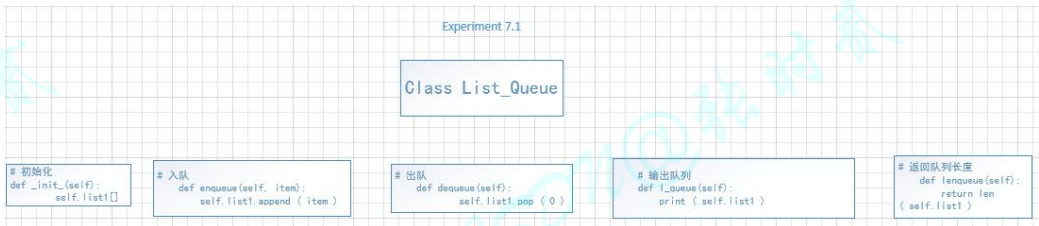


实验环境	严禁抄袭 仅供参考 Blog:zhangshier.vip
一、实验题目 实验七：异常处理和单元测试	
二、实验目的 <ol style="list-style-type: none"> 了解异常的基本概念和常用异常类。 掌握异常处理的格式、处理方法。 掌握断言语句的作用和使用方法。 了解单元测试的基本概念和作用。 掌握在 Python 中使用测试模块进行单元测试的方法和步骤。 	
三、实验内容 <ol style="list-style-type: none"> 编程实现如下功能： <ol style="list-style-type: none"> 定义一个利用列表实现队列的类 List_Queue，可以实现队列元素进入、删除、求队列长度等功能。 定义个异常处理类 List_Queue_Exception 对类 List_Queue 中可能出现的异常进行处理。 编程实现如下功能： <ol style="list-style-type: none"> 定义一个实现算术运算的类 Arithmetic_Operation，可以实现两个整数的加法、减法、乘法和除法运算。 定义一个测试类 Test_Arithmetic_Operation 对 Arithmetic_Operation 中的功能进行测试。 	
四、实验步骤 <ol style="list-style-type: none"> 题目一 <ol style="list-style-type: none"> 问题分析（含解决思路、使用的数据结构、程序流程图等） <p>队列（queue）是只允许在一端进行插入操作，而在另一端进行删除操作的线性表。一种先进先出（First In First Out）的线性表，简称 FIFO。允许插入的一端称为队尾，允许删除的一端称为队头。</p> <p>通过设计 List_Queue 类，利用集合保存队列数据，增 list.append()，删 list.pop()。设计 List_Queue_Exception 类，当队列为空且执行出队操作时，对异常处理类设计图</p>  算法/代码描述（基本要求源代码） <pre>class List_Queue: # 初始化</pre> 	

```

def __init__(self):
    self.list1 = [ ]
    print ( '初始化成功!' )

# 入队
def enqueue(self, item):
    self.list1.append ( item )
    print ( '添加成功!' )

# 出队
def dequeue(self):
    if len ( self.list1 ) > 0:
        print ( "出队列数据:", self.list1[0] )
        self.list1.pop ( 0 )
    else:
        raise List_Queue_Exception ()

# 返回队列长度
def lenqueue(self):
    return len ( self.list1 )

# 输出队列
def l_queue(self):
    print ( self.list1 )

class List_Queue_Exception ( BaseException ):
    def __init__(self):
        print ( "列表为空!" )

if __name__ == '__main__':
    list_queue = List_Queue ()
    print ( "-----" )
    print ( "*****1,入队*****" )
    print ( "*****2,出队*****" )
    print ( "*****3,队列长度*" )
    print ( "*****4,显示列表*" )
    print ( "*****0,退出*" )
    print ( "-----" )
    while True:

        x = int ( input ( "输入序号:" ) )
        try:

```

```

if x in [ 0, 1, 2, 3, 4 ]:

    if x == 0:
        print ( "已经退出" )
        break;
    elif x == 1:
        y = input ( "请输入输入的数据:" )
        list_queue.enqueue ( y )
    elif x == 2:
        list_queue.dequeue ()
    elif x == 3:
        print ( "队列长度为:", list_queue.lenqueue () )
    elif x == 4:
        list_queue.l_queue ()

else:
    print ( "输入有误!" )
except BaseException as ex:
    print ( ex )

```

(3) 运行结果（含执行结果验证、输出显示信息）

```

D:\Python3.9.10\python.exe "D:/太工/大三第二学期/Python/192062116 张帆/Experiment 7.1.py"
初始化成功!
-----
*****1,入队*****
*****2,出队*****
*****3,队列长度*
*****4,显示列表*
*****0,退出*
-----
输入序号:1
请输入输入的数据:1
添加成功!
输入序号:1
请输入输入的数据:2
添加成功!
输入序号:1
请输入输入的数据:3
添加成功!
输入序号:4
['1', '2', '3']
输入序号:3
队列长度为: 3
输入序号:2
出队列数据: 1
输入序号:4
['2', '3']
输入序号:0
已经退出

```

当队列为空，执行出队操作的异常处理

```

D:\Python3.9.10\python.exe "D:/太工/大三第二学期/Python/192062116 张帆/E
初始化成功!
-----
*****1,入队*****
*****2,出队*****
*****3,队列长度*
*****4,显示列表*
*****0,退出*
-----
输入序号:2
列表为空!

输入序号:0
已经退出

```

2、题目二

(1) 问题分析（含解决思路、使用的数据结构、程序流程图等）

在 `Arithmetic_Operation` 类中分别设计构造函数、`add()`、`sub()`、`mul()`、`div()`，在 `Test_Arithmetic_Operation` 类中利用单元测试框架 `unittest` 对 `Arithmetic_Operation` 类测试。

(2) 算法/代码描述（基本要求源代码）

```

import unittest

class Arithmetic_Operation:
    def add(self):
        return self.x + self.y

    def sub(self):
        return self.x - self.y

    def mul(self):
        return self.x*self.y

    def div(self):
        return self.x/self.y

    def __init__(self, x, y):
        self.x = x
        self.y = y

```

```

class Test_Arithmetic_Operation ( unittest.TestCase ):
    def setUp(self):
        self.op = Arithmetic_Operation ( 3, 5 )

    def test_add(self):
        if self.assertEqual ( self.op.add (), 8 ):
            print ( "正确" )

    def test_sub(self):
        self.assertEqual ( self.op.sub (), -2 )

    def test_mul(self):
        self.assertEqual ( self.op.mul (), 15 )

    def test_div(self):
        self.assertEqual ( self.op.div (), 0.6 )

if __name__ == '__main__':
    unittest.main ()

```

(3) 运行结果（含执行结果验证、输出显示信息）

```

Testing started at 9:41 ...
D:\Python3.9.10\python.exe "D:\Pycharm Professional 2020(64bit)\PyCharm 2020.1\plugins\python\he
Launching unittests with arguments python -m unittest D:/太工/大三第二学期/Python/192062116 张帆/Exp
|
Ran 4 tests in 0.003s

OK

进程已结束，退出代码 0

```

五、 出现的问题及解决的方法

在利用 unittest 框架时，出现报错：“No module name ‘Experiment 7’，文件名此时是‘Experiment 7.2’，在使用测试框架时，文件名不能包含特殊符号，改为‘Experiment 7’即可

Tests failed: 1 of 1 test - 2 ms

```
File "D:\Python3.9.10\lib\unittest\case.py", line 59, in testPartExecutor
    yield
File "D:\Python3.9.10\lib\unittest\case.py", line 592, in run
    self._callTestMethod(testMethod)
File "D:\Python3.9.10\lib\unittest\case.py", line 550, in _callTestMethod
    method()
File "D:\Python3.9.10\lib\unittest\loader.py", line 34, in testFailure
    raise self._exception
ImportError: Failed to import test module: Experiment 7
Traceback (most recent call last):
  File "D:\Python3.9.10\lib\unittest\loader.py", line 154, in loadTestsFromName
    module = __import__(module_name)
ModuleNotFoundError: No module named 'Experiment 7'
```

进程已结束，退出代码 1

断言失败

断言失败

断言失败

断言失败