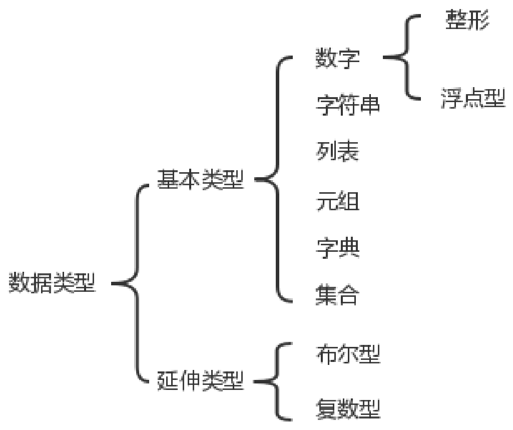
金融软件工程 第 7 讲作业

程洁帆 161220025

所选语言：python

一、 基本成分

1、 数据成分



2、 运算成分

算术运算符：+， -， *， /， //， %

关系运算符：==, !=, >, <, >=, <=

逻辑运算符：and, not, or

位运算符：&, ^, ~, |, >>, <<

赋值运算符：=, +=, -=, *=, /=, %=, //=, <<=, >>=, &=, |=, ^=, ~=

3、 控制成份

顺序结构

条件选择结构(if…elif…else…)

重复结构(for, while)

4、 传输成分

print ( )函数用于向标准输出设备(屏幕)写数据

input ( ) 函数用于从标准输入设备(键盘)上读数据

write()函数用于文件写入

read()函数用于文件读取

二、 语言特性

1、 语言设计特性

动态类型语言，无需类型声明，因而简单、易扩展。

面向过程，代码结构化，可复用。

可以自动管理内存，自动进行垃圾回收。

可嵌入或扩展支持 C 语言，适合混合语言开发。

2、 工程特性

无需编译及链接，便于快速开发。

类、模块、异常等特性，可用于大型项目的开发。

可移植性，因为 Python 是用 C 写的，又由于 C 的可移植性，使得 Python 可以运行在任何带有 ANSI C 编译器的平台上。

易维护性，因为 Python 本身就易于学习和开发。

可拓展性。

3、 应用特性

python 在网络爬虫、人工智能、科学计算、大数据、云计算、web 开发、金融领域等多有应用。

三、records 库（SQL for Humans™）注释分析

1、 简单介绍

records 是一个非常简单但功能强大的库，可用于对大多数关系数据库进行原始 SQL 查询。它的工作流程较为简单，界面也相对优雅。

2、 项目注释

程序名	来源	代码行	程序语言	序言性注释	功能性注释
records.py	kennethreitz/records	520	python	1. Given an object, return a boolean indicating whether it is an instance or subclass of :py:class:`Exception`. 2. A row, from a query, from a database. 3. Returns the list of column names from the query. 4. Returns the list of values from the query. 5. Returns the value for a given key, or default. 6. Returns the row as a dictionary, as ordered. 7. A Tablib Dataset containing the row. 8. Exports the row to the given format. 9. A set of excellent Records from a query. 10. Iterate over all rows, consuming the underlying generator only when	1. Ensure that lengths match properly. 2. Support for index-based lookup. 3. Support for string-based lookup. 4. Merge standard attrs with generated ones (from column names). 5. Other code may have iterated between yields, so always check the cache. 6. Throws StopIteration when done. Prevent StopIteration bubbling from generator, following https://www.python.org/dev/peps/pep-0479/ 7. Convert RecordCollection[1] into slice.

				necessary. 11. A Tablib Dataset representation of the RecordCollection. 12. Returns a list of all rows for the RecordCollection. If they haven't been fetched yet, consume the iterator and cache the results. 13. Returns a single record for the RecordCollection, or \`default\`. If \`default\` is an instance or subclass of Exception, then raise it instead of returning it. 14. Returns a single record for the RecordCollection, ensuring that it is the only record, or returns \`default\`. If \`default\` is an instance or subclass of Exception, then raise it instead of returning it. 15. Returns the first column of the first row, or \`default\`. 16. A Database. Encapsulates a url and an SQLAlchemy engine with a pool of connections. 17. Closes the Database. 18. Returns a list of table names for the connected database. 19. Get a connection to this Database. Connections are retrieved from a pool. 20. Executes the given SQL query against the Database. Parameters can, optionally, be provided. Returns a RecordCollection, which can be iterated over to get result rows as dictionaries. 21. Bulk insert or update. 22. Like Database.query, but takes a filename to load a	8. Export the RecordCollection to a given format (courtesy of Tablib). 9. If the RecordCollection is empty, just return the empty set, Check number of rows by typecasting to list 10. Set the column names as headers on Tablib Dataset. 11. By calling list it calls the __iter__ method 12. Try to get a record, or return/raise default. 13. Cast and return. 14. Try to get a record, or return/raise default. 15. Ensure that we don't have more than one row. 16. Cast and return. 17. If no db_url was provided, fallback to $DATABASE_URL. 18. Create an engine. 19. Setup SQLAlchemy for Database inspection. 20. Execute the given query. 21. Row-by-row Record generator. 22. Convert psycopg2 results to RecordCollection. 23. Fetch all results if desired. 24. If path doesn't exists 25. If it's a directory 26. Read the given .sql file into memory. 27. Defer processing to

				query from. 23. Like Database.bulk_query, but takes a filename to load a query from. 24. A context manager for executing a transaction on this Database. 25. A Database connection. 26. Executes the given SQL query against the connected Database. Parameters can, optionally, be provided. Returns a RecordCollection, which can be iterated over to get result rows as dictionaries. 27. Bulk insert or update. 28. Like Connection.query, but takes a filename to load a query from. 29. Like Connection.bulk_query, but takes a filename to load a query from. 30. Returns a transaction object. Call ``commit`` or ``rollback`` on the returned object as appropriate. 31. Receives a row, converts datetimes to strings.	self.query method. 28. If path doesn't exists 29. If it's a directory 30. Read the given .sql file into memory. 31. Parse the command-line arguments. 32. Create the Database. 33. Can't send an empty list if params aren't expected. 34. Execute the query, if it is a found file. 35. Execute the query, if it appears to be a query string. 36. Otherwise, say the file wasn't found. 37. Print results in desired format. 38. Run the CLI when executed directly.
setup.py	kennethreitz/records	95	python	1. Support setup.py publish. 2. Prints things in bold.	1. TODO: Add the rest.

3、 注释分析

1） 该项目以不同的注释方式直接区分了序言性注释（用三个单引号注释）和功能性注释（用井号注释），较为清晰。

2） 该项目明确地说明了函数的返回值，十分清晰。

3） 该项目对于边界条件的判断都给出了注释说明，清晰而又增加了程序的稳定性。

4） 该项目会在注释中说明两个函数间的关系。

四、代码修改

1、 修改前

由于原代码逻辑较为简单，仅考虑在函数前说明每个函数的作用，并对边界条件的判断

加以说明。由于代码较长，尽截取部分 CheckPage 类做以说明。

```python
#查账页面
class CheckPage(Toplevel):
    def __init__(self):
        super().__init__()
        self.title('查账')
        self.geometry('800x400')
        self.CheckUI()

    def CheckUI(self):
        frame = Frame(self)
        frame.pack()
        self.allbutton = Button(frame, text="显示全部", command=self.ShowAll)
        self.allbutton.grid(row = 0, column = 2)
        self.itembutton = Button(frame, text="按类别查询", command=self.ShowItem)
        self.itembutton.grid(row = 0, column = 1)
        self.datebutton = Button(frame, text="按日期查询", command=self.ShowDate)
        self.datebutton.grid(row = 0, column = 0)
        self.stabutton = Button(frame, text="统计", command=self.Statistics)
        self.stabutton.grid(row = 1, column = 0)
        self.revisebutton = Button(frame, text="修改", command=self.Revise)
        self.revisebutton.grid(row = 1, column = 1)
        self.deletebutton = Button(frame, text="删除", command=self.Delete)
        self.deletebutton.grid(row = 1, column = 2)
        self.writebutton = Button(frame, text='导出信息', command=self.Write)
        self.writebutton.grid(row = 2, column = 0)
        self.cancelbutton = Button(frame, text='取消', command=self.cancel)
        self.cancelbutton.grid(row = 2, column = 2)

        frame2 = Frame(self)
        self.tree = ttk.Treeview(frame2, show="headings", height=18)
        self.vbar = ttk.Scrollbar(frame2, orient=VERTICAL, command=self.tree.yview)
        self.tree.configure(yscrollcommand=self.vbar.set)
        self.tree['columns'] = ('编号','年','月','日','类别','金额')
        self.tree.column('编号', width=60, anchor="center")
        self.tree.column('年', width=60, anchor="center")
        self.tree.column('月', width=60, anchor="center")
        self.tree.column('日', width=60, anchor="center")
        self.tree.column('类别', width=60, anchor="center")
        self.tree.column('金额', width=60, anchor="center")
        self.tree.heading('编号', text='编号')
        self.tree.heading('年', text='年')
        self.tree.heading('月', text='月')
        self.tree.heading('日', text='日')
        self.tree.heading('类别', text='类别')
        self.tree.heading('金额', text='金额')
        self.tree.pack()
        frame2.pack()

    def Show(self, type, chosen=None):
        conn = sqlite3.connect('account.db')
        cur = conn.cursor()
        cur.execute('create table if not exists account (id integer primary key, date 
integer, year integer, month integer, day integer, item varchar(20), money double)')
        data = []
        if type == 0:
            data = cur.execute('select id,year,month,day,item,money from account where item 
= ?', (chosen,)).fetchall()
        elif type == 1:
            data = cur.execute("select id,year,month,day,item,money from 
account").fetchall()
        elif type == 2:
            data = cur.execute("select id,year,month,day,item,money from account where date 
between ? and ?", (chosen[0], chosen[1])).fetchall()
        if data == []:
            messagebox.showerror('错误', '无账单信息。')
            return
        for _ in map(self.tree.delete, self.tree.get_children("")):
            pass
        for line in data:
            self.tree.insert('', 'end', values=line)
        cur.close()
        conn.close()

    def ShowAll(self):
        self.Show(1)

    def ShowItem(self):
        res = ShowItemPage()
        self.wait_window(res)
        if res.info is None:
            return
        self.Show(0, res.info)

    def ShowDate(self):
        res = ShowDatePage()
        self.wait_window(res)
        #判断日期合法性
        if res.bdate == None:
            return
        elif res.bdate > res.edate:
            messagebox.showerror('错误', '请输入正确的日期！')
            ShowDatePage()
        else:
            self.Show(2, [res.bdate, res.edate])

    def Statistics(self):
        #略

    def Revise(self):
        res = RevisePage()
        self.wait_window(res)
        if res.info == None:
            return
        conn = sqlite3.connect('account.db')
        cur = conn.cursor()
        cur.execute('create table if not exists account (id integer primary key, date
integer, year integer, month integer, day integer, item varchar(20), money double)')
        cur.execute('update account set date=?,year=?,month=?,day=?,item=?,money=? where
id=?',
(res.info[1],res.info[2],res.info[3],res.info[4],res.info[5],res.info[6],res.info[0]))
        conn.commit()
        cur.close()
        conn.close()

    def Delete(self):
        res = DeletePage()
        self.wait_window(res)
        if res.info == None:
            return
        conn = sqlite3.connect('account.db')
        cur = conn.cursor()
        cur.execute('create table if not exists account (id integer primary key, date
integer, year integer, month integer, day integer, item varchar(20), money double)')
        cur.execute('delete from account where id=?',(res.info,))
        conn.commit()
        cur.close()
        conn.close()

    def Write(self):
        res = WritePage()
        self.wait_window(res)
        if res.info == None:
            return
        conn = sqlite3.connect('account.db')
        cur = conn.cursor()
        cur.execute('create table if not exists account (id integer primary key, date
integer, year integer, month integer, day integer, item varchar(20), money double)')
        datalist = cur.execute("select year,month,day,item,money from account").fetchall()
        if datalist == []:
            messagebox.showerror('错误','无账单信息。')
            return
        cur.close()
        conn.close()
        df = pd.DataFrame(data=datalist, columns=['year','month','day','item','money'])
        filename = res.info[0]+res.info[1]
        if res.info[1]=='.csv':
            df.to_csv(filename, index=False)
        if res.info[1]=='.txt':
            df.to_csv(filename, index=False)


    def cancel(self):
        self.destroy()
```

2、 修改后

```
'''查账页面'''
class CheckPage(Toplevel):
    def __init__(self):
        super().__init__()
        self.title('查账')
        self.geometry('800x400')
        self.CheckUI()

    '''主视图设计'''
    def CheckUI(self):
        frame = Frame(self)
        frame.pack()
        #按钮
        self.allbutton = Button(frame, text="显示全部", command=self.ShowAll)
        self.allbutton.grid(row = 0, column = 2)
        self.itembutton = Button(frame, text="按类别查询", command=self.ShowItem)
        self.itembutton.grid(row = 0, column = 1)
        self.datebutton = Button(frame, text="按日期查询", command=self.ShowDate)
        self.datebutton.grid(row = 0, column = 0)
        self.stabutton = Button(frame, text="统计", command=self.Statistics)
        self.stabutton.grid(row = 1, column = 0)
        self.revisebutton = Button(frame, text="修改", command=self.Revise)
        self.revisebutton.grid(row = 1, column = 1)
        self.deletebutton = Button(frame, text="删除", command=self.Delete)
        self.deletebutton.grid(row = 1, column = 2)
        self.writebutton = Button(frame, text='导出信息', command=self.Write)
        self.writebutton.grid(row = 2, column = 0)
        self.cancelbutton = Button(frame, text='取消', command=self.cancel)
        self.cancelbutton.grid(row = 2, column = 2)

        #表格绘制
        frame2 = Frame(self)
        self.tree = ttk.Treeview(frame2, show="headings", height=18)
        self.vbar = ttk.Scrollbar(frame2, orient=VERTICAL, command=self.tree.yview)
        self.tree.configure(yscrollcommand=self.vbar.set)
        self.tree['columns'] = ('编号','年','月','日','类别','金额')
        self.tree.column('编号', width=60, anchor="center")
        self.tree.column('年', width=60, anchor="center")
        self.tree.column('月', width=60, anchor="center")
        self.tree.column('日', width=60, anchor="center")
        self.tree.column('类别', width=60, anchor="center")
        self.tree.column('金额', width=60, anchor="center")
        self.tree.heading('编号', text='编号')
        self.tree.heading('年', text='年')
        self.tree.heading('月', text='月')
        self.tree.heading('日', text='日')
        self.tree.heading('类别', text='类别')
        self.tree.heading('金额', text='金额')
        self.tree.pack()
        frame2.pack()

    '''根据实际需求进行表格数据的填充'''
    def Show(self, type, chosen=None):
        conn = sqlite3.connect('account.db')
        cur = conn.cursor()
        cur.execute('create table if not exists account (id integer primary key, date 
integer, year integer, month integer, day integer, item varchar(20), money double)')
        data = []
        #按类别显示数据
        if type == 0:
            data = cur.execute('select id,year,month,day,item,money from account where item 
= ?', (chosen,)).fetchall()
        #显示全部数据
        elif type == 1:
            data = cur.execute("select id,year,month,day,item,money from
account").fetchall()
        #显示某一时间段内数据
        elif type == 2:
            data = cur.execute("select id,year,month,day,item,money from account where date
between ? and ?", (chosen[0], chosen[1])).fetchall()
        #无数据
        if data == []:
            messagebox.showerror('错误', '无账单信息。')
            return
        #数据显示
        for _ in map(self.tree.delete, self.tree.get_children("")):
            pass
        for line in data:
            self.tree.insert('', 'end', values=line)
        cur.close()
        conn.close()

    '''显示全部'''
    def ShowAll(self):
        self.Show(1)

    '''按类别显示'''
    def ShowItem(self):
        #读取类别信息，信息为空则返回
        res = ShowItemPage()
        self.wait_window(res)
        if res.info is None:
            return
        self.Show(0, res.info)

    '''按时间显示'''
    def ShowDate(self):
        res = ShowDatePage()
        self.wait_window(res)
        #判断日期合法性，日期信息为空则返回，错误则报错
        if res.bdate == None:
            return
        elif res.bdate > res.edate:
            messagebox.showerror('错误', '请输入正确的日期！')
            ShowDatePage()
        else:
            self.Show(2, [res.bdate, res.edate])
    '''统计'''
    def Statistics(self):
        #略

    '''修改'''
    def Revise(self):
        #读取修改信息，若无信息则返回
        res = RevisePage()
        self.wait_window(res)
        if res.info == None:
            return
        #数据修改
        conn = sqlite3.connect('account.db')
        cur = conn.cursor()
        cur.execute('create table if not exists account (id integer primary key, date 
integer, year integer, month integer, day integer, item varchar(20), money double)')
        cur.execute('update account set date=?,year=?,month=?,day=?,item=?,money=? where 
id=?',
(res.info[1],res.info[2],res.info[3],res.info[4],res.info[5],res.info[6],res.info[0]))
        conn.commit()
        cur.close()
        conn.close()

    '''删除'''
    def Delete(self):
        #读取删除信息，若无信息则返回
        res = DeletePage()
        self.wait_window(res)
        if res.info == None:
            return
        #数据删除
        conn = sqlite3.connect('account.db')
        cur = conn.cursor()
        cur.execute('create table if not exists account (id integer primary key, date 
integer, year integer, month integer, day integer, item varchar(20), money double)')
        cur.execute('delete from account where id=?',(res.info,))
        conn.commit()
        cur.close()
        conn.close()

    '''导出数据'''
    def Write(self):
        #读取导出文件名，若为空则返回
        res = WritePage()
        self.wait_window(res)
        if res.info == None:
            return
        #读取所有数据，若为空则报错
        conn = sqlite3.connect('account.db')
        cur = conn.cursor()
        cur.execute('create table if not exists account (id integer primary key, date 
integer, year integer, month integer, day integer, item varchar(20), money double)')
        datalist = cur.execute("select year,month,day,item,money from account").fetchall()
        if datalist == []:
            messagebox.showerror('错误','无账单信息。')
            return
        cur.close()
        conn.close()
        #导出到文件
        df = pd.DataFrame(data=datalist, columns=['year','month','day','item','money'])
        filename = res.info[0]+res.info[1]
        if res.info[1]=='.csv':
            df.to_csv(filename, index=False)
        if res.info[1]=='.txt':
            df.to_csv(filename, index=False)

    '''取消，返回'''
    def cancel(self):
        self.destroy()
```