

C0A21006 / ProjExD Public

[Code](#) [Issues](#) [Pull requests](#) [Actions](#) [Projects](#) [Wiki](#) [Security](#) [Insights](#)

main

...

[ProjExD](#) / [ex01](#) / [alphabet.py](#) / [Jump to](#)

C0A21006 ver.1.1

[History](#)

1 contributor

52 lines (44 sloc) 2.15 KB

...

```
1 import datetime
2 import random
3
4 NUM_OF_TRIALS = 5 #最大繰り返し数
5 NUM_OF_ALL_CHARS = 10 #対象文字数
6 NUM_OF_ABS_CHARS = 2 #欠損文字数
7
8 def main():
9     st = datetime.datetime.now() #時間計測開始
10    for i in range(NUM_OF_TRIALS):
11        seikai = shutsudai()
12        f = kaitou(seikai)
13        if f == 1: #完全正解なのでbreakする
14            break
15    ed = datetime.datetime.now() #時間計測終了
16    print(f"所要時間: {(ed-st).seconds}秒かかりました")
17
18 def shutsudai():
19     #全アルファベット文字のリスト
20     alphabets = [chr(c+65) for c in range(26)] #chr(コード)->文字
21     #全アルファベットからNUM_OF_ALL_CHARS個の文字をランダムに選ぶ: 対象文字
22     all_char_lst = random.sample(alphabets, NUM_OF_ALL_CHARS)
23     print(f"対象文字: {all_char_lst}")
24
25     #対象文字からNUM_OF_ABS_CHARS個の文字をランダムに選ぶ: 欠損文字
26     abs_char_lst = random.sample(alphabets, NUM_OF_ABS_CHARS)
27     print(f"欠損文字: {abs_char_lst}") #欠損文字の正解なので、後で非表示にする
28
29     #対象文字から欠損文字を除いたものを表示する: 表示文字
30     pre_char_lst = [c for c in all_char_lst if c not in abs_char_lst]
31     print(f"表示文字: {pre_char_lst}")
32
33     return abs_char_lst #正解欠損文字をmain関数に返す
34
35 def kaitou(seikai):
36     num = int(input("欠損文字はいくつあるでしょうか?"))
37     if num != NUM_OF_ABS_CHARS:
38         print("不正解です。またチャレンジしてください")
```

```
39         return 0 #文字数回答で不正解の場合
40     else:
41         print("正解です。それでは具体的に欠損文字を1つずつ入力してください")
42         for i in range(NUM_OF_ABS_CHARS):
43             c = input(f"{i+1}つ目の文字を入力してください：")
44             if c not in seikai:
45                 print("不正解です。またチャレンジしてください")
46                 return 0 #文字回答で不正解の場合
47             seikai.remove(c)
48         print("正解です。ゲームを終了します")
49         return 1 #完全正解の場合
50
51 if __name__ == "__main__":
52     main()
```