Assignment 2

This assignment set will make you an advanced python users and assumes a good share of self-learning and troubleshooting on your own.

You need **100 points** to pass.

You are not expected to solve every problem — the assignment is structured so that students with different levels of experience can focus on the problems most relevant to their skill level.
The problems start simple and gradually increase in difficulty.[1] If you are a novice, focus on solving the first set of problems; this should be sufficient to reach the passing threshold. If you are more experienced, you are encouraged to attempt the more challenging, open-ended, creative (and more fun) problems toward the end.

Help from large language models (LLMs) or other coding assistants is not discouraged. However, you are expected to understand and be able to explain every part of your solution. At the end of the week, everyone will be asked to explain one randomly chosen problem that they have solved, to demonstrate their understanding. Direct copying from your friend, the internet, or ChatGPT is therefore highly inadvisable.

[1] In theory, at least

Contents

1 Basics of programming

2 Setting up

Question 1 — 10 points

A quick recap:
In VSCode (see the note `editor.pdf`) make a `.py`-file that prints "Hello world". Run it in the built in terminal.

Question 2 — 10 points

Using the terminal, make a folder called (for example) "DatF", where you can keep all your DafF-related files. **Optionally** make an alias, so you can open the folder by simply writing `datf`.

Question 3 — 25 points

In this exercise you will work with a small data file called `nice_data.csv`. Your goal is simply to do the following:

- load the provided data,
- compute both a Gaussian and Cauchy fit[a],
- plot the data and, the fitted lines and the residuals.

Split into function
First create you solution, making sure that it works, and then to split it into four files:

- `utilities.py`: functions for loading the data,
- `fit_tools.py`: functions for analysing and fitting the data,
- `plot_tools.py`: functions for plotting the data,
- `main.py`: the main program where you import and use function from the above files.

In `main.py` the code should be minimal, predominately importing the functions from other files (eg. `from plot_tools import plot_data, plot_fit` etc.). The other files should be largely free from specific values, having each function be a general as possible.

Keep the code simple. The purpose of the exercise is to practice splitting code into modules, importing functions, and using them to analyze a small data set.

[a]using `scipy.curve_fit`

Question 4 — 20 points

Create a folder named `my_package`.

Either:
Create the files `__init__.py, math_utils.py, string_utils.py` as described in the lecture on packages. Add some interesting functions to `math_utils.py`, `string_utils.py`. Make a program (`.py` or a notebook), that imports your new package and runs the functions.

Or:
Make some of the files from Question 4 (above) into a package (as described in the lecture on packages) and show that it works.

Question 5 — 10 points

Make a folder called `mypythonlib` and make it part of your PYTHONPATH (as explained in the Week2 lecture).

Either:
Make a python file with an arbitrary function definition and show that you can call this function from anywhere.

Or:
Add the files from Question 3, and show that your `main.py` can now be run from anywhere.

2.1 Classes

Question 6 — 20 points

Back when your instructors were young, a toy called a Tamagotchi was the latest craze. Create a class `Tamagotchi`.

- Write an `__init__` method that takes a name for your pet and stores it.
- Inside `__init__`, set two attributes:

`food = 5`, `happiness = 5`.

- Add a method `status()` that prints a short summary in a nicely formatted way.

Try running the following code:

```
friend = Tamagotchi("Andy Goldfinch")
friend.status()
```

Once your code runs without error, extend your class by adding two methods:

- `feed(amount)`: increases food by `amount`.
- `play(minutes)`: increases happiness by `minutes` decreases food by `minutes`.

Both methods must use `self` to access and modify the instance attributes. After calling the methods once, calling `status()` should show the updated values.

Question 7 — 0 points

Extend the Tamagochi with a .display() function that shows a small ASCII-visualization of your pet. Make it reactive to the current status ie. if `happiness>10` it smiles or if `food<0` it is dead :(

3 Marie Antoinette's Garden Cellular Automata

Question 8 — 15 points

In 1777, the queen of France, Marie Antoinette[a], started construction on her royal gardens. The problem is: Plants grow!

In `garden.txt` the initial plan for the garden as originally intended by Richard Mique, can be seen.

In the file, three things are visible: Path (.), an empty bed (L), or a flowering bed (#).

The following happens to every part of the garden **simultaneously**:

- **If a bed is empty** (L) and there are no flowering plants adjacent to it, the flower will bloom ($L \rightarrow \#$).
- **If a bed is occupied** (#) and four or more beds adjacent to it are also occupied, the flower does not get enough nutrients, and it dies ($\# \rightarrow L$).
- **Otherwise**, the sites state does not change.

Also, the path (.) never move and plants never sprout here.

Please show how the garden looks as it is planted and after it has reached a steady state (any further updates does not change the garden). As a sanity check, the total number of plants at this point should be **N_plants!!**.

As always, sketching a pseudocode-version of your solution might be helpful.

This is a cellular automata... (write more)

Important hint: Python can print emoji to the terminal

[a] Better known for other work

4 Robot Vacuum Cleaner

Staying in the theme of automating the boring stuff, . The robotic vacuum cleaner (roomba among friends) has become popular over recent years.

When activated, RVC moves towards the bottom right corner, cleaning one point at a time. At each timestep RVC removes all the dust at the position it occupies. The cleaning continues until all dust is removed.

For the RVC questions below, we want you to hand in a small self-contained report with plots, answers to the questions and short explanations of the problems and your solutions.

4.1 Cleaning in an ordered pattern

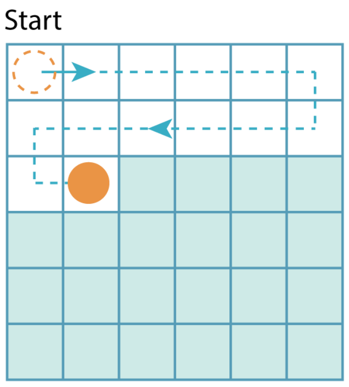

Figure 1: RVC (orange) cleaning in a predetermined ordered "snake"-like pattern, alternating between left-to-right and right-to-left movements in the rows (dashed trajectory).

We assume that the vacuum cleaner moves on a square grid, of size $N \times N$, one grid point at a time (see Fig. 1). Initially RVC is placed in the top left corner of the room and the dust is evenly distributed on the floor (with the same amount of dust d in each point).

Question 9 — 25 points

Outline your solution in pseudocode (bullet-points or diagrams).

Convert your outline into the python code.
Make a plot showing the total dust over time and show the state of the floor change (e.g. using Matlotlib's imshow). Discuss how you results agree or disagree with your expectations

4.2 Cleaning randomly

Chose a version of random:

1. Teleport to a random legal place.
2. At every time step, move in one of the local 8 directions
3. At every time step, move a constant distance (> 1) in a random direction
4. Move forward until hitting a wall, then choose a random direction (how they do in real life)

Question 10 — 25 points

Outline your solution in pseudocode (bullet-points or diagrams).

Convert your outline into the python code.

Make a plot showing the total dust over time. Make an animation (see the lecture slides) of the state of the floor over time.
Discuss how you results agree or disagree with your expectations

4.3 Adding obstacles

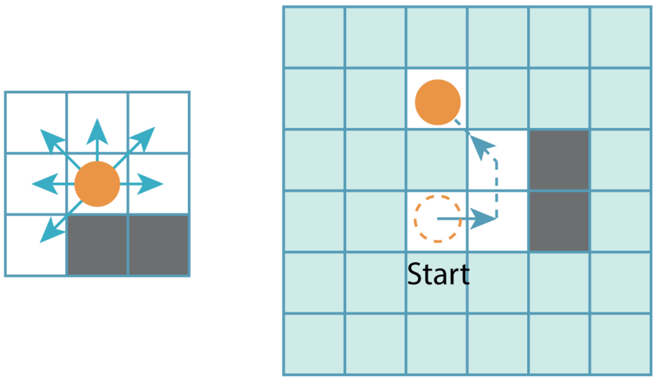

Figure 2: RVC (orange) cleaning in a predetermined ordered "snake"-like pattern, alternating between left-to-right and right-to-left movements in the rows (dashed trajectory).

In `obstacles_lvl1.txt` some obstacles are placed.

Question 11 — 35 points

Outline your solution in pseudocode (bullet-points or diagrams).

Convert your outline into the python code.

Make plots and animations.

Discuss how the random solution, the ordered solution, and your new solution compare.

4.4 Extra (15 points for each)

Question 12 — 15 points

Doing any of the following extensions to the report nets you 15 points (you can do multiple if you feel like learning more)

1. If the suction is imperfect (eg. it only removes a percentage of the dust), how is the results affected?
2. Implement another random movement type and compare
3. Make a Roomba-script that can avoid the obstacles in `obstacles_lvl2.txt`
4. Something else

End of the assignment.