CS10-SU16: Group 18
Jean Michel Diaz, Yujia Lai, Vedi Chaudhri

Final Project ReadMe: Chopsticks (in Python)

Link to view project: https://youtu.be/MmZz2wAtC98

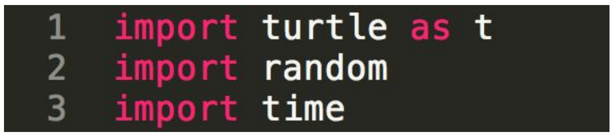
How to start the game:

1. Open Terminal
2. Go to the gamefolder using "cd"
3. Run the Chopsticks!.py file in interactive mode "Python3 -i Chopsticks.py"
4. Put the terminal next to the game window to see both at the same time

How to play the game:

1. The Instructions are the first thing you will see when you open the project. It will tell you what the game is about and how it works
2. During the game you will make your inputs in the terminal. The hands will be displayed in another window. You have to watch both windows parallel to play the game.
3. The game automatically checks if the game is game over. In case it is you will get the respective message in your Terminal.

Technical background:

1. The project is written in the Chopsticks.py file.
2. All graphics we used are located in the project folder as well.
3. We used the libraries turtle, time and random to implement the project. We imported the libraries right in the first lines:


```
import turtle as t
import random
import time
```

4. The .py file is structured in 3 areas to make it easier to understand:

    a. The initialization part: (1) import libraries, (2) set-up the screen, (3) initialize each hand as a turtle & (4) import/register graphics to use as shapes for turtle
    b. Setting global variables that we use later during the game.
    c. All functions that we use later during the game.
    d. The actual game, which makes use of the before defined variables and functions.

5. Overview of the most important functions:
    a. Computerturn() performs the computers turn in different variations depending on the level:
        i. easy(): Just picks the hands for the turn randomly.
        ii. medium(): Makes a winning move if one is possible and thereby makes use of howtowin(). Otherwise just random.
        iii. Hard(): Behaves like medium() just that it before doing a random move checks if it can do moves which do not result in a situation in which the player can get a computer hand out. Thereby the function createlistoffunctionsnottomake(handvalue) is used to check which moves can / can't be done.
    b. Playerturn() is waiting for user input and does the move the user wants to. Thereby it checks if the hands are available for that specific move. If the number on the hand is 5 and the hand is out, you will be asked again to choose a hand, until you chose one that is actually available.
    c. Finishturn() is used to update first the global valueonhand variable and then updates the shapes of the hands.
6. In the file "testing blocks.py" we test two important functions with a testing block similar to the one we used in Snap!.

Complexity:

This project is complex because we have different modes for artificial intelligence, which makes use of different algorithms like moves-not-to-make and how-to-win. The computer has to think in advance in order to make the current move. We largely make use of the powerful while loop in python, which is a big difference from the repeat until loop in Snap, to build the game structure, using abstraction--the actual game code is very precise and clear. Additionally, we had little python experience so it was difficult learning many functions on python. Graphics, for example, were very difficult in python. We had to learn how to use the turtle library.