装饰器

🌸 19377404 王临寒

STEP 1 实现耗时耗内存的类

对一个数组进行遍历并且输入到一个 txt 文件中，如果 range 足够大，就会消耗大量时间和内存。

```python
class Traverse:
    def __init__(self,range):
        self.range = range
        self.list = []

    @profile
    def traverse(self):
        for i in range(self.range):
            self.list.append(i)
        f = open('output.txt', 'wb')
        pickle.dump(self.list, f)
```

STEP 2 运行时间、运行进度、内存占用情况

STEP 2.1 运行时间

import 模块：

```python
from line_profiler import LineProfiler
```

此时需要在命令行中运行以下代码：

```python
kernprof -l -v main.py
```

输出结果：

```
(homework) D:\大三上\10现代程序设计技术\作业\第8次>kernprof -l -v main.py
Wrote profile results to main.py.lprof
Timer unit: 1e-06 s

Total time: 0.0084087 s
File: main.py
Function: traverse at line 11

Line #      Hits         Time  Per Hit   % Time  Line Contents
==============================================================
    11                                           @profile
    12                                           def traverse(self):
    13       101         42.5      0.4      0.5          for i in range(self.range):
    14       100         58.0      0.6      0.7              self.list.append(i)
    15         1        628.4    628.4      7.5          f = open('output.txt', 'wb')
    16         1       7679.8   7679.8     91.3          pickle.dump(self.list, f)
```

STEP 2.2 运行进度

import 模块：

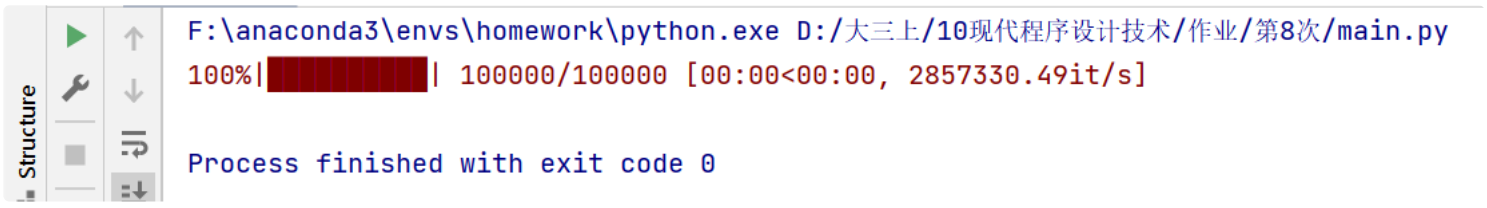
```python
from tqdm import tqdm
```

稍微修改 traverse 函数：

```python
    def traverse(self):
        total = self.range
        for i in tqdm(range(self.range)):
            self.list.append(i)
        f = open('output.txt', 'wb')
        pickle.dump(self.list, f)
```

Python

成功输出进度条：



STEP 2.3 内存占用情况

import 模块：

```python
from memory_profiler import profile
```

Python

此时的 traverse 函数：

```python
    @profile
    def traverse(self):
        total = self.range
        for i in range(self.range):
            self.list.append(i)
        f = open('output.txt', 'wb')
        pickle.dump(self.list, f)
```

Python

测试的 main 函数，可以在 pycharm 中直接运行。

```python
if __name__ == "__main__":
    example = Traverse(1000)
    example.traverse()
```

Python

输出结果：

```
Line #    Mem usage    Increment  Occurences   Line Contents
=============================================================
    11     59.4 MiB     59.4 MiB           1       @profile
    12                                             def traverse(self):
    13     59.4 MiB      0.0 MiB        1001           for i in range(self.range):
    14     59.4 MiB      0.0 MiB        1000               self.list.append(i)
    15     59.4 MiB      0.0 MiB           1           f = open('output.txt', 'wb')
    16     59.4 MiB      0.0 MiB           1           pickle.dump(self.list, f)
```

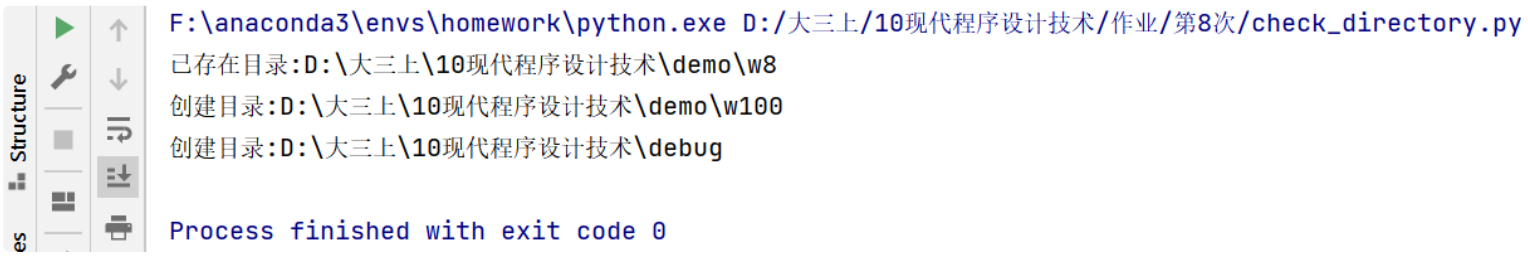
STEP 3 对参数路径进行检查

```python
# 导入模块
import os
from functools import wraps

# 装饰器函数
def check(func):
    @wraps(func)
    # 如果参数中有"\",则视为路径
    def wrapper(*args, **kwargs):
        dir_list = []
        # 所有的参数都要进行检查
        for i in args:
            if "\\" in str(i):
                dir_list.append(str(i))
        for j in kwargs.items():
            if "\\" in str(j):
                dir_list.append(str(j))
```

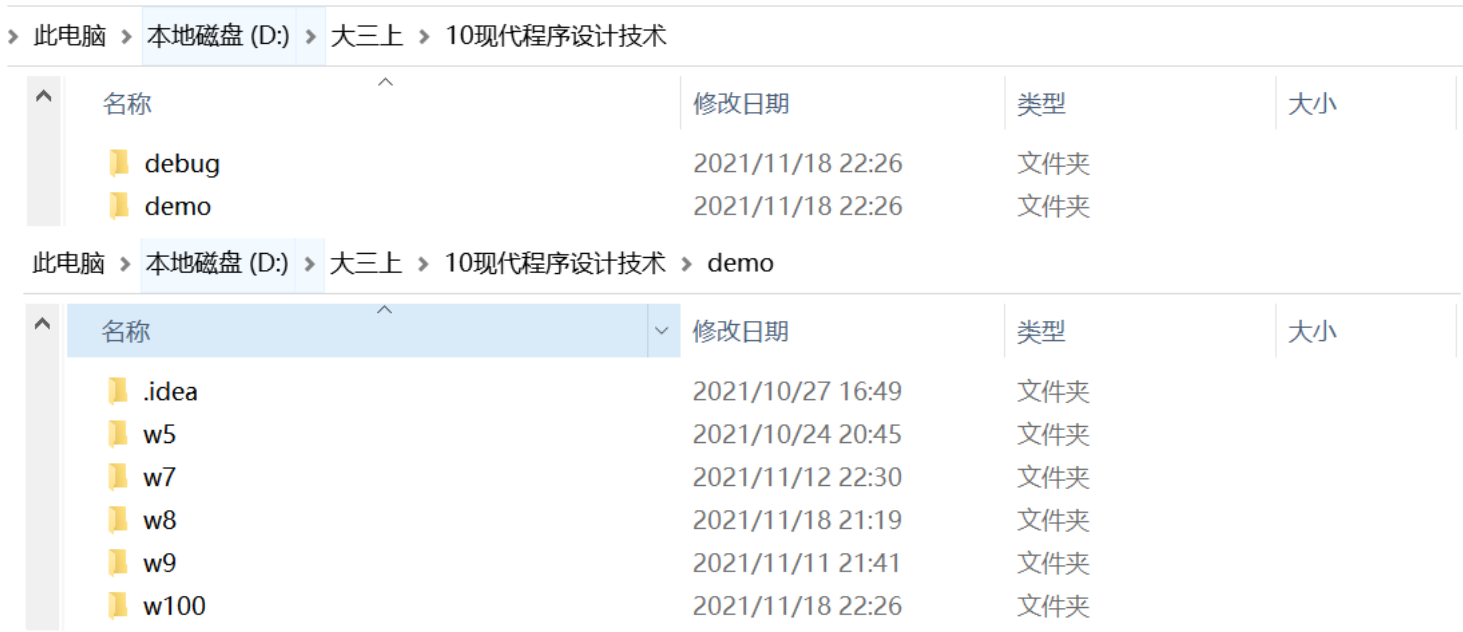
```python
        for dir in dir_list:
            if not os.path.exists(dir):
                os.makedirs(dir)
                print("创建目录:" + dir)
            else:
                print("已存在目录:" + dir)

    return wrapper

# 被装饰的函数
@check
def print_file(path):
    print(path)

# 用于测试的主函数
if __name__ == "__main__":
    print_file("D:\\大三上\\10现代程序设计技术\\demo\\w8")
    print_file("D:\\大三上\\10现代程序设计技术\\demo\\w100")
    print_file("D:\\大三上\\10现代程序设计技术\\debug")
```

Python

运行结果：



同时 debug 目录和 w100 目录被成功创建：



STEP 4 程序结束后播放声音

```python
from playsound import playsound
from functools import wraps


def music(func):
    @wraps(func)
    def wrapper(*args, **kwargs):
        func(*args, **kwargs)
        # 在程序运行完成之后播放音乐
        playsound(u"song.mp3")

    return wrapper


@music
def print_happy():
    print("happy!")


if __name__ == "__main__":
    print_happy()
```

Python

运行完成之后就可以听到美妙的音乐啦！