artisoc Cloudレシピブック

05. ポテンシャル法で移動しよう

（株）構造計画研究所
創造工学部
https://mas.kke.co.jp

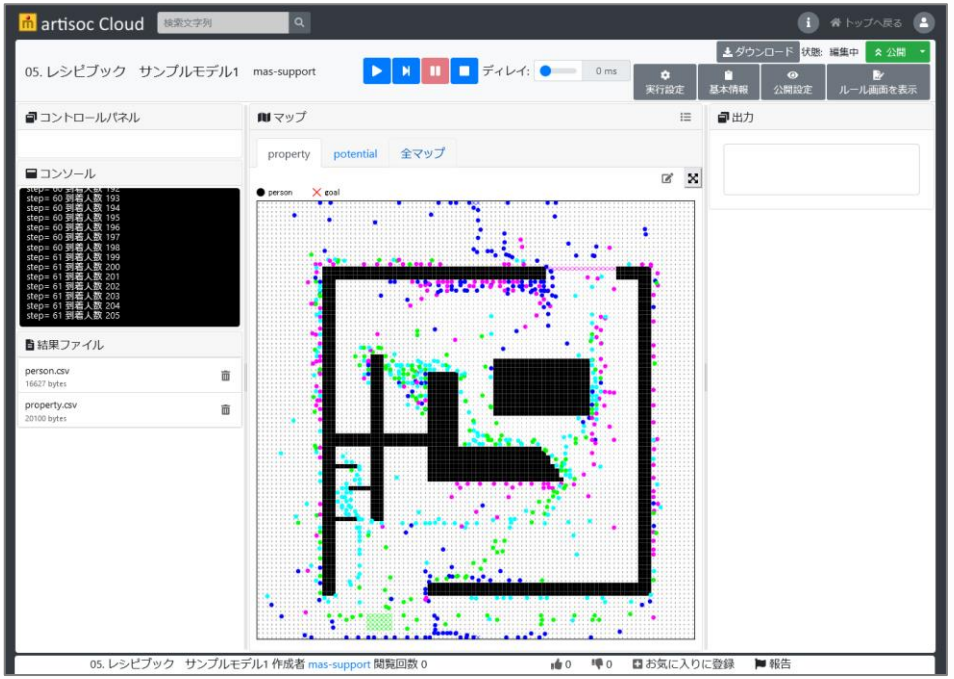
ポテンシャル法で移動しよう

構造計画研究所
KOZO KEIKAKU ENGINEERING Inc.

ダイクストラ法は道路上を最短経路で移動するときに便利でしたが、
ポテンシャル法は道路幅や広場のような面上を移動するときに便利です。

1. マップを作成する
2. 行動計画を作成する
3. モデルを定義する
4. Universeのルールを記述する
5. personのルールを記述する
6. 出力画面を定義する

[Tips] artisoc Cloudのライセンス



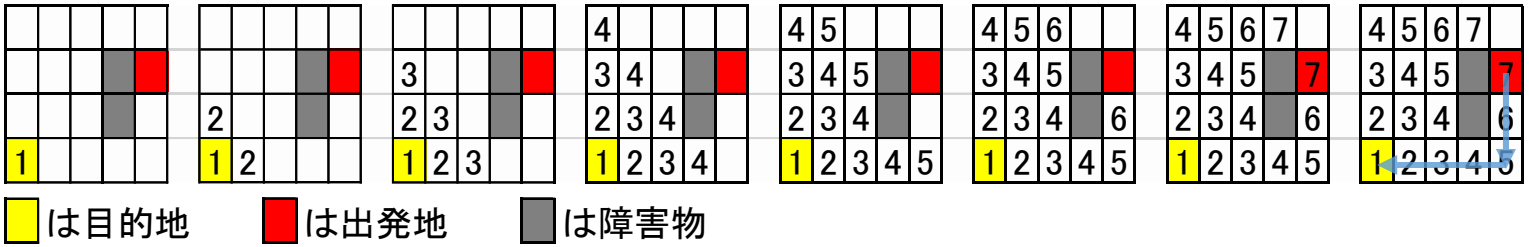
05. レシピブック　サンプルモデル1

ポテンシャル法とは？

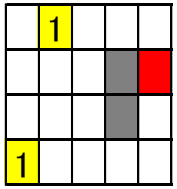
ボールが高いところから低いところへ転がっていく性質を利用した
探索方法です

ポテンシャル法で最短経路を求める手順

1. 空間をメッシュ状に切り、出発地と目的地のセルを決めます。
2. 目的地に「1」を代入し、その周り上下左右（４方向）に「＋１」した値を代入します。
3. 順番に値を決めていき、出発地の値が決まるまで繰り返します。
4. 出発地から値の小さいセルを順番に辿れば最短経路が見つかります。



※ 目的地が複数あるときはどうなるか試してみましょう。

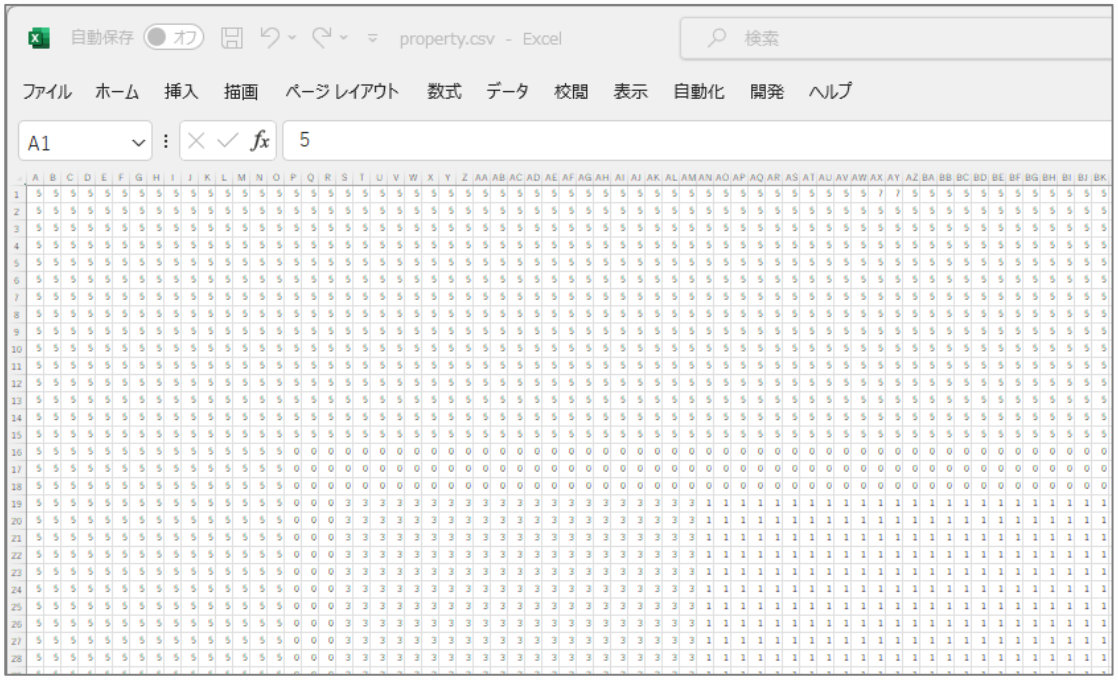


① マップを作成する（1）

構造計画研究所
KOZO KEIKAKU ENGINEERING Inc.

Excelを使ってマップを作成します

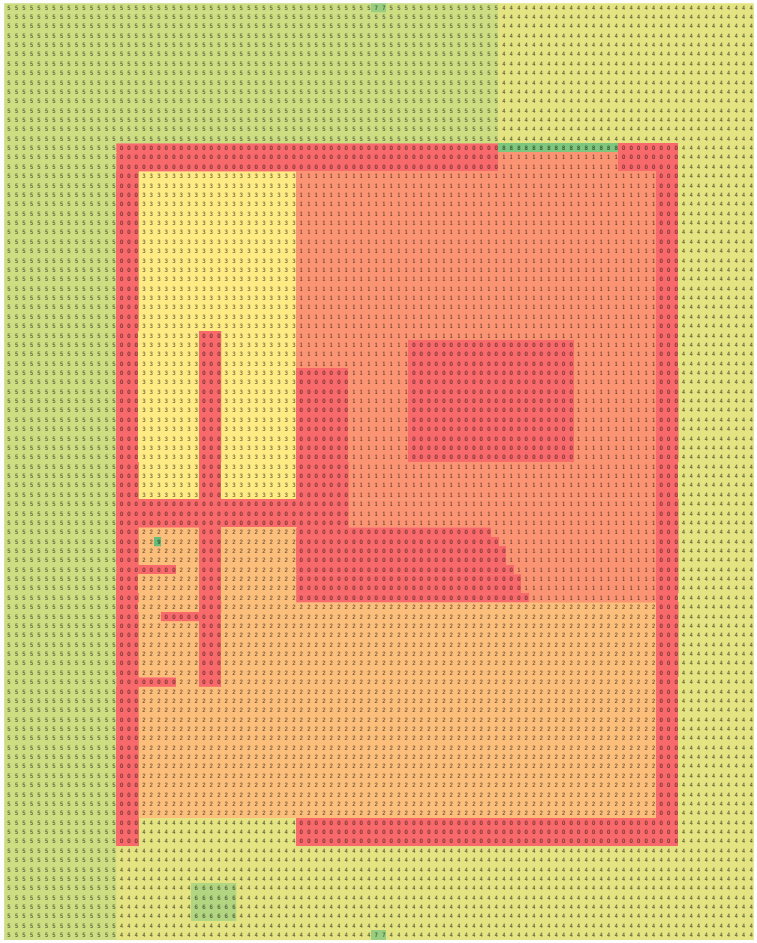
- property.csv をExcelで開きます。
- マップの大きさのプロパティ値（壁、床など）を定義します。
- プロパティ値は0が歩行不可、1以上が歩行可能です。
- プロパティ値を使って、出発地IDと目的地IDを割り当てます。



① マップを作成する（2）

構造計画研究所
KOZO KEIKAKU ENGINEERING Inc.

マップのプロパティ値は、以下の手順で作成します



1. マップの大きさ（100x100）を枠で囲みます。
2. 壁（歩行不可）を0で定義します。
3. 歩行可能エリアを1から5のエリアで分割します。
4. この1から5のエリアは出発地IDになります。
5. 目的地IDを6から9で配置します。
6. 指定するエリアは1セルでも飛び地でもOKです。

※ ホームタブ > 条件付き書式 > カラースケール で着色するとマップが見えやすくなります
※ 出発地IDと目的地IDの数が足りない場合は、モデル上で変更できます

② 行動計画を作成する

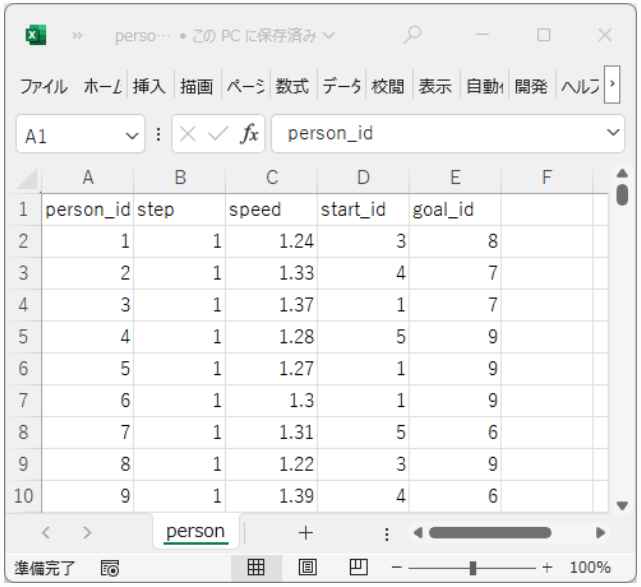
構造計画研究所
KOZO KEIKAKU ENGINEERING Inc.

Excelを使って行動計画を作成します

- person.csv をExcelで開きます。
- 以下の値を定義します。

person_id：	personのID
step：	発生するstep
speed：	personの移動速度
start_id：	マップで定義した出発地ID
goal_id：	マップで定義した目的地ID

※ 同じstep、同じstart_idで複数のpersonを生成できます

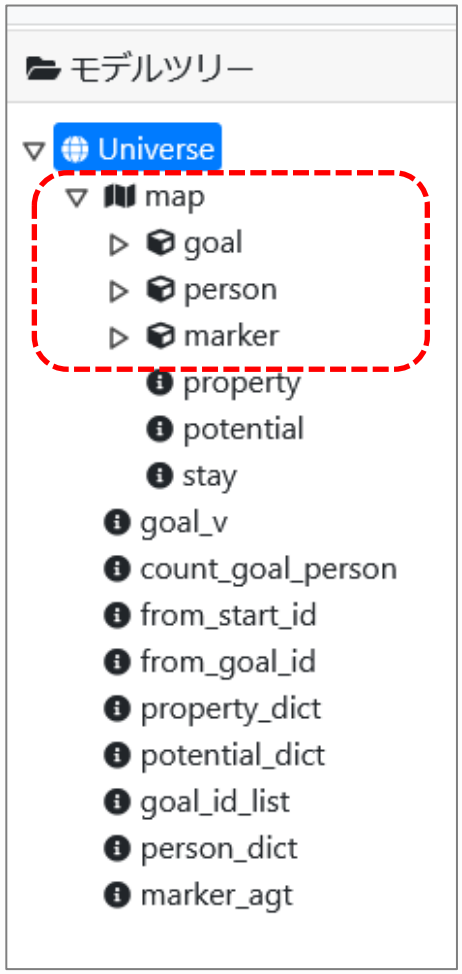


	A	B	C	D	E
1	person_id	step	speed	start_id	goal_id
2	1	1	1.24	3	8
3	2	1	1.33	4	7
4	3	1	1.37	1	7
5	4	1	1.28	5	9
6	5	1	1.27	1	9
7	6	1	1.3	1	9
8	7	1	1.31	5	6
9	8	1	1.22	3	9
10	9	1	1.39	4	6

③ モデルを定義する（1）

構造計画研究所
KOZO KEIKAKU ENGINEERING Inc.

モデルツリーで「空間」「エージェント」「変数」を定義します



- 空間（Universe.map）
  連続空間、大きさ100 x 100、レイヤ数1、ループしないで作成

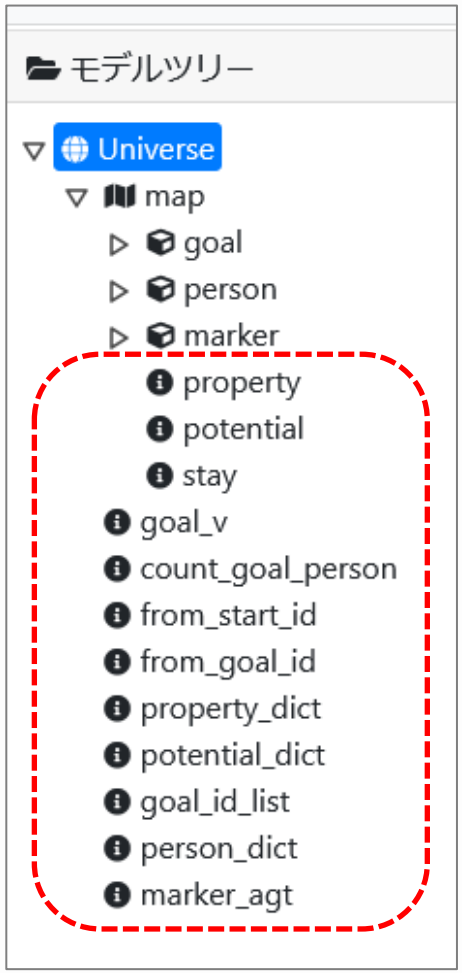
- エージェント（Universe.map）
  goal：　目的地、マップ出力で×を表示
  - エージェント変数
    - goal_id：　目的地ID
    - color：　マップ出力の表示色

  person：　歩行者エージェント
  - エージェント変数
    - person_id：　personのid
    - step：　発生するstep
    - speed：　移動速度
    - start_id：　出発地ID
    - goal_id：　目的地ID
    - color：　マップ出力の表示色

  marker ：　personを移動させるための移動マーカー

https://www.kke.co.jp

③ モデルを定義する（2）



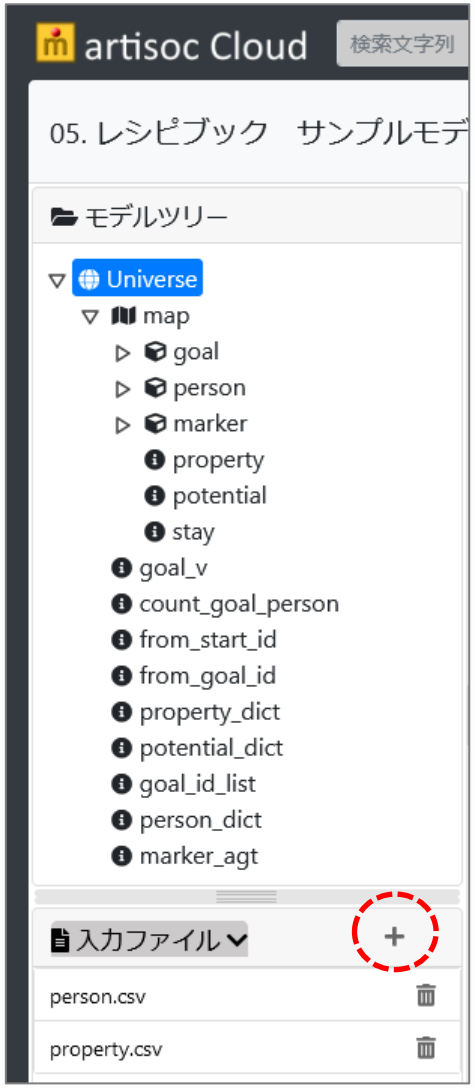
- 空間変数
  property :　プロパティ値を格納　※ property.csvから読込
  potential :　ポテンシャル値を格納　※モデル内で計算
  stay　:　personの存在有無を格納

- Universe変数
  goal_v :　目的地のポテンシャル値を格納
  count_goal_person :　目的地に到着した人数を格納
  from_start_id :　出発地IDの開始番号
  from_goal_id :　目的地IDの開始番号
  property_dict :　プロパティ値をキーに座標集合を格納
  potential_dict :　プロパティ値をキーにポテンシャルを格納
  goal_id_list :　目的地IDのリスト
  person_dict :　stepをキーに行動計画のリストを格納
  ※person.csvから読込
  marker_agt :　移動マーカーのエージェントを格納

③ モデルを定義する（3）

構造計画研究所
KOZO KEIKAKU ENGINEERING Inc.

- 入力ファイルの「＋」をクリックして、
  入力ファイルをインポートします。
  person.csv：　　行動計画をファイル入力
  property.csv：　　プロパティ値をファイル入力



④ Universeのルールを記述する（1）

構造計画研究所
KOZO KEIKAKU ENGINEERING Inc.

personが与えられた出発地から目的地へ移動し、移動の途中に他のpersonと相互作用するモデルを作成します

- univ_initは、変数の初期化やファイル読込、ポテンシャルの計算等を行います。

```
def univ_init(self):

    # 変数の初期化
    Universe.from_start_id = 1      # 出発地IDの開始番号：0:歩行不可として定義しているため1以上とする
    Universe.from_goal_id = 6       # 目的地IDの開始番号：6以上を目的地とする
    Universe.count_goal_person = 0  # 移動完了人数のクリア

    # personのファイル読込
    self.file_read_person('person.xlsx', 'person')

    # propertyのファイル読込
    self.file_read_property('property.xlsx', 'layer0')

    # potentialの計算
    self.set_potential()

    # 移動マーカーの生成
    Universe.marker_agt = create_agt(Universe.map.marker)

```

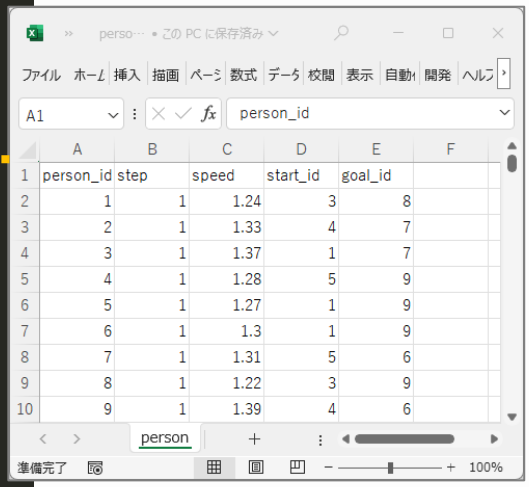
※ 出発地IDと目的地IDの数が足りない場合は、from_start_id、from_goal_idを調整してください

④ Universeのルールを記述する（2）

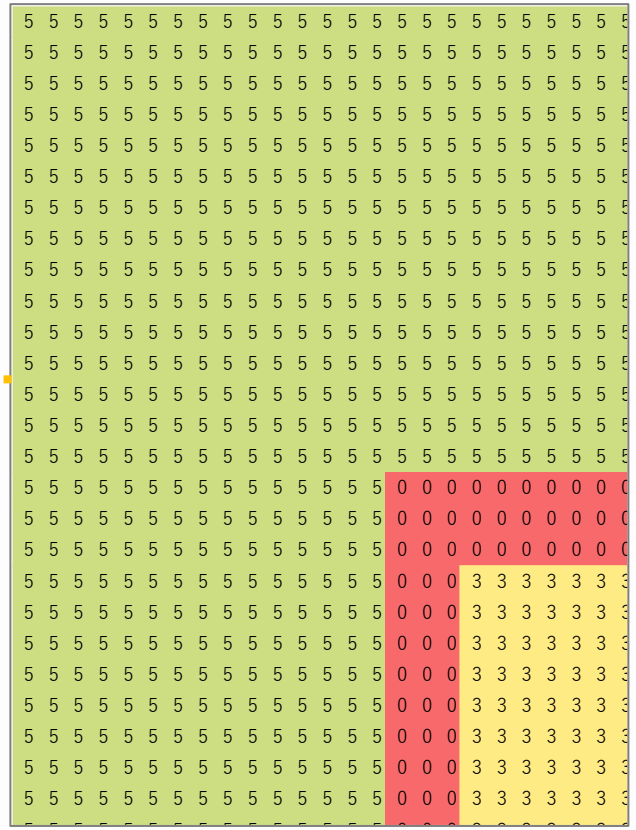
構造計画研究所
KOZO KEIKAKU ENGINEERING Inc.

- file_read_personは、person.csvを読み込み、step毎にまとめた上でperson_dictに格納します。

```python
# personファイルの入力
def file_read_person(self, input_filename):

    import csv

    # ファイルの読込
    Universe.person_dict = {}
    with open(input_filename, encoding='shift_jis') as f:
        csvreader = csv.reader(f)
        row_ct = 0
        for row in csvreader:
            if row_ct > 0:
                one_person_dict = {}
                one_person_dict['person_id'] = row[0]
                v_step = int(row[1])
                one_person_dict['step'] = v_step
                one_person_dict['speed'] = float(row[2])
                one_person_dict['start_id'] = int(row[3])
                one_person_dict['goal_id'] = int(row[4])

                if str(v_step) in Universe.person_dict:
                    Universe.person_dict[str(v_step)].append(one_person_dict)
                else:
                    Universe.person_dict[str(v_step)] = []
                    Universe.person_dict[str(v_step)].append(one_person_dict)
            row_ct += 1
```



	A	B	C	D	E
1	person_id	step	speed	start_id	goal_id
2	1	1	1.24	3	8
3	2	1	1.33	4	7
4	3	1	1.37	1	7
5	4	1	1.28	5	9
6	5	1	1.27	1	9
7	6	1	1.3	1	9
8	7	1	1.31	5	6
9	8	1	1.22	3	9
10	9	1	1.39	4	6



④ Universeのルールを記述する（3）

- file_read_propertyは、空間変数を初期化し、goal_v（ポテンシャルの最も小さい値）を計算します。（道路が複雑な場合は、より小さい値をgoal_vに設定してください）
- 次にproperty.csvを読み込み、プロパティID毎にまとめて property_dict に格納します。このとき、目的地IDを判別して goal_id_list を作成します。

```python
# propertyファイルの入力
def file_read_property(self, input_filename):

    import csv

    w = get_width_space(Universe.map)
    h = get_height_space(Universe.map)
    one_layer = 0

    # 空間変数の初期化
    for i in range(h):
        for j in range(w):
            Universe.map.property[j, i, one_layer] = 0
            Universe.map.potential[j, i, one_layer] = 0
            Universe.map.stay[j, i, one_layer] = 0

    # 目的地のポテンシャル値を計算
    Universe.goal_v = -1 * (w + h)

    # ファイルの読込
    Universe.property_dict = {}
    Universe.goal_id_list = []
    with open(input_filename, encoding='shift_jis') as f:
        csvreader = csv.reader(f)
        row_ct = 0
        for row in csvreader:
            for j in range(w):
                v = int(row[j])
                i = h - row_ct - 1
                key = str(j) + ',' + str(i) + ',' + str(one_layer)
                Universe.map.property[j, i, one_layer] = v

                # property_dictの作成
                if str(v) in Universe.property_dict:
                    Universe.property_dict[str(v)].append(key)
                else:
                    Universe.property_dict[str(v)] = []
                    Universe.property_dict[str(v)].append(key)

                # 目的地の場合
                if v >= Universe.from_goal_id:
                    if not v in Universe.goal_id_list:
                        Universe.goal_id_list.append(v)
                    self.create_goal(j, i, v)
                    v = Universe.goal_v
                Universe.map.potential[j, i, one_layer] = v

            row_ct += 1
```

5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5
5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5
5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5
5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5
5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5
5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5
5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5
5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5
5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5
5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5
5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5
5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5
5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5
5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5
5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5
5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	0	0	0	0	0	0	0	0	0
5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	0	0	0	0	0	0	0	0	0
5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	0	0	0	0	0	0	0	0	0
5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	0	0	0	3	3	3	3	3	3
5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	0	0	0	3	3	3	3	3	3
5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	0	0	0	3	3	3	3	3	3
5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	0	0	0	3	3	3	3	3	3
5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	0	0	0	3	3	3	3	3	3
5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	0	0	0	3	3	3	3	3	3
5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	0	0	0	3	3	3	3	3	3
5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	0	0	0	3	3	3	3	3	3

④ Universeのルールを記述する（4）

構造計画研究所
KOZO KEIKAKU ENGINEERING Inc.

- set_potentialは、goal_id_listに格納した目的地IDをcalc_potentialに渡して、目的地ID毎のポテンシャルを計算します。

```
150  # ポテンシャルの計算
151  def set_potential(self):
152
153      Universe.potential_dict = {}
154
155      # 目的地ID毎にポテンシャルを計算
156      for one_goal_id in Universe.goal_id_list:
157          self.calc_potential(one_goal_id)
```

- calc_potentialは、プロパティ値が0以外（歩行可能）のセルのポテンシャル値を求めます。計算結果はpotential_dictに格納します。

```
 83  # 任意の目的地IDのポテンシャルを計算
 84  def calc_potential(self, goal_id):
 85
 86      w = get_width_space(Universe.map)
 87      h = get_height_space(Universe.map)
 88      one_layer = 0
 89
 90      # 計算対象のセルを取得
 91      target_list = []
 92      cell_list = []
 93      for i in range(h):
 94          for j in range(w):
 95              v = Universe.map.property[j, i, one_layer]
 96              if v > 0:
 97                  key = str(j) + ',' + str(i) + ',' + str(one_layer)
 98                  if v == goal_id:
 99                      target_list.append(key)
100                      Universe.map.potential[j, i, one_layer] = Universe.goal_v
101                  else:
102                      cell_list.append(key)
103              else:
104                  Universe.map.potential[j, i, one_layer] = 0
```

④ Universeのルールを記述する（5）

構造計画研究所
KOZO KEIKAKU ENGINEERING Inc.

- univ_step_beginでpersonの生成、univ_step_endで終了条件をチェックします。

```
20  def univ_step_begin(self):
21
22      # step毎にpersonを生成
23      self.create_person_step()
24
25
26  def univ_step_end(self):
27
28      # 終了条件のチェック
29      if count_agt(Universe.map.person) == 0:
30          exit_simulation()
31
32
33  def univ_finish(self):
34      pass
```

※ 終了条件は、personの数が0になった場合で判定しているため、step=1でpersonを生成しない場合は、終了条件を変更してください。

④ Universeのルールを記述する（6）

構造計画研究所
KOZO KEIKAKU ENGINEERING Inc.

- create_person_stepは、person_dictに格納したstep毎の行動計画を使って、任意のstepでpersonを生成します。このとき、personを生成するセルは出発地IDと同じ値かつ他のpersonが存在しないセルを選択します。

```
37  # step毎にpersonを生成
38  def create_person_step(self):
39
40      import random
41
42      ct_step = str(count_step())
43      if ct_step in Universe.person_dict: # 該当のstepでpersonを生成する場合
44          person_list = Universe.person_dict[ct_step]
45          for one_person_list in person_list:
46              one_person = create_agt(Universe.map.person)
47              one_person.person_id = one_person_list['person_id']
48              one_person.step = one_person_list['step']
49              one_person.speed = one_person_list['speed']
50              one_person.start_id = one_person_list['start_id']
51              one_person.goal_id = one_person_list['goal_id']
52
53              start_cell_list = Universe.property_dict[str(one_person.start_id)]
54              for i in range(10):
55                  one_cell = random.choice(start_cell_list)   # 出発地IDのセルをランダムで取得
56                  pos_list = one_cell.split(',')
57                  px = int(pos_list[0])
58                  py = int(pos_list[1])
59                  p_layer = int(pos_list[2])
60                  if Universe.map.stay[px, py, p_layer] == 0: # 該当セルにpersonが不在の場合
61                      break
62              one_person.x = px
63              one_person.y = py
64              one_person.layer = p_layer
65              one_person.color = self.get_color(one_person.goal_id)
66              Universe.map.stay[px, py, p_layer] += 1
```

⑤ personのルールを記述する

構造計画研究所
KOZO KEIKAKU ENGINEERING Inc.

- personのagt_stepでは、自らの目的地のポテンシャルに沿って、移動します。
  周囲8方向のうち、歩行可能で他のpersonが存在しないセルの中からポテンシャル値が最も低いセルを選び、speedの距離だけ移動します。
  ゴールに到着すると到着人数を加算し、エージェントを消去します。

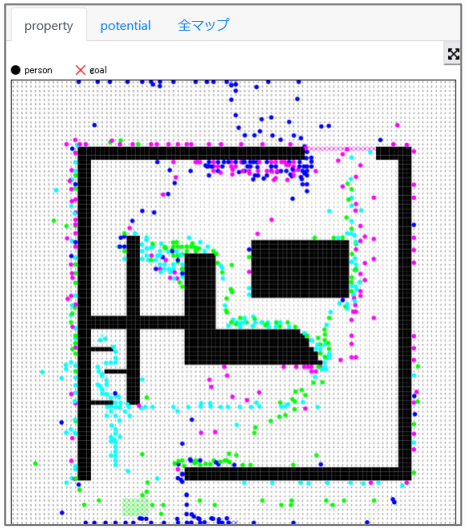
```
def agt_step(self):

    w = get_width_space(Universe.map)
    h = get_height_space(Universe.map)

    # 目的地のポテンシャルを取得
    potential_map = Universe.potential_dict[str(self.goal_id)]

    # ポテンシャル値の取得関数
    potential_dict = {}
    def get_potential(cx, cy, c_layer):
        if 0 <= cx and cx < w and 0 <= cy and cy < h:
            cx = int(cx)
            cy = int(cy)
            if Universe.map.stay[cx, cy, c_layer] == 0:
                key = str(cx) + ',' + str(cy) + ',' + str(c_layer)
                if potential_map[key] < 0:
                    potential_dict[key] = potential_map[key]

    distance = self.speed
    goal_flag = False
    for i in range(int(distance)+1):
        # 周囲8方向でポテンシャル値が低い方へ移動
        px = self.x
        py = self.y
        p_layer = self.layer
        get_potential(px-1, py, p_layer)   # 左
        get_potential(px+1, py, p_layer)   # 右
        get_potential(px, py+1, p_layer)   # 上
        get_potential(px, py-1, p_layer)   # 下
        get_potential(px-1, py+1, p_layer)   # 左上
        get_potential(px+1, py+1, p_layer)   # 右上
        get_potential(px-1, py-1, p_layer)   # 左下
        get_potential(px+1, py-1, p_layer)   # 右下
```

※ ポテンシャルの計算は周囲4方向で求めましたが、personの移動は周囲8方向を対象としています。これは、移動方向を周囲4方向にすると、personの動きがカクカクした動きになるため、周囲8方向にしています。

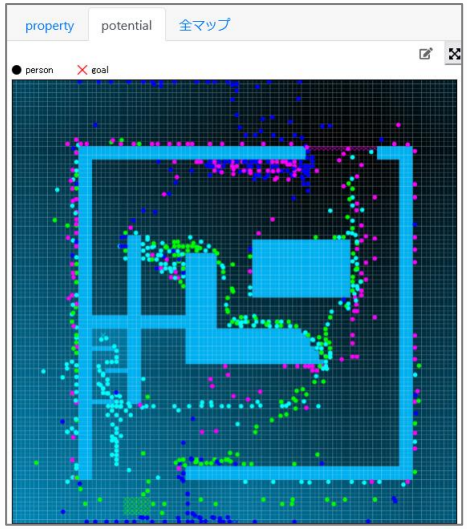
⑥ 出力画面を定義する

構造計画研究所
KOZO KEIKAKU ENGINEERING Inc.

マップ出力画面を定義します



- 出力設定 > property を表示します。
  マップ要素リストは、以下の通り定義しています。
  person： マーカー「円」、表示色「color」
  goal： マーカー「×」、表示色「color」
  property：グラデーション指定「0:黒」～「1:白」、
  変数表示「property」



- 出力設定 > potential を表示します。
  マップ要素リストは、以下の通り定義しています。
  person： マーカー「円」、表示色「color」
  goal： マーカー「×」、表示色「color」
  potential：グラデーション指定「-200:黒」～「0:青」

[Tips] artisoc Cloudのライセンスについて

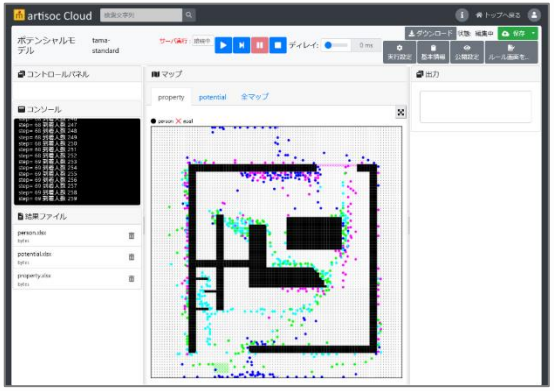
構造計画研究所
KOZO KEIKAKU ENGINEERING Inc.

standardライセンスにアップグレードすると
多くのpythonライブラリが利用できます

■ academicライセンス
- 教育機関の教員および学生向けの研究活動を支援するための無償ライセンス

■ standardライセンス
- 高度な研究やビジネス利用のための有償ライセンス
- 高速なサーバ上でシミュレーションを実行
- 多くのpythonライブラリが利用できます

※ openpyxlをインポートし、person.xlsxとproperty.xlsxを読み込んで動作します



05. レシピブック　サンプルモデル2

standardライセンスのみで利用可能なライブラリ

ライブラリ名	説明
ezdxf	dxfファイルの読み書き
feather-format	データ保存用の軽量なフォーマット
fiona	GISデータの読み書き
folium	leaflet.jsを使ったhtmlを作成できる
geojson	geojsonファイルの読み書き
geopandas	pandasのデータ形式であるDataFrameでGISデータを操作できるように拡張したライブラリ
geopy	世界中の住所、都市、国、ランドマークの座標を簡単に見つけることができる
glob	特定のパターンにマッチするファイルを取得できる
gym	強化学習のシミュレーション環境
libsumo	交通シミュレータSUMOのパッケージ
mercantile	XYZ スタイルの球状メルカトルタイルを操作するためのユーティリティ モジュール
openpyxl	Excelファイルの読み書き
osmnx	OpenStreetMap から道路ネットワークをダウンロード、モデル化、分析、視覚化するためのパッケージ
pgmpy	ベイジアンネットワークを構築するためのモジュール
pyproj	座標系・測地系の変換、緯度経度からの2点間の距離・方位角計算
pyrosm	OpenStreetMapファイル（*.pbf）の読み書き
pyshp	シェープファイル（*.shp）の読み書き
scikit-learn	機械学習ライブラリ
shapely	GISデータを操作および分析するためのライブラリ
simpy	離散イベントシミュレーションのフレームワーク
sympy	記号数学用のライブラリ
tensorflow	機械学習用フレームワーク