, "%a %b %d %H:%M:%S %Y")))


#随机数
import random   #先将笔记上的掌握，自主学习其他函数
print("随机的整数:",random.randint(1,10))

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

random.shuffle(mylist) #打乱列表

print(mylist)
```

10-24

3.异常(建议学生加班看)

1.异常语法

```
try:
   #有可能发生异常的代码
except #某个异常:
   #异常发生后要执行的代码
else:
    #如果没有异常发生,要执行的代码
finally:
    #不管有没有捕获到异常,最后都会执行这里的代码
```

```
try:
    a=1
    print(b)
except(NameError,SyntaxError):
    print("该程序有问题")
else:
    print("没有捕获到异常")
finally:
    print("我最后执行")
```

```
try:
    a=1
    print(b)
    class="Person"
except SyntaxError:
    print("语法有问题")
except NameError:
    print("没有这个变量")
else:
    print("没有捕获到异常")
finally:
    print("我最后执行")
```

```
try:
    a=1
    print(b)
except Exception as e:
    print("有异常",e.args)
else:
    print("未捕获到异常")
finally:
    print("最后执行")
```

作业

<1>.每隔59秒打印一下系统时间

10-27 (文件相关)

1.读取文件

```
#打开文件
f=open('1.txt','r')

#<1>.按字节数读取
#readstr1=f.read(20)
#print(readstr1)

#统计该文件有多少行
'''
line=0
#<2>.按行读取
while True:
    readStr2=f.readline()
    if not readStr2:
        break
    print(readStr2)
    line+=1

print(f"该文件有{line}行")
'''
readstr3=f.readlines() #读取文件的所有内容
print(readstr3)
#关闭文件
```

```
f.close()
```

2.写入文件

```
#打开方式
#“r"  以只读方式打开文件,文件不存在,则出错
#“r+” 以可读可写方式打开文件,文件不存在,则出错
#“w”  以只写方式打开文件,文件不存在，则新建,文件存在，则清空文件中的内容
#"w+" 以可读可写方式打开文件,文件不存在，则新建,文件存在，则清空文件中的内容
#"a"  以追加方式打开文件,文件不存在,则新建,文件存在,则追加到位文件的末尾
#"a+" 以可读可写方式打开文件,文件不存在,则新建,文件存在,则追加到位文件的末尾
#"x" 文件存在,则出错
#"b" 二进制  照片 视频，歌曲
#"rb"
#"t" 文本文档

#<1>.打开文件
#<1>.文件名(可包含路径)  <2>.打开方式  <3>.编码格式
#f=open("1.txt","r")
f=open("1.txt","a")

#<3>.写入文件
f.write("hellonanjing\r\n")

#<2>.关闭文件
f.close()
```

3.文件其他函数

```
f=open("1.txt","r")
f.seek(0,2)      #移动文件光标的位置  #0 从文件开头移动  1 从文件当前位置移动  2.从文件末尾移动
filesize=f.tell() #tell()获得文件光标的位置
print("filesize=",filesize)


#<1>.f=open("文件名", "打开方式",encoding="utf-8")
#<2>.关闭文件 f.close()
#<3>.f.read()  f.readline()  f.readlines()
#<4>.f.wirte()
#<5>.其他的两个函数 f.tell()  f.seek()
```

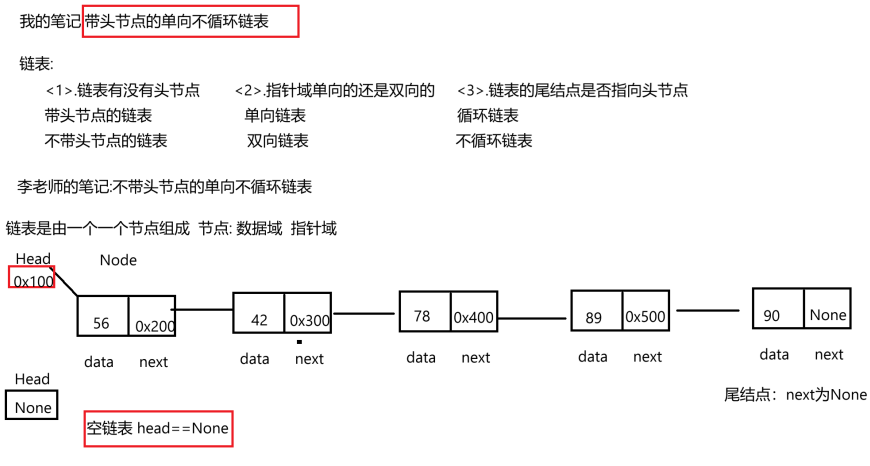
4.每个5秒将行号，系统时间写入文件

```
import  time
# 为了获取该文件有多少行  写一个子函数,实现文件的行数
def get_line_count(file_path):
    '''统计文件行数'''
    count = 0
    with open(file_path, "r") as f:
        while True:
            line = f.readline()
            if not line:
```

```
                break
            count += 1
    return count


file_path = "time.log"
while True:

    # 获取当前文件已有的行数
    line_num = get_line_count(file_path) + 1

    # 获取当前系统时间
    time_now = time.strftime("%Y-%m-%d %H:%M:%S", time.localtime())
    try:
        # 以追加方式写入文件
        with open(file_path, "a+", encoding="utf-8") as f:
            f.write(f"{line_num},{time_now}\n")

        print(f"写入第{line_num}行：{time_now}")
        time.sleep(5)

    except KeyboardInterrupt:
        print("\n手动中断程序，文件已保存。")
        break
    except FileNotFoundError:
        with open(file_path, "w", encoding="utf-8") as f:
            pass
    except Exception as e:
        print(f"错误：{e}")

        break
```

10-28

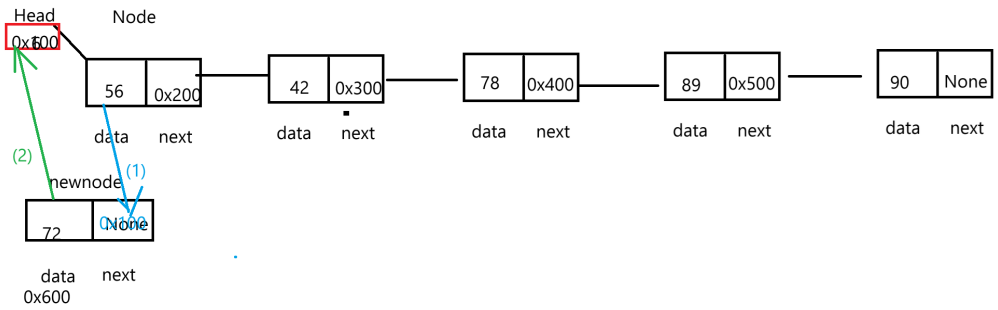
1.链表基本概念



```
#定义一个节点的类
class Node:
    def __init__(self, data):
        self.data = data    #数据域
        self.next = None    #指针域
```

2.链表插入

1.头插法

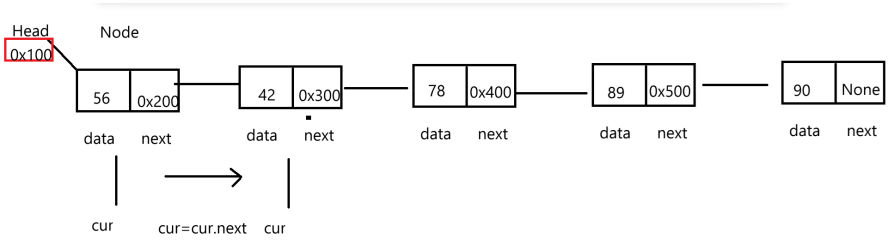


头插法:

newnode.next=self.__head (1)

self.__head= newnode (2)

3.遍历链表



遍历:

<1>.定义一个临时变量=第一个节点

cur=self.__head

<2>.在cur !=None 输出数据域

while cur!=None:

print(cur.data)

cur=cur.next

```
#定义一个节点的类
class Node:
    def __init__(self, data):
        self.data = data    #数据域
        self.next = None    #指针域


#定义一个链表类
class LinkedList:
    def __init__(self,node=None):
        self.__head = node

    #判断该链表是否为空链表
```

```
    def is_empty(self):
        return self.__head==None

    #判断该链表的元素个数
    #插入:
    #<1>.头插法
    def add(self,data):
        #<1>.创建新节点
        newnode=Node(data)
        #<2>.使用头插法
        #<2.1>.保护好头节点后的所有
        newnode.next = self.__head
        #<2.2>.更新头节点
        self.__head = newnode

    #<2>.尾插法
    def append(self,data):
        #<1>.创建新节点
        newnode=Node(data)
        #<1---1>.特殊处理，如果是空链表，直接将newnode赋值给self.__head
        if self.is_empty():
            self.__head = newnode

        #<2>.找到尾结点:next==None
        cur=self.__head
        while cur.next!=None:
            cur=cur.next

        #<3>.将新节点插入到尾结点之后
        cur.next=newnode
    #<3>.在任意位置插入     pos>length-1:尾插法   pos<0:头插法   else:在中间插入

    #链表的遍历
    def travel(self):
        #<1>.定义一个临时变量初始化第一个节点
        cur=self.__head
        #<2>.在cur!=None的情况下,输出数据域,循环移动cur
        while cur!=None:
            print(cur.data," ")
            cur=cur.next  #移动到下一个结点的指针
        print()

if __name__ == '__main__':
    linked_list=LinkedList()
    linked_list.add(10)  #头插法
    linked_list.add(20)
    linked_list.add(30)
    linked_list.add(40)
    linked_list.travel()
```