Reflection on problems and approach:

- Problem 1
  - Packages best dealt with by creating instances of them using classes.
  - Conditional statements to see if vouchers were valid
    - Switch statement could have potentially been used in hindsight for better scalability
  - Hurdles for solving the problem were minor, initially used discount code function using `Package.distance_in_km` & `Package.weight_in_km` etc. instead of nesting the `discount_code` function as a method within the class (& using `self`)
  - Was stuck on conditional statement for a moment as I was using `&` instead of `and` 😅
  - Notable resources:
    - https://www.w3schools.com/python/python_classes.asp
    - https://www.w3schools.com/python/python_operators.asp

Problem 2:

- Found it to be a lot more complicated
- Broke down the problem into parts, two main parts in attaining the goal of **maximising the packages in each delivery** and **returning the delivery time of each package**:
  - Selecting the best packages according to the delivery criteria
  - Dealing with the aspect of returning vehicles to estimate the delivery time of each package.
- How I approached it

- Read and tried to understand the problem, assessed further by looking at the sample inputs and outputs as well as the steps given. Did not spend too long on the steps portion of the problem initially, moved onto breaking down the delivery criteria and putting its logic into plain english.
- Started working on psuedocode/approach.

- Writing a function to select packages based on criteria:
  - Knew it could be done with a combination of loops, conditional statements and recording values to iterate through and check that the packages met certain conditions.
  - Had trouble with returning best combination, what I had would just return the first combination that it found and thus not meeting the second/third criteria.
  - Looked into python combinations. In doing so realised that it was likely not the most optimal way to solve the problem (considering big o notation, worst case scenario as the inputs grow starts to get very large). Was still not able to get function working as intended where it was able to meet all three criteria.
  - Turned to chatGPT for guidance on the specifics of the issue (meeting multiple criteria), received guidance which led to solving the problem
  - Tested the problem by running the function with the packages array and changing around package values/inserting more packages.
- Notable resources for `find_best_packages_combination()`:
  - https://www.geeksforgeeks.org/permutation-and-combination-in-python/?ref=ml_lbp
    - Also delved into various associated combination topics on sidebar
  - https://chatgpt.com/share/d29f3d55-a704-4c2d-b92a-f57f6c9e6c7c

- Trying to simulate the deliveries `run_delivery()`

- Got stuck on accounting for the van waiting time and choosing the soonest available van after both vans were out for delivery. Also initially approached the question wrong by adding the distances for delivery rather than using the max distance for the delivery (in initial pseudocode written I overlooked that they were on the same route). Recognised this by going over steps & sample inputs/outputs again while making sense of the question.
- I knew I had to:
    - Find the delivery with the maximum distance out of the selected packages combination and use that for the time unavailable calculation
    - Iterate through the packages and remove the ones that were delivered from the array being iterated over
    - Account for and record the delivery time of each package, particularly of the later deliveries taking into account time of vans prior return trips
- Account for the van being unavailable
    - First attempt involved setting a van as unavailable within the loop, removing it (-1) from the `vans_available` variable, and tallying the time using a current time counter (from steps given). Created a current time for each van and attempted using that approach but could not get it working as intended.
    - Attempted new approach of creating a van class and included the time available as a variable attached to each van object, which made more sense when writing it out as part of the approach.
        - Also had trouble turning this into actionable code that would select the soonest available vehicle to take the remaining selected packages, and then take the time available into account for the actual delivery time.
        - Used chatGPT to help with my approach, problem came together a lot quicker after understanding `min(vehicles, key=lambda v:v.available_after)`.

- Notable resources for running the van deliveries:
    - Conversation to help implement my approach:
        - https://chatgpt.com/share/d333546a-2cf1-48a8-b00a-b516ee528222
    - Understanding further:
        - https://blogboard.io/blog/knowledge/python-sorted-lambda/

- Ongoing small issues with some syntax, issues with using functions/methods with certain data types etc. Also had troubleshooting issues with displaying package info early on in the problem on, used `__repr__` to help with that. Outputs were also slightly off, assuming that the values needed to be rounded down to 2d.p.
    - https://stackoverflow.com/questions/62034663/repr-and-python-unpacking