Advanced Software Engineering

Pet Project

Game Decision Tree

Felicitas König

(Matr-No. 886778)

Contents

[1. Introduction 2](#_Toc777820)

[2. UML 2](#_Toc777821)

[2.1 Activity Diagram 2](#_Toc777822)

[2.2 Class Diagram 3](#_Toc777823)

[2.3 Component Diagram 3](#_Toc777824)

[3. Metrics 4](#_Toc777825)

[4. Clean Code Development 5](#_Toc777826)

[4.1 Meaningful Naming 5](#_Toc777827)

[4.2 Don’t repeat yourself 5](#_Toc777828)

[4.3 Comments 5](#_Toc777829)

[4.4 Exception Handling 6](#_Toc777830)

[4.5 Version Control 6](#_Toc777831)

[5. Build Management 7](#_Toc777832)

[6. Continuous Delivery 8](#_Toc777833)

[7. DSL 9](#_Toc777834)

[8. Functional Programming 9](#_Toc777835)

[8.1 Final Data Structures 9](#_Toc777836)

[8.2 (Mostly) Side Effect Free Functions 9](#_Toc777837)

[8.3 Higher Order Functions - Functions Parameters and Return Values 10](#_Toc777838)

[8.4 Closures 10](#_Toc777839)

[8.5 Anonymous Functions 10](#_Toc777840)

# Introduction

The Game Decision Tree is a little program which should help to find a decision regarding to the evening or weekend plans. For now, the game is designed for two players and includes two steps of decisions. First the players choose one of three categories that reflects their mood. If they have chosen the same one, the game comes directly to the next step. If the chosen categories are different tic tac toe is played. A coin toss decides who may start first. After the category is chosen via tic tac toe, the players select an actual activity within this category. Is this option again the same, there is no need to compromise. However, if the activities are different, the players have to find a decision via rock-paper-scissors. Finally, you will have an independently decided evening or weekend plan and hopefully have a lot of fun.

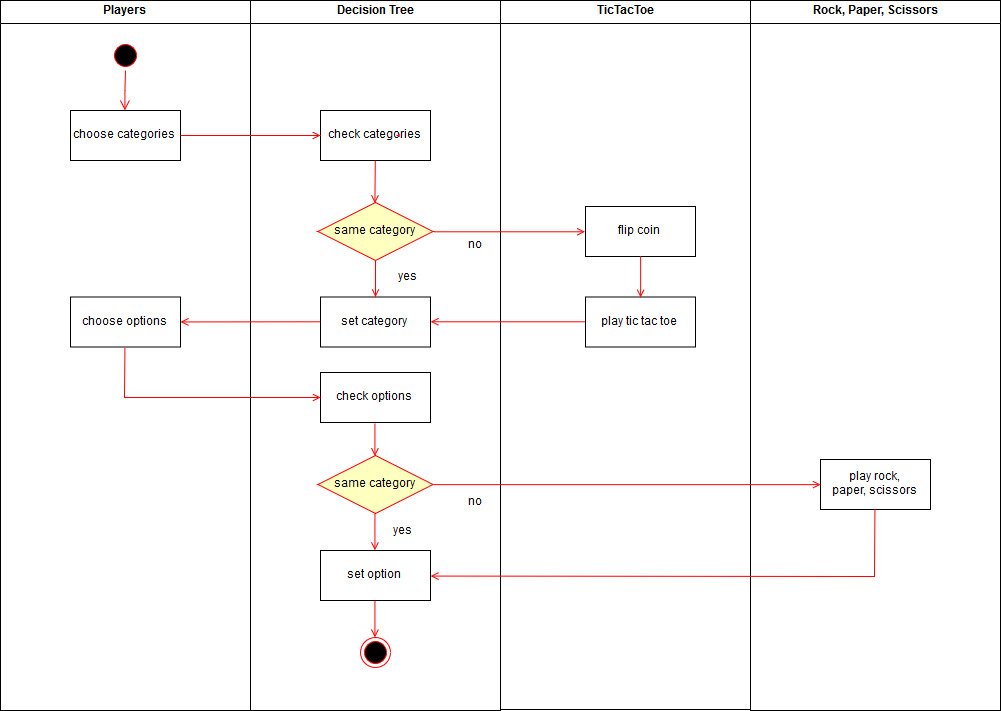
The Game Decision Tree is still a very rudimental version and could be extended by e.g. more players, a whole data base of activities or concrete events/restaurants/other places in a specific city.

# UML

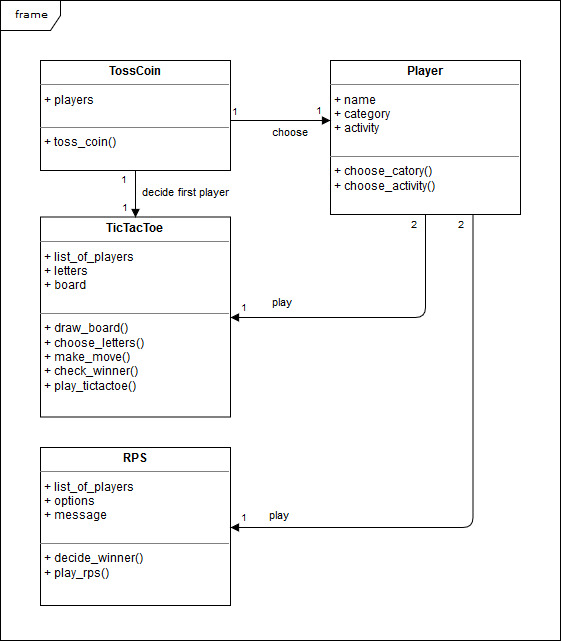
The UML diagrams a made with draw.io.

Files: pet-project\uml

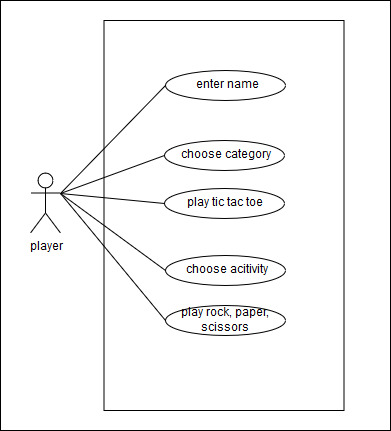
## 2.1 Activity Diagram



## 2.2 Class Diagram

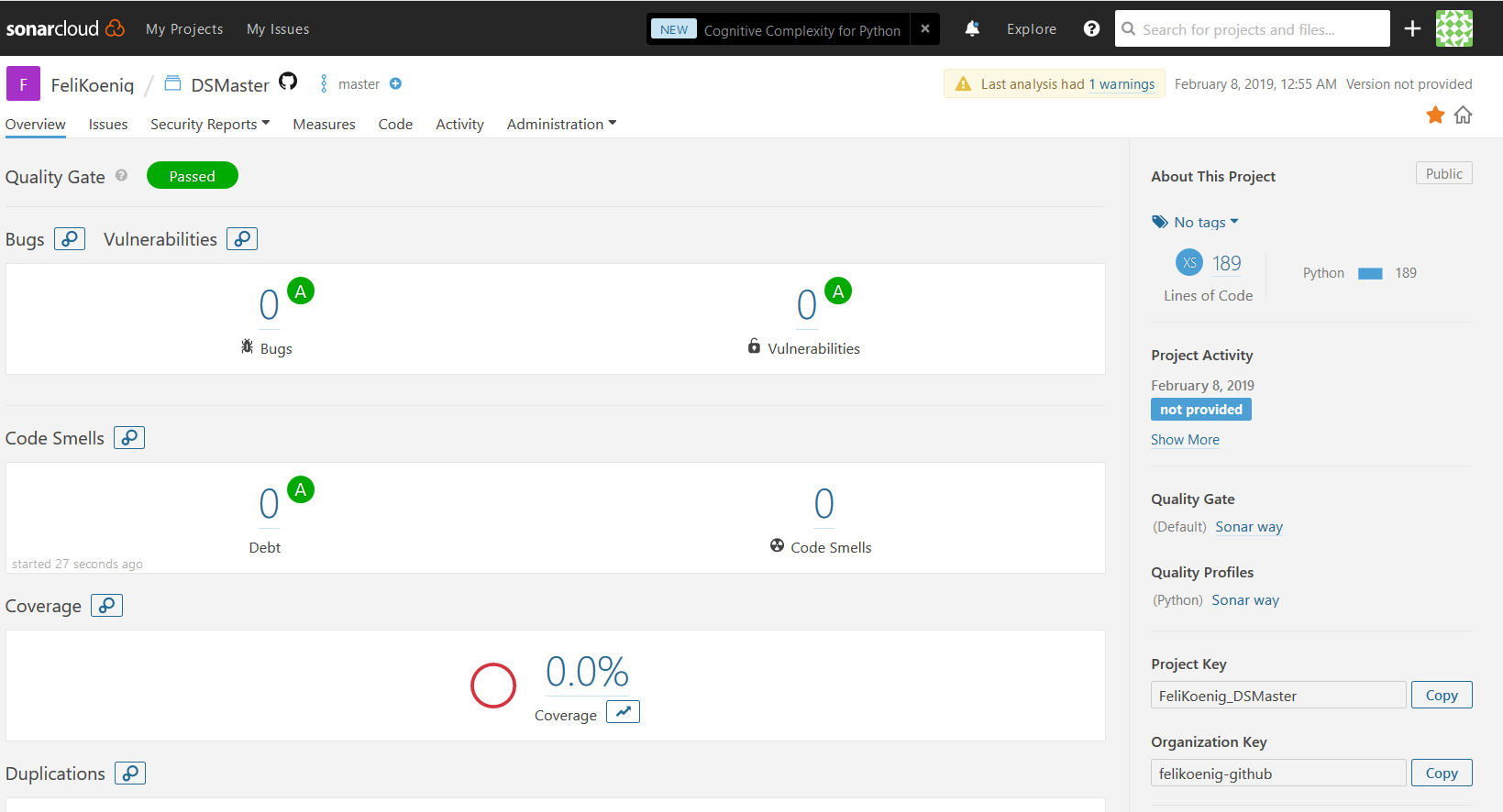


## 2.3 Component Diagram

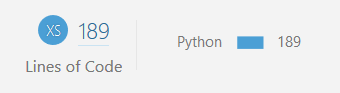


# Metrics

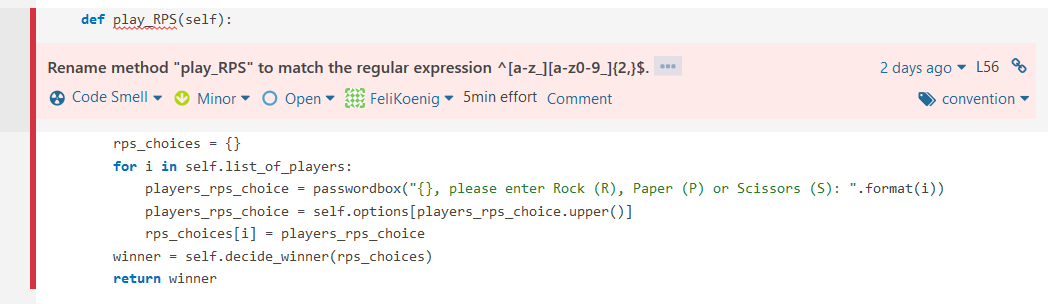
I used Sonarcloud to investigate the metrics of my python-code.



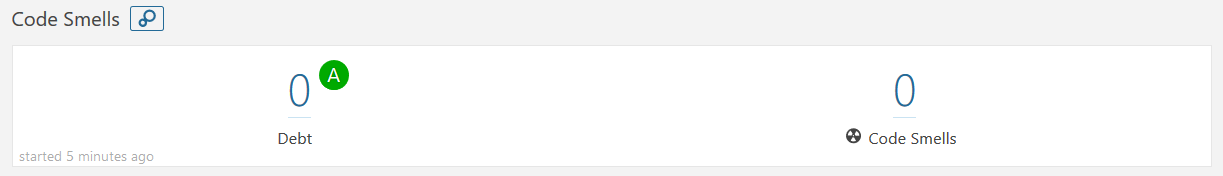
Lines of Code:



Code Smells:



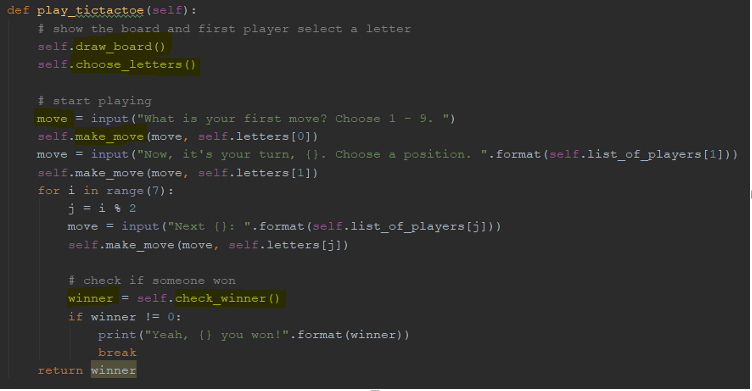
🡪 After correcting the Code Smells:



# Clean Code Development

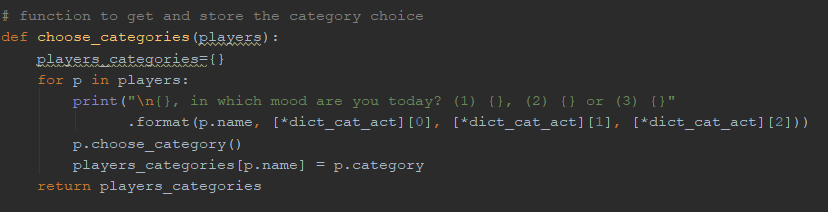
## Meaningful Naming

Functions and variables should be named as precise but still simple as possible. In the example below a function for playing Tic Tac Toe is shown. All steps could be clearly identified by their name.



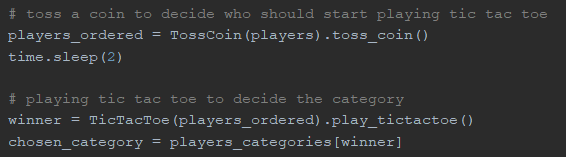
## Don’t repeat yourself

One should not repeat code artefacts, but use refactorization to avoid repetitions. In the example below I used a for-loop in connection with a class-function instead of repeating the code for each player.



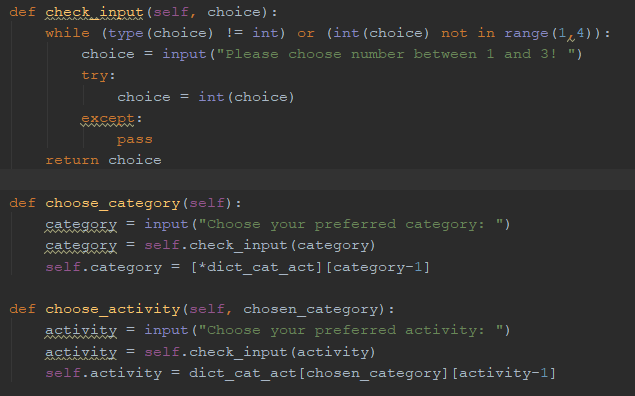
## Comments

Using meaningful comments are important for the understanding of the code. It saves a lot of time reading the code.



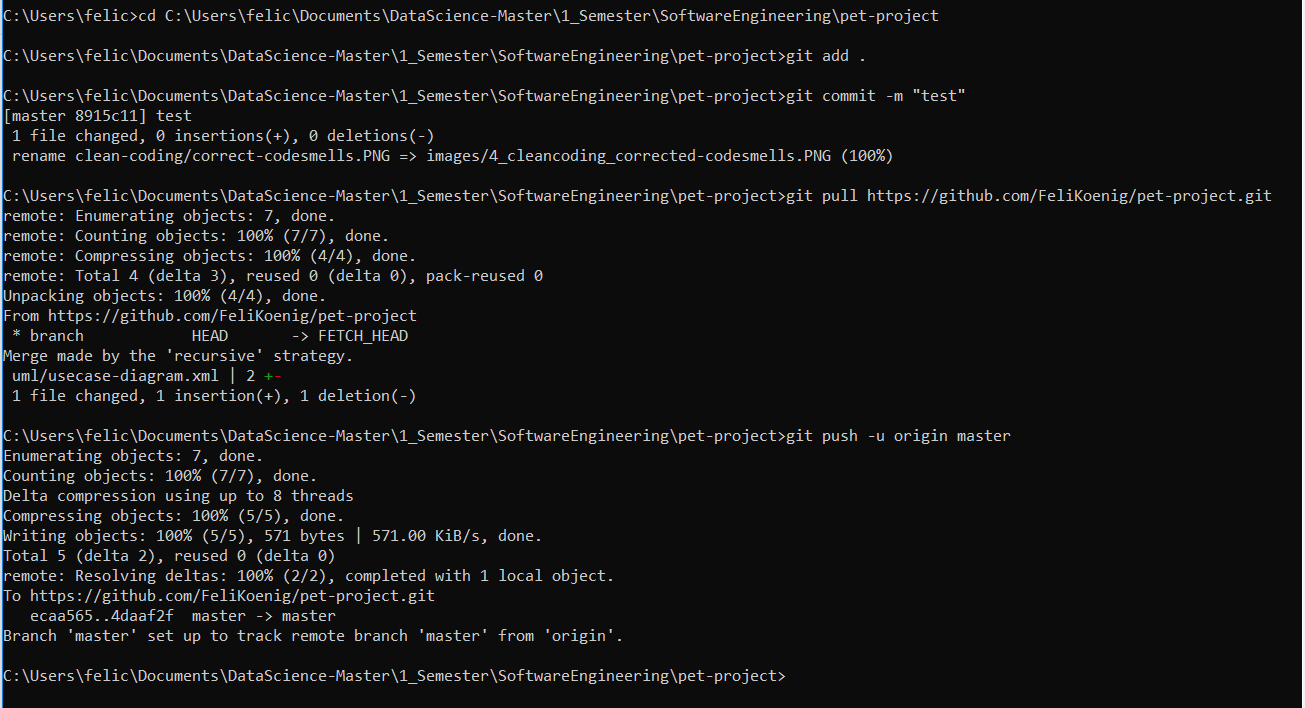
## Exception Handling

One should check inputs to be valid to avoid errors. In the example below I checked the input of a player’s choice and used also exception handling.



## Version Control

For version control I used Git and Github.

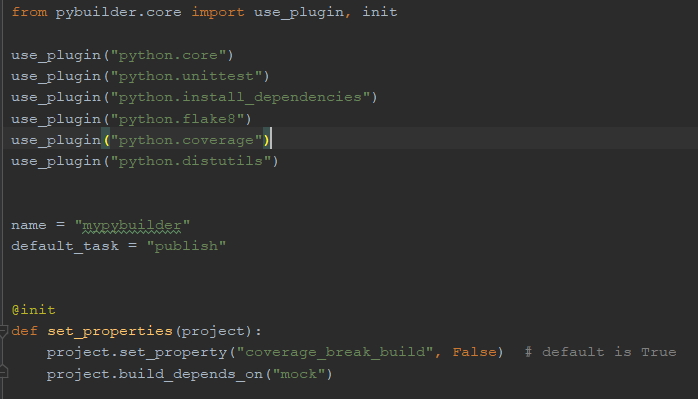


# Build Management

As a build management tool I used PyBuilder and went through the following tutorial:

<https://pythonhosted.org/pybuilder/walkthrough-new.html>

After installing the PyBuilder I created a build.py-file which includes some plugins used for the project and also some properties of it.

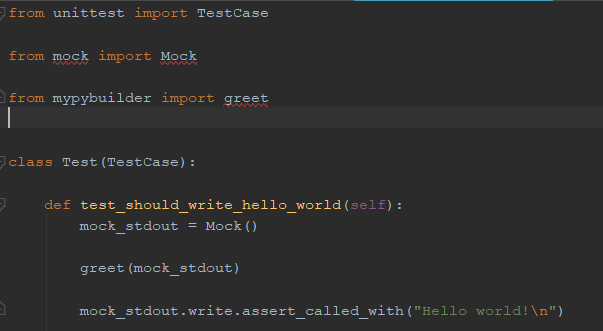


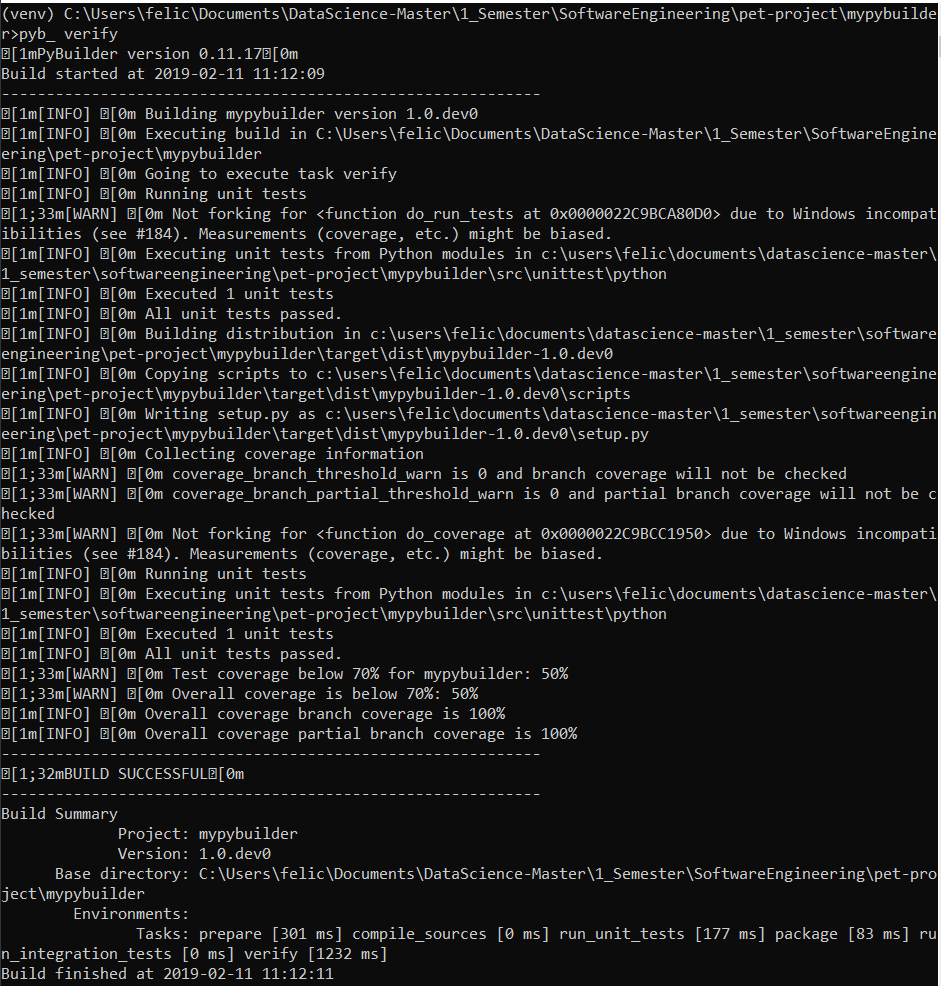
I implemented a function in the \_\_init\_\_.py-file, which I accessed through a unit test.

\_\_init\_\_.py



mypybuilder-test.py

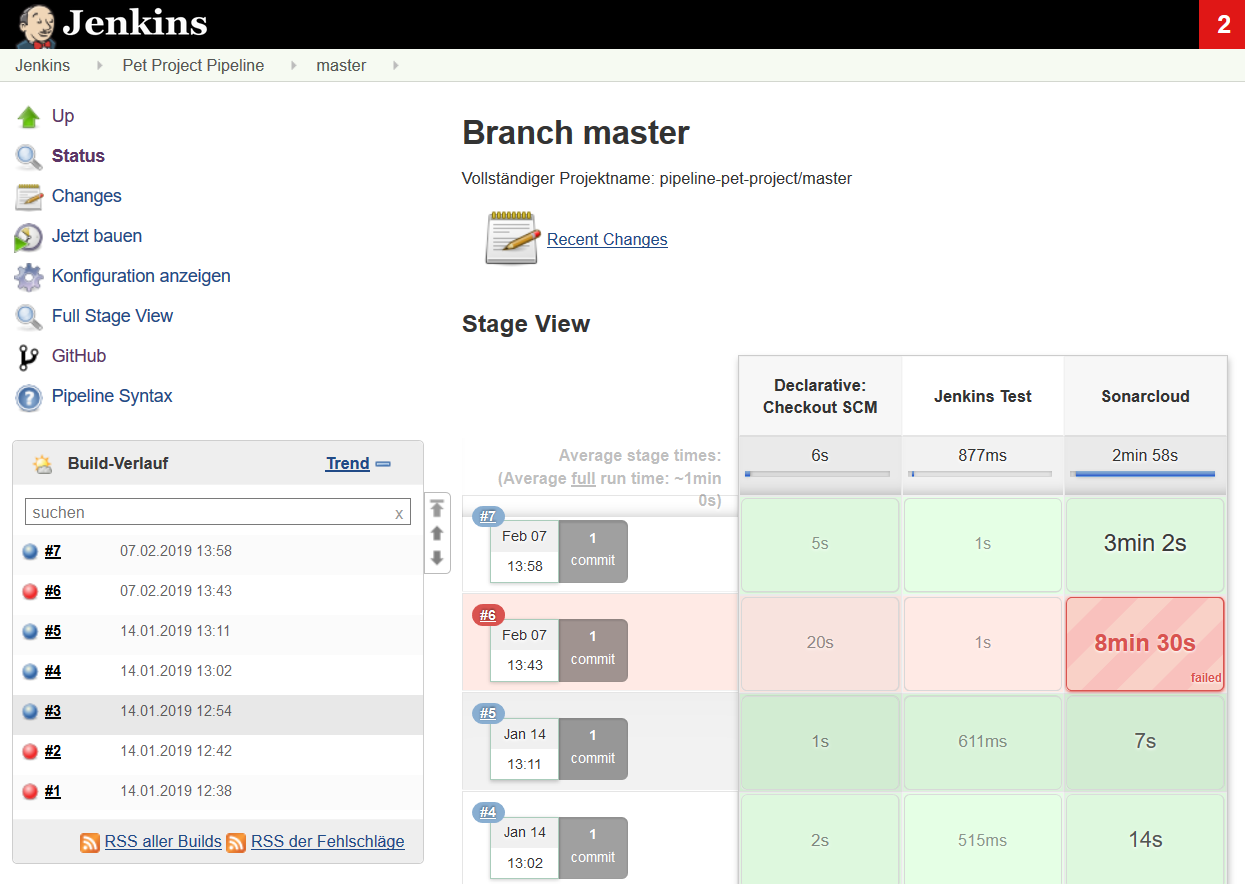




# Continuous Delivery

I created a Pipeline in Jenkins which for example includes to run Sonarcloud-program.

File: pet-project\Jenkins



# DSL

DSL (Domain Specific Languages) are languages that are used to communicate with computers in a certain domain. Within this specific domain, DSLs can be used for all purposes and by different users.

One well-known example for DSL is SQL (Structured Query Language).

In further development of the game decision tree one can use a database e.g. to store concrete proposals for restaurants, bars or other interesting places in town. Such a data base could be executed via SQL.

Example for a SQL query to find all restaurants in Charlottenburg:

SELECT \* FROM data\_table

WHERE type = “restaurant”

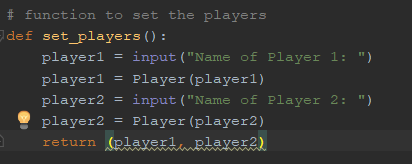
AND borough = “Charlottenburg”

ORDER BY name ASC

# Functional Programming

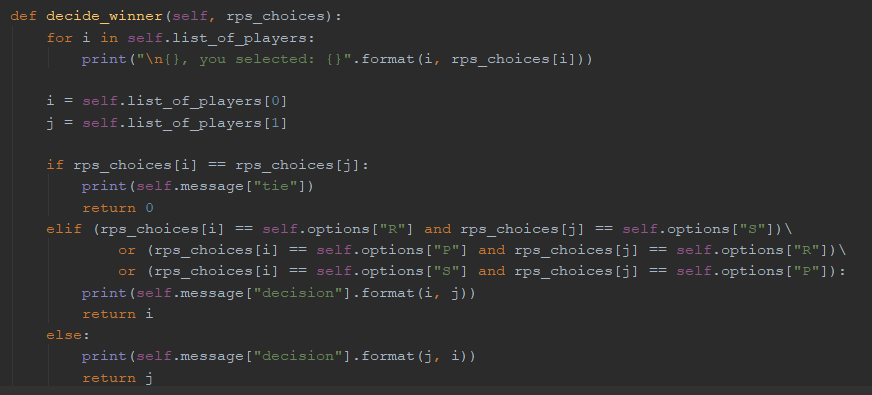
## Final Data Structures

A final data structure are e.g. variables which are assigned only once and could not be changed. In Python tuples are such immutable variable structures.

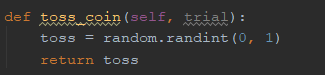


## (Mostly) Side Effect Free Functions

Side effect free functions returns with same input parameters always the same output. The example below detects the winner of Paper-Rock-Scissors, which is clearly deterministic.



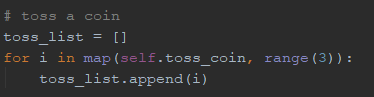
Since the project also includes a coin toss, its function produces a random return value.



## Higher Order Functions - Functions Parameters and Return Values

Higher order functions allow a function as input parameter or return value.

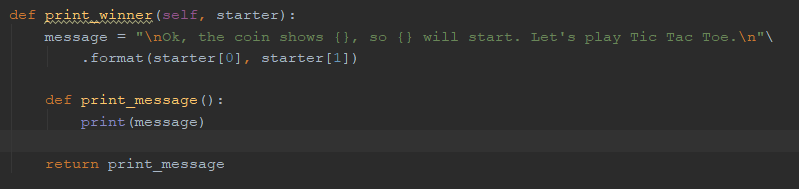
In Python map() is an example for a function which takes a function as parameter.

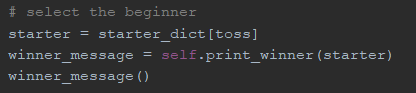


Another example for a function which returns a function is shown in the next section about closures.

## Closures

A closure is a function which remembers variables in enclosing scopes even if the block has finished executing. In the example below the inner function print\_message() is returned by the outer function print\_winner(). The returned inner function is assigned to the variable winner\_message and it persists within its variables after the outer function has been exited.





## Anonymous Functions

An anonymous function is declared without adding a name to it. An example for Python would be a lamda-function.

