# 2024 秋 Python 程序设计大作业

张城玮 (zhang-cw23@mails.tsinghua.edu.cn，2023013342)

2024 年 12 月 7 日

## 1 设计架构

本代码主要由两个类组成，分别为 TimeManagerApp 类以及 PomodoroDialog 类。TimeManagerApp 类为本代码的主要部分，实现了绝大部分功能。PomodoroDialog 类通过访问 TimeManagerApp 类的部分变量控制提示窗口的弹出，实现番茄钟功能。

## 2 核心代码讲解

### 2.1 库的调用

本代码调用的库均为 Python 内置的库，不需要额外安装。tkinter 库主要用来实现图形界面以及交互操作。time 库用来实现计时操作。datetime 函数用来将时间戳转化为本地时间。csv 库用来处理历史记录文件的存储、读取和删除。Path 函数用来获取文件地址。

```
import tkinter as tk
from tkinter import messagebox
from tkinter import simpledialog
import time
import csv
from datetime import datetime
from pathlib import Path
```

### 2.2 PomodoroDialog 类

#### 2.2.1 enable_pomodoro 函数与 disable_pomodoro 函数

```
def enable_pomodoro(self):
    self.pomodoro_enabled=True
    self.ask_for_gap()
    self.on_update(self.pomodoro_enabled,self.pomodoro_gap)
    self.destroy()

def disable_pomodoro(self):
    self.pomodoro_enabled=False
```

```
        self.on_update(self.pomodoro_enabled,self.pomodoro_gap)
        self.destroy()
```

这两个函数通过对话框的按键选择是否启用番茄钟功能，并且通过 self.on_update 函数对父类的相关变量值进行修改，最后关闭该对话框。

#### 2.2.2 ask_for_gap 函数

```
def ask_for_gap(self):
    gap_str=simpledialog.askstring(" 番茄钟间隔"," 请输入提醒间隔时间:",parent=self)
    if gap_str and gap_str.isdigit():
        self.pomodoro_gap=int(gap_str)
    else:
        messagebox.showerror(" 输入错误"," 请输入有效的分钟数！")
        self.ask_for_gap()
```

该函数主要弹出对话框，获取用户想要的间隔时长。假如用户没能输入有效的的间隔时间，则弹出错误提示并重新弹出对话框。

### 2.3 TimeManagerApp 类

#### 2.3.1 create_widgets 函数

该函数主要用于创建图形界面，一共有两个输入框，五个按键，以及一个计时工具。

#### 2.3.2 update_time_display 函数

```
def update_time_display(self):
    if self.task_start_time:
        elapsed_time=time.time()-self.task_start_time
        ...
        time_str=f'{hours:02}:{minutes:02}:{seconds:02}'
        ...
        self.root.after(1000,self.update_time_display)
```

该函数通过获取现在的时间戳减去任务开始时的时间戳，得到任务进行的时间，并将其转化为时分秒的形式。

#### 2.3.3 start_task 函数

```
    self.task_name=self.task_name_entry.get()
    self.task_note=self.task_note_entry.get()
    if not self.task_name:
```

```
        messagebox.showerror(" 错误"," 请输入任务名称！")
        return
```

这一段代码为获取任务的类型与备注。假如没有输入任务类型则会提示错误，并且要求重新输入。

```
    pomodoro_dialog=PomodoroDialog(self.root,self.update_pomodoro_settings)
    self.root.wait_window(pomodoro_dialog)

    if self.pomodoro_enabled:
        self.pomodoro_start_time=time.time()
        self.schedule_pomodoro_reminder()
```

这段代码创建了一个 PomodoroDialog 类的对象。该对象用于决定 self.pomodoro_enabled 变量是否改变（以及在 self.pomodoro_enabled 变量改变的基础上获取时间间隔），进而决定是否调用后面的 schedule_pomodoro_reminder 函数以及必要参数（即是否使用番茄钟功能）。

```
    self.task_start_time=time.time()
    self.update_time_display()
```

这段代码调用了时间更新的函数，实现了任务进行时的计时功能。

### 2.3.4 end_task 函数

```
task_end_time=time.time()
task_duration=task_end_time-self.task_start_time
task_duration_hours=change_seconds_into_hours_minutes_seconds(task_duration)[0]
task_duration_minutes=change_seconds_into_hours_minutes_seconds(task_duration)[1]
task_duration_seconds=change_seconds_into_hours_minutes_seconds(task_duration)[2]

task_record = {
    "task_name": self.task_name,
    "task_note":self.task_note,
    "start_time": datetime.fromtimestamp(self.task_start_time).strftime('%Y-%m-%d %H:%M:%S'),
    "end_time": datetime.fromtimestamp(task_end_time).strftime('%Y-%m-%d %H:%M:%S'),
    "duration": task_duration,
    "duration_hours": task_duration_hours,
    "duration_minutes": task_duration_minutes,
    duration_seconds": task_duration_seconds
    }

self.records.append(task_record)
self.save_record_to_file(task_record)
```

这一段代码用于将时间戳转化为时分秒形式的本地时间，并且调用 save_record_to_file 函数将其保存下来。该函数的其他部分为将变量初始化。

### 2.3.5 view_history 函数

```
history_window=tk.Toplevel(self.root)
history_window.title(" 历史记录")

text_widget=tk.Text(history_window,width=100,height=20)
text_widget.pack(padx=10, pady=10)

file_path=Path(__file__).parent/"task_records.csv"

if not file_path.exists():
    text_widget.insert(tk.END," 没有历史记录！ ")
    return

with open(file_path, mode="r") as file:
    reader=csv.DictReader(file)
    for idx, row in enumerate(reader):
        start_time=f"{row['start_time']}"
        end_time=f"{row['end_time']}"
        task_name=f"{row['task_name']}"
        task_note=f"{row['task_note']}"
        duration=\
        f"{row['duration_hours']}小时{row['duration_minutes']}分钟{row['duration_seconds']}秒"
        text_widget\
        .insert(tk.END, f"{start_time}-{end_time}|{task_name}|{task_note}|{duration}\n")

    text_widget.config(state=tk.DISABLED))
```

该函数访问历史记录文件得到相关数据，然后将其读取出后储存到相应变量中。在最后使用 TEXT 组件完成数据的展示，并且更改窗口的状态使其不能被编辑。

### 2.3.6 view_statistics 函数

整体结构与 view_history 函数相近，不同的地方在于该函数通过字典形式将不同的数据分开，最后遍历字典打印数据。

```
for row in reader:
    task_date=row['start_time'].split(' ')[0]
    task_duration=float(row['duration'])
    task_category=row['task_name']
```

```
    if task_date not in task_duration_by_day:
        task_duration_by_day[task_date]=0
        task_duration_by_day[task_date]+=task_duration

        if task_date not in task_category_duration_by_day:
        task_category_duration_by_day[task_date]={}

        if task_category not in task_category_duration_by_day[task_date]:
        task_category_duration_by_day[task_date][task_category]=0
        task_category_duration_by_day[task_date][task_category]+=task_duration
```

如上代码所示，同一天的任务时间、同一天同一类型的任务时间被分开并储存在相应的字典中。最后遍历字典打印结果即可。

#### 2.3.7 delete_history 函数

```
confirmation=messagebox.askyesno(" 确认删除"," 您确定要删除所有历史记录吗？")
if confirmation:
    file_path=Path(__file__).parent/"task_records.csv"
        if file_path.exists():
            file_path.unlink()
            messagebox.showinfo(" 成功"," 历史记录已成功删除！ ")
        else:
            messagebox.showerror(" 错误"," 没有历史记录！ ")
```

此函数结构简单，首先确定是否删除记录，其次确定是否存在历史记录。若存在历史记录则删除，若不存在则返回错误信息。

# 3 使用方法

## 3.1 任务执行

1. 假如想要执行任务，则需要先输入任务名称（必须!），如有需要也可以输入备注。
2. 随后点击“开始任务”按钮，则会弹出“是否启用番茄钟”的询问界面。假如需要使用番茄钟，则点击“启用”按钮，继续输入自己想要的时间间隔；假如不需要使用，则点击“关闭”按钮。
3. 完成上述操作后则任务已经开始。在窗口下沿可以看到任务的进行时间。假如启用番茄钟功能，则到了预定时间会弹出提示窗口。
4. 假如想要结束任务，则点击“结束任务”按钮。

### 3.2 历史记录与统计

1. 如果想要查询历史记录，则点击“查看历史记录”按钮即可。
2. 如果想要查询历史统计数据，则点击“查看历史统计”按钮即可。
3. 如果想要删除历史数据，则点击“删除历史数据按钮”，此时会弹出确认删除的界面，点击“确认”按钮即可。