Hangman Documentation

Release 1.0.0

Yuankun, Jessica, Liesa

November 05, 2013

CONTENTS

This is a short documentation about how to write a game called hangman using Python

Contents:

CHAPTER

ONE

INSTRUCTION

1.1 How to play Hangman

Hangman is a very simple game of guessing a word one letter at a time.

This game only contains **english words** with at least 3 letters and the player has got a limeted number of chances.

1.1.1 Let's get started!

The game ist going to start with the comment:

> "Welcome to our game"

- **At first you'll be asked to press any key to start the game**
  - **press any key**

    or
  - to **exit** type "exit"“
- If you decided to play the game will show you the following options:
  - please give a letter:
    - here you have to guess a letter in order to complete the word
    - the history will show you:
      1. how many tries you've got left
      2. all letters you've guessed
  - enter **888** to exit the game

    or
  - enter **222** to show the answer

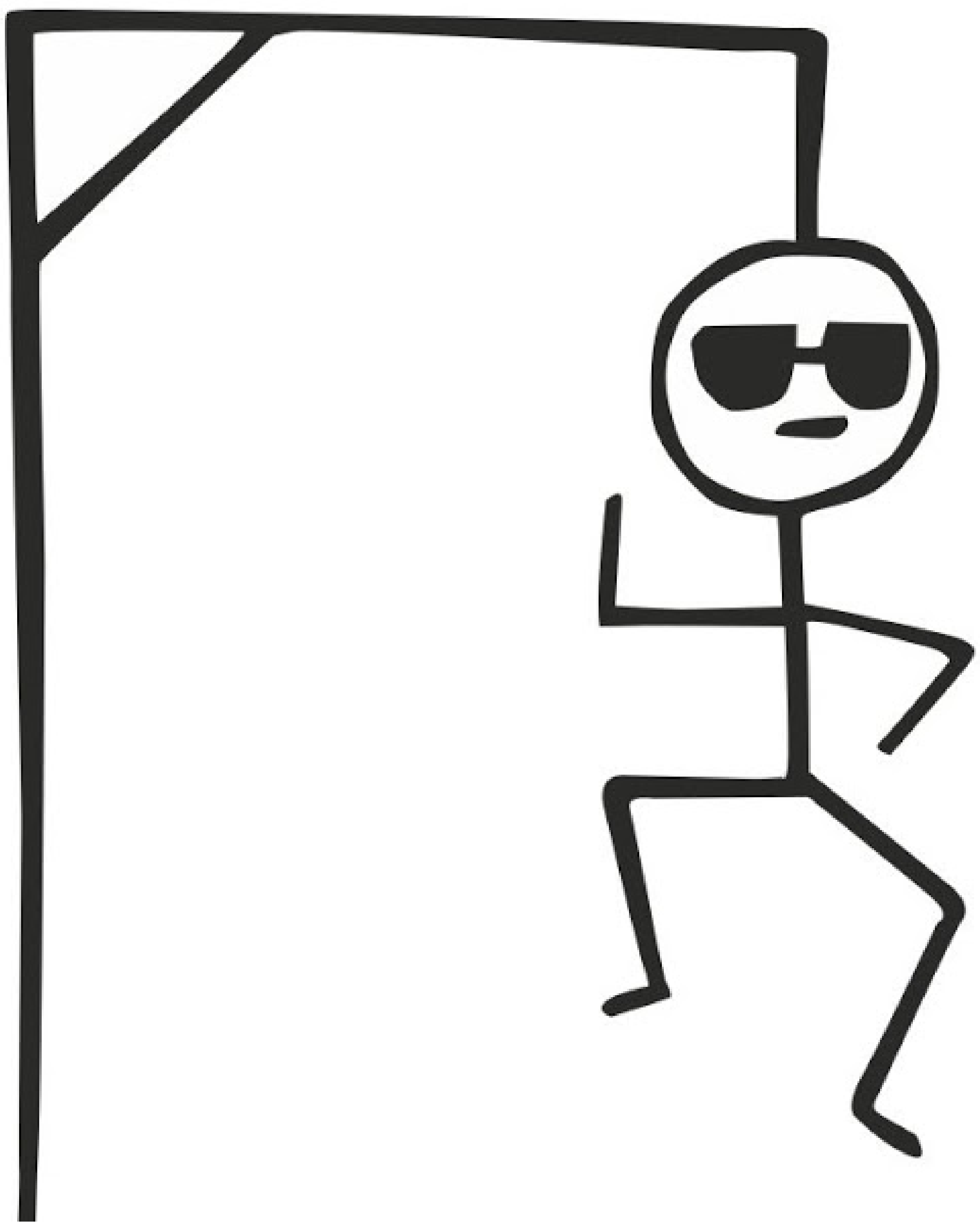
CHAPTER TWO

PROJECT

2.1 codename “Sphinx”

This project contains two important parts:

1. a documentation using Sphinx
2. **development of a source code**
    - in mudular programming
    - coding-style Python (PEP-008)



CHAPTER

THREE

DOCUMENTATION

3.1 Module

Interbuild modules in python:

```
import re
"""regular expression"""
import random
"""create random number"""
import urllib2
"""for the text"""
import os
"""to clean the screen"""
import time
"""timemodule"""
```

3.2 Functions

Main function:

```
def initialGame():
        """configurates the game"""
        wordsList = makeWordsList()
        #print(wordsList[:10])
if askTOStart():
                startGame(wordsList)
```

MakeWordsList:

```
def makeWordsList(fileurl = 'http://www.gutenberg.org/cache/epub/11/pg11.txt',
                                        language = 'english',
                                        minLength = 3):
        """wordlist accepts 3 arguments, the Url, the language, and the minimum length"""
        """important to use try-except, becuase of the nltk-module"""
        """language argument is a default argument"""
        """creates a stopwordlist"""
        """regexpTookenizer to get a clean amount of words"""
        try:
                import nltk
                from nltk.tokenize import RegexpTokenizer
                stopWords = set(nltk.corpus.stopwords.words( language ))
                f = urllib2.urlopen(fileurl)
                tokenizer = RegexpTokenizer(r'\w+')
                cleanWords = list(set(tokenizer.tokenize(f.read()))- stopWords)
        except ImportError :
                print("Error importing nltk, initial game from default txt.")
                with open('input.txt') as f:
                        txt = f.read()
                        cleanWords = txt.split(" ")
        finally:
                        result = [x for x in cleanWords if len(x) >= minLength]
        return result
```

Start:

```
def startGame(wordsList):
        """has got the WordsList as its argument"""
        targetWord = pickTarget(wordsList)
        #print(targetWord)
        guess(wordsList, targetWord)
```

Target:

```
def pickTarget(wordsList):
        """chooses a random word from the generated WordsList"""
        index = random.randint(0,len(wordsList)-1)
        word = wordsList[index]
        return word
```

Guessing:

```
def guess(wordsList, targetWord):
        """everytime just one letter and max. 10 tries"""
        """targetWord is a list"""
        """history as an empty list which saves all guessed letters"""
        """while True= infinit loop"""
        targetWord = targetWord.lower()
        targetWord = list(targetWord)
        returnWord = ['*' for w in targetWord]
        bingoTime = 0
        retryTime = 10
        history = []
        showAnswer = True
        while True:
                clear()
                makeTui(targetWord,bingoTime, returnWord, retryTime, history, showAnswer)
                inputLetter = askLetter().lower()
                if inputLetter == "222":
                                showAnswer = True
                                makeTui(targetWord,bingoTime, returnWord, retryTime, history, showAns
                                askForOneMoreTime(1,wordsList)
                for i in range(0, len(targetWord)):
                        if inputLetter == targetWord[i]:
                                returnWord[i] = inputLetter
                #print(returnWord)
                if ''.join(returnWord) != ''.join(targetWord):
                        history.append(inputLetter)
                        retryTime -= 1
                        if retryTime == 0:
                                askForOneMoreTime(2, wordsList)
                if ''.join(returnWord) == ''.join(targetWord):
                        askForOneMoreTime(1, wordsList)
```

Tui:

```
def makeTui(targetWord, bingoTime, returnWord, retryTime, history, showAnswer):
        """text user interface, for a simple output"""
        info = "{t}\n"
        #info += "Hier ist your No. {bingoTime} word to guess: {s}"
        info += "\t [ {returnWord} ]\t{s}"
        info += "Your have {Time} chance to try. {s}"
        info += "Your history: [ {history} ] . {s}"
        if showAnswer:
                info += "The answer is [ {answer} ]. {s}"
        info += "{t}\n"

        print(info.format(s='\n',t='-'*50,
                                                returnWord = ' '.join(returnWord),
                                                #bingoTime = number + 1,Time = retryTime,
                                                answer = ''.join(targetWord),
                                                history = ",".join(history)))
```

AskLetter:

```
def askLetter():
        """saves the input in the variable inputLetter"""
        print("Enter [888] to exit game;[222] to showAnswer")
        inputLetter = raw_input("Please give a letter: ")
        if inputLetter == "888":
                exit()
        if inputLetter == "222":
                return "222"
        while not re.match("^[a-z]$", inputLetter):
                print("Error! Only one letters from a-z allowed!\n")
                inputLetter = askLetter()
        return inputLetter
```

Ask to start:

```
def askToStart():
        """variable=start"""
        """exit()is a python build-in function which is going to quit python"""
        """if true= the game will start and return to startGame"""
        print("Welcome to our game!")
        start = raw_input("Please press anykey to start game,\nor input [exit] to quit: ")
        if start.lower == "exit":
                print("F**k, why dont you cantinue! I HATE YOU! TMF")
                exit()
        if len(start) > 0 :
                return True
```

Time:

```
def askForOneMoreTime(n, wordsList):
        """1. if-loop: if the user guessed right"""
        """2. if-loop: if the user guessed wrong and the user has to wait 2 sec. to start over with
        if n == 1:
                print("Yeah! You guess right!")
                oneMoreTime = raw_input("Exit? Enter [Y] to exit or [N] to retry one more time: ").lo
                if oneMoreTime in ['y','n']:
                        if oneMoreTime == 'y':
                                exit()
                        if oneMoreTime == 'n':
                                askForOneMoreTime(2, wordsList)
        if n == 2:
                print("Now wait for 2 seconds skip to next word")
                time.sleep(2)
                startGame(wordsList)
```

Clear:

```
def clear():
        import os
        os.system('cls' if os.name=='nt' else 'clear')


intialGame()
```

CHAPTER

FOUR

INDICES AND TABLES

- *search*