**西安电子科技大学**

**《Python程序设计》 课程实验报告**

**实验名称 hangman 游戏实现**

姓名 王越洋 学号 22009200894

|  |
| --- |
| 1. 实验目的   (1)熟练掌握 Python 语言中的循环语句和条件语句。  (2)熟悉字符串的 split()方法。  (3)掌握成员运算符的使用。  (4)熟悉模块 string 中常用的常量。   1. 实验环境   Anconda环境   1. 实验基本原理及步骤   编程实现一个经典的文字游戏 Hangman。第二个玩家是电脑，它将随机生成一个单词。首先我们会实现一些简单的帮助函数，接着实现 hangman 函数，然后通过电脑和玩家的交互来进行整个游戏。  步骤：  (1)读取 words.txt 文件，将单词存储在列表中。  (2)表中随机选择一个单词作为目标单词。  (3)定义函数用于显示当前猜中的字符和未猜中的字符。  (4)在 hangman 函数中使用循环来进行游戏的交互和判断。  (5)在每一轮中，获取玩家的猜测字符并进行相应的处理和判断。  (6)根据猜测结果更新玩家当前猜中的字符列表和未猜中的字符列表。  (7)判断游戏结束的条件，并根据游戏结果进行输出提示。   1. 实验数据记录 2. 实验任务如下：   (1)从输入文件 words.txt 中随机选择一个单词作为目标单词。  (2)在游戏开始之前，告知玩家目标单词的长度。  (3)每一轮，玩家猜一个字符。  (4)立即给玩家反馈所猜的字符是否出现在目标单词中。  (5)每轮结束后，显示玩家当前猜中的字符，以及还未猜过的字符。  (6)每个玩家最多有 8 次猜测机会，每次猜错扣除一次机会。  (7)如果玩家猜的字符之前已经猜过，则提醒玩家重新猜一次。  (8)当玩家猜出单词或机会用尽时，游戏结束。  (9)如果游戏结束时玩家未猜出单词，则显示目标单词。   1. 程序及运行结果(完整截图)：   程序代码：   1. import random 2. def choose\_word(words): 3. """从单词列表中随机选择一个单词""" 4. return random.choice(words) 5. def display\_word(word, guessed\_letters): 6. """显示当前猜中的字符和未猜中的字符""" 7. guessed\_word = '' 8. for letter in word: 9. if letter in guessed\_letters: 10. guessed\_word += letter 11. else: 12. guessed\_word += '\_' 13. return guessed\_word 14. def hangman(): 15. with open('words.txt', 'r') as file: 16. words = file.read().split() 17. word = choose\_word(words) 18. guessed\_letters = [] 19. maxwe = 8 20. chances = maxwe 21. guessed\_right\_letters = [] 22. left\_letters = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j', 'k', 'l', 'm', 'n', 'o', 'p', 'q', 'r', 's', 't', 23. 'u', 'v', 'w', 'x', 'y', 'z'] 24. guessed\_word2 = ["\_", "\_", "\_", "\_", "\_", "\_", "\_", "\_"] 25. print("Welcome to Hangman!") 26. print("The word has {} letters.".format(len(word))) 27. for i in range(100): 28. print("Chances left: {}".format(chances)) 29. print("Guessed  right letters: {}".format(", ".join(guessed\_right\_letters))) 30. print("The letters you have not use: {}".format(", ".join(left\_letters))) 31. guess = input("Guess a letter: ").lower() 32. if chances == 7: 33. print(" +---+\n") 34. print("     |\n") 35. print(" 0   |\n") 36. print("     |\n") 37. print("     |\n") 38. print("     |\n") 39. print("========\n") 40. elif chances == 6: 41. print(" +---+\n") 42. print(" |   |\n") 43. print(" 0   |\n") 44. print("     |\n") 45. print("     |\n") 46. print("     |\n") 47. print("========\n") 48. elif chances == 5: 49. print(" +---+\n") 50. print(" |   |\n") 51. print(" 0   |\n") 52. print("/    |\n") 53. print("     |\n") 54. print("     |\n") 55. print("========\n") 56. elif chances == 4: 57. print(" +---+\n") 58. print(" |   |\n") 59. print(" 0   |\n") 60. print("/|   |\n") 61. print("     |\n") 62. print("     |\n") 63. print("========\n") 64. elif chances == 3: 65. print(" +---+\n") 66. print(" |   |\n") 67. print(" 0   |\n") 68. print("/|\  |\n") 69. print("     |\n") 70. print("     |\n") 71. print("========\n") 72. elif chances == 2: 73. print(" +---+\n") 74. print(" |   |\n") 75. print(" 0   |\n") 76. print("/|\  |\n") 77. print("/ \  |\n") 78. print("     |\n") 79. print("========\n") 80. elif chances == 1: 81. print(" +---+\n") 82. print(" |   |\n") 83. print("[0   |\n") 84. print("/|\  |\n") 85. print("/ \  |\n") 86. print("     |\n") 87. print("========\n") 88. elif chances == 0: 89. print(" +---+\n") 90. print(" |   |\n") 91. print("[0]  |\n") 92. print("/|\  |\n") 93. print("/ \  |\n") 94. print("     |\n") 95. print("========\n") 96. if len(guess) != 1 or not guess.isalpha(): 97. print("Invalid input. Please enter a single letter.") 98. continue 99. if guess in guessed\_letters: 100. print("You have already guessed that letter. Try again.") 101. continue 102. guessed\_letters.append(guess) 103. if guess not in word: 104. chances -= 1 105. left\_letters.remove(guess) 106. print("Wrong guess!") 107. else: 108. left\_letters.remove(guess) 109. guessed\_word = display\_word(word, guessed\_letters) 110. guessed\_right\_letters.append(guess) 111. print("Correct guess!") 112. print(guessed\_word) 113. if "\_" not in guessed\_word: 114. print("Congratulations! You guessed the word{}!".format(word)) 115. break 116. if chances == 0: 117. print("Game over! The word was: {}".format(word)) 118. break 119. hangman()   运行截图：        思路：  首先使用 choose\_word 函数从单词列表中随机选择一个单词作为目标单词。  初始化猜测的字母列表 guessed\_letters、剩余机会次数 chances、已猜对的字母列表 guessed\_right\_letters 和剩余可猜测的字母列表 left\_letters。  打印欢迎信息和目标单词的长度。  进入游戏循环，在每一轮中：  打印剩余机会次数、已猜对的字母和剩余可猜测的字母列表。  提示玩家输入一个字符进行猜测。  检查输入的字符是否有效。  如果字符已经猜过，提示玩家重新猜测。  将猜测的字符加入到 guessed\_letters 列表中。  检查猜测的字符是否出现在目标单词中。  根据猜测情况更新剩余机会次数、剩余可猜测的字母列表和已猜对的字母列表。  检查是否胜利（即目标单词已完全猜中）：  如果胜利，打印胜利信息，结束游戏循环。  检查是否失败（即剩余机会次数已用完）：  如果失败，打印失败信息和目标单词，结束游戏循环。  3.问题记录：  (1)输出已猜出字母的位置需要进行判定  (2)剩余字母需要从列表中删除  (3)结束条件需要考虑到该单词中可能存在两个以上的相同字母，故可利用”\_”数目判定  (4)输出图案时要注意chances的次数匹配   1. 实验结果分析   游戏开始时，会随机选择一个单词作为目标单词，并告知玩家目标单词的长度。  玩家每轮可以输入一个字母进行猜测，程序会验证输入的有效性。  如果猜测的字母已经猜过，程序会提示玩家重新猜测。  每轮猜测后，程序会给出反馈，告知玩家猜测的字母是否出现在目标单词中。  程序会记录玩家的猜测字母列表，以及已经猜对的字母列表。  每轮结束后，程序会显示玩家当前猜对的字母以及剩余未猜对的字母。  玩家最多有 8 次猜测机会，每次猜错会减少一次机会。  如果玩家成功猜出目标单词，则显示胜利信息。  如果玩家的机会用完仍未猜出目标单词，则显示失败信息，并显示正确的目标单词。  实验结果可能有以下几种情况：  (1)玩家成功猜出目标单词，游戏显示胜利信息。  (2)玩家的机会用完仍未猜出目标单词，游戏显示失败信息，并显示正确的目标单词。  (3)玩家在给定的机会内猜对了部分字母，但未完全猜出目标单词。  (4)玩家输入无效的字符，程序提示无效输入并要求重新输入。  (5)玩家重复猜测已经猜过的字母，程序提示已猜过，并要求重新输入。  根据这些结果，玩家可以根据提示和反馈来逐步猜测目标单词，同时要注意机会的使用和有效输入的判断。游戏结果会根据玩家的猜测和运气而有所不同，每次游戏都是独立的，所以结果也会有一定的随机性。   1. 实验代码和相关的输入输出文件（附件） |