

ANGLAIS  
DEUXIÈME ANNÉE

---

# SLANGED PROJECT

---

Loïc Mazou  
Tony Zhou

# Table des matières

<b>Introduction (anglais)</b>	<b>1</b>
<b>Documentation utilisateur (anglais)</b>	<b>2</b>
1.1 Description . . . . .	2
1.2 Browsing the games . . . . .	2
1.3 Launching the app . . . . .	2
1.4 Hangman . . . . .	3
1.4.1 Summary . . . . .	3
1.4.2 Scoring . . . . .	3
1.4.3 Control . . . . .	3
1.5 GuessWord . . . . .	4
1.5.1 Summary . . . . .	4
1.5.2 Scoring . . . . .	4
1.5.3 Control . . . . .	4
1.6 MatchWord . . . . .	5
1.6.1 Summary . . . . .	5
1.6.2 Scoring . . . . .	5
1.6.3 Control . . . . .	5
1.7 Dictionary . . . . .	5
<b>Documentation technique (français)</b>	<b>6</b>
2.1 Prérequis . . . . .	6
2.2 Implémentation . . . . .	6
2.2.1 Programmation orientée objet (POO) . . . . .	6
2.2.2 Choix du langage . . . . .	7
2.2.3 Pattern MVC . . . . .	7
2.2.4 Utilisation de l'API . . . . .	7
2.3 Difficultés/Problèmes/Solutions . . . . .	8
2.3.1 Turtle . . . . .	8
2.3.2 CrossWord . . . . .	8
2.3.3 Difficulté subjective . . . . .	9
<b>Conclusion (anglais)</b>	<b>10</b>

# Introduction (anglais)

Even though hundreds of languages are still being used as of now, English is still considered to be the most relevant. It's a common ground that allows people to communicate efficiently. As time goes by, new requirements appear in every domain. This is also true regarding communication. With different people within different contexts, the way for one to express themselves evolves.

There is a need to keep up with the language changes. The risk of being left behind might cause trouble while communicating. While not essential, the skills are still useful to exchange with one another.

Slanged is our own way of tackling this problem. Many are the people who don't understand the colloquialism used by the netizens. Although there are tools/resources that allows one to look for a particular expression, it is usually too much of a hassle to actually do it.



FIGURE 1 – Common English slang (from MentalFloss)

Our project is primarily meant for people who would like to improve their English in that particular field. Of course, if one just wants to have fun, they can also enjoy our application.

# Documentation utilisateur (anglais)

## 1.1 Description

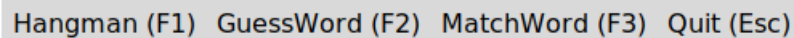
Slanged is an application meant to learn the English slang used by the people on the Internet. It uses Urban Dictionary's API to get words. Throughout several games, it attempts to make the user remember the definitions of some terms.

There are three different games in total :

- Hangman
- GuessWord
- MatchWord

## 1.2 Browsing the games

To select a game, the user may click on the different items available. Otherwise, they can use the F1, F2 or F3 keys to play respectively Hangman, GuessWord and MatchWord. Once done with the application, they may leave it by clicking on **Quit** or pressing the **Escape** key.



Hangman (F1) GuessWord (F2) MatchWord (F3) Quit (Esc)

FIGURE 1.2 – Slanged's menubar

## 1.3 Launching the app

Assuming the requirements mentioned later on this document are met, the instructions to launch the app are the following :

- go to the project directory where `slanged.py` is located
- open up a terminal from this folder
- input the following command :  
`python slated.py`

## 1.4 Hangman

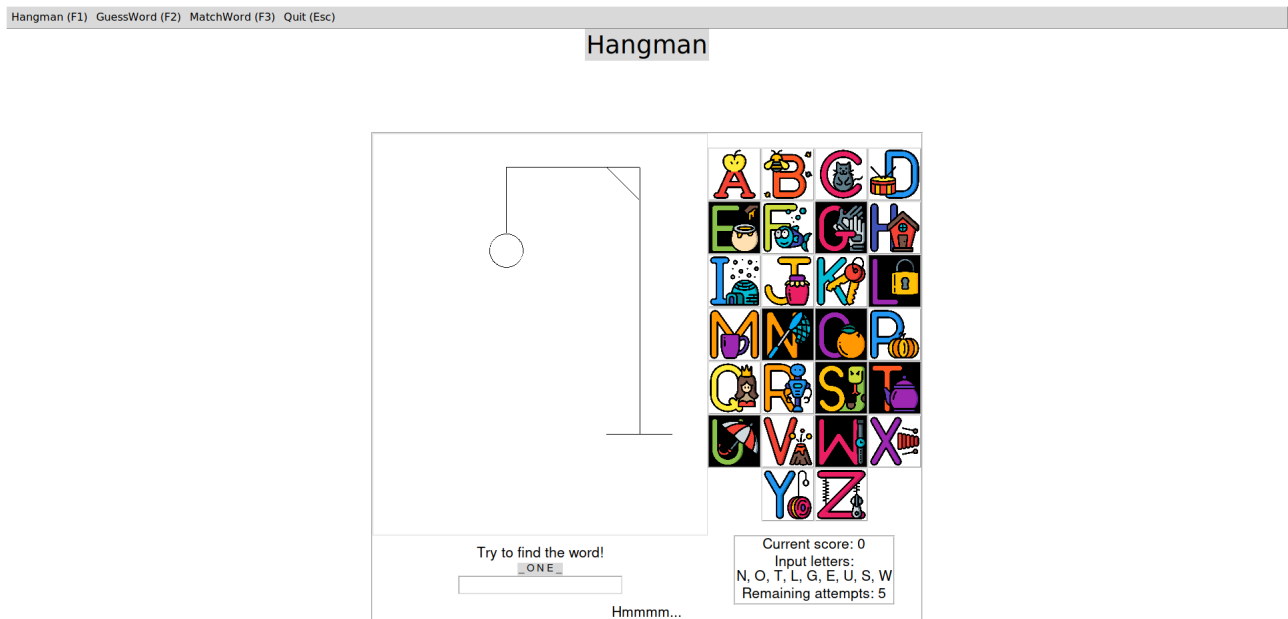


FIGURE 1.3 – Hangman

### 1.4.1 Summary

A random word is chosen for the user to find. The user only knows its length.

The player has to click the letters (which are buttons) to make a guess. Or, if they are confident enough, they can try to type in a word in the entry. They have 11 tries.

### 1.4.2 Scoring

The game's score uses the following rules :

- if the word is found, then the player gets the number of remaining lives as points
- if not, the player gets a penalty of 5 points (can't have a negative score)

### 1.4.3 Control

The user has to use their mouse as a way to choose the letters to fill in. They can also select the entry to type in the word using the keyboard. They can submit their proposition by pressing the **Return** key.

After a round, the application will ask to press the **R** key to restart a game. The user can also press a letter from the app's keyboard.

## 1.5 GuessWord

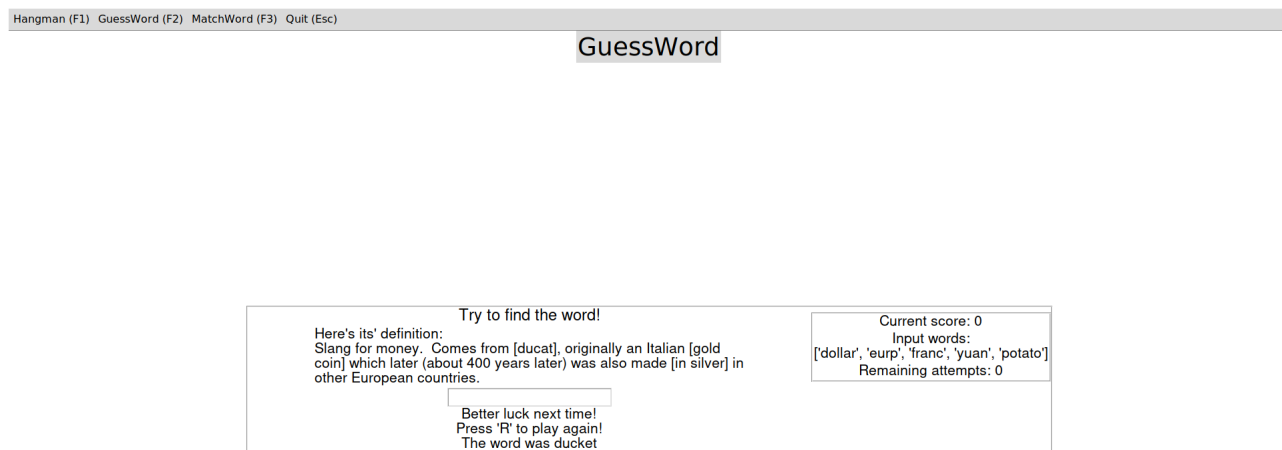


FIGURE 1.4 – Guessword

### 1.5.1 Summary

A random word is chosen for the user to find. The user only knows its definition (which is displayed on the screen).

The player has to type in a word in the entry in order to make a guess. They have 5 lives.

### 1.5.2 Scoring

The game's score uses the following rules :

- if the word is found, then the player gets the number of remaining lives as points
- if not, the player gets a penalty of 5 points (can't have a negative score)

### 1.5.3 Control

The user has to type the word using their own keyboard. They can submit their proposition by pressing the **Return** key.

After a round, the application will ask to press the **R** key to restart a game.

## 1.6 MatchWord

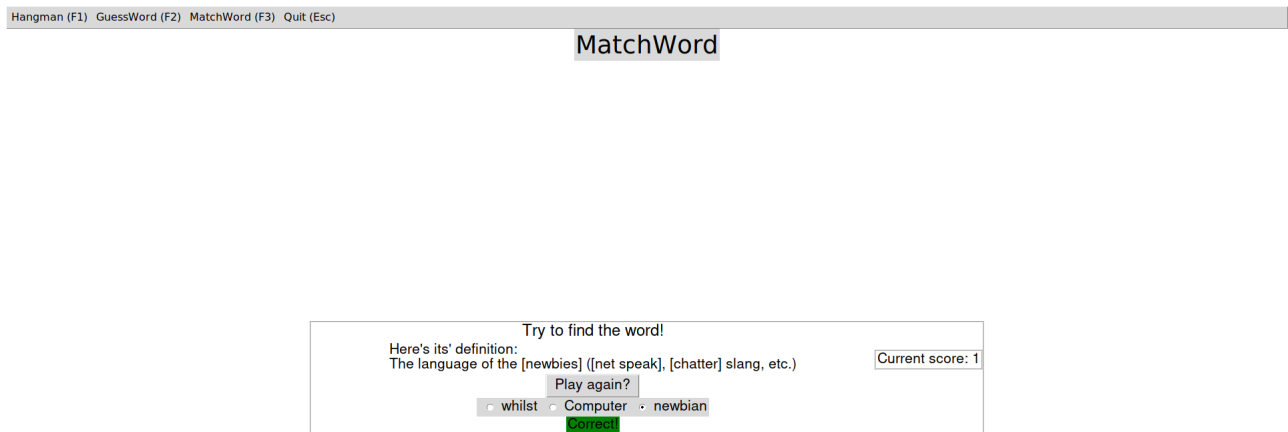


FIGURE 1.5 – Matchword

### 1.6.1 Summary

A random word is chosen for the user to find. The user only knows its definition (which is displayed on the screen).

Basically an MCQ where the user has to pick the corresponding word for the definition.

### 1.6.2 Scoring

The game's score uses the following rules :

- if the word is found, then the player gets 1 point
- no penalty when not found

### 1.6.3 Control

The user has to tick one of the checkboxes, then press the **Submit** button.

After a round, the application will ask to press the **Play again** button to restart a game. The previous **Submit** button becomes unavailable during this event.

## 1.7 Dictionary

The user can review words they encountered during the playthrough. Every single word is saved in a file `vocabulary.txt`, located at the root folder of the app.

It appends automatically the word and the definition without doing anything, except playing the game.

# Documentation technique (français)

## 2.1 Prérequis

Pour pouvoir utiliser notre application, il est nécessaire d'avoir quelques éléments installés sur son ordinateur :

- Python version 3
- connexion à Internet

L'application fonctionne sur tout type d'ordinateur, peu importe le système d'exploitation.

## 2.2 Implémentation

### 2.2.1 Programmation orientée objet (POO)

Tout comme les langues, l'application se veut modulable et évolutive. Pour cette raison, nous avons choisi d'établir une programmation par objet plutôt que procédurale. De plus, cette méthode est facilement compréhensible : elle se rapporte à de nombreux exemples de la vie réels.

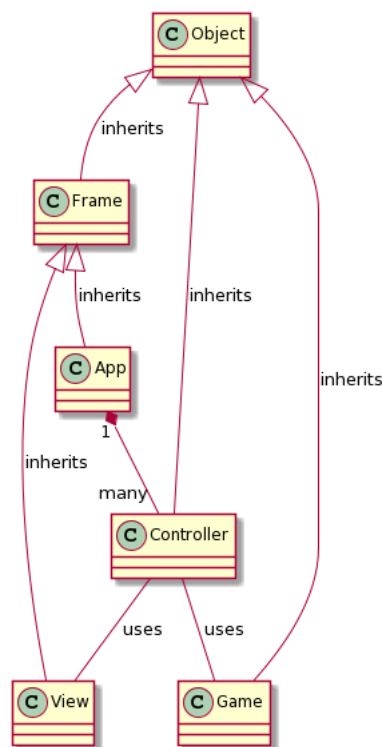


FIGURE 2.6 – Diagramme de classes basique



### 2.2.2 Choix du langage

D'ordinaire, lorsqu'on parle de POO, le langage de prédilection est **Java**. Cependant, durant notre cursus, nous avons déjà beaucoup utilisé ce langage pour coder ce type de programme. Nous avons donc décidé de nous laisser un défi en choisissant quelque chose de moins maîtrisé.

Nous avons déjà côtoyé **Python** au cours de notre scolarité. Cependant, nous ne l'avions jamais utilisé pour coder en POO. La syntaxe étant différente, nous avons dû apprendre des nouvelles utilisations possibles de ce langage.

Pour tout ce qui de l'interface graphique, l'équivalent de **JavaFX** est pour **Python** le package **Tkinter**. Nous utiliserons ce module pour créer l'ensemble de nos classes. Cela nous permet de garder un thème uniforme sur l'ensemble de l'application.

### 2.2.3 Pattern MVC

Nous avons choisi d'utiliser ce pattern car il correspond aux besoins de notre application.

- les modèles contenant les données seront les jeux en question
- les vues correspondront à chacune des différentes pages
- à chaque couple de modèle/vue sera associé un contrôleur

Dans le modèle, on considère les données suivantes :

- le mot (objet créé par l'API, décrit ultérieurement)
- le score
- le nombre de vies

L'objectif principal de ce pattern est de découpler les données des vues et des actions, limitant ainsi leur inter-dépendance. Cela permet l'évolutivité et la modularité des différentes classes.

### 2.2.4 Utilisation de l'API

Pour récupérer des mots, nous souhaitons utiliser une base de données complète contenant en particulier leurs définitions. La solution d'écrire nos propres mots/définitions nous est venue à l'esprit, mais le manque de diversité (et éventuellement de connaissances) aurait limité l'originalité du vocabulaire.

Nous avons donc décidé d'utiliser une ressource en ligne, à savoir une API.

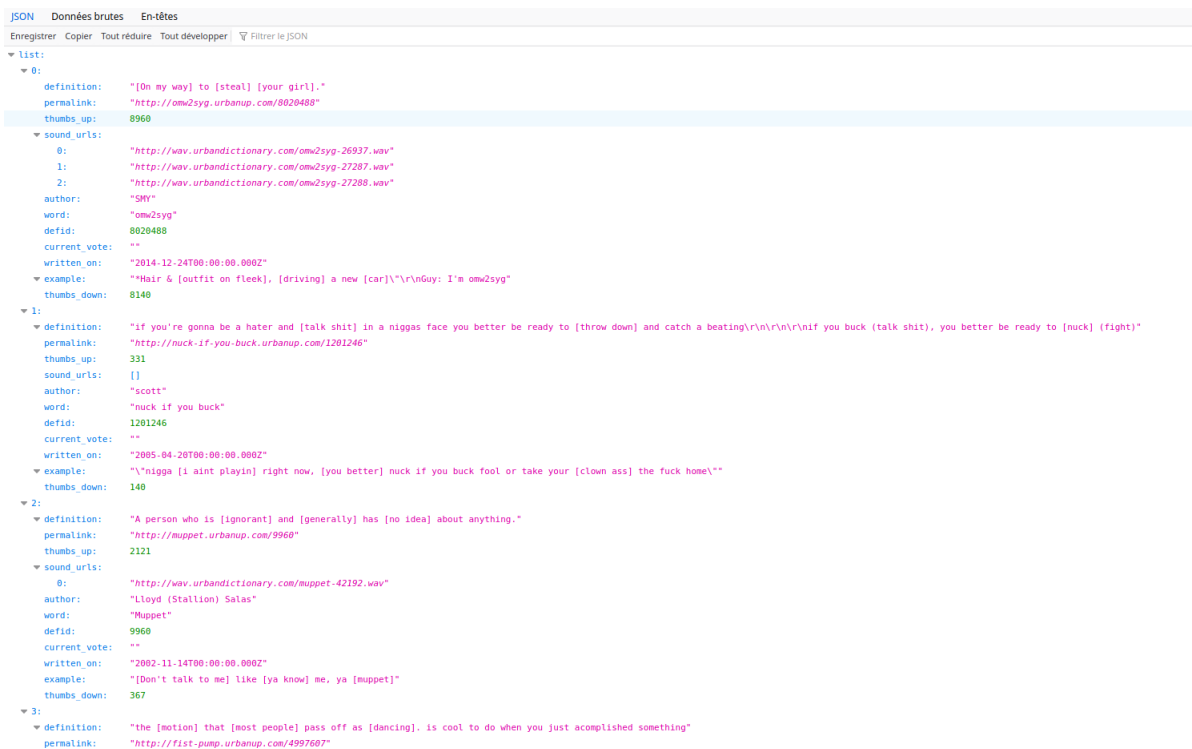


FIGURE 2.7 – Interface de l'API

Pour faire nos requêtes à l'API d'Urban Dictionary, nous utilisons le module `requests`. Nous récupérons un objet au format `json` contenant la réponse du serveur. On récupère une liste de 10 mots au hasard, dans laquelle nous en sélectionnons une aléatoirement.

En utilisant un parser spécialement écrit pour ces données, nous extrayons ce qui nous intéresse et le stockons dans un de nos modèles.

## 2.3 Difficultés/Problèmes/Solutions

### 2.3.1 Turtle

Pour rendre notre jeu plus attrayant, nous avons décidé d'utiliser des images pour illustrer leurs événements internes. En particulier, pour le pendu, nous avons voulu dessiner un stickman évoltif. L'idée a été d'abord d'utiliser le module `Turtle`, qui est un package de base pour dessiner des figures. Or nous nous sommes rendus compte que celui-ci n'est pas compatible avec le module `Tkinter`, utilisé pour créer l'interface graphique de l'application.

Nous avons donc réimplémenté une nouvelle classe `Turtle` simplifiée, qui nous permet de dessiner notre pendu.

### 2.3.2 CrossWord

Initialement, il était prévu de créer un jeu de mots-croisés généré automatiquement. L'algorithme était conçu pour de la programmation procédurale, mais quand est venu le moment de l'écrire pour la POO, les problèmes ont été nombreux.

Il faut en particulier gérer toutes les entrées possibles pour les mots (bouton, label) sur l'interface et les lier aux données du modèle. Cela nous a finalement semblé trop ambitieux pour le temps que nous avons pour ce projet.

### 2.3.3 Difficulté subjective

Les mots à deviner sont parfois assez difficiles à trouver. Nous avons déjà essayé de les filtrer avant de les proposer. Au niveau de la récupération du mot, si celui-ci contient des espaces, des symboles ou des chiffres, la requête est réémise.

Il semble que certaines mots (et leurs définitions) soient trop explicites. Certains sont tout simplement vulgaires.

Il arrive également que la définition contienne le mot caché, ce qui défait la raison d'être du jeu.

# Conclusion (anglais)

Overall, this application lets you discover words that one may not have known about. It is always a great deal of fun when someone finds about anything new.

We definitely learned a lot about coding, but we also got new vocabulary to brag about. While the application might seem outdated/simple, there is always room for it to be improved. Some ideas we had were to include slang from other languages, or to add other databases. Maybe we could add a multiplayer option.

Anyway, the point being that the possibilities are limitless since we designed our application to be future-proof, albeit kind of lacking.

Hopefully the people who try our product will pick up one or two words just like we did.