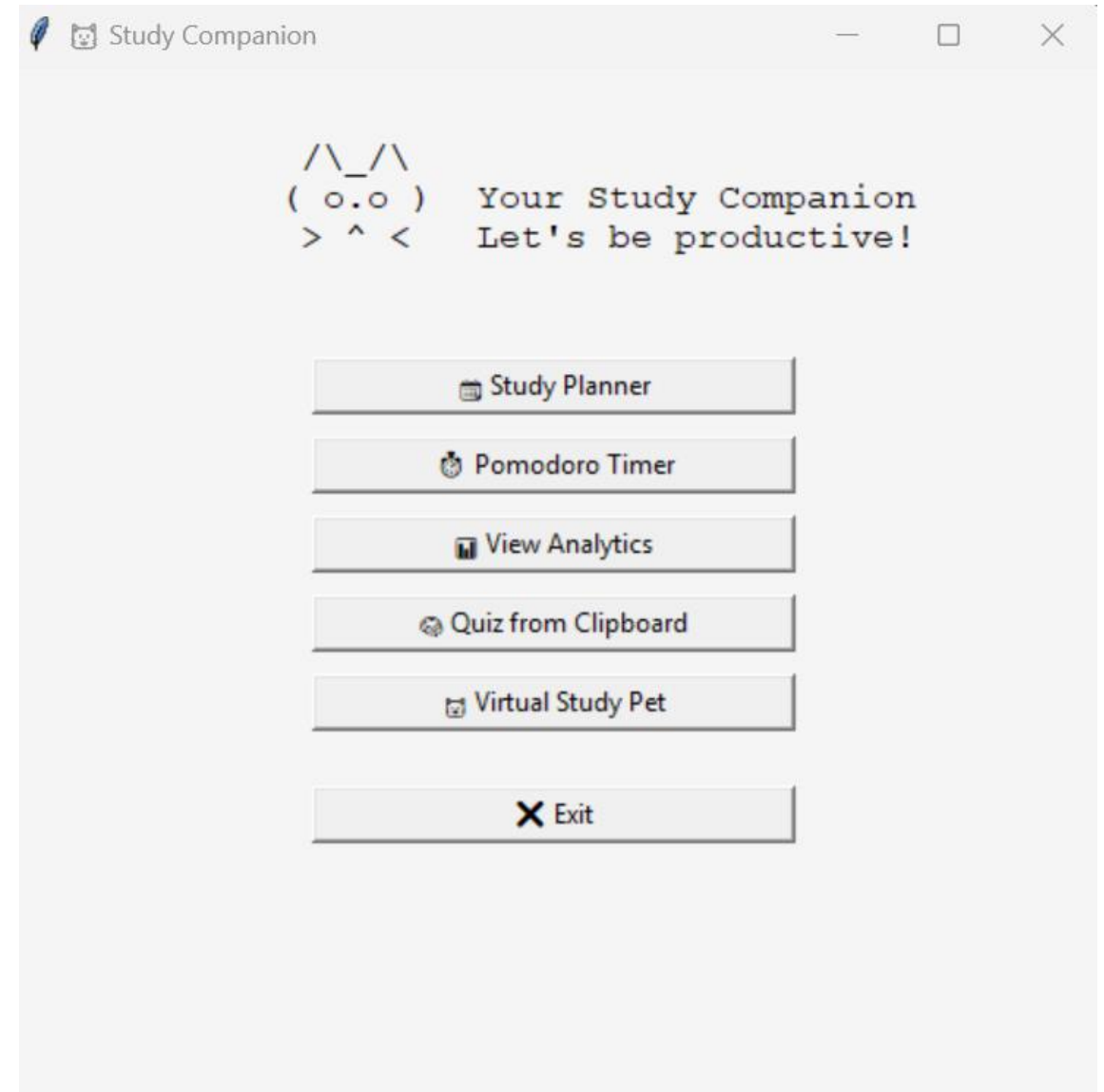


Software Carpentry Final Project

By: Jana AlMadani & Maya Beyer

# Study Companion



# Overview

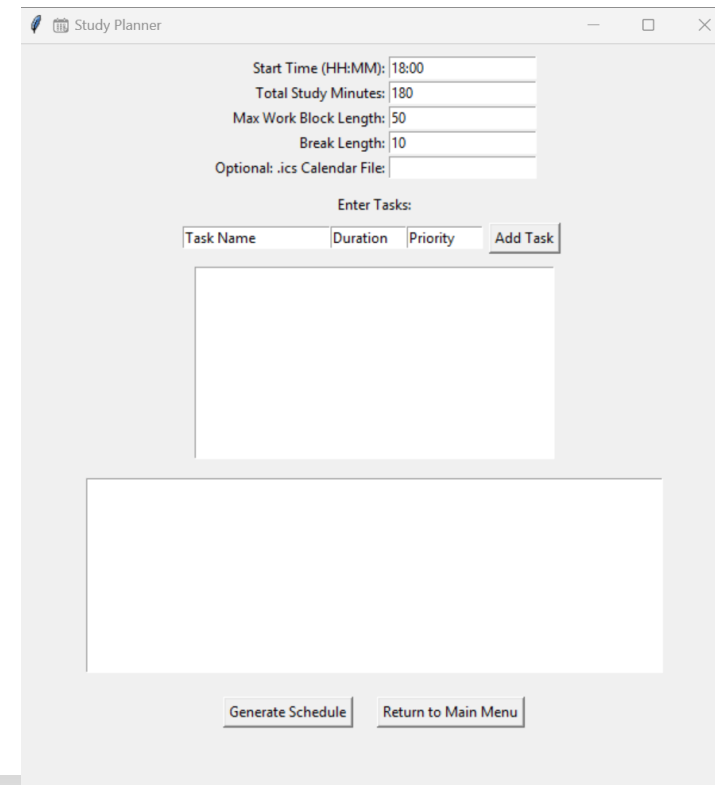
- Python-based productivity tool to support focused studying.
- Modular system combining 6 productivity functions in one program with GUI.
- Integrates planning, timed work cycles, analytics, and auto-generated quizzes.
- Helps organize tasks, stay focused, and review material effectively.
- Includes optional features for motivation and calendar awareness.



# Part I: Smart Study Planner

- Defines a *Task* dataclass to store task name, duration, priority, deadline.
- *build\_schedule()* generates a timeline.
  - Sorts tasks by priority & deadline.
  - Creates timed work blocks using *timedelta*.
  - Inserts breaks between blocks.
  - Avoids calendar conflicts from .ics and exports.ics file for calendar import
- Uses Python's *datetime* module to build continuous schedule.
- Outputs plan via *print\_schedule()* in readable timeline format.

```
Study Plan
-----
18:00 - 18:50 Biochem exam review
18:50 - 19:00 Break
19:00 - 19:40 Biochem exam review
19:40 - 20:30 Chinese translation homework
20:30 - 20:40 Break
20:40 - 20:50 Chinese translation homework
20:50 - 21:00 Alternative Energy reading
-----
Total window length: 180 min
```



The screenshot shows a window titled "Study Planner" with a light gray background. At the top, there are five input fields with labels: "Start Time (HH:MM):", "Total Study Minutes:", "Max Work Block Length:", "Break Length:", and "Optional .ics Calendar File:". Below these fields is a section titled "Enter Tasks:" which contains a table with three columns: "Task Name", "Duration", and "Priority". To the right of the "Priority" column is an "Add Task" button. Below the table is a large empty rectangular box for task entry. At the bottom of the window, there are two buttons: "Generate Schedule" and "Return to Main Menu".

# Part II: Pomodoro Timer

- Implements timed work/break cycles for a selected task.
- Uses Two Dataclasses: *PomodoroConfig* (settings) and *PomodoroResult* (summary).
- Core logic in *run\_pomodoro*, *run\_interval*, and *sleep\_unit*.
- Logs each session to *sessions.csv* for later analytics.
- Generates work versus break pie chart with Matplotlib (*plot\_session\_pie*).
- Two Moes: Demo Mode & Real-Time Mode

```
=== Cycle 1/2 ===
--- Work for 2 minute(s) ---
Work time remaining: ~1 min
Work time remaining: ~1 min
Work time remaining: ~0 min
Work time remaining: ~0 min
Work time remaining: ~0 min
*** Work interval finished! ***

--- Break for 1 minute(s) ---
Break time remaining: ~0 min
Break time remaining: ~0 min
Break time remaining: ~0 min
Break time remaining: ~0 min
Break time remaining: ~0 min
*** Break interval finished! ***

=== Cycle 2/2 ===

--- Work for 2 minute(s) ---
Work time remaining: ~1 min
Work time remaining: ~1 min
Work time remaining: ~0 min
Work time remaining: ~0 min
Work time remaining: ~0 min
*** Work interval finished! ***

Pomodoro Summary
-----
Task:           Example: Biochem exam review
Cycles completed: 2
Focused work time: 4 min
Break time:      1 min
Total session time: 5 min
Started at:      2025-12-07 17:55
Finished at:     2025-12-07 17:56

Session logged to: sessions.csv
Session summary pie chart saved to: session_summary.png
```

Pomodoro Timer Se...

Task Name:

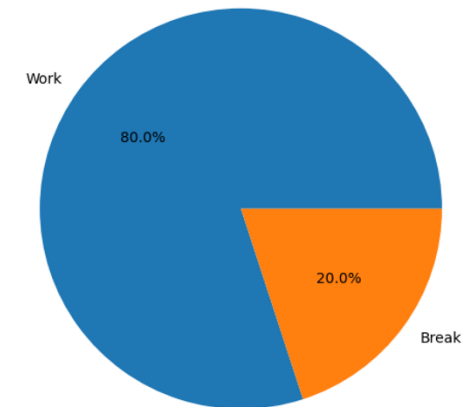
Work Minutes:

Break Minutes:

Number of Cycles:

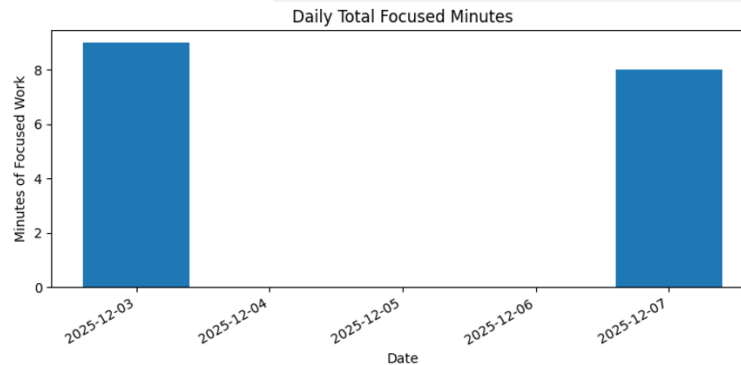
☐ Demo Mode (Fast)

Pomodoro Session: Example: Biochem exam review



sessions						
date	task_name	cycles_completed	work_minutes	break_minutes	started_at	finished_at
2025-12-03	Example: Biochem exam review	2	4	1	2025-12-03T16:34:57	2025-12-03T16:35:13
2025-12-03	Example: Biochem exam review	2	4	1	2025-12-03T16:40:02	2025-12-03T16:40:17
2025-12-03	Chem	1	1	0	2025-12-03T16:40:57	2025-12-03T16:41:57
2025-12-07	Example: Biochem exam review	2	4	1	2025-12-07T13:28:40	2025-12-07T13:28:55
2025-12-07	Example: Biochem exam review	2	4	1	2025-12-07T17:55:53	2025-12-07T17:56:08

# Part III: Productivity Analytics Dashboard



```
=== Study Companion - Productivity Analytics Dashboard ===
Mode: REAL DATA

--- Text Summary ---
Study streak (longest consecutive days): 1 day(s)
Average focused minutes per session:    3.4 min
Most common study start hour:          16:00

Total focused minutes (today):           0 min
Total focused minutes (this week):       0 min
Total focused minutes (all time):        17 min

Generating charts...
Saved: analytics_daily_focus.png
Saved: analytics_time_of_day.png

Analytics complete. Open the PNG files to view the charts.
```

- Reads session history from *sessions.csv* to computer key. Metrics like study streak, average session length, peak study hour, total focused minutes.
- Converts times into Python *datetime* objects.
- Creates visualizations using Matplotlib (daily focus chart + time-of-day histogram).
- Includes a demo mode that generates fake data for testing.
- Uses simple Python data processing; reading CSV, aggregating minutes, counting hours, and summarizing patterns.

```
Summary Bullet Points:
- Guanxi is a Chinese term that refers to the network of relationships a person builds
maintains over time.
- In many contexts, guanxi can help individuals access resources, opportunities, or se
more quickly.
- However, reliance on guanxi can also create inequality, because people without stron
connections may be left out or receive lower-quality treatment.
```

Questions:

Q1: Fill in the blank:  
Guanxi is a Chinese term that refers to the network of \_\_\_\_ a person builds and maintai  
time.

Answer: relationships

Q2: Fill in the blank:  
In many contexts, guanxi can help individuals access resources, \_\_\_\_, or services more  
Answer: opportunities

Q3: Fill in the blank:  
However, reliance on guanxi can also create inequality, because people without strong  
connections may be left out or receive \_\_\_\_ treatment.  
Answer: lower-quality

Using built-in demo passage about guanxi.

=== Summary Bullet Points ===

```
- Guanxi is a Chinese term that refers to the network of relationships a person builds and maintains over time.
- In many contexts, guanxi can help individuals access resources, opportunities, or services more quickly.
- However, reliance on guanxi can also create inequality, because people without strong connections may be left out or receive lower-quality treatment.
```

=== Quiz Time ===

Question 1:

Fill in the blank: Guanxi is a Chinese term that refers to the network of \_\_\_\_ a  
person builds and maintains over time.  
Your answer: relationships  
Correct.

Question 2:

Fill in the blank: In many contexts, guanxi can help individuals access  
resources, \_\_\_\_, or services more quickly.  
Your answer: opportunities  
Correct.

Question 3:

Fill in the blank: However, reliance on guanxi can also create inequality,  
because people without strong connections may be left out or receive \_\_\_\_  
treatment.  
Your answer: better  
Incorrect. Correct answer: lower-quality

You answered 2 out of 3 correctly.

Clipboard Quiz Generator

Paste or write a passage below:

Paste from Clipboard

Generate Quiz

Return to Main Menu

Output:

## Part IV: Quiz Generator

- Takes text from demo mode or user-provided input.
- Cleans and splits text into sentences.
- Generates bullet-point summaries.
- Pulls out important words and scores sentences based on number of keywords and time appeared in text.
- Creates fill-in-the-blank quiz questions.
- Allows interactive quiz and saves results to a .txt file.
- Uses helper functions for keyword extraction and sentence scoring.

# Part V: Calendar Sync

- The user is prompted to upload a .ics file, which is parsed using the ics Python Package.
- Busy times are stored as (start, end) blocks.
- During scheduling, we check each study task:
  - if it overlaps with a calendar event -> we shift it.

18:00 - 18:30 Study: Math

18:30 - 19:00 Break

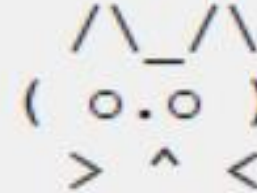
(Skipped 19:00-20:00 because of a calendar event)

20:00 - 20:45 Study: Biology

# Part VI: Virtual Study Pet

---

- Reads `sessions.csv` from Pomodoro logs.
- Tracks the total focused time and longest study streak
- Calculates the pet's:
- **Mood:** tired 🤔 / happy 😊 / proud 🐱
- **Level:** increases every 2 hours of focus.
- Displays ASCII art of a pet.

An ASCII art representation of a pet's face. The eyes are represented by two small circles with a dot in the middle, enclosed in parentheses. The mouth is a simple caret (^) shape. The ears are represented by two diagonal lines (/ \) on the top left and top right. The nose is a small horizontal line (-) between the eyes.

Your pet's current level: 1  
Mood: 🤔  
Study streak: 2 day(s)

[Return to Main Menu](#)



# Study Companion (Main Launcher)

## Planner

- **Functions:**
  - build\_schedule() – schedules tasks with breaks and calendar conflicts
  - run\_example\_scenario() – demo schedule
  - run\_interactive\_scenario() – custom GUI planner
  - run\_planner\_gui() – main GUI function
- **Imports:**
  - calendar\_sync.py (ICS import)
  - calendar\_export.py (ICS export)

## Pomodoro

- **Functions:**
  - run\_pomodoro() – runs the timer and logs result
  - run\_interactive() – GUI to configure timer
  - append\_to\_log() – logs session to sessions.csv
  - plot\_session\_pie() – creates pie chart summary
- **Supports: Demo mode, CLI, and GUI**

## Analytics

- **Functions:**
  - load\_sessions() – reads sessions.csv
  - generate\_demo\_sessions() – mock data for testing
  - run\_analytics\_gui() – tkinter GUI
- **Uses matplotlib for bar chart of time spent per task**

## Quiz Generator

- QuizQuestion dataclass
- **Functions:**
  - load\_pomodoro\_history() – loads session data from sessions.csv
  - calculate\_pet\_stats() – computes stats for level/mood
  - export\_to\_text() – saves quiz to .txt
- **Supports clipboard input using pyperclip**

## Virtual Pet

- QuizQuestion dataclass
- **Functions:**
  - load\_pomodoro\_history() – loads session data from sessions.csv
  - calculate\_pet\_stats() – computes stats for level/mood
  - run\_pet\_dashboard\_gui() – pet GUI with ASCII art
- **Pet levels up with more focus time and cycles**

```
gpu_data:
    __init__(self):
        gpu = gpuInfo.get_gpu(0)
        self.load = int(gpu.query_load() * 100)
        self.gpu_clock = int(round(gpu.query_clock() / 1000))
        self.gpu_memory_usage = round(gpu.query_memory_usage())
        self.gpu_gtt_usage = round(gpu.query_gtt_usage())
        self.power = gpu.query_power()
        self.voltage = round(gpu.query_graphics_voltage())
        fans = sensors_fans()
        for name, value in fans.items():
            setattr(self, name, value[0][1])
```

Thank You!