# Università degli Studi di Pavia

**Bachelor degree in Artificial Intelligence**


## Planisuss — Final exam project — A.Y. 2022/23


**Stefano Ferrari**


**Computer programming, Algorithms and Data str., Mod. 1**

# Contents

date: February 7, 2023
version: 0.95

The final exam project consists in the design and implementation of a simulation of a fictitious world called "Planisuss", freely inspired by Wa-Tor [1][2] and Conway's Game of Life [3][4]. The simulation is intended to provide data to be interactively visualized using the library `matplotlib` [5] (see Sect. 7.2). In order to allow some flexibility in the design, the specifications below described will refer to the simulation parameters through suitable constants, summarized in Sect. 6. The constants are represented using in uppercase typewriter font (e.g., `CONST`).

The Planisuss world is constituted of a single continent which is populated by three species: Vegetob, Erbast, and Carviz. In the simulation, several individuals of the species interact evolving their population under the rules explained in the next sections.

# 1. The Planisuss world

Planisuss is regularly structured in geographical units called *cells*. Cells are organized in a regular grid structure and their position can be identified by a bidimensional coordinate.

The size of Planisuss is `NUMCELLS` × `NUMCELLS` cells.

The cells can be occupied by water or ground. Cells on the boundary of the grid are always occupied by water. All the other cells are suitably assigned either to water or ground at the beginning of the simulation.

Each ground cell can host individuals of the three species, while water cells are uninhabitable. A suitable procedure initializes the content of the cells at the beginning of the simulation.

All the Erbast in a cell constitute a herd. Similarly, all the Carviz in a cell constitute a pride.

The basic events on Planisusshappen in discrete time units called *days*. For practical purposes, the simulation can be terminated after a predefined number of days, `NUMDAYS`.

# 2. The ecosystem

Three species populates Planisuss:

**Vegetob** (*pl.* Vegetob) is a vegetable species. Spontaneously grows on the ground with a regular cycle. Vegetob is the nutrient of Erbast.

**Erbast** (*pl.* Erbast) is a herbivore species. Erbast eat Vegetob. They can move on the continent to find better living conditions. Individuals can group together, forming a *herd*.

**Carviz** (*pl.* Carviz) is a carnivore species. Carviz predate Erbast. They can move on the continent to find better living conditions. Individuals can group together, forming a *pride*.

Erbast and Carviz are animal species.

## 2.1. Vegetob

Vegetob are characterized by their density in a cell. The density can have a value between 0 and 100.

## 2.2. Erbast

Erbast are characterized by the following properties:

- *Energy*: represents the strength of the individual. It is consumed for the activities (movement and fight) and can be increased by grazing. When the energy value reaches 0, the Erbast dies.
- *Lifetime*: duration of the life of the Erbast expressed in days. Its value is set at the birth and does not change.
- *Age*: number of days from birth. When the age reaches the lifetime values, the individual terminates its existence.
- *Social attitude*: measures the likelihood of an individual to join to a herd. It is represented by a value in [0, 1].

Erbast have knowledge about the status of the cells around their position up to a distance of `NEIGHBORHOOD` cells. Erbast in the same cell form a herd.

## 2.3. Carviz

Carviz are characterized by the following properties:

- *Energy*: represents the strength of the individual. It is consumed for the activities (movement and fight) and can be increased by hunting. When the energy value reaches 0, the Carviz dies.
- *Lifetime*: duration of the life of the Carviz expressed in days. Its value is set at the birth and does not change.
- *Age*: number of days from birth. When the age reaches the lifetime values, the individual terminates its existence.
- *Social attitude*: measures the likelihood of an individual to join to a pride. It is represented by a value in [0, 1].

Carviz have knowledge about the status of the cells around their position up to a distance of `NEIGHBORHOOD` cells. Carviz in the same cell form a pride.

## 2.4. Ecosystem limits

Some design choices must be made to proper implementing the simulation:

**Cell capacity** While the Vegetob density is specified as number in [0, 100], its nature as integer or real number can be questioned. The simplest choice would be using an integer to represent the density, but this limits the minimum change in this property to be equal to 1.

Another possible limitation is the number of individuals constituting a herd or a pride. This has the practical motivation in limiting the computational complexity and the modelling motivation that only a limited number of individuals can crowd a limited area. However, if not set, other factors related to the dynamics of the system can indirectly limit the population numerosity. In the following, these limits will be referred as `MAX_HERD` and

`MAX_PRIDE` for the herd and pride, respectively. In case these constants are not used or defined, they can be interpreted as an indefinitely large value. In Python, the maximum number of elements of a **`list`** depends on the architecture of the used platform, but they are really quite large for the purposes of this project (in the order of $10^9$ for 32-bit and $10^{18}$ for 64-bit systems).

**Energy limit** A limitation of the Energy is not explicitly needed (and integer can be proper), but can be used to avoid a divergent trajectory of the species, which may involve the extinction of one of the two animal species. It will be referred as `MAX_ENERGY`.

**Lifetime limit** Similarly to the Energy, the lack of limitation of the Lifespan value can involve a world with a strange dynamics. It will be referred as `MAX_LIFE`.

# 3. A day on Planisuss

The time on Planisussis structured in units called *day*. A day is articulated in the following phases:

**Growing** Vegetob grow everywhere of a fixed quantity (`GROWING`).

**Movement** The individuals of animal species (Erbast and Carviz) decide if move in another area. Movement is articulated as individual and social group (herd or pride) movement.

**Grazing** Erbast which did not move, can graze the Vegetob in the area.

**Struggle** Carviz insisting on the same area can fight or hunt.

**Spawning** Individuals of animal species can generate their offspring.

Conventionally, long periods of time on Planisuss are measured in months, years, decades and centuries, where a month is 10 days long, a year is 10 months long, a decade is 10 years long, and a century is 10 decades long.

## 3.1. Growing

The Vegetob density is increased by `GROWING`. If a cell is completely surrounded by cells having the maximum Vegetob density, the animals present in the cell are overwhelmed by the Vegetob and terminated.

## 3.2. Movement

In the Movement phase, individuals and social groups evaluate the possibility to move in another cell. Based on suitable rules, the most appealing cell in the neighborhood is identified. The evaluation is carried on first by social group basis (herd and pride).

All the Erbast in a cell at the beginning of the day form a herd. Similarly, all the Carviz in a cell constitute a pride. Social groups can have memories and strategies: proper values can be stored and used in the evaluation and planning. For instance, they can be provided with the coordinate of the last cell visited to avoid to move back, or the density of nutrients of cells visible in previous days.

Once the herd and the pride of the cell made a decision (stay or move), the individuals can choose if they will follow the social group or made a different decision, splitting by the social group.

Splitting decision can be formed considering the properties of the individual (e.g., the herd will move, while the individual having a low value of energy may stay; or, on the contrary, the herd will stay, but strong Erbast may want to move, due to the lack of Vegetob in the cell) and is weighted with the social attitude of the individual.

Movements take place for all the cells at the same time and are instantaneous. The movement costs to each individual one point of Energy.

## 3.3. Grazing

The Erbast that did not move, can graze to increment their Energy. The grazing decreases the Vegetob density of the cell. Every Erbast can have 1 point of Energy for 1 point of Vegetob density. If the Vegetob density is lower than the number of Erbast, 1 point is assigned to those Erbast having the lowest value of Energy, up to exhaustion of the Vegetob of the cell.

Optionally, the Social attitude of those individuals that did not eat (due to lack of Vegetob) can be decreased.

## 3.4. Struggle

After the movements, the social groups in the cells may need to be reorganized. In fact, different social groups of the same species can reach the same cell.

Erbast in the cell joint to the same herd. If two or more herds moves in the same cell, the herds are fused (and their memories can be joined).

If more than one Carviz pride reach the cell, they evaluate the joining in a single pride. The evaluation is made on pride-basis, using the social attitude of their members. If one of the prides decide not to join, a fight takes place. In case of more than two prides reaching the same cell, the above procedure is applied iteratively to pairs of prides (i.e., starting from those with less individuals). The prides that decided to join can form the single pride before starting the fight.

### Fight

The fight between two prides is a last-blood match. In its simplest form, a random number is drawn and each pride has a winning probability proportional to the sum of the Energy on its components.

A more complex scheme may consider one-to-one matches between the champions of the pride (those having the higher Energy), until one of the prides has no more components.

Optionally, the Social attitude of the winning pride components can be increased.

### Hunt

When only one pride is present in the cell, a hunt takes place. The pride identifies the stronger Erbast in the cell and combat with him. Similarly to the fight, several scheme can be adopted:

- No combat: The prey is always took down. No Energy consumption for the pride.
- Single assault: The pride has only one opportunity to take down the prey. A random number is drawn and the probability of success depends on the relative value of the cumulative Energy of the pride and the Energy of the prey.

- Last blood: The single assault scheme is repeated, but every fruitless assault costs some Energy to the pride (e.g., one Energy from a random individual).

In case of success, the Energy of the prey is shared by the pride individuals, increasing their energy value (spare energy points are assigned to the Carviz with the lowest Energy).

Optionally, the events can change the Social attitude of the individuals. For instance, the Social attitude of the winning pride components and the surviving herd components can be increased. Alternatively, if no prey is hunt (due to absence of preys or to failure of the hunt) the Social attitude of the pride individuals can decrease.

### 3.5. Spawning

The Spawning is the last phase of the day. The Age value of the animals are increased by one day.

Those that reach an age multiple of 10 (one month) have their Energy decreased by `AGING`.

Those that reach their Lifetime are terminated by spawning two offsprings. The offsping properties are set with the following rules:

- Age: set to 0.
- Energy: the sum of the Energy of the offsprings is equal to the Energy of the parent.
- Other properties: the average of the properties of the offsprings are equal to the corresponding properties of the parent.
  - Hence, the sum of the property values of the offsprings is the double of the corresponding property of the parent.

In case `MAX_HERD` and `MAX_PRIDE` are defined, the spawning is allowed only if the social group can be enlarged to include the offsprings. Any rule can be used to decide which individuals can spawn.

## 4. Data visualization

The status and the dynamics of Planisuss can be suitably represented in the user interface of the simulation program. While global properties (such as, for instance, the population of a species) can be represented as a single number (at a given time or on the average), more elaborated visual representations should be used to display trends and local properties on larger scale. For instance, the population vs. time can be represented as a time plot graph, while a pseudocolors map can be used to provide an overall representation of the instantaneous population of Planisuss.

The visualization may be interactive, allowing the user to operate on the simulation speed (basically it could be the ability to reset/pause/restart the simulation), to change the displayed properties, to change the parameters of the elements in the simulation, and to limit the visualization to a given portion of the graph (e.g., zooming on a map or a plot).

The `matplotlib` library provides the functionalities to display data and to build an interactive interface [5][6].

Among others the following visualization graphs can be implemented:

- Overall map: it allows to represent graphically the current status of the simulated world by mapping the parameters of interest. Optionally, zooming on an area can be also provided.

- Population plots: properties of the inhabitants (e.g., average or cumulative Energy) can be plotted versus the simulation time to show trends. Also, the plots may show the relationship of a pair of properties (e.g., the numerosity of Erbast wrt. Carviz).
- Trajectories: shows the position of the populations on the map the wrt. time.
- Cell inspection: the properties of the inhabitants of a given cell can be represented in detail. Optionally the events occurring in the different phases of the day may be depicted.

Besides realtime visualization, replay of an already runned simulation can be provided. This would require the logging of the populations properties of interest, such as numerosity or average Energy.

## 4.1. Maps

The most suitable representation of the global status of Planisuss is probably by means of a map. Since Planisuss has a planar topology and its surface is structured as a regular square grid, the map can be realized as a digital image, where each pixel corresponds to a cell of the world grid.

The set of values that a single property of a cell can assume can then be mapped on a color set to be visualized: the property of each cell is codified as a color and this color is assigned to the corresponding pixel of the map image. For instance, the density of the Vegetob can range between 0 and 100. Hence, a suitable function to map an integer number in $[0, 100]$ to a color can be used to represent the Vegetob density as a `NUMCELLS` $\times$ `NUMCELLS` image.

In color images, the color of a pixel can be described using the combination of the three primary colors (red, green, and blue). The shades of the primary colors are usually represented as values in $[0, 1]$ (being 0 the absence of the color and 1 its maximum intensity), or as integers in $\{0, \ldots, 255\}$ (being 0 the absence of the color and 255 its maximum intensity). The matrix containing the all the coefficients of a given primary color is called a channel. Hence, a color image can be represented by three matrices that are the red (R), green (G), and blue (B) channels.

For instance, the mapping for representing the Vegetob density is then a function $[0, 100] \rightarrow [0, 1]^3$.

A useful map can show the numerosity of the population using one channel for each species. For example:

- R: Carviz
- G: Erbast
- B: water/Vegetob density

Two problems need to be addressed: the mapping of the population of Carviz and Erbast in $[0, 1]$ and the mapping of two features (type of the cell as water or ground, and the Vegetob density).

For the first mapping, if `MAX_HERD` and `MAX_PRIDE` have been defined, the function can simply map 0 in 0 and `MAX_HERD` (or `MAX_PRIDE`) to 1 (hence a normalization with respect to the maximum value). If the constants are set to a very high value, the mapping can be non-linear (e.g., logarithmic). Otherwise, if `MAX_HERD` and `MAX_PRIDE` have not been defined (or if its value is very high with respect to the current situation), the scaling can be done with respect to the maximum number of elements of the corresponding social group at the given time.

The mapping to mix two features (that are mutually exclusive) can be done using the first half of the channel spectrum for the water and the second for the Vegetob density. Since the population

in water cells are always zero, this mapping will assign zero to all the channels of water cells, representing them as black pixels. A ground cell having zero individuals of any species would be represented as a mid-blue pixel.

An alternative visualization can consist in several single-feature maps.

Also, if the grid size is small with respect to the resolution of the screen (or the portion of the screen dedicated to the map), the cell can be represented as a group of pixels, where each element of the group represents a single feature.

Details on how a map can be realized and visualized using `matplotlib` can be found in [6].

## 4.2. Plots

While a map can represent a geographical distribution of the properties of the cells (property value vs. the cell position), a plot can represent the relationship between two entities. The simplest plot family represents a property value vs. the time. For instance, the overall numerosity of a species wrt the simulation time (e.g., measured in simulated days). Different graphical elements (like the color, the thickness, or the style of a line) can be used to distinguish different properties on the same plot.

Another way to compare and relate two properties is plotting one vs. the other. For instance, the numerosity of two species can be plotted on a Cartesian axis, where the values of these properties at the same simulation time are the coordinates of the points on the graph. The line joining the points in the time order provides a representation of the trajectory of the simulation in the domain of the properties.

Three-dimensional plots can also be considered.

## 4.3. Trajectories

The movements of some social groups or individual of interests can be tracked and shown on the map. In this case, the information does not consist in the content of the cells, but in time and position of the element of interest. Hence, the color of the cells of the map can just reflect if the cell is water or ground. Instead, additional information (e.g., the numerosity of a herd) can be added using the line appearance (e.g., color or thickness).

## 4.4. Cells details

The events happening in a single cell can be represented in several forms. Besides the content of a cell at a single instant, other information could be of interest. For instance, the average population of the species or the relative frequency with which the cell remains unpopulated. These quantities can be represented as time plots.

Optionally, the description can have a finer resolution, by involving the day phase events.

## 4.5. Logs and replay

Some information generated during the simulation can be saved to be re-processed later. The usage of the logged values may be of different nature.

The first is simply the replay of the evolution of the simulated word. Different level of detail can be considered for this goal. For instance, at the least level of detail, the overall population can

be considered. This information can be represented as time plots or property vs. property plots, as described in Section 4.2.

Higher level of detail logging can consider grid-level information, storing also the geographical positions and the content of each cell.

The full logging of the properties of the simulated world allows a full replay of the simulation as well as the evaluation of alternative evolutions. Another possible usage of the logged data may be reconstruct the life of individual of interest (e.g., the strongest Carviz, or the the one generating the largest family tree).

The logging for archival and replay purposes can be different than the logging for in-simulation inspection or visualization. The former can be obtained from the latter by discarding some information.

### 4.6. Simulation controls

The execution of the simulation may be realized allowing the user a minimal interaction.

Basically, the user can start, pause, and terminate a simulation. Other useful interactions may be slowing down or speeding up the simulation.

Also, in case the simulation are tuned to run for a long time, a *Save/Load* functionality should be provided. In this case, all the information characterizing the status of the simulation must be stored and a procedure to recall them and assign the proper value to the program variables has to be devised. No particular format is required in this specification, and any library (such as `pickle`) can be used for this purpose.

Additionally, the user may be allowed to change some simulation parameters, from a global parameter to the content of a single cell.

## 5. Notes

### 5.1. Differences with Game of Life and Wa-Tor

This project consists in the design and implementation of a simulation freely inspired by Wa-Tor [1][2] and Conway's Game of Life [3][4]. A short comparison with these two simulated world may help. In both of them, the world is represented as a square grid and in each cell only one individual (if any) may exist.

**Game of Life** Each cell can be populated or unpopulated. At any time step, a cell is evaluated and, based on its status and its neighbors status, the status (populated/unpopulated) can change.

**Wa-Tor** The world has the topology of a torus and two species: shark and fish. At each time step, an individual can randomly move in a neighboring cell, provided that it is unpopulated. A shark can also move in a cell populated by a fish: in this case, the fish is removed from the simulation and the shark gains a certain quantity of energy. This property control the shark population: each time step, the energy is decreased and a shark dies when its energy reaches zero. After a given number of time steps, the individuals may reproduce. It happens by generating a new individual in an empty neighboring cell, resetting the counter for the reproduction.

The Planisuss's rules are hence an extension of the Wa-Tor's rules. Planisuss differs from Wa-Tor by the complexity of the rules (in Wa-Tor they are either deterministic or based on randomness, while Planisuss allows the use of some goodness function), the structure of the ecosystem, and the condivision of the grid elements. In fact, the main difference is that in Planisuss a single cell may host more than one individual, also of different species. Also, in Planisuss three species are present, although one (Vegetob) has the role of nutrient for the prey species (Erbast) and its dynamic is very simple (like the cells in Game of Life).

Another difference is the presence of social groups that allow to simplify some of the operations by deciding some collective behavior for a large number of individuals instead of computing the (complex) decision procedure for each of them.

Also, predators (Carviz) can clash for the dominance of the same territory.

## 5.2. Computational complexity

The present specification allows great flexibility in the definition of the procedures driving the evolution of the simulated ecosystem. This choice is motivated by the possibility to have implemented variegated spectrum of dynamics and to allow anyone to set the more proper level of detail in the simulation.

Several factors influence the computational time required by the simulation:

**Time span** The duration of the simulation can be measured by the number of simulated days it lasts. The computational complexity of the simulation is directly proportional to the number of simulated days, `NUMDAYS`.

**Grid size** Each time unit, several procedures have to be evaluated for each grid cell. Hence the computational time is directly proportional to the grid size, $\texttt{NUMCELLS}^2$.

**Decision policy** In every cell, at most one herd and one pride are present. Every social group must take some decision during the day. The first one is the movement. After that, each individual has to decide whether follow the social group or operate a different decision. The complexity of the decision policy for the social group can be relatively high, but it has a mild impact on the total complexity (it occurs only for $\texttt{NUMCELLS}^2$ times). The complexity of the individual decision, instead may have a higher impact: it should not be made as exceedingly high.

**Struggle** The struggle phase may have some impact in the complexity of the single cell, but probably it represents a relatively rare event. Hence, its impact on the total computational cost should be mild.

The computational cost of the simulation is then proportional to $\texttt{NUMDAYS}\times\texttt{NUMCELLS}^2$. For a $1\,000 \times 1\,000$ grid and for 100 years (i.e., $10^4$ days) simulation, the computational cost is about $10^{10}$ times the cost of the decision policy. Considering that the current CPU has a clock speed in the order of $10^9$ operation per second, each operation in the decision policy may add tenth of seconds in the physical duration of the simulation.

Hence, simulations on larger grids or for a larger number of years may require a substantial amount of machine time.

A compromise between the size of the grid, the duration of the simulation, and the complexity of the daily procedure (i.e., the smartness of the species) may be necessary.

### 5.3. Social groups

The collective behavior is a factor that allowed the surviving and the evolution of species (and also on Planisuss can help weak individuals), but it is a feature that allows to run a simulation with a large number of individuals.

This effect is more evident if the Social attitude property is large. In this case, once of the social group decision about the movement has been made, if most of the individuals will follow the group, the computational cost for the individual decision is negligible.

In order to keep low the computational cost of the simulation, the individual decision procedures should be kept as simple as possible. On the other hand, the social group decisions may use more computational resources to made the group decision smarter. A simple technique can be storing the coordinate of the last visited cells, in order to avoid to move back in a territory that could have been already exploited. The stored cells may cover regions that do not belong to the explorable neighborhood of `NEIGHBORHOOD` radius.

### 5.4. Variants

The specification of the dynamics of Planisuss are intentionally flexible to allow anyone to experiment the preferred characterization and degree of complexity.

More complex scenarios may include geological or climatic phenomena. For instance, a tidal cycle can be devised, which periodically allows coastal cells to become water, and viceversa.

Another option, to allow a gaming user interaction, is to provide the user of "God-mode" interaction tools, which strongly change the evolution of the populations. For instance, a period of drought can stop the growing of Erbast (in given regions or globally), meteors can destroy the life in a region, or on the contrary, in a region a given species can have some advantage (e.g., increase Lifetime).

### 5.5. Development suggestions

The simulation of Planisuss has the purpose of providing data to the interactive data visualization, using the library `matplotlib` [5] (see Sect. 7.2).

A complex set of rules for the evolution of the ecosystem may provide a rich and interesting scenario for the visualization, but its development should not shade the focus, that is the data visualization that make observable what is going on the simulated world.

In order to get familiar with `matplotlib` (and the problem of simulation), Game of Life and Wa-Tor can be used in the first development rounds. Once the visualization tools have been tuned, the complexity of Planisuss can be implemented.

Similarly, the simplest options described in the present document can be considered (e.g., randomness instead of smartness), but structuring the classes with attributes and methods needed for more complex functionalities.

## 6. Constants

In this section, some parameters that characterize the simulation are reported as constants. Their values should be chosen to customize the dynamics of the designed simulation. Referring to them in the code allows to modify easily the overall dynamics of the simulation. A proposal for an implementation of these parameters is further described in Sect. 7.3.

`NUMCELLS`: Size of matrix representing the Planisuss map. It is the number of elements in each row and each column. A possible variant may consider a rectangular map, represented as a `NUMCELLS_R` $\times$ `NUMCELLS_C` matrix.

`NEIGHBORHOOD`: The radius of the region that a social group can evaluate to decide the movement. Its value could be different for Erbast and Carviz (`NEIGHBORHOOD_E` and `NEIGHBORHOOD_C`, respectively). Its value may impact the cost of the movement decision.

`MAX_HERD` and `MAX_PRIDE`: The maximum numerosity of the social groups. These values can be undefined and not used in the program. If used, they should be interpreted as a soft-max value. For instance, if the herds move in the same cell, they merge together, and their number may exceeds `MAX_HERD`. This should be allowed, and the herd splitting can be forced in the next day.

`NUMDAYS`: Length of the simulation. Although this value is not required (the termination could be directly controlled by the user), it can be set in order to easy the simulation management and the plotting.

`MAX_ENERGY` and `MAX_LIFE`: Respectively the maximum value of Energy and Lifetime. They are not required, but a proper value can avoid simulation divergency. In fact, a large value of Lifetime allows an individual to collect high value of Energy and allows the generation of stronger and more long-lived one of its offspring, strengthen this property in the next generations. The two specie may have different value for these constants (`MAX_ENERGY_E` and `MAX_LIFE_E` for Erbast, and (`MAX_ENERGY_C` and `MAX_LIFE_C` for Carviz).

`AGING`: The quantity of energy lost each month. It is not required (it can be zero) and can be different for the two species (`AGING_E` and `AGING_C` for Erbast and Carviz, respectively).

`GROWING`: The quantity of Vegetob density that grows in every cell at the beginning of the day.

# 7. Resources

## 7.1. The `matplotlib` library

`matplotlib` is a free and open-source cross-platform library for interactive data visualization using Python [5].

A short tutorial on its use (with code examples) can be found in [6].

## 7.2. The `numpy` library

`numpy` is a library for scientific computing in Python [7]. It provides the data structures for the data visualization in `matplotlib`.

A short tutorial on its use (with code examples) can be found in [6].

## 7.3. Constants

The value of the constants described in Sect. 6 should be chosen accordingly to several factors. A possible implementation is proposed in [8].

Every morning in Planisuss, a Erbast wakes up. It knows it must outrun the fastest Carviz or it will be killed.

Every morning in Planisuss, a Carviz wakes up. It knows it must run faster than the slowest Erbast, or it will starve.

It doesn't matter whether you're the Carviz or a Erbast—when the sun comes up, you'd better be running.

Unless you are a Vegetob.

---

Planisuss proverb

# References


[1] A. K. Dewdney, "Computer recreations: Sharks and fish wage an ecological war on the toroidal planet wa-tor," *Scientific American*, vol. 251, pp. 14–22, December 1984.

[2] "Wa-tor." `https://en.wikipedia.org/wiki/Wa-Tor`.

[3] M. Gardner, "The fantastic combinations of John Conway's new solitaire game "life"," *Scientific American*, vol. 223, pp. 120–123, October 1970.

[4] "Conway's game of life." `https://en.wikipedia.org/wiki/Conway%27s_Game_of_Life`.

[5] "Matplotlib website." `https://matplotlib.org/`.

[6] S. Ferrari, "Numpy and matplotlib minimal tutorial." `https://elearning.unipv.it/mod/folder/view.php?id=67083`, 2023.

[7] "NumPy website." `https://numpy.org/`.

[8] "Constants for planisuss." `https://elearning.unipv.it/pluginfile.php/269900/mod_folder/content/0/planisuss_constants.py`, 2023.