

1. 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，以测试如下的装饰器（用类或函数实现），如大的数据结构生成、遍历、写入文件序列化等。

测试随机生成一个长度为 n 的列表，进行排序，并且将结果写到 txt 文件中

```
import tqdm
from tqdm import tqdm
import random
import memory_profiler
import os
from playsound import playsound
import urllib.request

filepath=r"C:\Users\86186\Desktop\现代程序设计\8\result.txt"

class sort_list:
    def __init__(self,n,filepath) -> None:
        self.n=n
        self.filepath=filepath
    def random_sort(self):
        res=sorted ([random.random() for i in range (self.n)])
        with open(self.filepath,'w',encoding='utf-8') as f:
            f.write(str(res))
```

2. 如果要知道程序的运行时间、运行进度、内存占用情况，请利用 line_profiler、memory_profiler、tqdm 等装饰器实现相关功能，要求在程序执行结束后，打印程序的内存占用和运行时间。

(1) 测试程序运行时间

利用 line_profiler 装饰器测试程序运行的时间，利用命令行打印结果

```
@profile
#@memory_profiler.profile
#@tqdm_decorator
#@path_check
#@end_mark
def test(n,filepath):
    t=sort_list(n,filepath)
    t.random_sort()

test(200000,filepath)
```

运行结果：

```

C:\Users\86186\Desktop\现代程序设计\8>kernprof -v -l random.py
Wrote profile results to random.py.lprof
Timer unit: 1e-06 s

Total time: 0.229049 s
File: random.py
Function: test at line 55

```

Line #	Hits	Time	Per Hit	% Time	Line Contents
55					@profile
56					#@memory_profiler.profile
57					#@tqdm_decorator
58					#@path_check
59					#@end_mark
60					def test(n,filepath):
61	1	2.8	2.8	0.0	t=sort_list(n,filepath)
62	1	229046.3	229046.3	100.0	t.random_sort()

(2) 测试程序运行占用内存

利用 memory_profiler 装饰器测试程序内存占用情况

```

@memory_profiler.profile
#@tqdm_decorator
#@path_check
#@end_mark
def test(n,filepath):
    t=sort_list(n,filepath)
    t.random_sort()

test(200000,filepath)

```

运行结果：

```

Filename: c:\Users\86186\Desktop\现代程序设计\8\random.py

```

Line #	Mem usage	Increment	Occurences	Line Contents
51	58.4 MiB	58.4 MiB	1	@memory_profiler.profile
52				#@tqdm_decorator
53				#@path_check
54				#@end_mark
55				def test(n,filepath):
56	58.4 MiB	0.0 MiB	1	t=sort_list(n,filepath)
57	63.9 MiB	5.5 MiB	1	t.random_sort()

(3) 显示程序的运行进度
利用 tqdm 工具进行手动更新

```
def tqdm_decorator(func):
    def wrapper(*args,**kwargs):
        with tqdm(total=200000,desc='Example', leave=True, ncols=100, unit='B', unit_scale=True) as pbar:
            for i in range(200):
                func(*args,**kwargs)
                pbar.update(1000)
    return wrapper
```

运行结果:

```
@tqdm_decorator
#@path_check
#@end_mark
def test(n,filepath):
    t=sort_list(n,filepath)
    t.random_sort()

test(200000,filepath)
```

```
PS C:\Users\86186\Desktop\现代程序设计> c:: cd 'c:\Users\86186\Desktop\现代程序设计'; & 'c:\Users\86186\AppData\Local\Programs\Python\Python37\python.exe' 'c:\Users\86186\.vscode\extensions\ms-python.python-2021.11.1422169775\pythonFiles\lib\python\debugpy\launcher' '2912' '-...' 'c:\Users\86186\Desktop\现代程序设计\0\random.py'
Example: 6% | 13.0k/200k [00:03:00:51, 3.62kB/s]
```





- 在程序处理结束后, 通常需要将模型或者数据处理结果保存下来。但是, 有时会因为路径设置错误(忘记新建文件夹)等原因导致文件无法存储, 浪费大量的时间重复运行程序。一种解决方法是在执行程序前对参数中的路径进行检查。要求利用装饰器函数实现这一功能, 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。

检查文件是否存在, 不存在返回错误并且创建文件。

```
def path_check(func):
    def wrapper(*args,**kwargs):
        if not os.path.exists(args[1]):
            print("File not found!create now.")
            f=open(args[1],'w',encoding='utf-8')
            f.close()
        return func(*args,**kwargs)
    return wrapper
```

运行结果:

```
86186\.vscode\extensions\ms-python.python-2021.11.1422169775\pythonFiles\lib\python\debugpy\launcher
File not found!create now.
PS C:\Users\86186\Desktop\现代程序设计>
```






	__pycache__	2021/11/20 10:45	文件夹	
	random.py	2021/11/20 11:31	Python File	2 KB
	random.py.lprof	2021/11/20 11:03	LPROF 文件	1 KB
	result.txt	2021/11/20 11:31	文本文档	3,959 KB

4. 在程序运行结束后，可以给用户发送一个通知，比如播放一段音乐等。要求实现对应的装饰器类，在被装饰的函数执行结束后，可以主动播放声音（了解并使用一下 `playsound` 或其他声音文件处理的库）。

在运行结束后播放“yes”语音，因为 `playsound` 在播放后不考虑文件的占用情况，所以从网上下载音频生成临时文件，播放后删除。

```
def end_mark(func):
    def wrapper(*arg,**kwargs):
        func(*arg,**kwargs)
        curr_path = os.path.dirname(os.path.realpath(__file__))
        url = 'http://dict.youdao.com/dictvoice?type=0&audio=yes'
        tmp_file = 'tmp_voice.mp3'
        tmp_path = os.path.join(curr_path, tmp_file)
        urllib.request.urlretrieve(url, tmp_path)
        playsound(tmp_path)
        os.remove(tmp_path)
    return wrapper
```

运行结果：

	__pycache__	2021/11/20 10:45	文件夹	
	random.py	2021/11/20 11:35	Python File	2 KB
	random.py.lprof	2021/11/20 11:03	LPROF 文件	1 KB
	result.txt	2021/11/20 11:35	文本文档	3,960 KB
	tmp_voice.mp3	2021/11/20 11:35	MP3 文件	29 KB

可以看到下载的音频文件，会在程序运行后播放