The game „Mäxchen“

# Abstract

///General game information

**Table of Contents**

[Abstract 1](#__RefHeading___Toc356_2567421638)

[1. How the game works 1](#__RefHeading___Toc358_2567421638)

[1.1 The rules 1](#__RefHeading___Toc360_2567421638)

[1.2 The values 2](#__RefHeading___Toc362_2567421638)

[1.3 Point loss 2](#__RefHeading___Toc364_2567421638)

[The program/User manual 2](#__RefHeading___Toc366_2567421638)

[Upgrades 3](#__RefHeading___Toc368_2567421638)

[1. The turn order 3](#__RefHeading___Toc370_2567421638)

[2. Not ordered numbers 3](#__RefHeading___Toc372_2567421638)

[3. The settings 4](#__RefHeading___Toc374_2567421638)

[4. Visual dices 4](#__RefHeading___Toc376_2567421638)

[5. Game history 4](#__RefHeading___Toc378_2567421638)

[6. Cheats 4](#__RefHeading___Toc380_2567421638)

[7. The AI 5](#__RefHeading___Toc382_2567421638)

[Change the points 5](#__RefHeading___Toc384_2567421638)

[Change the “order\_digits” option 5](#__RefHeading___Toc386_2567421638)

# 1. How the game works

## 1.1 The rules

Mäxchen is a turn-based game. Player A rolls two dices, where the larger one is the first and the smaller one the second digit of his number (e.g. Alpha rolls a 4 and a 6. The 6 is the first digit, the 4 the second digit, which results in his number being 64). The clue is: Only he knows what he rolled.

He then claims that he rolled a number – this claimed number can be higher, lower or equal to his rolled one, the important thing is that it is larger (in value, more on that later) than the number that was previously rolled or better the one that was claimed to be rolled.

Player B now has to decide. If B believes A, it’s her turn to roll and the game goes on. However, if she doesn’t believe A, the dices – and therefore the real number – gets revealed. If the claimed number has a higher value than the one A rolled, he got caught in a lie and loses points. If however the number has a lower value or is equal to what he rolled, he was falsely accused by B, therefore B loses points. Point loss is based on the number rolled, not the number claimed.

After this, a new round begins.

## 1.2 The values

Most numbers are ordered by their normal values. However, there are a few exceptions to this rule:

* A 42 is called a Hamburger and has the highest value overall (since it’s the answer).
* A 21 is called a Mäxchen and has the second highest value, only behind the Hamburger (it’s only half of the answer).
* Doubles have a higher value than any other normal number, but are below Mäxchen and Hamburger. The Doubles itself are ordered by their normal value, meaning that a 1-1 is smaller than a 2-2 which is smaller than a 3-3 and so on.

## 1.3 Point loss

The point loss is based on the number that is revealed, not the one that was claimed to be rolled.

* A revealed Hamburger inflicts 3 points.
* A revealed Mäxchen inflicts 2 points.
* Every other number just inflicts 1 point.

# 2. How the game works

One simple heads-up:  
After some tests, we are quite unsure why this happens, but the program seems to have some problems with the python shell from IDLE. We tested it in Pycharm as well as running it through the console which both works without any problems, but for some reasons Mäxchen doesn‘t like IDLE or vice-versa. So just use the console or PyCharm to get the full experience.

So you are using the console? Great! The game starts with an explanation of the settings part. The part is for choosing the rules you like to play with. But don’t worry, there are already pre-set rules so you don’t have to set all rules every game again. Just change the ones you like. For help, type help. To change a setting, press the number. For the more settings information, check [The settings](#_The_settings).

After setting up the rules you can start the game typing “q”. Before you can play, you have to set up the players. Type the number of players you want, then enter the names and let the fun begin! For bots see [The AI](#_The_AI). The random play order is shown before game start.

The game runs in circles. The active player can reveal the number he tossed. After 5 seconds, it is hidden again and he/she has to type the tossed number. It is allowed to lie. The next player has to decide: Is the active one telling the truth? If he/she types “yes”, nothing special happens. Then its his/her turn to throw the dices. If he/she types “no”, the dices are shown. If the told number is bigger than the tossed one, the active player loses points depending on the value of the number. Otherwise the other one loses points. Now it is his/her turn and it goes so on.

If a player has less than 1 point, he/she is out of the game. The other ones still play, until just one remains. This is the winner of the game! The tables with the points in all rounds are displayed at the end. See [Game history](#_Game_history) for more information.

The game does only accept valid input. So you cannot set the Hamburger to “Test123”. Problematic input is just in the name section possible. But there is a hint that you have to choose unique names. If not, the game has no problems. The users are just confused because they do not know who’s turn it is (making the game unplayable in a certain way).

# Upgrades

# The turn order

Normally, the turn order is always the same. But here you have another option. You can change the play direction at game start. And (the more interesting option) you can define that the play order is reversed every time a Mäxchen and/or a Hamburger, as you wish, is revealed. The program handles it with the "play\_order"-item of the settings dictionary

# Not ordered numbers

You can change your rules, so the numbers are not ordered. This means that the first number is allowed to be smaller than the second one. In the game, less Mäxchen and Hamburger will be tossed because of 12 and 24 are no longer converted to 21 and 42. This setting just turns off ordering the numbers. The comparing of numbers is not changed.

# The settings

...

# Visual dices

When the dices are revealed (somebody does not believe a number), the tossed number is shown as the dices. The mechanic is not that complex. Every digit from 1 to 6 has a representation on a dice. They are saved in ui\_help.py. When they are needed, they can be displayed simply in 3 lines in the console.

# Game history

It is interesting to see the evolution of the player points. For this, we implemented a statistic that is shows at the end of the game. For every player, a table is shown, which displays the points in all rounds. Here are just the rounds mentioned, in which a player loses points (If not cheating). Otherwise, it could be very boring to read those tables. The points are saved after every round a player does not believe the last number. They are not in one big table, because the names can be very long so the tables would be very large. The console or the python shell has most of the time a small width. The table would be really ugly. For a better look, the tables are formatted in a way, the points are always in the same columns. The rounds must be lesser than 10000 to make sure the table looks that nice. But first, nobody with a brain would play 10000 or more rounds and second, the game does not crash. It just looks not that nice anymore.

# Cheats

In this game, you are able to cheat. When entering your name at game start, you just have to put the cheat code in front of your name. The cheat codes will not be part of your name later. Please keep in mind that code is case sensitive. Only one active code per game is possible.

The cheat codes are:

* HamToTheBurger: When using this cheat, you are playing with loaded dices. With a chance of 50% you toss a Hamburger (Standard rules: 42).
* MegaMax: This cheat works like the HamToTheBurger cheat, but instead a Hamburger, a Mäxchen (Standard rules: 21) is tossed with a chance of 50%.
* GodKing: The user of this cheat becomes godlike. He/She is not able to lose any points, whatever happens.

# The AI

...

# Change the points

You can change the points in an elegant way: You just change the settings at the game start. The menu points 6 to 8 make it easy to modify the points.

If you really want to change the source code, go to the “points\_worth”-function in mäxchen.py and feel free to change the numbers. You even can calculate the points for all numbers if you want to. Wheather it makes sense for the game is another question...

# Change the “order\_digits” option

///For code and settings (reference)

*Sources? (Delete print e.g.)*

[*https://pynative.com/python-random-shuffle/*](https://pynative.com/python-random-shuffle/) *for shuffle*

*Tests? (e.g. Tables, bots)*