INF200_H21_Ju05

June 2, 2022

1 INF200 Lecture No Ju05

1.0.1 Hans Ekkehard Plesser / NMBU

1.0.2 3 June 2022

1.1 Today's topics

- Towards implementing migration
- Visualization
  - Plotting in PyCharm
  - Plotting in Python scripts and the GIL
  - Updating data in plots
  - A time counter in a figure
  - An island map
  - RandVis: Making movies
- Exam and evaluation criteria

---

2 Towards implementing migration

Free discussion in class.

---

3 Plotting in PyCharm

- Newer versions of PyCharm have a "Scientific" mode that supports plotting inside the PyCharm window
- That did not work well for me
- To turn it off, go to `Preferences > Tools > Python Scientific` and remove the check mark for `Show plots in tool window`

4 Plotting in Python scripts and the GIL

- Parallel programs
  - A program may run internally on multiple *threads*
  - *Threads* are essentially multiple "workers" inside a single program working in parallel

  - Problem: Two workers may change the same data at the same time: Chaos!
  - Solution in CPython: The GIL
- GIL: Global Interpreter Lock
  - Present in CPython and several other Python implementations
  - In each Python program, there is *exactly one* GIL (think of it as a baton, "stafettpinne")
  - Only the thread holding the GIL can work, all others are paused
- Plotting and the GIL
  - Usually, a figure window is managed by its own thread
  - The thread can only draw to the screen while it has the GIL
  - In a loop like (`plt` is `matplotlib.pyplot`) `for alpha in range(100): plt.plot(x, sin(alpha*x))` the graphics thread may never get access to the GIL
  - We must *pause* the main thread to ensure figure updating `for alpha in range(100): plt.plot(x, sin(alpha*x)) plt.pause(1e-6)`
  - Pause duration is in seconds and can be very short
- Other (more or less) relevant plot commands
  - `plt.show()` hands over the GIL to the figure thread, the main thread is frozen until the figure window is closed; usually to be used as the last line in a script, so the user can look at the figure(s) generated
  - `plt.ion()` ("interactive on") ensures that figures are updated automatically after each `plt` command (e.g. change in axis limits); meaningful when working interactively in an IPython shell or updating in a loop while displaying "live",
  - `plt.ioff()` turns interactive mode off again,
  - `plt.draw()` updates figure; when not in interactive mode, the effect of many `plt` commands will become visible only upon `plt.draw()`
  - `FigureCanvasBase.flush_events()` is called on a figure canvas and ensures all pending "events", e.g., changes to the drawing, are executed and the figure is updated
- Detailed behavior can depend on operating system, Matplotlib backend and the exact way the Python script is run
- References:
  - Matplotlib Interactive Figures and Asynchronous Programming
  - Python Wiki on GIL
  - Wikipedia article on GIL
  - David Beazly: Understanding GIL

5 Updating data in plots

- Example program: `plotting/plot_update.py`
  - Creates list of random numbers, updates plot every time a new number is added
  - Note the slow-down when running function `replot(1000)`
  - Then run just function `update(1000)`: no slow-down
- Plotting using `Figure` and `Axes` objects
  - Plotting using `plt....()` functions is easy, but limits our possibilities
  - By using the equivalent methods on `Figure`, `Axes`, `Line`, etc objects, we get much more control
- Reason:
  - `plt.plot()` adds new coordinate system (`Artist`) to figure each time it is called

  - Matplotlib must render all `Artists` each time the plot is updated
- Solution:
  - Call `plt.plot()` once, afterwards just update the values

```
def update(n_steps):
    fig = plt.figure()
    ax = fig.add_subplot(1, 1, 1)
    ax.set_xlim(0, n_steps)
    ax.set_ylim(0, 1)

    line = ax.plot(np.arange(n_steps),
                   np.full(n_steps, np.nan), 'b-')[0]

    for n in range(n_steps):
        ydata = line.get_ydata()
        ydata[n] = np.random.random()
        line.set_ydata(ydata)
        fig.canvas.flush_events()
        plt.pause(1e-6)
```

  - Notes:
    * We start with a line having as many data points as we want to have in the end
    * We set all y-values to `np.nan` (not-a-number), a special value that is *not plotted*
    * We must set x and y axis limits before we plot
    * `line.set_ydata(ydata)` updates the data in the figure
    * The `fig.canvas.flush_events()` call ensures that everything is plotted in the figure. This can slow down the code. Without it, though, there is a risk that not everything is shown on the screen. Another approach is to make the `pause()` longer.

6 A time counter in a figure

- Example program: `plotting/time_counter.py`
  - Creates several axes for plotting (not actually used in this example)
  - Adds one axes at prescribed location, turns coordinate axis off
  - Adds text
  - Updates text in loop

```
fig = plt.figure()

# normal subplots
ax1 = fig.add_subplot(2, 3, 1)
ax2 = fig.add_subplot(2, 3, 3)
ax3 = fig.add_subplot(2, 3, 4)
ax4 = fig.add_subplot(2, 3, 5)
ax5 = fig.add_subplot(2, 3, 6)

# axes for text
axt = fig.add_axes([0.4, 0.8, 0.2, 0.2])  # llx, lly, w, h
axt.axis('off')  # turn off coordinate system
```

```
template = 'Count: {:5}'
txt = axt.text(0.5, 0.5, template.format(0),
               horizontalalignment='center',
               verticalalignment='center',
               transform=axt.transAxes)  # relative coordinates

plt.pause(1e-6)  # pause required to make figure visible

input('Press ENTER to begin counting')

for k in range(40):
    txt.set_text(template.format(k))
    plt.pause(0.1)  # longer pause so we see timer running
```

- For more fine-grained control of subplot placement, see Matplotlib's `GridSpec` (https://matplotlib.org/stable/tutorials/intermediate/gridspec.html#sphx-glr-tutorials-intermediate-gridspec-py)
- You can also place text directly into a figure; see Matplotlib Gallery for examples (https://matplotlib.org/stable/tutorials/text/text_intro.html#sphx-glr-tutorials-text-text-intro-py)

7 An island map

- Example program: `plotting/mapping.py`
  - Starts from map given as multiline string
  - Creates nested list, where each cell on island is represented by RGB color triplet
    - Three nesting levels: row - column - color component
  - Draws displays map using `imshow()`
  - Manually adds legend by adding `Rectangle` patches

---

8 RandVis: Making movies

8.1 Animations

- Matplotlib has built-in animation support
  - Useful for interactive animation on the screen
  - Can also create movies
  - See, e.g., the Matplotlib Animation Tutorial by Jake Vanderplas
- Do it yourself
  - We can do the same thing by regularly updating figures, as shown above

8.2 Making movies from animations

- Principles
  - Movies are sequences of images
  - Write all images for a movie to file first
  - Use *external application* to combine images to movie

- With `matplotlib.animation`
  - Write movie by calling `save()` on `Animation` object
  - Removes temporary files automatically when movie is ready
  - To see available movie writers (movie file formats):
    ```
    import matplotlib.animation as animation
    print(animation.writers.list())
    ```
- Manually
  - Write figures to file with `savefig()` after updating
  - Usual image format: PNG (Portable network graphics)
  - Enumerate files from 0, with leading digits:
    ```
    img_00000.png
    img_00001.png
    ```
  - Use external application to combine figures to movie

8.3 External applications to convert images to movies

8.3.1 Animated GIF

- Very robust (plays anywhere), but file may be *very* large
- Use ImageMagick
  - For OSX and Win, you can download installation package
  - For OSX, you can also install using `brew install ImageMagick`
  - Package also supports all kinds of image file format transformations and image manipulations

8.3.2 Video formats such as MP4, AVI, ...

- Typically much smaller files
- High levels of compression may lead to artefacts
- Not as robust, if you choose unsuitable parameters, the resulting file may not play on some devices or player programs
- Use FFMPEG
  - Should be available in your `inf200` conda environment
  - To check if `ffmpeg` is available on your computer
    1. Start `Terminal` in PyCharm
    2. Run `conda activate inf200` if necessary
    3. Run `ffmpeg -version`
    4. If version information is printed, all is well
  - If `ffmpeg` is not yet available, install it using `conda install ffmpeg` while your `inf200` environment is active
- For a versatile player, see VLC

8.3.3 Example of simulation with manual movie production

- `RandVis` package under `examples` in course repository

8.3.4 DO NOT COMMIT IMAGE OR MOVIE FILES TO YOUR REPOSITORY!!!

- Files will take a lot of place
- Re-running your simulations will create different files even if the simulations are the same

- Files are compressed binary files and cannot be compared/diff'ed meaningfully
  - If you commit images over and over, all images will collect in your repo
  - Requires a lot of space
- **Make sure Git ignores directories with graphics!**

---

9 Exam information

9.0.1 Time and Place

- **Monday 20 June and Tuesday 21 June 2022**
- The detailed schedule will be posted later
- Exams will be held at REALTEK in room TF1-305 (Zeta)

9.0.2 Format

- Both students in a group are examined together
- Short presentation of your results (**five minutes**)
  - You may use **one** PDF file (a few slides) and **one** short animation (MP4 or GIF)
  - These must be submitted by **Friday, 17 June, 15.00 CEST**
- Discussion with examiners (20 minutes)
  - Your code will available on screen during discussion
- There will be two separate exam commissions working in parallel
- You can speak English or Norwegian

9.0.3 Content of presentation/discussion

- You are a software developer presenting your product to your client
- Explain to the client/examiner how you have solved the task
  - Overall structure of the code
  - Examples of how you solved specific aspects/problems
- Persuade the client/examiner that your code is trustworthy
  - How did you ensure quality?
  - Is code in maintainable shape (tests/documentation etc)
  - Do you as a developer know your stuff?
- Pursuade the client that your code is productive
  - Ease of use. Is it easy to use and manipulate the code you have written?
  - Performance: Is the code fast?
- Show some interesting results
- Remember, 5 minutes is not very long!
  - Choose in advance what you want to spend your time on
  - Do not make too many or too verbose slides

9.1 Evaluation criteria

9.1.1 Criteria for code

Weighting

Code (70%)

- Delivery 20%
- Quality 20%
- Tests 15%
- Documentation 10%
- Bonus 5%

Exam (30%)

- Presentation 10%
- Discussion 20%

Evaluation

- Generally, joint evaluation for code, individual evaluation for exam
- Deviation possible if there is clear indication that partners differ very strongly in contributions and competence
- In each category, A-F logic is applied:
  - On a 0-100 point scale, "just passing" is usually at 40% (lower end of E)
  - Thus, e.g., for *Delivery*, a "just passing" code will give 40% of 20%, i.e., 8% to the total, while a perfect delivery would give 20% to the total.
- General definition of grades (https://www.uhr.no/_f/p1/i4bfb251a-5e7c-4e34-916b-85478c61a800/karaktersystemet_generelle_kvalitative_beskrivelser.pdf):
  - A: An excellent performance, clearly outstanding. The candidate demonstrates excellent judgement and a very high degree of independent thinking.
  - B: A very good performance. The candidate demonstrates sound judgement and a high degree of independent thinking.
  - C: A good performance in most areas. The candidate demonstrates a reasonable degree of judgement and independent thinking in the most important areas.
  - D: A satisfactory performance, but with significant shortcomings. The candidate demonstrates a limited degree of judgement and independent thinking.
  - E: A performance that meets the minimum criteria, but no more. The candidate demonstrates a very limited degree of judgement and independent thinking.

Requirements in different categories

Delivery

- Does your code deliver on all requirements described in Sec. 1–3 of the task description?
- Full score (20%) requires that all is in place and works.
- Minimum requirement to pass is herbivores, carnivores, migration on an island with one landscape type, plotting of total number of animals of each species, and ability to run via the BioSim class (eg `check_sim`, `test_biosim_interface`)

Quality

- Is your code well written?
  - Sensible organisation of code into classes?
  - Well-chosen class and variable names?
  - Is the encapsulation principle in object-oriented programming observed?
  - Do you follow PEP8 rules (but with lines up to 100 characters allowed)?
- Are there any bugs?
- Is your code well readable?
- Is your code reasonably efficient?

Tests

- Do you have unit tests for all methodes in classes for animals and landscape types?
- Do you have tests for the island and simulation classes?
- You do **not** need to write tests for visualisation.
- Are your tests meaningful and sufficiently comprehensive?
- Are the tests well written?
- Do you use some advanced techniques, such as parameterization, fixtures and statistical tests?
- Full score requires comprehensive tests including good use of parameterization, fixtures and statistical tests.

Documentation

- Do you have meaningfull docstrings for all classes and methods?
- Are docstrings reasonably formatted?
- Do you have documentation text describing the parts of BioSim and how to use them?
- For a "B"-level score, you must also use at least one advanced Sphinx features, e.g., example code, note boxes, mathematics, or figures
- For an "A"-level score, you must use several advanced Sphinx features including math and figures.

Bonus

- In this category, you can earn up to 5% extra for features beyond the requirements of Sec 1–3 or extraordinarily well-written code.

9.1.2 Criteria for presentation

- Your ability to present and explain your code
  - Overall structure
  - What solutions you chose and why
  - Why your code is trustworthy
- Your ability to discuss your code
  - Do you understand your own code?
  - Are you aware of its limitations?
  - Can you discuss improvements or extensions to your code?