第4回演習課題

提出時の注意

- student_id及びファイル名に学籍番号を設定すること。
- Pythonファイル中の型指定を守ること。
- 提出用のPythonファイルのグローバルスコープには、余分な関数呼び出しを書かないこと。デバッグ用に書いたものはすべてコメントアウトすること。Pythonファイルを実行した際に何も実行されないのが正しい状態です。

```
def func(a, b):
    ...
func(1, 2) <--これが呼び出しです。これをグローバルスコープに書かないでください。
```

演習１：ISBN

国際標準図書番号（ISBN）は、図書に割り当てられる10桁の番号です。右端の数字はチェックサムであり、d_iを（右から）i桁目の数字としたとき、$d_1 + 2d_2 + 3d_3 + \ldots + 10d_{10}$が11の倍数となるよう設計されています。$d_1$は、0から10のどれかの値を取り、10のときは"X"で表現します。9桁のintを引数にとり、チェックサムを計算の上10桁のISBNを出力する関数を書いてください。

例：020131452のチェックサムd_1は以下より5になります。

$(d_1 + 2*2 + 3*5 + 4*4 + 5*1 + 6*3 + 7*1 + 8*0 + 9*2 + 10*0)\%11 = 0$

テストケース：

```
isbn(20131452)
'0201314525'
isbn(12345672)
'012345672X'
```

```
def isbn(n):
    """
    引数 n: int
    返値 str
    """
    # 以下にプログラムを記述。
    return XXX #string型
```

演習２：文字列操作

文字列`text`、長さ`length`、文字`char`を引数にとり、

- `text`が`length`以上の場合、頭から`length`文字列を返す
- `text`が`length`より小さい場合、`char`で埋めて`length`の長さの文字列を返す

関数`shape_text`を作ってください。

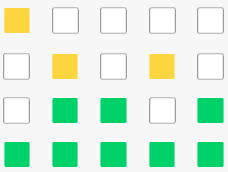
テストケース：

```
shape_text("abc", 5, "?")
abc??
shape_text("abcdef", 5, "?")
abcde
```

```
def shape_text(text, length, char):
    """
    引数 text: str, length: int, char: str(1文字)
    返値 str
    """
    # 以下にプログラムを記述。
    return XXX #String型
```

演習３：Wordleを作ろう

Twitterなどで次のような投稿を見たことはありますか？



これはwordleというゲームです。初見の方は[こちら]で遊んでみてください。この演習ではwordleを自分で作ってみます。

ステップ１

２つの文字列`correct_ans, ans`を引数にとり、`ans`の各文字について次のルールに従って組み立てられる文字列をprintする関数`wordle_check`を作ってください。

- 文字が`correct_ans`に含まれていて位置も同じ場合「🟩」

- 文字が`correct_ans`に含まれているが位置が違う場合「🟨」
- 文字が`correct_ans`に含まれていない場合「⬜」

また、

- `correct_ans`と`ans`の文字の長さが等しくないとき、演習2の`shape_text`を使用して文字列を`correct_ans`と同じ長さにすること
- その際、足りない文字列は"?"で埋めること
- それぞれの🟩（全色）のあとには半角スペースを入れること（最後の⬜も含む）

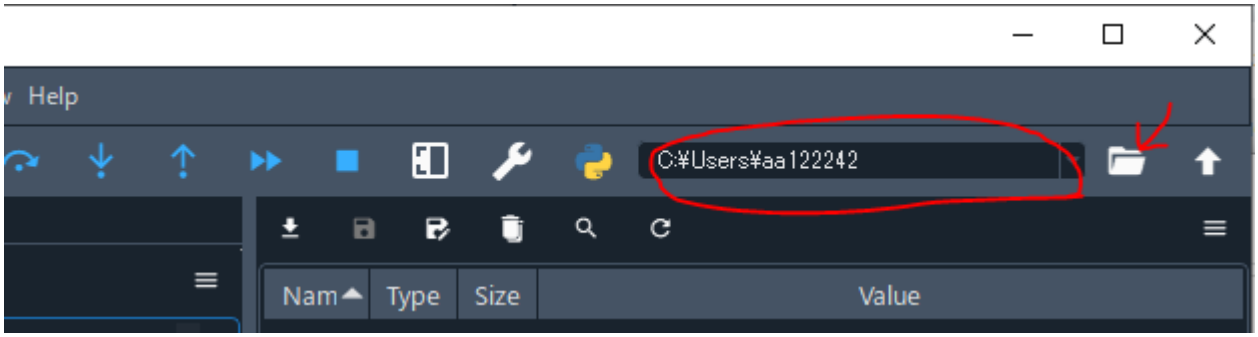
テストケース：

```
wordle_check("school", "ocohod")
🟨🟩🟨🟨🟩⬜
wordle_check("school", "sch")
🟩🟩🟩⬜⬜⬜
```

```
def wordle_check(correct_ans, ans):
    """
    引数 correct_ans: str, ans: str
    返値 なし
    """
    pass # 以下にプログラムを記述。この行は削除すること。
```

ステップ 2

ステップ 1 で作った関数を使って、最も基本的な部分のみを実装したバージョンである wordle_basicを作ります。 特定の長さの単語のリストを読み込む関数`load_words_of_length(length)`とランダムな単語を選択する関数`choose_word(wordlist)`を用意しました。実行にあたっては、必ず「words.txt」が同じフォルダにあることを確認してください。Spyderの右上のフォルダアイコン（下図の赤矢印）をクリックして、Pythonファイルとwords.txtが入っているフォルダを選択し、丸で囲まれたボックスがそのフォルダを示していることを確認してください。（Spyder以外を使っている人は、ワーキングディレクトリを適切に設定してください。）



wordle_basic()は次のような挙動をします。

- load_words_of_length(5)で、５文字の文字列のリストを取得します。
- choose_word(wordlist)で、wordlistから一つ単語を選びます。
- 試行回数は６回です。
- 一回の試行で次のことをします。
    - "Attempt #1: "のように入力を求めます。1の部分は試行回数です。
    - 入力文字列をwordle_checkで評価し、結果をプリントさせます。
    - もし、正解なら、ループを抜け、"Correct!"と表示します。
- ６回とも不正解の場合、"Correct answer was {correct_ans}"と表示します。

とりあえずは、ユーザーの入力が６文字以上の文字列だったり、英単語でなかった場合は無視して、wordle_checkを試してみましょう！プログラミングは、絵と同じで、（例えば自転車を描くことに当てはめて）タイヤからみっちり細かく描き始めるより、全体の自転車のラフスケッチから入ることがとても重要です。

ここで、choose_wordは正解の文字列をランダムに選ぶため、デバッグが大変かもしれません。その場合は、SEEDに番号を指定すると、毎回決まった文字列が選ばれるようになります。

テストケース：

```
SEED=1
Loading word list from file...
   4738 words loaded.
Attempt #1: place
⬜ ⬜ ⬜ 🟩 ⬜
Attempt #2: device
🟩 ⬜ ⬜ 🟨 🟨
Attempt #3: disco
🟩 🟩 🟩 🟩 🟩
Correct!

SEED=1
Loading word list from file...
   4738 words loaded.
Attempt #1: dis
🟩 🟩 🟩 ⬜ ⬜
Attempt #2: discoaaa
🟩 🟩 🟩 🟩 🟩
Attempt #3: d i s
🟩 ⬜ 🟨 ⬜ 🟩
Attempt #4:
⬜ ⬜ ⬜ ⬜ ⬜
```

```
Attempt #5: co
🟨 🟨 ⬜ ⬜ ⬜
Attempt #6: c
🟨 ⬜ ⬜ ⬜ ⬜
Correct answer was disco
```

```
import random
WORDLIST_FILENAME = "words.txt"
SEED = 1
def load_words_of_length(length):
    """
    Returns a list of valid words of length l. Words are strings of lowercase letters.

    Depending on the size of the word list, this function may
    take a while to finish.
    """
    print("Loading word list from file...")
    # inFile: file
    inFile = open(WORDLIST_FILENAME, 'r')
    # line: string
    line = inFile.readline()
    # wordlist: list of strings
    wordlist = sorted([word for word in line.split() if len(word) == length])
    print("  ", len(wordlist), "words loaded.")
    return wordlist

def choose_word(wordlist):
    """
    wordlist (list): list of words (strings)

    Returns a word from wordlist at random
    """
    if SEED:
        random.seed(SEED)
    return random.choice(wordlist)

def wordle_basic():
    """
    引数・返値 なし
    """
    pass # 以下にプログラムを記述。この行は削除すること。
```

ステップ 3

二分探索

wordle_basicでは、入力された文字列が存在するものなのか判別しませんでした。さて、ここではルールを標準のものから少しかえ、５文字未満の文字列が入力されたときも、それをprefixとして持つ（つまり、その文字列から始まる）単語が存在すれば、正しい入力として受け付けたいと思います。

二分探索を用いて、単語リスト`wordlist`に変数`prefix`から始まる単語が存在するか判別する関数`binary_prefix_search`を書いてください。

リストはまだやってない？？大丈夫！この演習では以下２つの基本的な操作（要素の取得、長さの取得）を知っていればOKです。

```
# リストはn個の要素（型は任意）の順序付けされた配列です。添字は0からn-1までで
a = [0, 2, 4, 8] # というリストがあった場合、その要素は
a[1]
2
# 長さは
len(a)
4
```

二分探索を要素を取得しながらリストの添字に対して行えば、あとは授業でやったアルゴリズムが応用できます。

ヒント：

- Pythonでは、文字列の大小比較を行うことができます。

  ```
  print("a" < "b") # True
  print("aa" < "ab") # True
  print("abc" < "abcd") # True
  ```

- prefixのみ比較したい場合どうしたらよいだろう？
- リストの長さは、`len()`で取得できます。
- 二分探索のヒント
  1. 探索範囲の中心にある項目と探したい項目を比較
  2. 配列の中心にある項目が探したい項目より大きければ中心より左側を、小さければ中心より右側を新たな探索範囲とする。もし同じであれば見つけたい項目があったことになるので探索は終了
  3. 新たな探索範囲に対して、1番の処理から同じことを繰り返す

テストケース：

```
binary_prefix_search(["a", "ab", "abcd", "b", "cd"], "ab")
True
binary_prefix_search(["a", "ab", "abcd", "b", "cd"], "abc")
True
binary_prefix_search(["a", "ab", "abcd", "b", "cd"], "abcde")
False
binary_prefix_search(["a", "ab", "abcd", "b", "cd"], "ad")
False
```

```
def binary_prefix_search(array, prefix):
    """
    引数 array: list[str], prefix: str
    返値 bool
    """
    # 初期探索範囲は配列の左端から右端まで
    left_index = 0
    right_index = len(array) - 1

    while True:
        if ?:
            # 探索範囲がなくなったときは見つからなかったものとする
            return False

        center_index = ?
        if ?:
            right_index = ?
        elif ?:
            left_index = ?
        else:
            # 探索終了
            return True
```

ステップ４

ステップ２で作った基本版をもとに、`binary_prefix_search`を使用して、wordleの入力が`load_words_of_length(5)`に含まれるか判別する改良版`wordle_standard`を作ってください。

- 入力文字列長が5より大きい場合と、word listに含まれない場合、"Invalid input"と表示し、残り試行数を減少させずに、再度入力を求める。
- 入力文字列が5以下の場合かつ、prefixとしてword listに含まれる場合、残り試行数を減少させて`wordle_check`を実行。

テストケース：

```
SEED=1
Loading word list from file...
```

```
   4738 words loaded.
Attempt #1: ddd
Invalid input
Attempt #1: discodi
Invalid input
Attempt #1: disc
🟩 🟩 🟩 🟩 ⬜
Attempt #2: disco
🟩 🟩 🟩 🟩 🟩
Correct!
```

```
def wordle_standard():
    """
    引数・返値 なし
    """
    pass # 以下にプログラムを記述。この行は削除すること。
```

力試し課題：カレンダー再び

この課題は提出しないでください。

この課題の難易度は中級です。プログラミング初学者でもいままでの知識で解くことができるので、積極的に挑戦してください。

問題：以下のようなカレンダーを出力する関数drawCalenderを作ってください。

テストケース：

```
drawCalender(2020, 2)
2020/2
 S  M Tu  W Th  F  S
                   1
 2  3  4  5  6  7  8
 9 10 11 12 13 14 15
16 17 18 19 20 21 22
23 24 25 26 27 28 29

drawCalender(2000, 2)
2000/2
 S  M Tu  W Th  F  S
       1  2  3  4  5
 6  7  8  9 10 11 12
```

```
13 14 15 16 17 18 19
20 21 22 23 24 25 26
27 28 29


drawCalender(2100, 2)
2100/2
 S  M Tu  W Th  F  S
    1  2  3  4  5  6
 7  8  9 10 11 12 13
14 15 16 17 18 19 20
21 22 23 24 25 26 27
28
```

```
# こちらに自由に関数を定義してよい。

def drawCalender(year, month):
    pass # 以下にプログラムを記述。この行は削除すること。
```

暇つぶし用課題：いい（１１）感じのn進数

この課題は提出しないでください。

問題：3以上の整数xのn進数表現を$(x)_n$と表すとき、$(x)_n$がすべて1で表現されることを「いい感じになる」と定義します。$(x)_n$がいい感じになる最小のnを求めてください。なお、$x < 10^{18}, n >= 2$である。

例：

$$(13)_{10} \to (111)_3$$
$$(15)_{10} \to (1111)_2$$
$$(20)_{10} \to (11)_{19}$$

したがって、

```
calc_smallest_good_n(13)
3
calc_smallest_good_n(15)
2
calc_smallest_good_n(20)
19
```

必要に応じて、以下の関数を用いても良い。

```
import math
math.log(a, b) # bを底とするaの対数を返す。
```

テストケース：

```
calc_smallest_good_n(23227344339325621)
12345
check_answer(calc_smallest_good_n)
# テストケースに全問正解の場合、メッセージが現れます
```

```
import math

def check_answer(fn):
    import hashlib
    test_input = [23227344339325621, 13247354345325761, 9668886503657665,
                  75047317648499560, 85037317649971934, 28531167061]
    x = "".join(map(str, map(fn, test_input)))
    cypher = 113362408063145597568669130889891273767
    key = hashlib.md5(bytes(x, 'utf-8')).hexdigest()
    decrypted_hex = cypher ^ int(key, 16)
    try:
        return bytes.fromhex(hex(decrypted_hex)[2:]).decode()
    except:
        return "Decoding error"
```