Test cases for „Hnefatafl“

TEST CASES for choose level:

Test cases under optimal circumstances:

File exists and has the right syntax: eg. brandub from moodle(level)

Program asks for a file: „Choose your file: “ and searches for it after input, in this example „brandub“. Program reads the file „brandub“ and saves the varibales. Then it closes „brandub“ and prints the board with the given variables.

Test cases under non optimal circumstances:

File does not exist:

Program asks for a file: „Choose your file: “ and searches for it after input, in this example the file does not exist and the program prints „File does not exist or has wrong syntax.“. You are given the opportunity to choose a new file: „Choose your file: “.

File does exist but with false syntax: eg. SIZE is a (string, float, negative integer or empty), SIZE > 11

Program asks for a file: „Choose your file: “ and searches for it after input. The program reads the wrong syntax and prints the error „File does not exist or has wrong syntax.“. You are given the opportunity to choose a new file: „Choose your file: “.

File does exist but with false syntax: eg. EMPTY is a (integer, float or empty)

Program asks for a file: „Choose your file: “ and searches for it after input. The program reads the wrong syntax and prints the error „File does not exist or has wrong syntax.“. You are given the opportunity to choose a new file: „Choose your file: “.

File does exist but with false syntax: eg. DECORATE:

<char> is: (integer); <int> is: (string, float, negative integer, integer > size(coordinate would be out of index))

Program asks for a file: „Choose your file: “ and searches for it after input. The program reads the wrong syntax and prints the error „File does not exist or has wrong syntax.“. You are given the opportunity to choose a new file: „Choose your file: “.

File does exist but with false syntax: eg. PLAYER_0 -> Look DECORATE

File does exist but with false syntax: eg. PLAYER_1 -> Look DECORATE

File does exist but with false syntax: eg. KING

<char> is (integer); <int> is: (string, float, negative integer, integer > size(coordinate would be out of index)); tuple is a list

For the the first or the second cases: Look DECORATE

For the third: Program auto corrects this issue and prints the board with the given variables

After starting the game, you get to the first window. You have to choose the file.
The test cases are the same we tested already at the top of the document.

Functions that require alphabetical inputs:

Testcases for alphabetical inputs:

1234567 abc123 123abc abc „ “(blank)
99999999999999999999999999, aaaaaaaaaaaaaaaaaaaa

begin_game()	program prints (INVALID_INPUT) and wants new input
single_game()	clearscreen() and wants a new input
names()	abc123, 123abc, abc and aaaaaaaaaaaaaaaaaaaa are correct the rest activates the message (Names have at least 1 letter.) and orders a new input
gameover()	clearscreen() and wants a new input

Functions that require alphabetical/numeric inputs:

Testcases for alphabetical/numeric inputs:

1234567 abc123 123abc abc „ “(blank)
99999999999999999999999999, aaaaaaaaaaaaaaaaaaaa
a75 y7 y100 ab 12b 29c

options()	program resets the frame ui and asks for a valid input in every case
turn()	all inputs are recognized as not valid by try – except; programm gives out the message that it wants valid inputs

Testcases for backward n:

1. its the beginning of the game, n bigger than turns made
2. pawn gets removed in earlier turn
3. you cant travel farther back in time (you are in the beginning)

For 1: program prints an error message and is asking for new coordinates

For 2. program prints the board of the former turn, changes the player and asks for new coordinates/backward n/forward n

For 3. program acts like in case one with the difference that you can forward n.

Testcases for forward n:

1. backward n wasnt executed

2. pawn gets removed in „future“ turn
3. you cant travel farther forth
4. there was a valid move after backward was executed

For 1: program prints an error message and is asking for new coordinates

For 2. program prints the board of the „future“ turn, changes the player and asks for new coordinates/ backward n/forward n

For 3. program acts like in case one

For 4: program acts like in case one

Testcases for coordinates:

1. move on your own field
2. move on an enemy or „x“
3. move over another pawn

For 1: program prints an error message and is asking for new coordinates

For 2: program acts like in case one

For 3: program acts like in case one

Testcase for Race:

abc123, 123abc, abc and aaaaaaaaaaaaaaaaaaaa

activates the program to ask the Question again and orders a new input.

Only h or c can make the program to continue