

现代程设第八周作业

童文宣 19377215

请用函数或类实现不同功能的装饰器并进行充分的测试。

1. 请用函数实现装饰器。部分函数往往需要将模型或者数据处理结果保存下来，但实际调用时却因为路径设置错误等原因导致文件无法存储，浪费大量的时间重复运行程序。请实现一个装饰器，接收函数的路径参数，检查路径对应文件夹是否存在，若不存在，则给出提示，并在提示后由系统自动创建对应的文件夹。
2. 请用类实现一个装饰器。部分函数可能需要花费较长时间才能完成，请实现一个装饰类，其能够在被装饰函数结束后通过声音给用户发出通知。了解并使用一下playsound或其他声音文件处理的库。另外，可否对根据返回值的类型，比如整数，元组等，来实现不同的声音通知？如果返回值有多个，可否多次按类型依次播放？
3. 请用类或者函数实现一个装饰器。部分函数可能会在运行过程中输出大量的中间状态或者中间结果，这些信息往往在程序出问题利于调试，但由于输出内容过多，可能在控制台无法全部查看。请实现一个装饰器，其能够将装饰函数在运行过程中的所有的输出（通过print）全部保存在特定的一个文件中。
4. 实现一个类，在其中提供一些方法模拟耗时耗内存的一些操作，如大的数据结构生成、遍历、写入文件序列化等，并通过其体验line_profiler、memory_profiler、tqdm、pysnoper等装饰器的相关功能。

```
import functools
from functools import wraps
from functools import reduce
import sys
from memory_profiler import profile
from line_profiler import LineProfiler
from tqdm import tqdm
import os
import time
from playsound import playsound
```

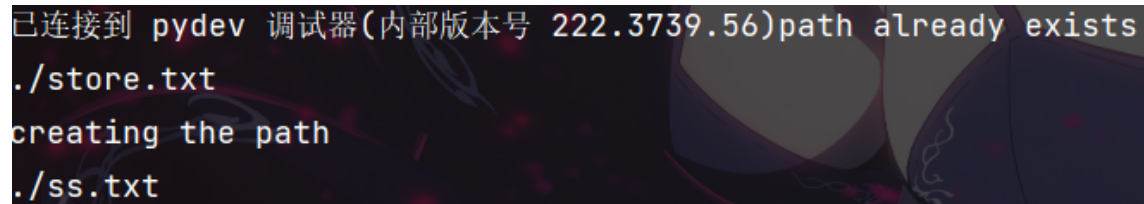
路径检查，生成

```
#检查是否存在相应路径
def path_examine(path, func):
    @wraps(func)
    def wrapper(*args, **kwargs):
        if os.path.exists(path)==True:
            print("path already exists")
        else:
            os.mkdir(path)#生成的是文件夹
            #with open 的方式也可以
            print("creating the path")
            return func(*args, **kwargs)
    return wrapper

path="./store.txt"
path2="./ss.txt"
ex1=functools.partial(path_examine,path)
ex2=functools.partial(path_examine,path2)
@ex1
def pri(path):
    print(path)
pri(path)

@ex2
```

```
def pri(path):
    print(path)
pri(path2)
```



已连接到 pydev 调试器(内部版本号 222.3739.56)path already exists
./store.txt
creating the path
./ss.txt

生成路径的方式采用这个更好

```
with open(path,"r") as f:
    pass
```

音乐通知程序运行，且返回的类型

```
#使用类来实现声音提醒代码处理
class Music:
    def __init__(self):
        pass
    def __call__(self, func):
        @wraps(func)
        def wrapper(*args, **kwargs):
            a=func(*args, **kwargs)
            if isinstance(a,str):
                playsound('./str.mp3')
            elif isinstance(a,int):
                playsound('./num.mp3')
            else:
                playsound("./els.mp3")
            return wrapper

@Music()
def fun1():
    for i in range(4):
        print(i)
    print(type(i))
    return i
fun1()
time.sleep(5)

@Music()
def fun2():
    for i in 'abcdef':
        print(i)
    print(type(i))
    return i
fun2()
time.sleep(5)
```

```
@Music()
def fun3():
    dic={'1':1, '2':2, '3':3, '4':4}
    dic=zip(dic.keys(),dic.values())
    for i in dic:
        print(i)
    print(type(i))
    return i
fun3()
time.sleep(5)
```

```
0
1
2
3
<class 'int'>
a
b
c
d
e
f
<class 'str'>
('1', 1)
('2', 2)
('3', 3)
('4', 4)
<class 'tuple'>
```

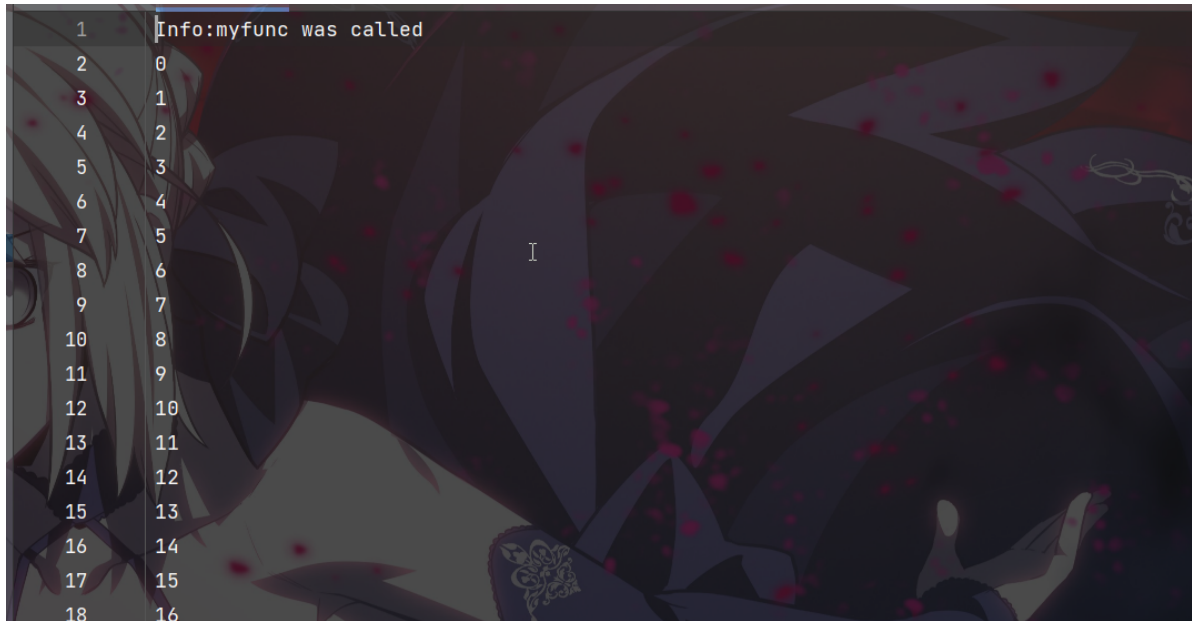
保存结果

```
#保存打印结果的装饰器
class storage:
    def __init__(self,path):
        self.path=path
    def __call__(self,func):
        @wraps(func)
        def wrapper(*args,**kwargs):
            __console=sys.stdout#保存屏幕的sys.out
            info='Info:'+func.__name__+" was called"
            with open(self.path,'a') as f:
                f.write(info+'\n')
```

```

        sys.stdout=f#将打印位置定位到文件中
        func(*args,**kwargs)
        sys.stdout=__console#还原位置
        return func(*args,**kwargs)
    ##如果需要使用func函数则返回函数即可，如果不需要使用则不返回也行
    return wrapper
@storage('./store.txt')
def myfunc():
    for i in range(100000):
        st=str(i)
        print(st)
#进行代码进度与时间的显示
myfunc()

```



时间、内存的测试

```

class Vis:
    def __init__(self):
        pass
    @profile#该方法使用的是命令行操作
    def test_time(self):
        for i in range(100):
            a=[1]*(10**6)
            b=[2]*(10**5)

    def sum(self,x,y):
        return x+y

    def mult(self,x,y):
        return x*y

    def ceshi(self):
        lis=[1]*(10**6)
        a=reduce(self.mult,lis)
        b=reduce(self.sum,lis)
        return a,b

```

```

def jin_du(self):#显示进度条
    lis=[1]*(10**3)
    bar=tqdm(lis)#转化成进度条的形式，跑了多少将会返回在进度条中
    j = 1
    for i in bar:##遍历表示跑了多少进度条
        bar.set_description("Now get "+ "No.".format(j))
        j=j+1
        time.sleep(0.001)

t=vis()
t.jin_du()
a,b=t.ceshi()
print(a)
print(b)
lp = LineProfiler()
lp_wrapper = lp(t.ceshi)
lp_wrapper()#
lp.print_stats()

```

```

D:\python\python.exe "D:/pycharm/PyCharm Community Edition 2022.2.1/plugins/python-ce/helpers/pydev/pydevd.py" -
127.0.0.1 --port 52809 --file D:/经管大三/现代程序设计/week8/week8.py
Now get No.: 74%|██████████| 735/1000 [00:11<00:04, 64.25it/s]

```

```

1
1000000
Timer unit: 1e-07 s

Total time: 1.86751 s
File: D:/经管大三/现代程序设计/week8/week8.py
Function: ceshi at line 126

Line #      Hits         Time  Per Hit   % Time  Line Contents
=====
126          1         31848.0    31848.0     0.2      def ceshi(self):
127          1    12207669.0   12207669.0    65.4          lis=[1]*(10**6)
128          1    6435587.0    6435587.0    34.5          a=reduce(self.mult,lis)
129          1         19.0         19.0     0.0          b=reduce(self.sum,lis)
130          1         19.0         19.0     0.0          return a,b

```