日商プログラミング検定 EXPERT(Python) サンプル問題

知識科目

第1問 （知識 4 択：20 問）

1. 結果が False になる式はどれか。次の中から選びなさい。

① bool([])
② bool(1.0)
③ not False
④ bool('4A')

2. 以下の式の実行結果はどれか。次の中から最も適切なものを選びなさい。

```
[[1,2,3] , [4,5,6] , [7,8,9]][1:2]
```

① [[4, 5, 6]]
② [[2, 5, 8]]
③ [4, 5, 6]
④ [2, 5, 8]

3. 以下の式を実行したとき、エラーにならず結果が求まるものは何個あるか。次の中から選びなさい。

```
[1, 2] + 3
[1, 2] + [3]
[1, 2] * 3
[1, 2] / 3
```

① 0 個
② 1 個
③ 2 個
④ 3 個

4. 以下のプログラム片を実行したとき、図の結果を得るために(A)に入れる句はどれか。次の中から最も適切なものを選びなさい。

```
x = {'A':12.5, 'F':11.0, 'S':9.7, 'T':9.5}
for i , v in      (A)      :
  print('{}:{}'.format(i , v))
```

【実行結果】
```
A:12.5
F:11.0
S:9.7
T:9.5
```

① x.items()
② range(len(x))
③ enumerate(x)
④ x

5. 以下の式の実行結果はどれか。次の中から選びなさい。

```
print('{:#02}{:#02}'.format(12,7))
```

① 1207
② 12 7
③ 127
④ 12 , 7

6. 以下のプログラム片の実行結果はどれか。次の中から選びなさい。

```
list(range(-3 , 4 , 3))
```

① [-3, 0, 3]
② [-3, -2, -1, 0, 1, 2, 3]
③ [-3, -2, -1, 0, 1, 2, 3, 4]
④ [-3, 0, 3, 4]

7. 以下のプログラム片を実行したあとの変数 x の内容はどれか。次の中から選びなさい。

```
x = [[5.4, 9.3, 6.0], [3.4, 3.5, 3.5], [7.03, 7.05, 4.65]]
x.sort()
```

① [[3.4, 3.5, 3.5], [5.4, 9.3, 6.0], [7.03, 7.05, 4.65]]
② [[3.4, 3.5, 3.5], [5.4, 6.0, 9.3], [4.65, 7.03, 7.05]]
③ [[3.4, 3.5, 3.5], [4.65, 7.03, 7.05], [5.4, 6.0, 9.3]]
④ [[3.4, 4.65, 3.5], [3.5, 5.4, 6.0], [7.03, 7.05, 9.3]]

8. 以下のリスト data1 内のすべての値を 2 倍してリスト data2 を求める式はどれか。次から最も適切なものを選びなさい。

```
data1 = [13.86, 9.46, 5.61]
```

① data2 = [x*2 for x in data1]
② data2 = data1 * 2
③ data2 = [data1]*2
④ data2 = x*2 for x in data1

9. 以下のプログラム片を実行し、結果が True となる type と code の組み合わせはどれか。次の中から選びなさい。

```
if not(type != 'A' or code<3):
  print(True)
else:
  print(False)
```

① type:'A' code:3
② type:'A' code:2
③ type:'B' code:3
④ type:'B' code:4

10. 以下のプログラム片を実行したとき、「B」と表示する key の値はどれか。次の中から選びなさい。

```
data = [10.31, 8.50, 5.30, 3.00, 7.30, 3.30]
for x in data:
  if x == key:
    print('A')
    break
else:
  print('B')
```

① 3.2
② 3.0
③ 10.31
④ 「B」と表示されることはない

11. 次のプログラム片を実行して得られるリスト all の要素数はいくつか。次の中から選びなさい。

```
colors = ['red', 'blue', 'yellow', 'cyan']
all = ['{}-{}'.format(c1, c2) for c1 in colors for c2 in colors]
```

① 16 個
② 8 個
③ 4 個
④ 2 個

12. 以下のプログラム片の実行が終了したときの変数 i とリスト x についての説明として正しいものはどれか。次の中から最も適切なものを選びなさい。

```
def inc(a, n):
  n += 1
```

```
  a[n] += 1

i = 2
x = [0,0,0,0,0]
inc(x , i)
```

① i の値は変化しないが、x の要素の１つは変化している。
② i の値は１加算されるが、x の要素は変化しない。
③ i の値は１加算され、x の要素の１つは変化している。
④ i の値も x のどの要素も変化しない。

13. 以下の関数を呼び出す文として実行可能であるものはどれか。次の中から正しい組み合わせを選びなさい。

```
def get_score(bv , mul = 1) :
  return bv * mul

(ア)get_score(8.0 , 1.1)
(イ)get_score(8.0)
(ウ)get_score(8.0 , 'A')
(エ)get_score()
```

① （ア）と（イ）
② （ア）と（ウ）
③ （ア）
④ （ア）と（イ）と（エ）

14. 以下のプログラム片を実行した結果はどれか。次の中から最もふわさしいものを選びなさい。

```
def disp(*s):
  for one in s:
    print(one)

disp(['red' , 'green'])
disp('red' , 'green')
```

①
```
['red', 'green']
red
green
```

②
```
['red', 'green']
red , green
```

③
```
['red', 'green']
['red', 'green']
```

④
```
red
green
red
green
```

15. 以下のプログラム片を実行したとき、変数 n には何が格納されているか。次の中から選びなさい。

```
class Sample :
  def __init__(self , s) :
    self.x = [s] * 4
  def add_x(self , s) :
    self.x.append(s)
  def get_n(self) :
    return len(self.x)

sample = Sample('A')
sample.add_x('S')
n = sample.get_n()
```

①5
②['S']
③4
④['A' , 'A' , 'A' , 'A']

16. 以下のプログラムを実行した。①は何回実行されたか。次の中から最も適切なものを選びなさい。

```
s_data = ['0', '1', '-', '2']
i_data = []
for c in s_data:
    try :
        x = int(c)
    except Exception :
        x = -1          //①
    finally :
        i_data.append(x)
```

①0回
②1回
③2回
④3回

17. 以下の式を実行したとき、c の構造はどれか。次から最も適切なものを選びなさい。

```
import numpy as np
b = np.arange(1,7)
c = b.reshape(2,3)
```

①

1	2	3
4	5	6

②

1	2
3	4
5	6

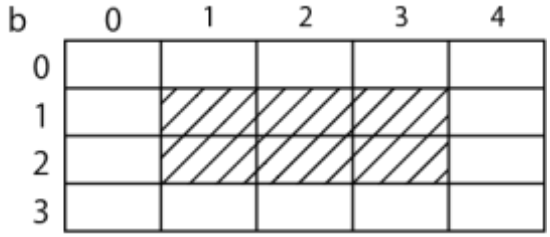
③

1	2	3	4	5	6

④

1
2
3
4
5
6

18.　以下の図は Numpy の配列 b の構造を表している。斜線部を取り出す式はどれか。次の中から選びなさい。



①b[1:3 , 1:4]
②b[1:3 , 1:2]
③b[1:4 , 1:3]
④b[1:2 , 1:3]

19.　次のような DataFrame が変数 df に格納されているとき、変数 df1 に'place'が札幌である行を取り出すことができる文はどれか。次の中から選びなさい。

	year	place
0	2012	札幌
1	2013	埼玉
2	2014	長野
3	2015	札幌

① df1 = df[df['place'] == '札幌']
② df1 = df[place == '札幌']
③ df1 = df['place' == '札幌']
④ df1 = [df['place'] == '札幌']

20.　データフレーム df に図のような情報が格納されているとき、次の式を実行して得られる結果についての説明はどれか。次の中から最も適切なものを選びなさい。

	score1	score2
0	106.33	216.07
1	110.95	219.46
2	102.63	183.73

```
df.apply(lambda row : row['score1'] + row['score2'] , axis=1)
```

① 行ごとの合計を求める。
② 列ごとの合計を求める。
③ すべての数値の合計を求める。
④ 第 0 行の合計を求める。

第2問 （穴埋め　2 問）

【問題1】

二値画像において、連続している黒い画素に同じラベル（番号）を付与するプログラムである。二値画像とは、白または黒の 2 種類のみで構成される画像のことであり、その一つひとつを画素という。図 1 のサンプル画像では、縦 20 横 20 の 400 個の画素に白または黒が配置されている。注目画素に対して隣接する上下左右の画素に同じラベルを付与する。これを 4 連結という（**図** 2）。図 1 のサンプル画像では、1 から 5 までのラベルが付与され、5 つの領域に分けることができる。

画像情報は白の画素を「1」、黒の画素を「0」とする 2 次元配列 img で与えられる。また、縦・横ともに最も外側の画素はすべて黒であるものとする。

同じサイズの 2 次元配列 labelling に、付与されたラベルを格納する。黒の画素にはラベル 0 を付与する。図 1 のサンプル画像では、labelling の各要素には 0 から 5 のいずれかが格納される。このような処理をラベリングという。

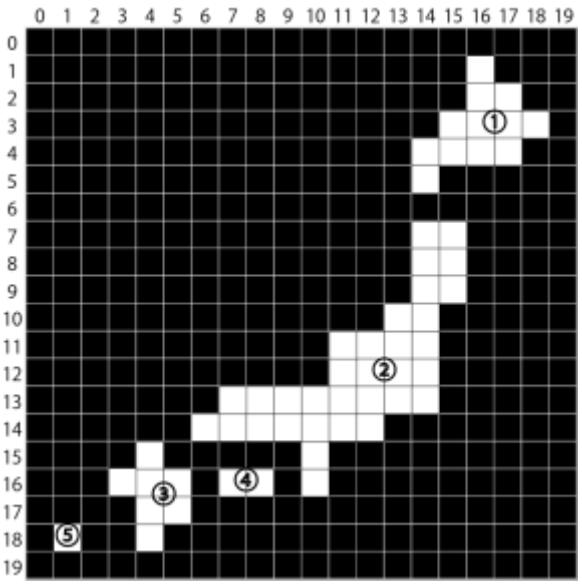


図 1 サンプル画像

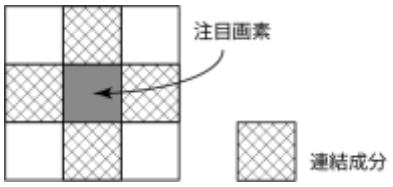


図 2　4 連結

ラベリングの手順は以下のとおりである。

1. 2 次元配列 labeling のすべての要素を 0 で初期する。対応を参照するリスト tbl を用意し、初期値として 0 を与える。初期状態における tbl の要素数は 1 個である。

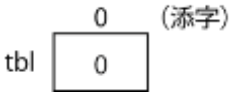


図 3　参照リストの初期化

2.　２次元配列 img を順にスキャンし、みつかった「1」の画素を注目画素とする。

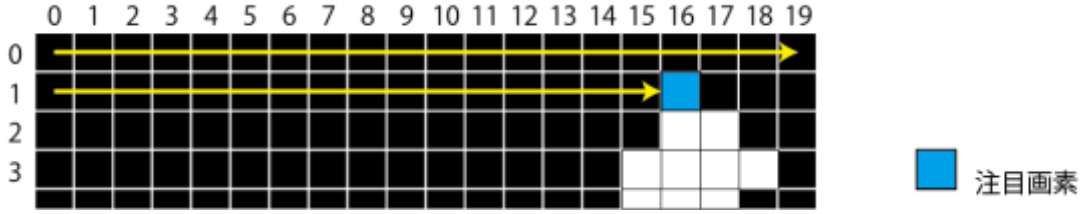
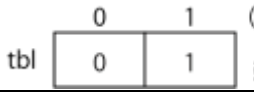
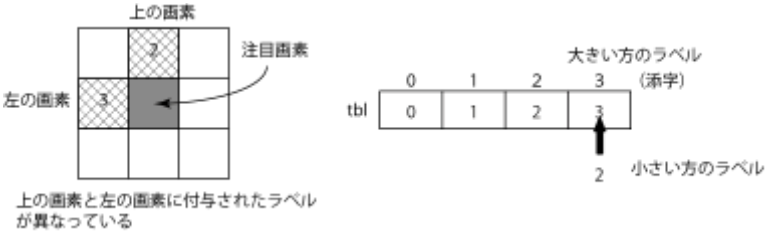


図 4　手順 2

3.　注目画素に対し、注目画素の上の画素と左の画素の状態に応じて表 1 の処理を行う。

表 1　注目画素に対する処理

上の画素	左の画素	処理
0	0	新規のラベルを付与。参照リストにラベルを追加する。 
1	0	上の画素と同じラベルを付与。
1	1	上の画素と同じラベルを付与。付与されたラベルが左の画素と異なる場合には、大きい方のラベルを添字とする参照リストの要素を小さい方のラベルの値で書き換える。 
0	1	左の画素と同じラベルを付与。

4.　すべての画素に対し、2.と 3.を行う。

5.　連続領域でありながら、異なるラベルが付与された画素について、参照リストにしたがってラベルを補正する。図 5 はその一例である。これに先立って、参照リストを統合する（図 6)。

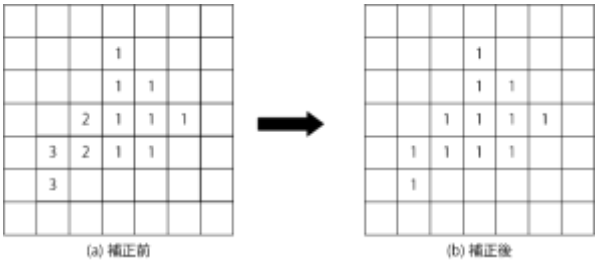


図 5 ラベルの補正

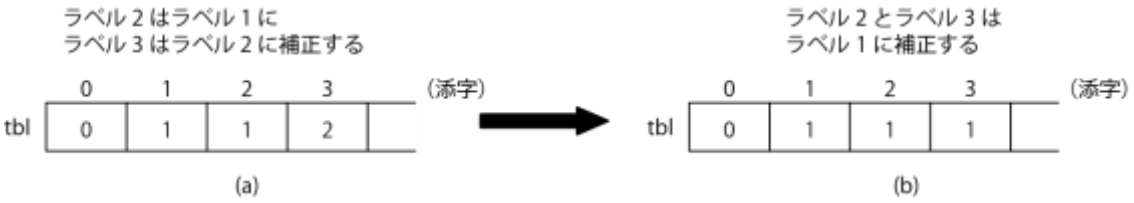


図 6　参照リストの統合

二次元配列 img と labeling は Numpy の配列を利用する。

Python では、以下の関数が利用できる。

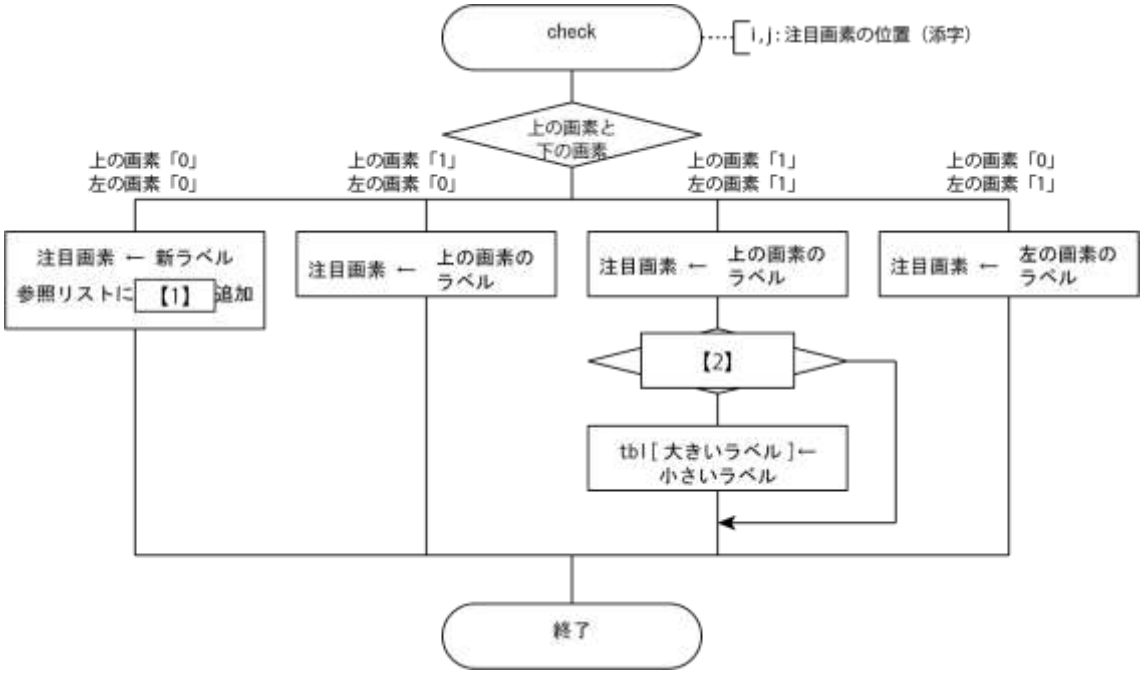
numpy.zeros(shape , dtype=データ型) shape : 縦の画素数と横の画素数のタプル shape で指定したサイズの配列を用意し、すべての要素の値を 0 にする。
list.append(x) list の要素の最後に x を追加する。
list.index(x) list の要素の中から値 x を探し、添字を返す。
max(arg1 , arg2) 最大の値を返す。
min(arg1 , arg2) 最小の値を返す。

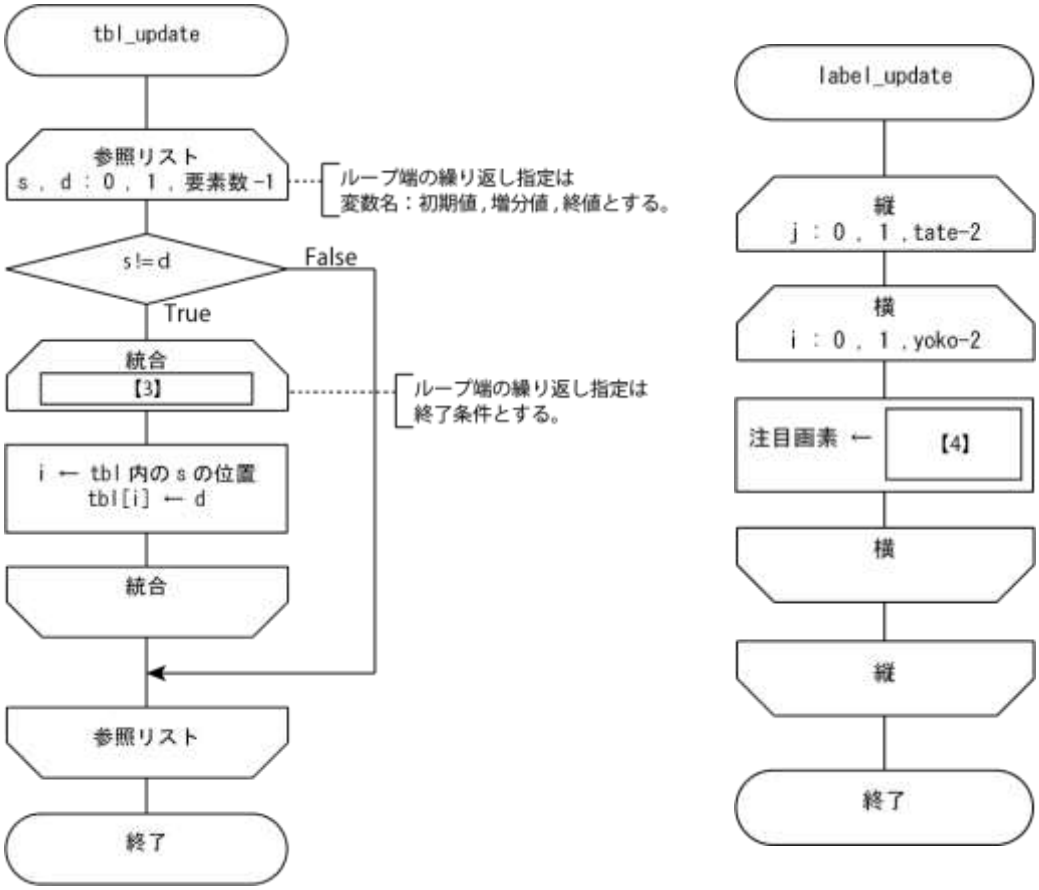
以下の関数を定義している。

check(j , i) j , i : 画素を指定するための縦方向・横方向の添字 指定された 1 つの画素について、手順 3 を実装する。
get_label() img のすべての要素について手順 3 を呼び出す。
tbl_update()

参照リストを更新する（手順 5）
label_update() 参照リストにしたがってラベルを補正する（手順 5）
tbl_compress() ラベルを連番にする

【流れ図】





【プログラム】

```
import numpy as np
tate = 20
yoko = 20
img = np.array([
          <<省略>>
       ])
labeling = np.zeros((tate , yoko) , dtype=np.uint8)
tbl = [0]


def check(j , i) :
    if labeling[j-1][i] == 0  and labeling[j][i-1] == 0 :
        labeling[j][i] = len(tbl)
        tbl.append(      【1】      )
```

```
    elif  labeling[j-1][i] > 0 and labeling[j][i-1] == 0 :
        labeling[j][i] = labeling[j-1][i]
    elif  labeling[j-1][i] > 0 and  labeling[j][i-1] > 0 :
        labeling[j][i] = labeling[j-1][i]
        if 【2】 :
            index = max(labeling[j][i-1] , labeling[j][i])
            tbl[index] = min(labeling[j][i-1] , labeling[j][i])
    else :
        labeling[j][i] = labeling[j][i-1]

def get_label() :
    for j in range(1,tate-1) :
        for i in range(1,yoko-1) :
            if img[j][i] > 0 :
                check(j , i)

def tbl_upadte() :
    for s , d in enumerate(tbl) :
        if s != d :
            【3】 :
                i = tbl.index(s)
                tbl[i] = d

def label_upadte() :
    for j in range(1,tate-1) :
        for i in range(1,yoko-1) :
            labeling[j][i] = 【4】

def disp_label() :
    for j in range(tate) :
        for i in range(yoko) :
            print(labeling[j][i] , end=' ')
        print()

def tbl_compress() :
    tbl_s = set(tbl)
```

```
    tbl2 = list(tbl_s)
    for i , k in enumerate(tbl2) :
        tbl[k] = i

get_label()
tbl_upadte()
label_upadte()
tbl_compress()
label_upadte()
disp_label()
```

次の中から、上の空欄【１】～【４】に入る最も適切なものを選びなさい。

【選択肢】

【１】	（１）	len(tbl)	（２）	i
	（３）	j	（４）	0
【２】	（１）	labeling[j][i-1] != labeling[j][i]		
	（２）	labeling[j][i-1] != labeling[j-1][i-1]		
	（３）	labeling[j][i-1] == labeling[j+1][i]		
	（４）	labeling[j][i-1] == labeling[j-1][i]		
【３】	（１）	while s in tbl	（２）	for s in tbl
	（３）	for i , s in enumerate(tbl)	（４）	while s != d
【４】	（１）	tbl[labeling[j][i]]	（２）	tbl[j][i]
	（３）	tbl[j]	（４）	tbl[i]

【問題2】

音符を LED で表現するプログラムをシミュレートする。

鍵盤の一つひとつに LED が付いており、音を LED の発光で表現する。各鍵盤には図 1 に示す番号が付いている。ここでは、0 番から 12 番まで 13 個の LED の点灯・消灯を制御する。

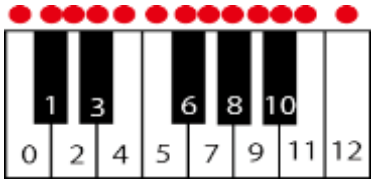


図 1 鍵盤の番号

音符は、開始拍と音階、それに音の長さで表現する。例えば、図 2 の譜面の例では、図 3 のように LED を点灯させる。第 0 拍のように音のないときもあれば、第 8 拍のように複数の音が重なることもある。音の長さは 8 分音符を 1 とする。

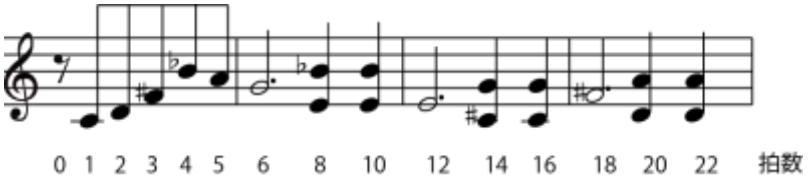


図 2　譜面の例

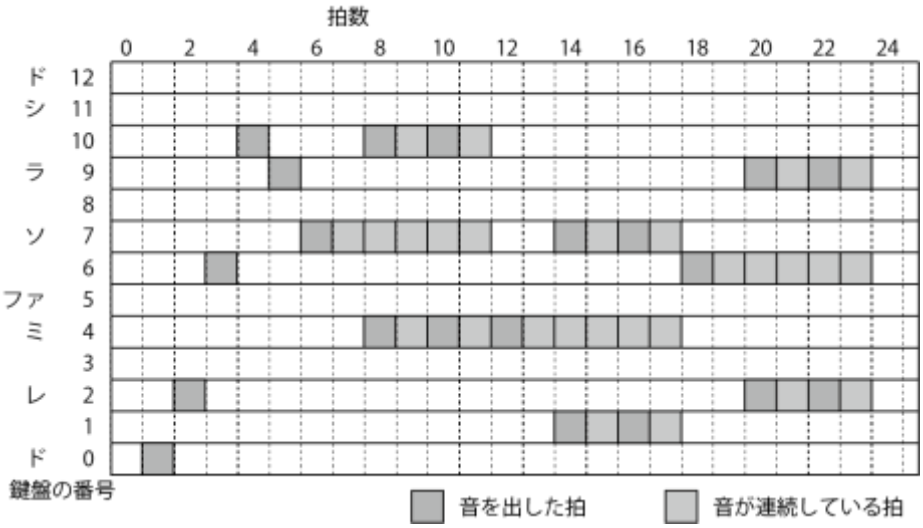


図 3　LED 点灯の様子

図の例では、第 0 拍では音がなく、LED はすべて消灯している。第 8 拍は、第 6 拍からの音の続きがあり、さらに 2 音が追加されて同時に 3 つの LED が点灯する。

13 個の LED は仮想レジスタで制御するものとする。仮想レジスタは 1 ビットずつ LED 1 個に対応しており、レジスタの内容を信号ピンに出力することで LED を点灯したり消

灯したりするが、仮想のため、レジスタの内容を表示することでシミュレートする。レジスタの各ビットが「1」のとき LED を点灯、「0」のとき LED を消灯する。第 1 拍ではレジスタは図のように「ド」の音に対応する LED のみ点灯する

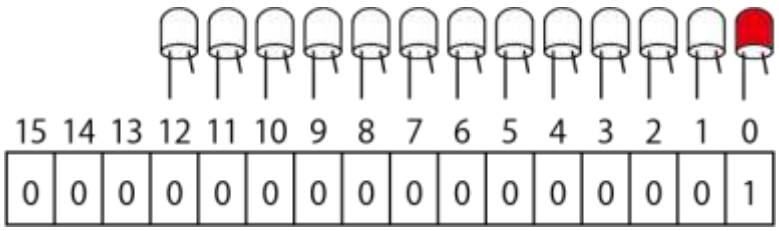


図 4　第 1 拍のレジスタと LED の状態

LED の制御は Music クラスに実装する。音符 1 つの情報は表の辞書に格納し、これがリストを形成している。

表 1　音符を 1 つ定義する辞書

key	説明
start_time	開始拍数
onkai	音の番号（図 1 による）
length	音の長さ。八分音符を 1 とする。

例えば図 2 の第 1 拍は 8 分音符のドであり、リストの内容は図のとおりである。

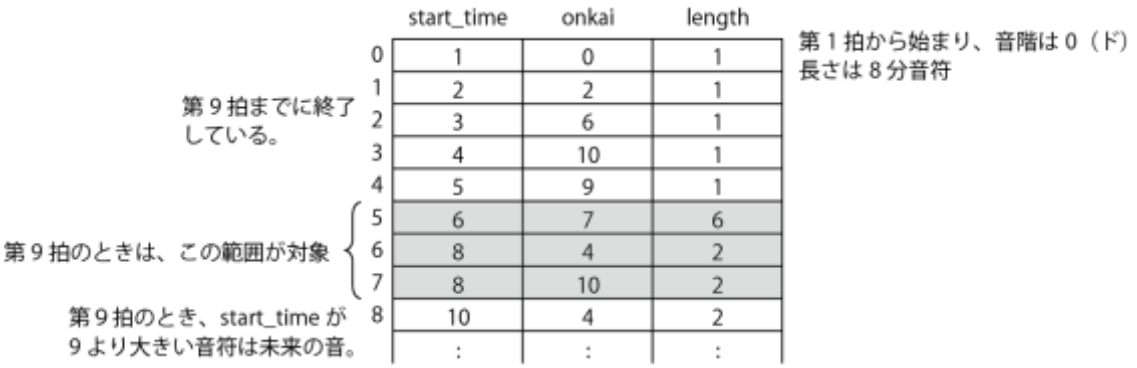


図 5　音符リストの例

Music クラスでは、変数 counter に拍数を格納し、要求に応じて 1 拍ずつ次々にレジスタを構築して提供する。音符リストの中から該当する辞書のみを処理の対象とする。図では影をつけた部分が第 9 拍の対象となる。

Music クラスの仕様は表のとおりである。

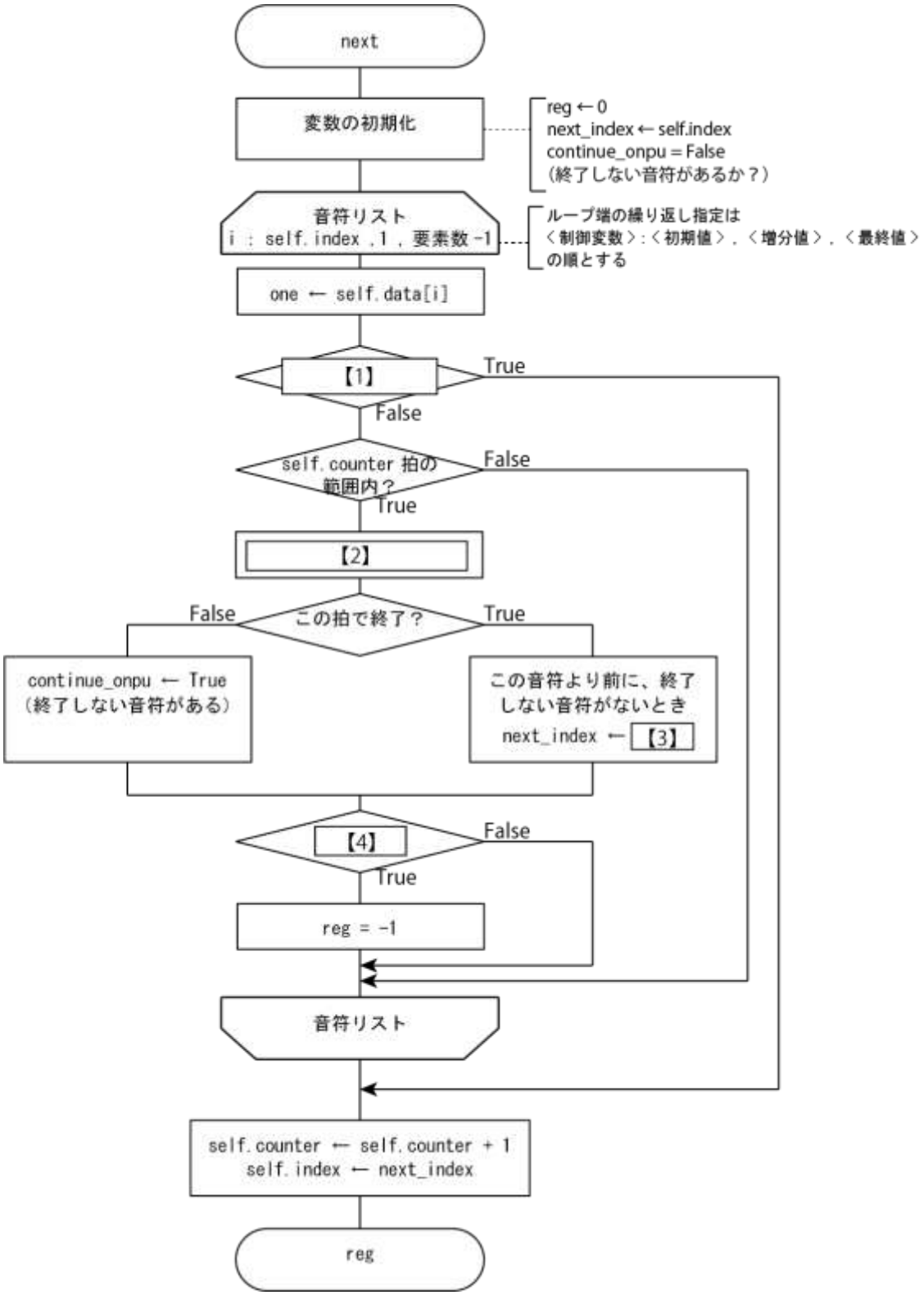
表２　Music クラスの仕様

属性	
data	音符データ。表の辞書を要素とするリスト。
counter	進行した拍数。１拍進行するたびに１加算する。
index	処理対象の最初の音符を指す添字
メソッド	
戻り値	メソッドと説明
	def __init__(self , data) data : リスト化した音符データ 辞書を要素とするリストを構築する。
なし	def start(self) counter と index を 0 に初期化する。
レジスタ	def next(self) 次の拍における仮想レジスタを返す。

Python では以下の関数が利用できる。

list.append(x) list の最後に x を追加する。
len(list) list の要素数を返す。

【流れ図】


next
変数の初期化
reg ← 0
next_index ← self.index
continue_onpu = False
（終了しない音符があるか？）
音符リスト
i : self.index , 1 , 要素数 -1
ループ端の繰り返し指定は
〈制御変数〉:〈初期値〉,〈増分値〉,〈最終値〉
の順とする
one ← self.data[i]
【1】
True
False
self.counter 拍の
範囲内？
False
True
【2】
False
この拍で終了？
True
continue_onpu ← True
（終了しない音符がある）
この音符より前に、終了
しない音符がないとき
next_index ← 【3】
【4】
False
True
reg = -1
音符リスト
self.counter ← self.counter + 1
self.index ← next_index
reg

【プログラム】

```
data = [
            [1,0,1], [2,2,1], [3,6,1], [4,10,1], [5,9,1],
            [6,7,6], [8,4,2], [8,10,2], [10,4,2], [10,10,2],
            [12,4,6], [14,1,2], [14,7,2], [16,1,2], [16,7,2],
            [18,6,6], [20,2,2], [20,9,2], [22,2,2], [22,9,2]
        ]

class Music :
    def __init__(self , data) :
        self.data = []
        for one in data :
            one_data = {'start_time' : one[0] , 'onkai' : one[1] , 'length' : one[2]}
            self.data.append(one_data)

    def start(self) :
        self.counter = 0
        self.index = 0

    def next(self) :
        reg = 0
        next_index = self.index
        continue_onpu = False
        for i in range(self.index , len(self.data)) :
            one = self.data[i]
            if 【1】 :
                break
            if one['start_time'] + one['length'] > self.counter :
                【2】
                if one['start_time'] + one['length'] -1 > self.counter :
                    continue_onpu = True
                else :
                    if not continue_onpu :
                        next_index = 【3】
        else :
            if 【4】 :
```

```
                reg = -1
        self.counter += 1
        self.index = next_index
        return reg

music = Music(data)
music.start()
reg = music.next()
while  reg >= 0 :
    print('{:016b}'.format(reg))
    reg = music.next()
```

次の中から、上の空欄【１】～【４】に入る最も適切なものを選びなさい。

【１】	（１）	one['start_time'] > self.counter
	（２）	one['start_time'] < self.counter
	（３）	one['start_time'] == self.counter
	（４）	one['start_time'] != self.counter
【２】	（１）	reg |= (1 << one['onkai'])
	（２）	reg &= (1 << one['onkai'])
	（３）	reg ^= (1 << one['onkai'])
	（４）	reg ~= (1 << one['onkai'])
【３】	（１）	i + 1
	（２）	self.index + 1
	（３）	self.counter + 1
	（４）	one['start_time'] + 1
【４】	（１）	reg == 0
	（２）	reg > 0
	（３）	continue_onpu
	（４）	not continue_onpu

第3問 （読解　1 問）

【問題】

感染症の陽性者一人ひとりのデータを収めた CSV ファイルがある。このデータを集計してグラフを描くプログラムである。ファイル名は data.csv であり、正しく保存されているものとする。

	A	B	C	D	E
1	No	公表_年月日	患者_年代	患者_性別	
2	1	2020/1/24	40代	男性	
3	2	2020/1/25	30代	女性	
4	3	2020/1/30	30代	女性	
	196636	2021/7/13	70代	男性	
197522	196637	2021/7/24		女性	
197523	196638	2021/7/24	30代		
197524	196639	2021/7/24	10代	女性	
197525	196640	2021/7/24	20代	女性	
197526	196641	2021/7/24	40代	男性	
197527	196642	2021/7/24	70代	男性	
197528	196643	2021/7/24	60代	女性	
197529	196644	2021/7/24	40代	男性	
197530					

図 1　データ例

データ項目は以下のとおりである。

表 1　　データの項目

名称	内容	備考
No	番号	連番
公表_年月日	日付	4 桁の年/月/日
患者_年代	年代	10 歳未満から 100 歳以上まで 10 歳刻み　非公開
患者_性別	性別	男性　女性　　非公開

Python では、次の関数が利用できる。

パッケージ	関数の形式と機能
datetime.datetime	strptime(date_string , format) 指定の形式で日付と時刻を表す文字列を datetime 型に変換する。 形式'%Y/%m/%d'は「4 桁の年/月/日」を指定する。
datetime.timedelta	timedelta(days=days) 日数を datetime 型に変換する。
pandas	read_csv(path) path で指定したファイルから読み込んで DataFrame を生成する
DataFrame	dropna(subset=columns) subset で指定した列が NaN である行を削除する。
DataFrame	apply(func , axis=axis) 各行または列について func で指定された式を実行する。axis=0 の時は行、axis=1 の時は列。
DataFrame	groupby（by) by で指定した列で集約する。
DataFrame	count() 行数を求める。

【プログラム】

```
import pandas as pd
import matplotlib.pyplot as plt
from datetime import datetime as dt
from datetime import timedelta

def date_div(today_str , start) :
    today = dt.strptime(today_str , '%Y/%m/%d')
    n = int((today - start).days//7)
    return start + timedelta(days = (n * 7 + 3))

df = pd.read_csv('data.csv')
df = df.dropna(subset=['公表_年月日'])

start_str = '2020/1/19'
start_date = dt.strptime(start_str , '%Y/%m/%d')

df['date_Wed'] = df['公表_年月日'].apply(lambda x : (date_div(x , start_date)) )
date_n = df.groupby('date_Wed')['No'].count() / 7

plt.plot(date_n)
```

上のプログラムにより描画されるグラフに関する説明として、最もふさわしいものを次から選びなさい。なお、2020 年 1 月 19 日は日曜日である。

【選択肢】

【1】	1 週間の平均の患者数の推移グラフ。
【2】	毎週水曜日の感染者の推移グラフ。
【3】	毎日の感染者数の推移グラフ。
【4】	毎週日曜日の感染者の推移グラフ。

データの出典：東京都新型コロナウイルス感染症対策サイト
https://stopcovid19.metro.tokyo.lg.jp/

実技科目

【問題１】

スタート地点からゴール地点に移動するにあたり、複数のルートの中から、もっとも短時間での移動が見込まれるルートを選択する。ただし、途中に休憩できる施設があるルートでなければならない。例えば、下表のように５つのルートがある場合、ルートＣを選択する。

表１　ルート一覧の例

	ルート A	ルート B	ルート C	ルート D	ルート E
休憩施設	なし	あり	あり	なし	あり
見込み時間	3.8 時間	4.5 時間	4.2 時間	3.7 時間	4.4 時間

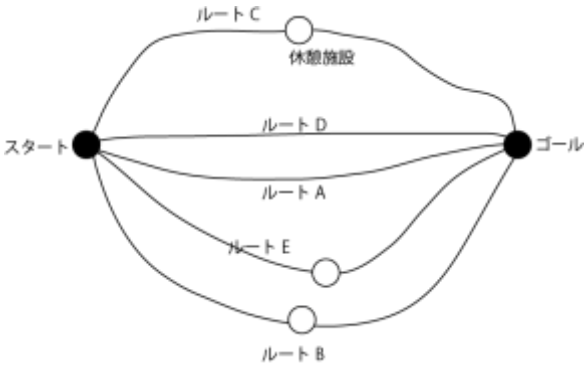


図１　ルート例

各地点に関する情報は、表２のような辞書に収められており、ルート数を要素数とするリストを構成している。

表２　辞書の構成

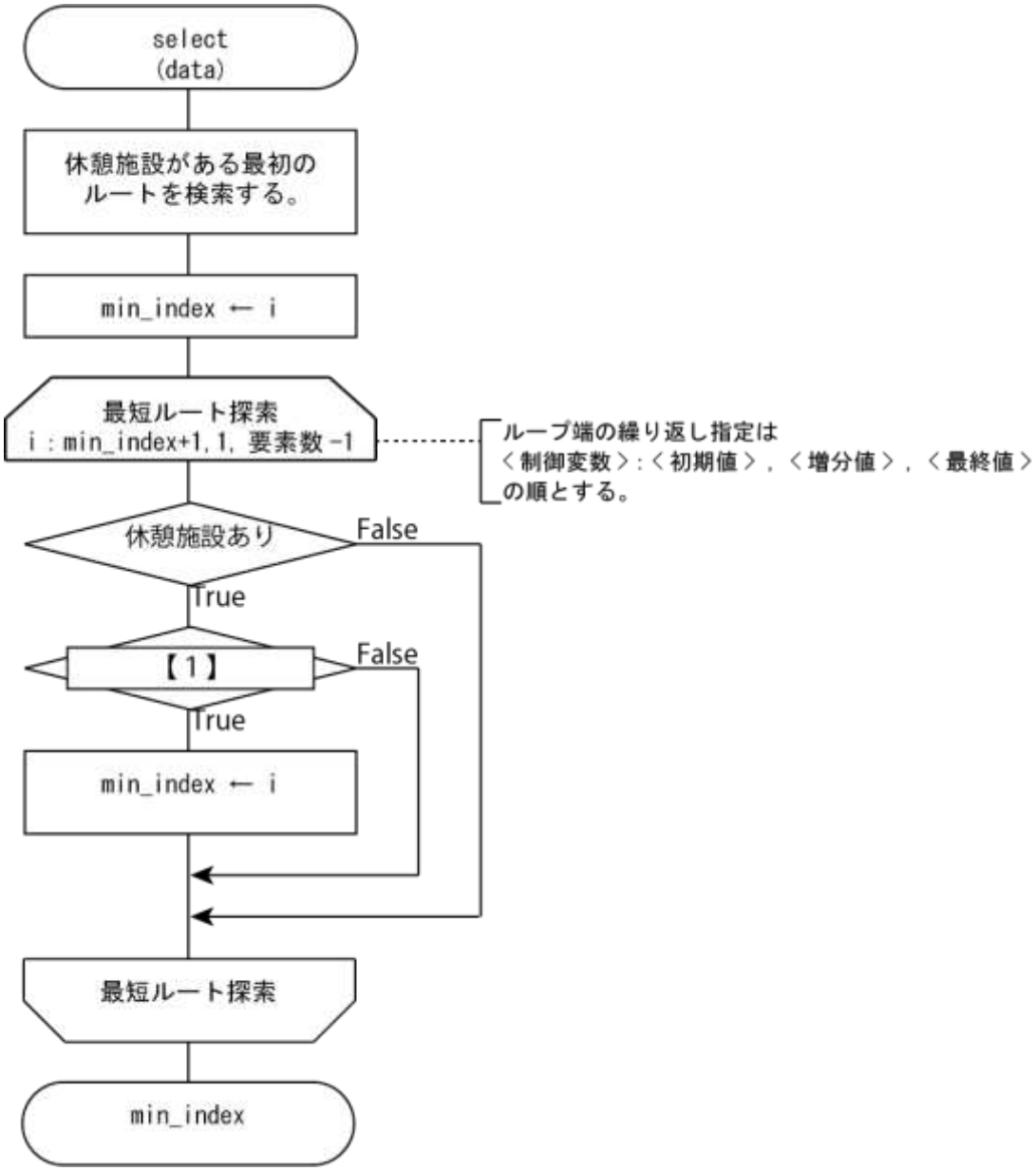
キー	説明
name	ルート名
restPlace	休憩施設　False:なし　True:あり
requiredTime	見込み時間（単位：時間）

関数 select は、以下の仕様である。

表３　関数 select の仕様

戻り値の型	関数の形式と機能
int 型	select(data) 条件に該当するルートのインデックス値を求める。 data：ルート情報を収めたリスト

【流れ図】



以下のプログラムの空欄に入る適当なプログラム片を入力してプログラムを完成させなさい。

```
sample = [
    {'name':'Route-A' , 'restPlace':False , 'requiredTime':3.8} ,
    {'name':'Route-B' , 'restPlace':True , 'requiredTime':4.5} ,
    {'name':'Route-C' , 'restPlace':True , 'requiredTime':4.2} ,
    {'name':'Route-D' , 'restPlace':False , 'requiredTime':3.7} ,
    {'name':'Route-E' , 'restPlace':True , 'requiredTime':4.4} ,
]

def select(data):
    for i , x in enumerate(data) :
        if x['restPlace'] :
            break

    min_index = i
    for i in range(min_index+1 , len(data)) :
        if data[i]['restPlace']:
            if 【1】 :
                min_index = i
    return min_index

i = select(sample)
print('ルート:{}　見込み時間:{}時間'.format(sample[i]['name'] , sample[i]['requiredTime']))
```

【問題２】

図１のような迷路をスタート地点からゴール地点に最短で移動する経路をすべて列挙するプログラムである。■は壁であり、進むことができない。また、最短で移動するために、右と下にのみ移動できる。

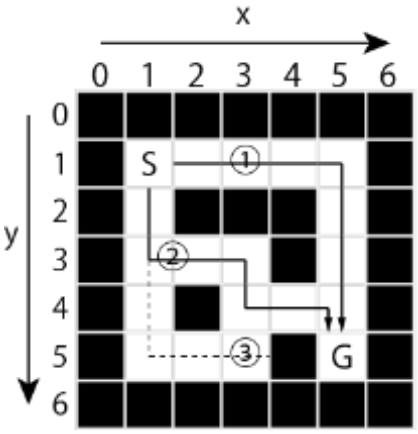


図 1　迷路の例

図１の例では、①と②はゴールにたどり着くことができるが、③は、途中で行き止まりになってしまい、ゴールにたどり着くことができない。

プログラムでは、壁の構成は、リスト maze_data で与えられ、maze_data[y][x]が True の位置には壁があり、False の位置は通路である。スタート地点 S は、[1][1]、ゴール地点 G は[5][5]である。また、地点を表す x と y の組は、辞書を利用している。辞書のキーは表１のとおりである。

通過した位置は、リスト route に記録していく。ゴールに到達したときは、route の内容を先頭から順に表示する。その後、順に戻りながら、次のルートを探す。このとき、route の内容を１つずつ破棄する。

利用するリストは表２のとおりである。

表 1　位置を示す辞書

キー	説明
x	横方向の添字
y	縦方向の添字

表 2　利用するリスト

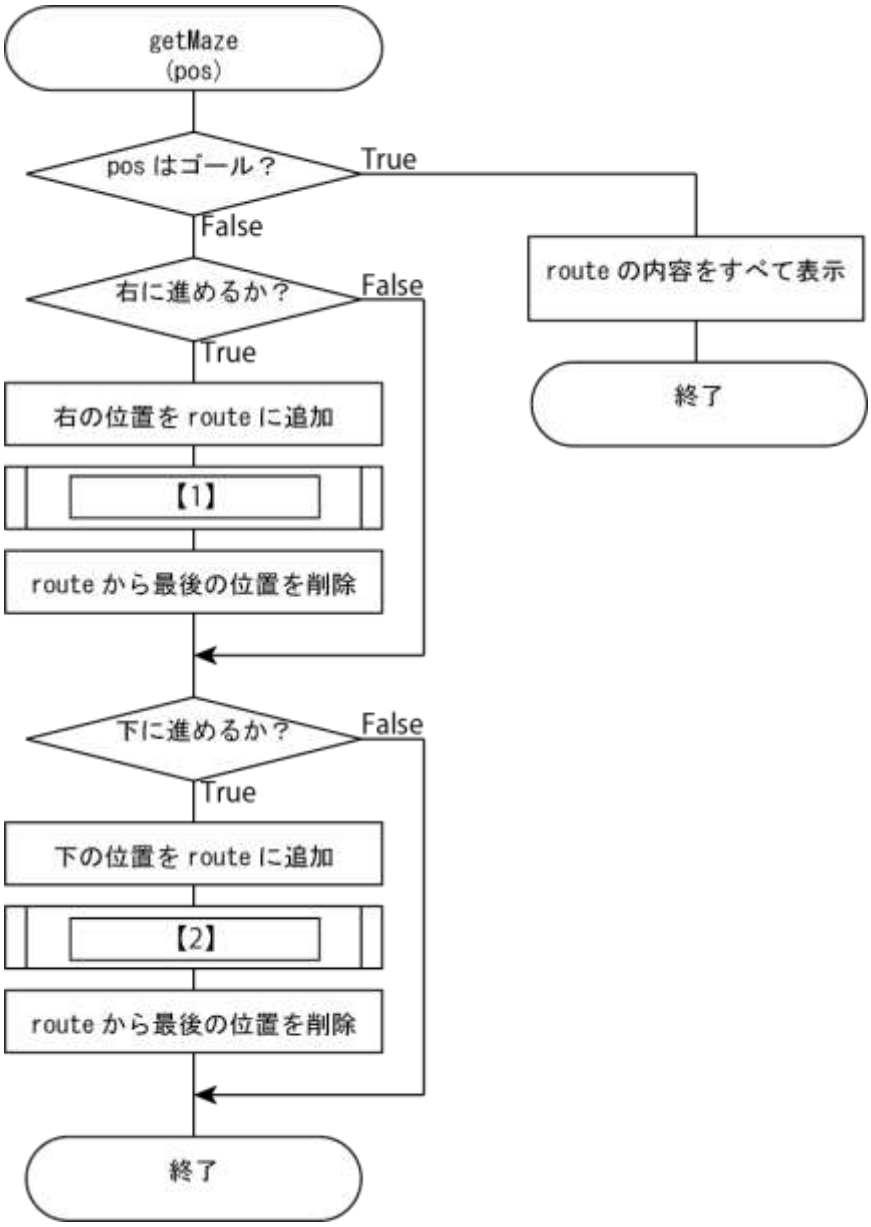
型	配列名	説明
リスト	maze_data[7][7]	壁の位置を格納するリスト False : 通路　True : 壁
リスト	route	経路（x と y の辞書）を格納するリスト

プログラムで定義している関数の仕様は以下のとおりである。

表 3　定義している関数

戻り値の型	関数の形式と機能
bool	isGoal(pos) pos がゴールであれば True　ゴールでなければ False を返す。
bool	checkGo(pos , direction) direction　1：右　2：下 pos の右または下の壁の状態を返す。 通行できるときは True 壁があって通行できないときは False を返す。
なし	printRoute() 経路が格納されているリスト route の内容をすべて表示する。
なし	getMaze(pos) スタート位置からゴール位置までの経路を表示する。 pos：開始位置を指定する。

【流れ図】



以下のプログラムの空欄に入る適当なプログラム片を入力してプログラムを完成させなさい。

```
maze_data = [
    [False , False , False , False , False , False , False] ,
    [False , True  , True  , True  , True  , True  , False] ,
    [False , True  , False , False , False , True  , False] ,
    [False , True  , True  , True  , False , True  , False] ,
```

```
    [False , True  , False , True  , True  , True  , False] ,
    [False , True  , True  , True  , False , True  , False] ,
    [False , False , False , False , False , False , False]
]

def isGoal(pos):
    if pos['x'] == 5 and pos['y'] == 5 :
        return True
    else :
        return False

def checkGo(pos , direction):
    if direction == 1:
        return maze_data[pos['y']][pos['x']+1]
    elif direction == 2:
        return maze_data[pos['y']+1][pos['x']]
    else :
        return False

def printRoute() :
    print("ROUTE")
    for pos in route:
        print('x:{}  y:{}'.format(pos['x'] , pos['y']))

def getMaze(pos):
    if isGoal(pos):
        printRoute()
        return

    if checkGo(pos , 1):
        route.append({'x':pos['x']+1 , 'y':pos['y']})
        【1】
        route.pop()
    if checkGo(pos , 2):
        route.append({'x':pos['x'] , 'y':pos['y']+1})
        【2】
```

```
        route.pop()

route = [{'x':1 , 'y':1}]
getMaze({'x':1 , 'y':1})
```

【問題３】

シャーレの中でバクテリアがどのように増殖するのかをシミュレートするプログラムである。シャーレを微小な空間に分け、その小さな一つひとつの空間にバクテリアが最大１匹存在できるものとする。図１では初期状態として、5 匹のバクテリアが存在している。

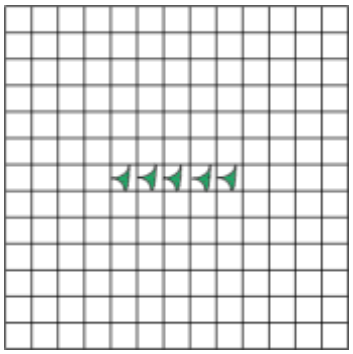


図１　微小空間にいるバクテリア

バクテリアは、環境により次のように変化する。

① バクテリアは、図２に示す近傍８個のセルのうち２個または３個のセルにバクテリアが存在すれば、生き続けられる。しかし、２個未満のときは、寂しくて死んでしまう。また、４個以上の時は、過密すぎて死んでしまう。

② バクテリアが存在しないセルでは、図３に示す近傍８個のセルのうち３個のセルにバクテリアがいれば、分裂により新たに誕生することができる。

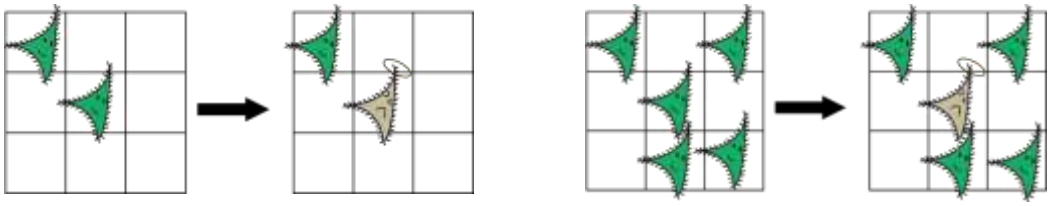


図２　周りのバクテリアが少なくても多くても死んでしまう

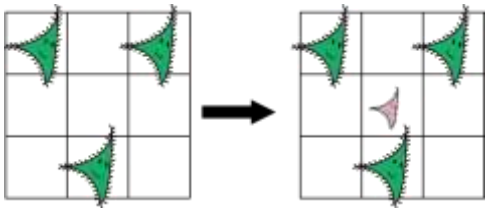


図３　近傍が３匹であれば赤ちゃん誕生

以上のルールに従って、時間が進んだとき、バクテリアがどのように変化しているかをシミュレートする。ただし、領域の外側にはバクテリアは生存せず、生誕することもできないものとする。

プログラムでは、13×13 の領域に分けており、変数 TATE と YOKO で定義している。各微小空間のバクテリアの状態は World クラス内のリスト cell に記録しており、バクテリアが生きている状態を True、バクテリアがいない状態を False とする。図 1 の初期状態を表したものが図 4 である。ここで、j 行 i 列の空間は

```
cell[j*YOKO+i]
```

で指定することができる。図では黒い要素は True が、白い要素は False が格納されていることを表している。

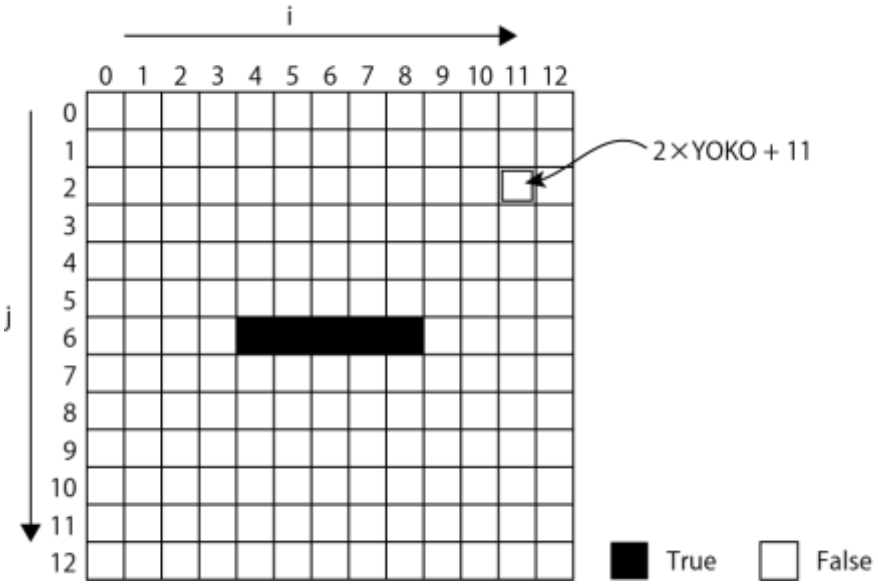


図 4　初期状態

バクテリアの状態を 10 回更新したときの様子を表示する。

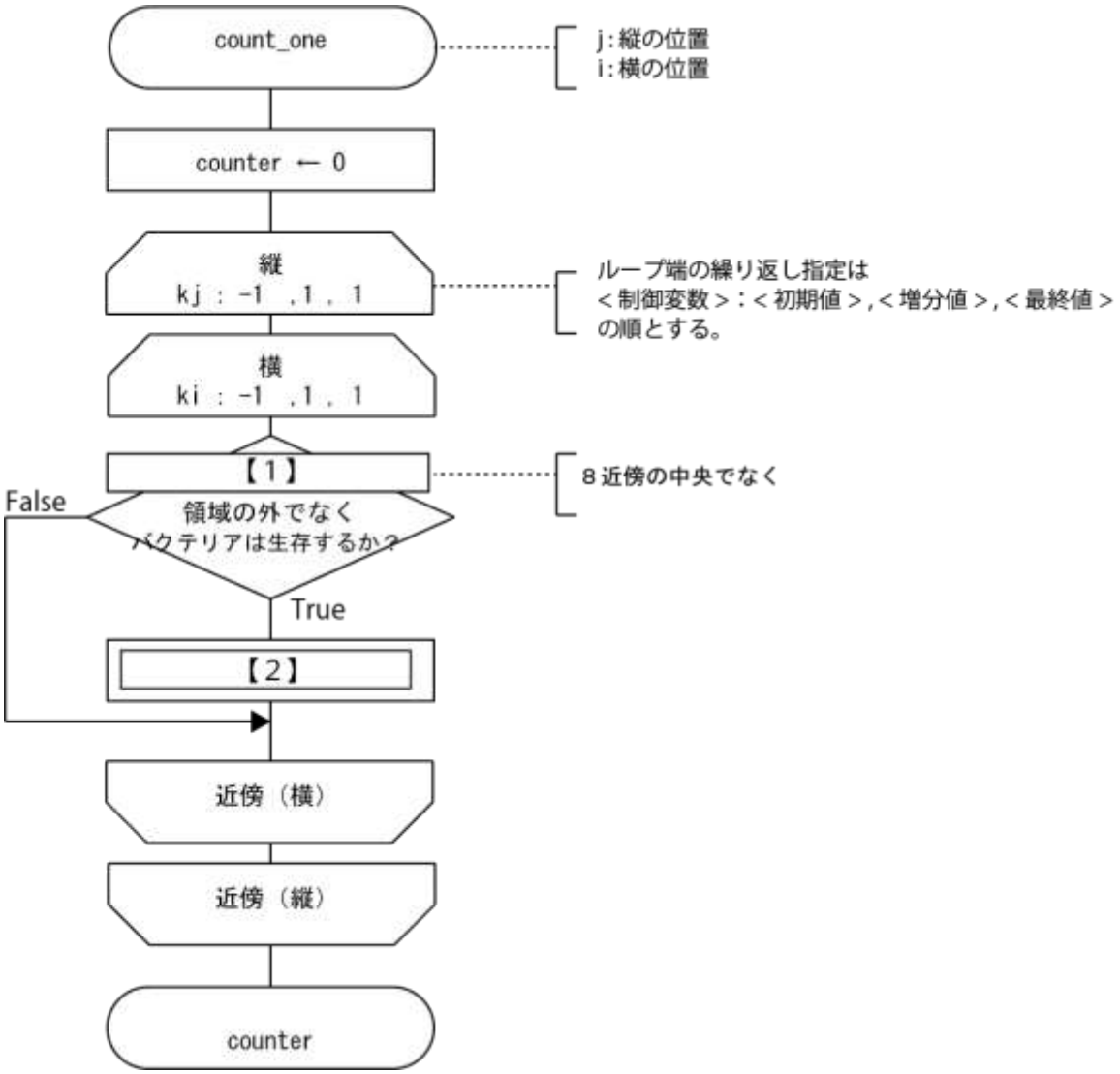
World クラスの仕様は表 3 のとおりである。

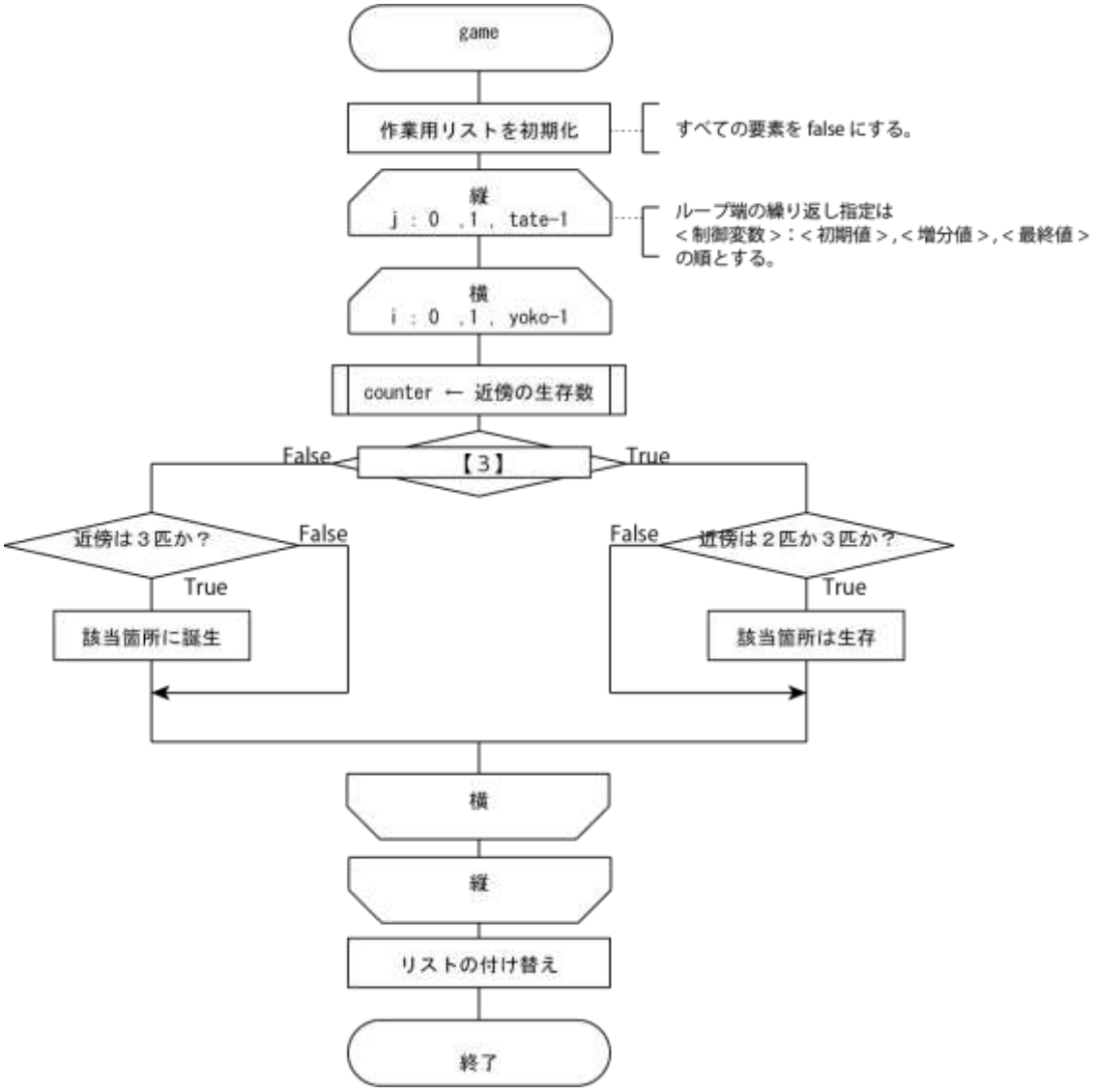
表 3　World クラス

属性		
tate	int 型	縦方向の要素数
yoko	int 型	横方向の要素数
cell	リスト	tate×yoko 個の要素を持つ。バクテリアが生存している要素は True、生存していない要素は False
メソッド		
戻り値の型	メソッドと説明	
なし	def __init__(self , n1 , n2) n1：縦の要素数 n2：横の要素数 cell をすべて False で初期化する。	

なし	set(self , j , i) j : 縦の要素番号 i : 横の要素番号 指定の要素を True にする。
なし	disp(self) cell のすべての要素を表示する。True の要素は「1 」を、False の要素は「 」を表示する。
int 型	count_one(self , j , i) j 行 i 列の要素の周りのバクテリアの数を返す。
なし	game(self) バクテリアの状態を 1 回更新する。

【流れ図】





以下のプログラムの空欄に入る適当なプログラム片を入力してプログラムを完成させなさい。

```
class World :
    def __init__(self , n1 , n2) :
        self.tate = n1
        self.yoko = n2
        self.cell = [False] * (n1 * n2)

    def set(self , j , i) :
        self.cell[j * self.yoko + i] = True

    def disp(self) :
        for k in range(len(self.cell)) :
```

```
            if self.cell[k] :
                print('1 ' , end='')
            else :
                print('  ' , end='')
            if  k % self.yoko == self.yoko - 1 :
                print()

    def count_one(self , j , i) :
        counter = 0
        for kj in range(-1 , 2) :
            for ki in range(-1 , 2) :
                if 【1】 :
                    if ((j+kj) >= 0 and (j+kj) < self.yoko and (i+ki) >= 0 and
                                (i+ki) < self.tate) :
                        if self.cell[(j+kj) * self.yoko + (i+ki)] :
                            【2】
        return counter

    def game(self) :
        next = [False] * (self.tate * self.yoko)
        for j in range(self.tate) :
            for i in range(self.yoko) :
                counter = self.count_one(j , i)
                if 【3】 :
                    if counter == 2 or counter == 3 :
                        next[j * self.yoko + i] = True
                    else :
                        if counter == 3 :
                            next[j * self.yoko + i] = True
        self.cell = next

TATE = 13
YOKO = 13
world = World(TATE , YOKO)
init = [[TATE//2 , YOKO//2-2] , [TATE//2 , YOKO//2-1] , [TATE//2 , YOKO//2] ,
        [TATE//2 , YOKO//2+1] , [TATE//2 , YOKO//2+2]]
for index in init :
    world.set(index[0] , index[1])
for k in range(9) :
    world.game()
world.disp()
```