Designing a Scheduler for a Day in the Life of a Minervan (Part I)

- Please refer to the CS110 course guide on submitting your assignment materials.
- **Please read through all instructions carefully before working on the assignment (the detailed prompts address many questions raised by students in the past).**

An overview

Revisiting the activity scheduler from the "Heaps and priority queues" lesson—Your activity scheduler receives a list of tasks to be performed daily, with the following minimal schema:

`id`	`description`	`duration`	`dependencies`	`status`

Figure 1. A task is defined by the five data fields depicted here.

- `id`: a unique task identifier (that other tasks can reference).
- `description`: a short description of the task (e.g., visit the Neues Museum, dine gogigui, get a Fahrkarten, go to an exhibition at the Dongdaemun Design Plaza).
- `duration`: how long a task takes in minutes.
- `dependencies`: task `id`'s that must be completed before the current task (e.g., one cannot eat gogigui unless one has arrived at a restaurant, has gotten a table, has ordered, etc.).
- `status`: the state of a task; possible values are `not_yet_started`, `in_progress`, or `completed`.

tpress

You can include more data fields, provided you explain what they represent and why they are relevant. The scheduler will plan the execution of the tasks, outputting a step-by-step execution **and** reporting the total time required to execute them. That is what a scheduler should do!

Criteria for scheduling tasks—Your program will start by determining which tasks are ready to execute. How will your implementation differ from the one discussed in our Forum session? You will need to:

1. write your max- or min-heap Class implementation (rather than using the `heapq` module)
2. compute each task priority (in our class, we assumed that every task was equally important, which is unrealistic). There are at least two aspects that might change the priority of a task:
    a. some tasks might be required to happen at a specific time (for example, classes).
    b. tasks without dependencies may have higher priority than those with dependencies.

Tasks priority value—It's your choice to define the priority criterion for the situations described above; you just need to explain why it's a sensible choice.

Deliverables

Setting up.

Prepare a table containing at least **eight** tasks you typically do daily:

- **at least five** "regular" tasks:
    - e.g., if you go grocery shopping (one task), you may need to collect shopping bags from your room, leave the residence, and take a bus (three other tasks).
- **at least three** rotation-city-centered tasks.
    - explain how these allow you to immerse into the city's culture deeply; for inspiration, please consult the City Calendar or the Events section in the Community Portal, where all the City Experiences will be up to date.

Preparing your algorithmic strategy 🎥.

In a < 4-minute video (you must provide the **shareable link and check it is accessible at the time of submission**), where your face is visible at all times, provide answers to the following questions.

A. Formulate a simple argument for why a priority queue is a particularly well-suited data structure to prioritize tasks in this context. Why don't we simply sort a list of tasks?
    a. An important consideration is to gauge whether you need one or more instances of priority queues. For instance, tasks can be time-flexible, and others might be fixed in time (you can prepare for class at any time before it starts, but the session time is fixed).
    b. Would you separate these tasks into different priority queues? Why or why not?
    c. Feel free to provide an overview, e.g. "at this stage in the algorithm, I will work with ____, and later on, I will use ____ to do _____."

B. Describe how your scheduler will work at a very high level. Consider explaining the algorithmic approach to scheduling to a peer who has not taken CS110 before.
    a. Focus on how the algorithm digests several tasks and returns an output (no jargon).
    b. Your description must be clear and concise enough that anyone fluent in Python can implement it. You may consider displaying resources to support your description (for instance, a flowchart or schematics) which other students have found helpful.

C. Explain how you have defined and computed the priority value of each task. This value is a profit/preference level, which you should compute (possibly as a function of all the other tasks and their properties included in the schedule). This whitepaper is a good opportunity to apply #utility and motivate your utility function. Here are a few more notes:
    a. The user will not *provide* the priority value for a given task. Instead, the code should calculate it and then use it as a key in the priority queue. Think about it—sometimes, you can only grasp how something is more important than something else when you consider several attributes of both to make a comparison. What criterion goes into the decision to prioritize one task over another? The same applies to scheduling tasks.
    b. The higher the priority value of a task, the more urgent it is to complete it first. The choice of employing a max or min-heap should reflect that. Explain your choice.
    c. Writing "More important tasks have higher priorities" is not insightful. "Brushing my teeth has a priority of 70, and dressing up a priority of 60 because I want to brush my teeth before putting some clothes on" is not sufficiently explanatory either. **Refer to the whitepaper mentioned earlier for explicit examples of computing utility values.**
    d. You must present a transparent methodology for computing the priority values. If task A has dependencies and task B has none, it seems reasonable to induce that task B will have higher priority than task A. You may need to consider other task features too.

Please take the time to make your argument clear and concise, it will make it easier for you to implement the algorithm. The 4-minute limit on the video recording will help you focus on what is most important.

Working on your Python implementation.

A. Provide **your own** OOP implementation of the MaxHeapq or MinHeapq (depending on your choice of the previous question). The skeleton, basic code provided below is as a starting point, and you should add any methods/attributes that you deem necessary to expand the application of this data structure to the context of the scheduler.

a. As for the methods already included below, please do not change their names.
b. Provide at least three test cases to demonstrate that your code works as intended.
c. You have already worked on this code in preparation for one of the Forum sessions. If you reutilize your implementation, you need to cite your own work.

```
class MaxHeapq (or MinHeapq):
    """
    YOUR DOCSTRING
    """

    def __init__(self):
        """
        YOUR DOCSTRING
        """
        ### YOUR CODE HERE

    def left(self, i):
        """
        YOUR DOCSTRING
        """
        ### YOUR CODE HERE

    def right(self, i):
        """
        YOUR DOCSTRING
        """
        ### YOUR CODE HERE

    def parent(self, i):
        """
        YOUR DOCSTRING
        """
        ### YOUR CODE HERE

    def heappush(self, key):
        """
        YOUR DOCSTRING
        """
        ### YOUR CODE HERE

    def heapify(self, i):
        """
        YOUR DOCSTRING
        """
        ### YOUR CODE HERE

    def heappop(self):
        """
        YOUR DOCSTRING
        """
        ### YOUR CODE HERE
```

B. Program an activity scheduler in Python, which receives the list of tasks above as input and returns a schedule for you to follow. Please use the code we have discussed in our session, “Heaps and priority queues,” as a basic skeleton that you are expected to improve on.

- Please refrain from using any external Python library for this question besides `pandas`, `math`, and the `random` modules.[1] The `heapq` is not allowed.
- You can include a method to compute the priority value of each task.
- Below is an **example** of what the input and output ***may*** look like—the output provides an order for the tasks, the starting and ending times, and a priority value assigned to each task.

```
task_0 = Task(id = 0, description = 'Get up, rise and shine!', duration = 10, dependencies = None)
task_1 = Task(id = 1, description = 'Get dressed and ready for the day.', duration = 15, dependencies = task_0]
task_2 = Task(id = 2, description = 'Eat a healthy breakfast.', duration = 40, dependencies = task_0]
my_tasks = [task_0, task_1, task_2]

simple_scheduler = MyScheduler(my_tasks, init_time = 8*60, final_time = None)
simple_scheduler.run()

⏱ t=8h00, started executing task "Get up, rise and shine!"
   ✅ t=8h10, completed "get up", with priority=50
⏱ t=8h10, started executing task "Get dressed and ready for the day."
   ✅ t=8h25, completed "get dressed and ready", with priority=40
⏱ t=8h25, started executing task "eat a healthy breakfast"
   ✅ t=9h05, completed "Eat a healthy breakfast.", with priority=30

🏁All the tasks have been completed in 65 minutes (1h05min), with an overall utility value of 120.
```

C. In addition to the actual scheduler, provide at least one simple example to demonstrate how:
   a. Your scheduler prioritizes tasks based on their priority value.
   b. Changing the order of the input tasks yields the same output (the order of the scheduled tasks should be uniquely determined by their priority values). Use assert statements.

D. Is the schedule that your code returns feasible? (E.g., having breakfast at 11pm might not be ideal). Do you anticipate running into trouble while test-driving this code? Briefly explain.

Let's test-drive your scheduler. 🎥📸🤳
It's time to take this schedule for a spin and explore your rotation city efficiently. Take a camera or your phone with you to take photos **or** record short clips of the critical stages of this adventure.

- Your face must be visible while performing a specific activity at the pre-determined time (for instance, if you are meant to be at a train station at 5 pm, then your face, the time on your or the station's watch and the station itself should all be visible in the image; if you arrive at the station earlier or later, this is something you should include in your report too).
- If you decide to generate a video, all the concatenated short clips should produce less than 4 minutes of total video time.[2] Upload your video to GDrive and add a link to it in your report; the settings should be such that anyone within the minerva.edu network has access.[3]

[1] If you think you will need other libraries, please check with your course instructor first, before working on your implementation, to avoid penalties on your #PythonProgramming assessment.
[2] Blending the short clips in a single video is optional. Please note that you are not expected to spend time editing the videos; video recordings are meant to document your work, not to be your primary deliverable.
[3] Here is an example of what the collection of short clips can look like. Somto Umeh (M25) suggests using Splice, Capcut or TikTok, but you are free to use whichever you prefer. Please reach out to your professor if you have any questions.

- If you decide to take photos, include them in a captioned sequence in your report.

Let's analyse your algorithm.
Supported by your test drive and the media work you generated:

A. Produce a critical analysis of the scheduler, highlighting the following:
    - the advantages of following the output it returns
    - any failure modes you have run into and any patches you added (e.g., the scheduler told you to take bus #30 at 4:30 pm in a specific location, but you missed it; you then had to jump to the next task without completing the one you should have).

   Note: The scheduler does not need to be perfect! The goal is to engage with your critical spirit and be honest about the strengths and limitations of the scheduler you have designed.
B. Examine the general efficiency of the **schedule** (the output from a given input) and include any explicit reference to the metrics you employed to determine this. You should consider evaluating how many tasks are done during the day or how much utility we can get from the tasks scheduled; however, other metrics can be more meaningful, so be as creative as you would like.
C. Perform experiments to determine and interpret the **scheduler's efficiency** (the actual program). Be creative in designing those experiments and contrast those results with the efficiency that could be derived theoretically from examining the code lines.

Getting back to the board.
Provide a high-level description (200-300 words) of how the scheduler should be improved (with an appropriate discussion on data structures; no Python implementation required). Include a word count.

Appendices.
Part I: Reflection
In < 80 words, identify a specific concept or discussion from class that clarified your problem-solving approach for this assignment. Briefly explain how it assisted you.

Part II: LO and HC applications

A. Justify the application of the core course LOs application (there are 6) in <50 words each (please include a word count and use that metric to refine your justification). Use the previous feedback on these LOs to demonstrate to the instructor how robust your applications are.
B. Identify one HC you have explicitly applied in this assignment and reflect on your application in 50-70 words (please include a word count and again use that metric to refine your justification).

Part III: Python code
All of the code you have produced for this assignment.