Projet Informatique

IN104

Natalia Díaz Rodríguez, ENSTA ParisTech

Logistics

- Moodle Course page includes material, course and project defense date
- Project preferences: Make 3 choices (in decreasing order of preference)
  - Deadline to choose one of the 8 topics: 21 March midnight (1st come, 1st served)
  - Beware: may not be possible to accommodate everyone; we need balanced groups, respond timely today!

Logistics (II)

- IN104 must help you learn manage your autonomy and independence
  - Essential today for an engineer
  - 1 ECTS = 25h of work (all included) -> Expect Min. 2h of extra work at home required
- What are normal grades from past years?
  - No Exam -> Everything between 10 -12 and 20.
  - Up to you to get beyond average
    - A self-learning project
    - Internet is to be used

Logistics (III): Evaluation

1. 25% **Source code**: features, tests, code documentation, etc. Source code must be documented, the report including a link to the repository must be in a .zip file. The repository will contain the sources, as well as a plain text file README.md which indicates the actual operational features and limitations. Python code should compile, the teaching assistants are not supposed to make significant corrections for it to compile: a code with few features, but that compiles and does not crash will be preferred to a more complete code but which is not directly operational!
2. 25% **Defense** (10 minutes presentation, 5 min. questions): The formal quality of the presentation will be an important element. The defense needs to include a demonstration on the basis of the source code, an analysis of the difficulties encountered and implemented solutions. It will not include a presentation of the problem or the method of resolution that the teaching assistant obviously knows already well. The defense is open to everyone (subject to the acceptance of the pair that will present). The chronological order of defense will be given by the list of each group.
3. 20% **Practice Analysis**: You will return a critical (max. 5 pages) report before the defense in which you analyze and criticize the progress of your project and its success and failure factors. This evaluation component also includes the oral treatment of this question during the defense.
4. 30% **Continuous Progress Evaluation** of the practical work (mid-term evaluation) during the practical lab sessions (based on git commits).

Timetable

1. 19/3/2019 14.45-18.00: Projects presentations (5 min each) and Intro to GIT practical
2. 25/3/2019 08.30-11.45: Object Oriented Programming (OOP) and Python. Project 0: Unix/Python/ OOP Tutorial: Submit via Gradescope. Deadline: 1 week (see exact time time in Gradescope)
3. 26/3/2019: 14.45-18.00 Test Driven Development (TDD). TDD Practical.
4. 2/4/2019: 14.45-18.00 Work on Project week
5. 9/4/2019: 14.45-18.00 Work on Project week
6. 16/4/2019: 14.45-18.00 Work on Project week
7. 23/4/2019: 14.45-18.00 Work on Project week. [TD-Dependent] Mid-term progress report: Deadline: 22/4/2019
8. 7/5/2019: 14.45-18.00 Work on Project week
9. 22/5/2019: 8.30-11.45 Work on Project and Report (Autonomous week without TD supervision): Deadline to hand in report containing link to project repository: 23 May 23.59.
10. 27/5/2019: 8.30-11.45 Project Defenses (15min per team: 10 min +5 questions) (Soutenances)

Course expectations

- This is your (first) independent project course
  - i.e., autonomous work expected
  - Time in lab is not expected to be sufficient to hand in a high quality project and report
  - Expected total work hours: at the very least, same amount of lab time, at home
- **Report: 1 single PDF containing link to repository**:
  - Quality over Quantity
  - Results (plots, insights, conclusions and learnings):

more important than lengthy reports explaining the problem

If you feel lost

- Establish milestones with your partner
    - Do GIT code reviews (of each other, if needed)
- Force yourself: pose the right question you want to answer
- The TA is there to support you!

If you feel lost

Mid-term report hand in:

- Use the class to show to your TA, min. first 50% of your project report in the equator of the course
    - makes sure you are on track
    - can help you not derail. You can include:
        - Project Requirements
        - Software Architecture
        - Classes, Tests, Plots, Versions, Documentation...

Course & Team Python Conventions

PEP-8: Defines Python coding practices https://www.python.org/dev/peps/pep-0008

- **Indentation**: 4 spaces (good editors will replace <TAB> by 4 spaces)
- **Variables**: `lower_case_with_underscores`
- **Functions**: `lower_case_with_underscores()`
- **Classes**: `UpperCamelCase`
- **Attributes**: `lower_case_with_underscores`
- **Protected attributes**: `_prefixed_with_1_underscore`
- **Constants**: `ALL_CAPS`
- **Modules**: `lowercase` (single word)

PEP-257: Documentation conventions.

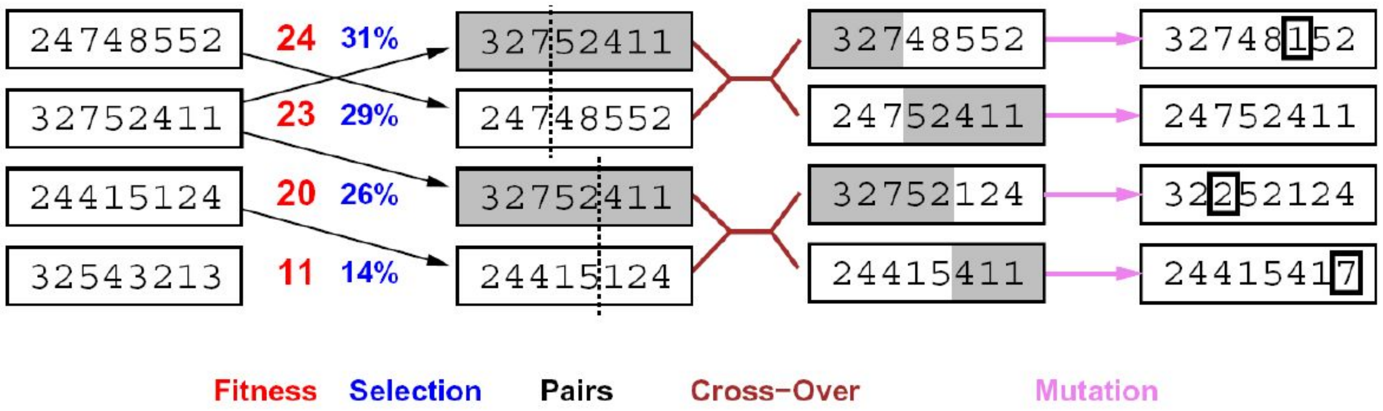
- Prescribes the function or method's effect as a command ("Do this", "Return that:").
- Docstring should NOT be a "signature" (reiterating the function/method parameters, which can be obtained by instrospection).

5 min Project Pitchs Time!

Project 1: GENETIC ALGORITHMS

Natalia Díaz Rodríguez

- Commonly used to generate high-quality solutions to **optimization and search** problems by relying on bio-inspired **operators**
- A **metaheuristic** belonging to the larger class of evolutionary algorithms (EA).
  - Inspired by a natural selection metaphor:
    - Keep best N hypotheses at each step (**selection**) based on a fitness function.
    - Have a pairwise **crossover** operator, with optional **mutation** to give variety

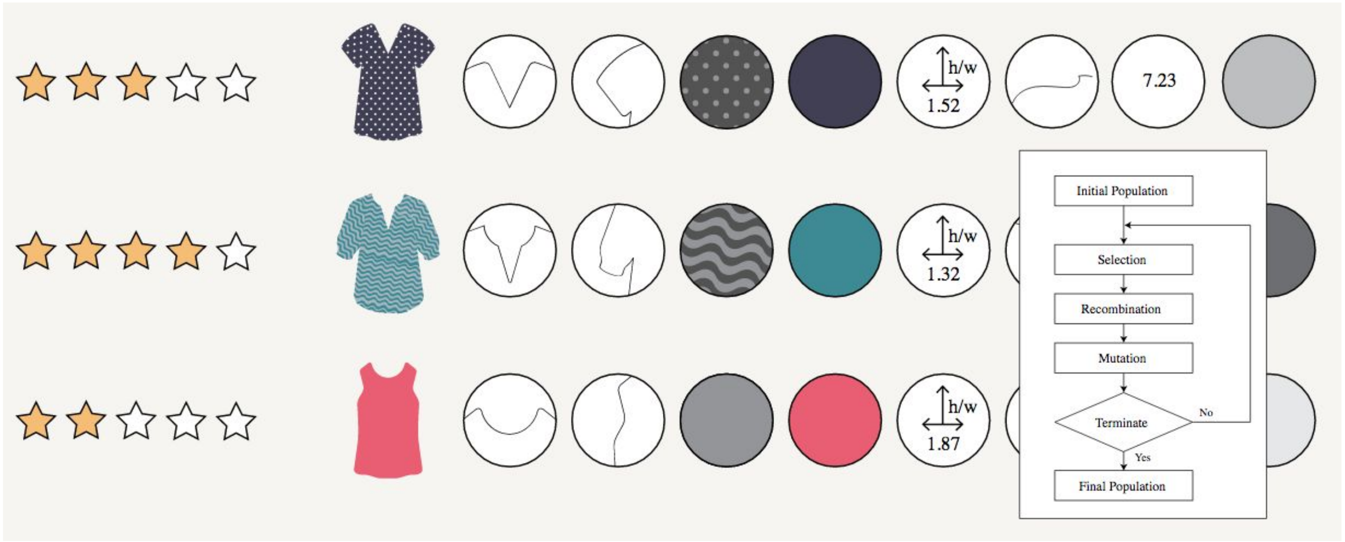


Project 1: GENETIC ALGORITHMS

Natalia Díaz Rodríguez

A Visual Guide to Evolution Strategies

OCTOBER 29, 2017



Project 1: GENETIC ALGORITHMS

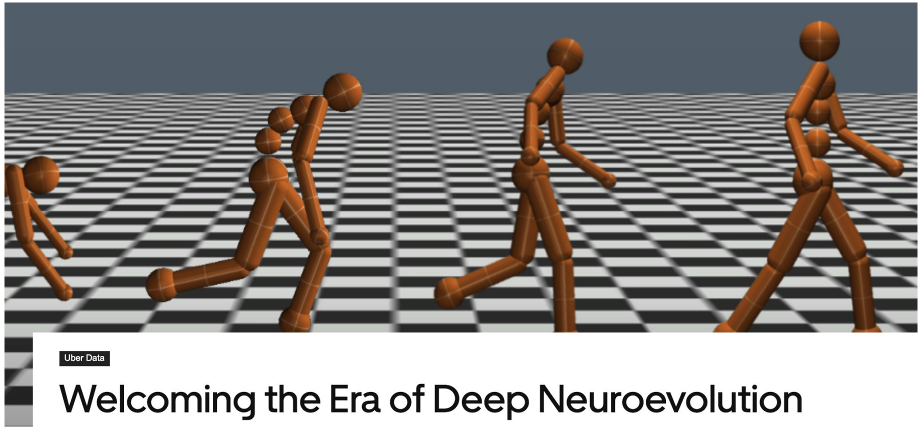
Natalia Díaz Rodríguez

Intro reads:

- GA: https://en.wikipedia.org/wiki/Genetic_algorithm
- EA: https://en.wikipedia.org/wiki/Evolutionary_algorithm
- Visual Evolution Strategies: http://blog.otoro.net/2017/10/29/visual-evolution-strategies/
- The Era of Deep Neuroevolution (Uber AI Labs): https://eng.uber.com/deep-neuroevolution/
- An intro to genetic algorithms and an example of real industry application on data-driven fashion design https://multithreaded.stitchfix.com/blog/2016/07/14/data-driven-fashion-design/

Contact: natalia.diaz(at)ensta-paristech(dot)fr

https://github.com/NataliaDiaz/IN_104-Projet-Informatique



Welcoming the Era of Deep Neuroevolution

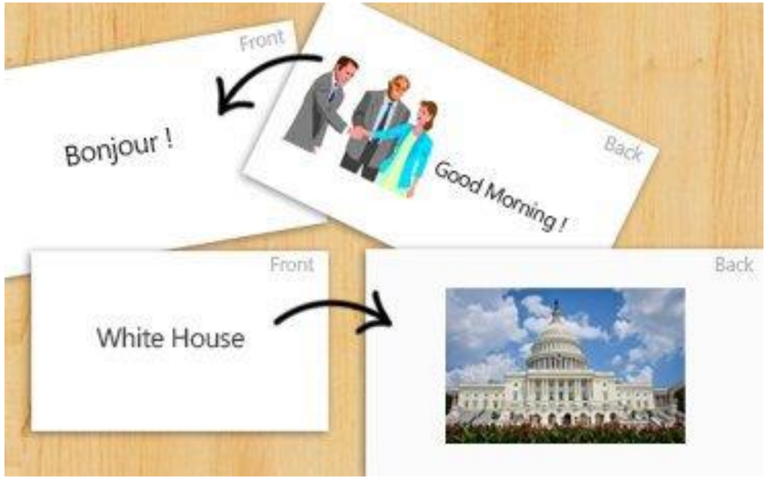
Project 2: Flashcard system
Roxana Agrigoroaie

Task:

Create your own flashcard system

Why:

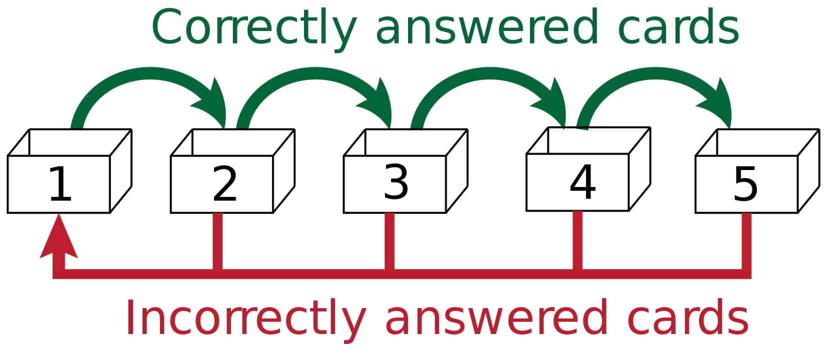
- Educational
- Practice your Python skills



Project 2: Flashcard system

Two main parts:

- **The flashcard management system**:
  - Add / edit / delete cards
  - Customize your cards
- **The training system**:
  - Implement spaced repetition
  - Training your skills
  - Select the subjects/number of cards that you want to practice
  - Have an overview of your performance



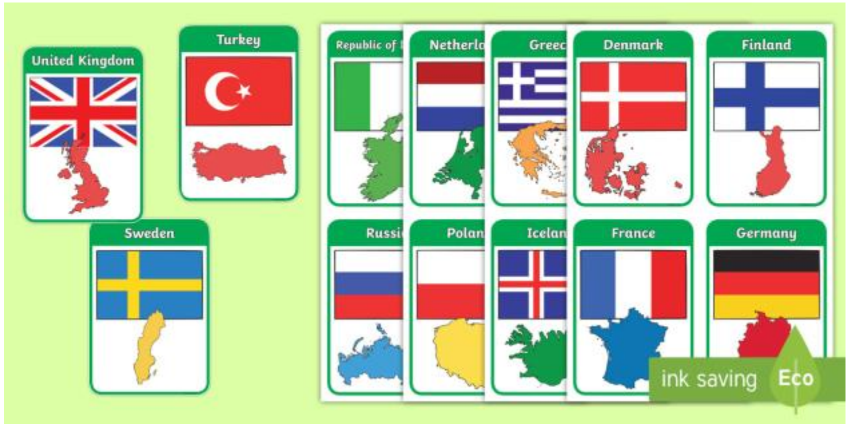
Project 2: Flashcard system

Training system:

- Using only the terminal
- Developing a basic GUI

Possible subjects:

- Learning a foreign language
- Learning the capitals of the world
- Learning some animals/plants
- Practicing for a school exam



Project 3: Hanabi AI

Jean-Didier Garaud jean-didier.garaud@onera.fr

A **cooperative** game (2-5 players)

Spiel des Jahres 2013

Very simple to explain, very tricky to play well

Challenging for AI design:

- Imperfect information
- AI has to cooperate with human
- Coordination, trust, empathy

Currently a research subject (google summer, deepmind, ...)

Project 3: Hanabi AI How to play

The goal is simply to play all 25 cards (1 to 5 in each color) in the correct order.

BUT : You don't see your cards,
You only see your partners'!

Each turn you must do 1 (and only 1) of:

- Give a clue
- Play a card
- Discard a card

Project 3: Hanabi AI

OO Programming with Python3:

use an existing library and extend it:

https://github.com/JDGaraudEnsta/hanabi

Task: develop one or more AI’s, based on:

- deterministic strategies
- human conventions & strategies
- or academic papers (hat strategy, deep learning)

Let it play against with itself or a human player

```
[jd@ananke hanabi]$ hanabi
Let's start a new game
Here are the starting hands:
[Y4 Y3 R4 B2 Y2, R2 Y1 R1 G1 W4]

Alice this is what you remember:   ** ** ** ** **
        this is what you see:       R2 Y1 R1 G1 W4
What do you want to play?
        (d)iscard a card (12345)
        give a (c)lue (RBGWY 12345)
        (p)lay a card (12345)
        e(x)amine the piles
hanabi> c1
Alice gives a clue: 1

Benji this is what you remember: ** *1 *1 *1 **
        this is what you see:     Y4 Y3 R4 B2 Y2
What do you want to play?
```

Project 3: Hanabi AI

Online resources:

[BGA](https://fr.boardgamearena.com/#!gamepanel?game=hanabi)

[Hanabi strategies](https://github.com/Zamiell/hanabi-conventions)

[HanSim: the hat guessing strategy]
(https://d0474d97-a-62cb3a1a-s-sites.googlegroups.com/site/rmgpgrwc/research-pap
ers/Hanabi_final.pdf)

[deepmind Hanabi] (https://arxiv.org/abs/1902.00506)

Project 4: Checkers AI Clément Masson

Why implementing a Checkers AI ?

- Fun application of programming:
  You can play against your AI !

- Challenging application :
  competition with other groups !

https://sites.google.com/view/ensta-in104-checkers
masson.cle@gmail.com

Project 4: Checkers AI Clément Masson

How implementing a Checkers AI ?

With ***minimax*** algorithm:

- Relevant for any 2 players sequential Game with perfect information (chess, checkers, go …)
- Simple theory but
- Many variants with +/- efficiency and complexity

https://www.youtube.com/watch?v=5oXyibEgJr0

Project 4: Checkers AI

Clément Masson

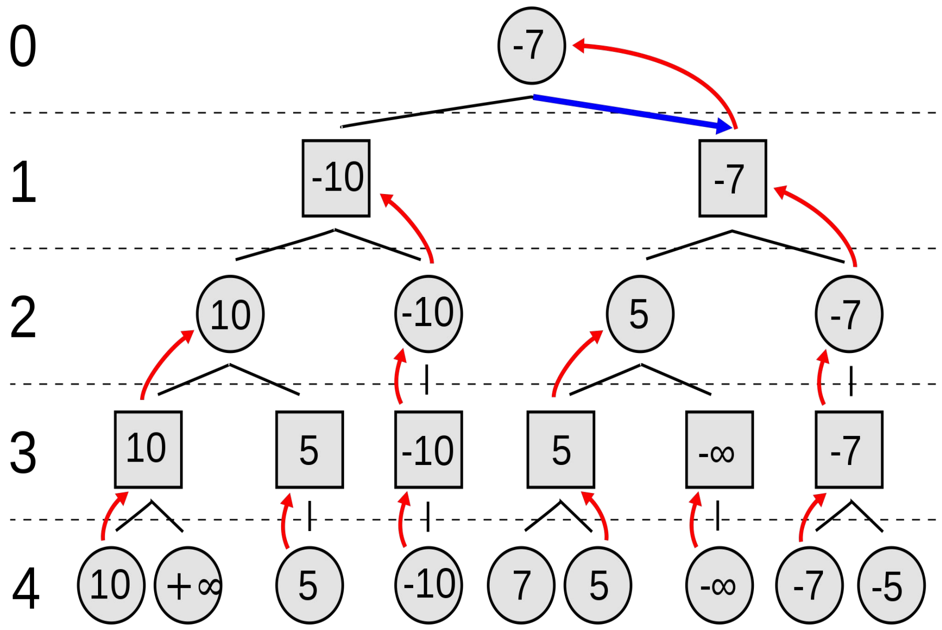
What is required to implement minimax ?

Theoretical aspects

- Trees
- Recursivity
- ...

Practical aspects :

- Object Oriented Programming
- Constrained execution time
- Profiling
- Debugging



https://en.wikipedia.org/wiki/Minimax

Project 5: Labyrinthe Florence Carton

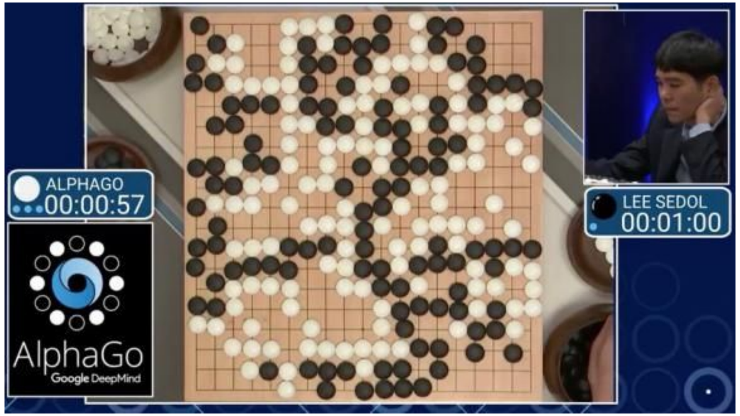
Intelligence artificielle

Apprendre à se déplacer dans un labyrinthe avec de l’apprentissage par renforcement

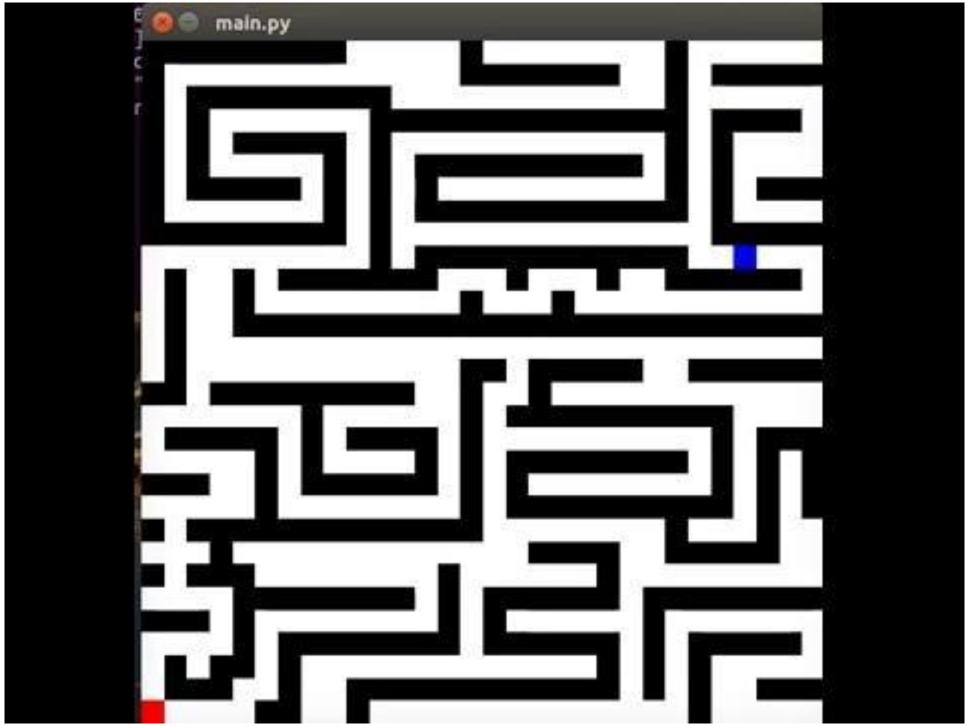
Project 5: Labyrinthe Florence Carton

Apprentissage par renforcement :

- Classe d’algorithme de Machine Learning
- Objectif : apprendre à partir d'expériences
- Mars 2016 : Alpha Go

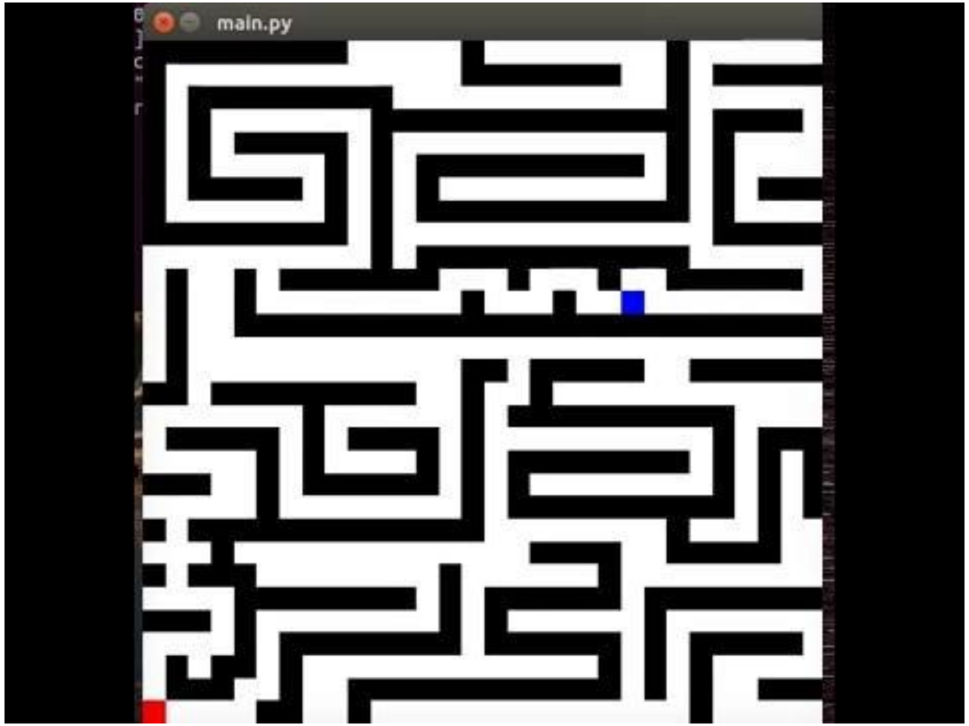


Project 5: Labyrinthe Florence Carton



1ère itération

Project 5: Labyrinthe Florence Carton

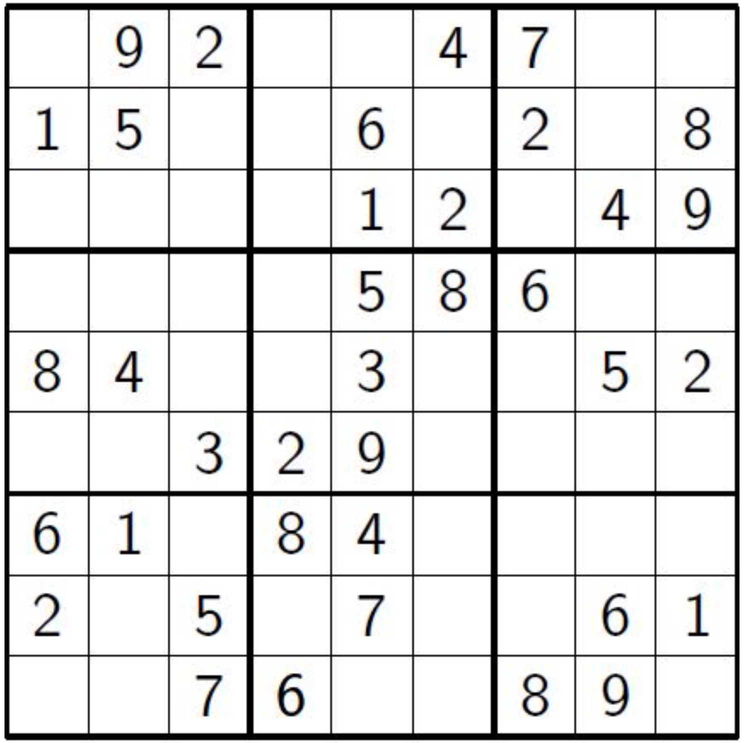


1000ème itération

Project 6: Solveur de Sudoku Ugo Vollhardt

Création d'un solveur de Sudoku :

1. Algorithme basique de résolution
2. Algorithme de forçage récursif
3. Algorithmes de discrimination rapide
4. Interface Graphique
5. Génération aléatoire de grille



	9	2			4	7		
1	5			6		2		8
				1	2		4	9
				5	8	6		
8	4			3			5	2
		3	2	9				
6	1		8	4				
2		5		7			6	1
		7	6			8	9	

Project 6: Solveur de Sudoku Ugo Vollhardt

Langages possibles : C/C++ et Python

Notions mobilisées :

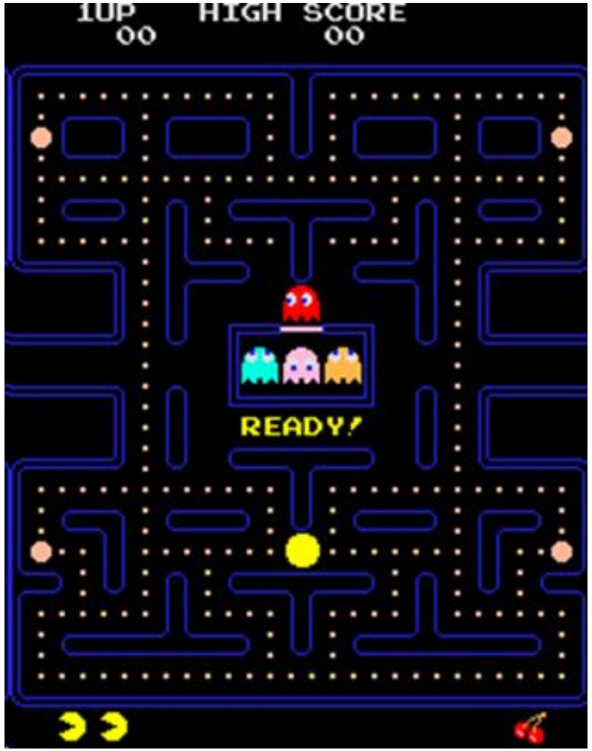
- Maîtrise des structures de données,
- Accès lecture/écriture fichier,
- Manipulation simple de tableau,
- Notions de récursivité ,
- Compilation pour C/C++.

Introduction à la librairie Qt pour l'interface Graphique

Project 7: Pacman game

Stephen Creff

- A good old fashion game,
  - in the popular culture since the 1980s
- Make and manage your project
  - Define the systems requirements,
  - Choose the development cycle to use
  - Choose your software language,
  - Design & implement algorithms to
    - control the Pac-Man &
    - control the Ghosts
  - Define your GUI,
  - Test & validate your system.



Screenshot of the Namco GUI

Project 7: Pacman game Stephen Creff

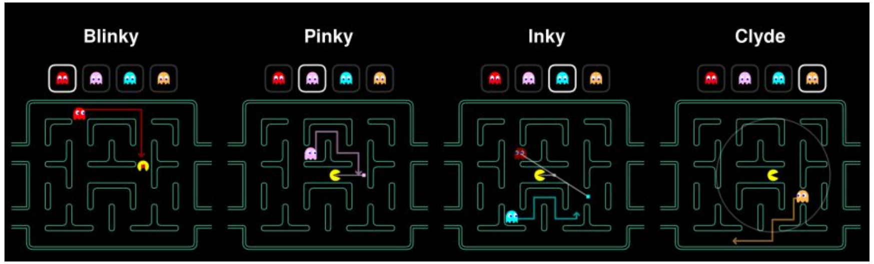
Basic objectives to achieve

Develop a software product and validate minimum game requirements:

- GUI
- Single stage game
- All ghosts have the same “chasing” behavior
- Partial traceability between deliverables (Software requirements, software functions and tests),
- Some tests pass

Explore ghosts behaviors

- Define strategies, define levels (stages) …
- Develop one or more AI’s for Blinky, Pinky,



Information about the gameplay https://en.wikipedia.org/wiki/Pac-Man

Project 8 : Search Engines

Guillaume LORRE

Explosion of the number of multimedia documents -> Needs for fast ways to access relevant information

We use search engines everyday but how are they implemented in practice?

What are the main factors for their relevance performance and efficiency?

Project 8 : Search Engines

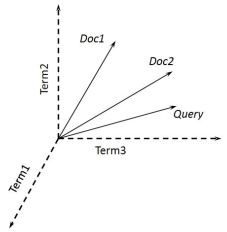
Guillaume LORRE

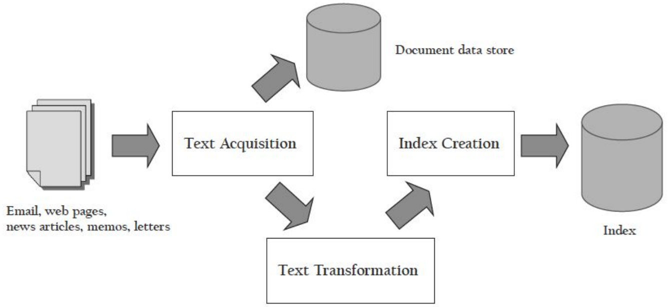
2 main parts : indexing and searching

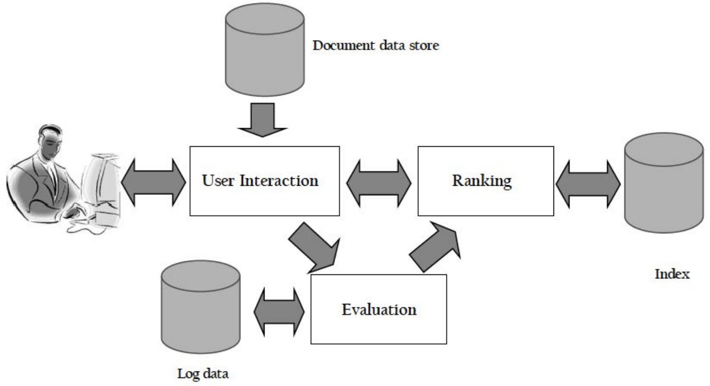
Focus on :

- Text transformation
- Index creation
- Ranking

Model : Vector Space







Project 8 : Search Engines

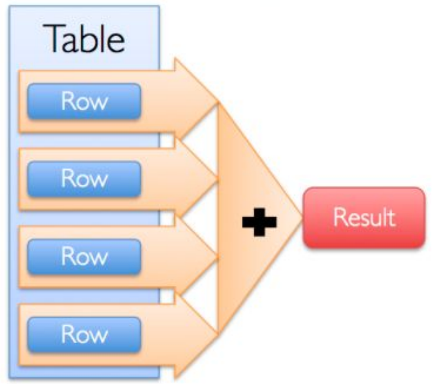
Guillaume LORRE

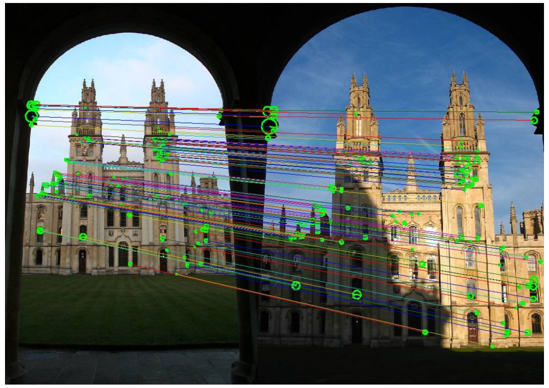
First part : Basic search engine implementation

Second part : Improvements

- Parallel index creation
- Index file compression
- Faster search

Or extension to images for the most motivated groups





Intro to GIT

Link to slides: https://github.com/NataliaDiaz/IN_104-Projet-Informatique

Lecture 1: Intro to GIT

Lecture 2: Intro to Object Oriented Programming (OOP)

Lecture 3: Test Driven Development (TDD) in Python

References & Acknowledgements

- Jean-Didier Garaud. Coding Best Practices
- Resources to learn Scientific Python: scipy, pandas, numpy: https://github.com/paris-saclay-cds/data-science-workshop-2019
- González Duque, R. Python para todos(on line).
- Lutz, M CC. (2011). Programming Python (4.a ed.). EE. UU.: O'Reilly Media.
- Lutz, M (2013). Learning Python (5.a ed.). EE. UU.: O'Reilly Media.
- Martelli, A (2017). Python in a Nutshell (3.a ed.). EE. UU.: O'Reilly Media.
- Pilgrim, M (2011). Dive Into Python 3(online). APress. http://www.diveintopython3.net
- Python Software Foundation. Python Official Webpage. <http://www.python.org>
- Photography: Sandy Skoglund