

1. 需求分析

1.1. 问题描述

仿真程序中需要“人”和“检测点”，并且各自有其属性；随机产生若干人、随机去检测点、检测点根据排队人数不同有不同状态；充分体现模块化以及面向对象思想

1.2. 对问题的理解

对于人：

- 概率可调地随机生成是否阳性
- 每人具有身份识别码 id，这里直接按排队顺序生成
- 随机生成合理的中文姓名
- 随机生成合理的中国手机号码
- 生成人数可以在配置文件中编辑、两个人去排队的时间间隔也应当可以在配置文件里设置
- 每人记录排队状态（未排队、排队中、采样完成，等待结果、核算结果已出、检测试管不足，未能采样）、排队时间、排队用时、（是否）完成采样、核酸检测地点、核酸检测结果（未完成、阴性、混管异常）

对于检测点：

- 设置两个空列表，用于储存排队中的“人”、还有等待攒满（以十混一为例）混管的“人”
- 可以自定义每个人做核酸用时（随机范围）
- 可以自定义混管数量（几混一）
- 可以根据排队人数变化上报拥挤状况
- 可以根据剩余名额报警

作业要求未提及但认为需要做的功能：

- “人”数据按 csv 格式再保存一份，方便数据可视化
- 数据实时保存，方便程序停止后继续先前保存的数据继续运行，也为了方便停止程序中途查看 csv 表格（暂时没想边看表格边读写数据）
- 人的生成数量和监测点生成通过 config 文件自定义，因为我认为有意义的数字才能反映有意义的将结果，即使纯随机也需要有一个范围，这里我选择数量自定义，真正赋予这个程序一个用途

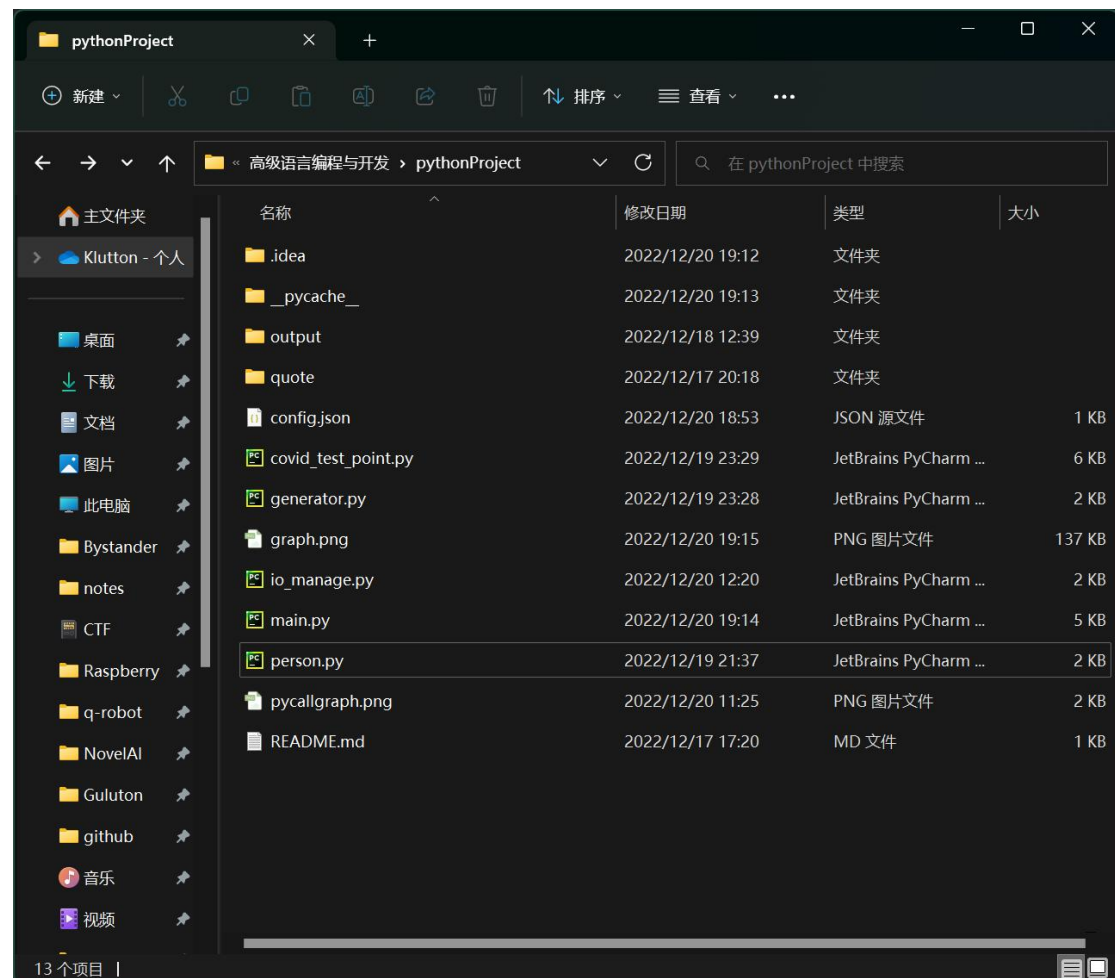
2. 程序设计

2.1. 概要设计

- 编写主程序用于调用其他程序
- 编写 person 模块用于储存不同状态以及 Person 类的属性
- 编写 covid_test_point 模块用于写 Point 类，类中包含多种函数，包括 Person 类的出队、入队函数；核酸检测（以十混一为例）的函数用于检查是否有阳性；一些自身指标检测的函数用于播报
- 编写 io_manage 模块用于读取、储存数据，数据文件以 dat 二进制文件储存；可视化表格名单以 csv 文件另存一份
- 编写 generate 模块用于生成并返回 Person 列表、Point 列表，该模块可以调用函数生成合理的姓名和手机号码
- Config 配置文件用 json 文件储存

2.2. 详细设计

使用的文件如下：



代码流程可以看代码清单运行结果的截图

首先在 config.json 里编辑参数

person:

interval: 等待 interval 秒后下一个人去排队

generate_number: 生成人数

positive_possibility: 生成人为阳性的概率

point:

generate_number: 生成检测点数量

time_cost_range: 生成检测点每次核酸检测用时范围，在生成时会根据此范围生成一个新的范围，先在此范围生成一个浮点数，以此浮点数为基准， $\pm 3s$ 为该核酸检测点的用时范围

capacity_range: 检测点的试管数生成范围

mixed_sample_amount: 每个试管混合样本数（几混一）

people_directory: 排队人员的 dat 文件以及 csv 文件的保存路径（文件夹）

point_directory: 核酸检测点的 dat 文件保存路径（文件夹）

telnum_directory: 生成电话号码配置文件，如果想要自定义需要遵守默认文件的格式，生成电话号码时会随机选择一行之后在 prefix 文件后随机生成四位数作为此人的电话号码（如果有重复，重新生成，上限是绝对够中国人人口数量的，虽然随机算法没有优化，但足够支撑万级人数的模拟） p.s. 后两列没用，该 csv 文件来自于 GitHub，并非个人整理

restart: 填 true false，是否重新开始（false 就接着先前保存的文件继续模拟，若导入失败会有响应报错）

	A	B	C	D	E	F	G	H
1	sid	prefix	province	city	service	prefixx	prefixxx	
2	1	1300000	山东	济南	中国联通	531	250000	
3	2	1300001	江苏	常州	中国联通	519	213000	
4	3	1300002	安徽	巢湖	中国联通	565	238000	
5	4	1300003	四川	宜宾	中国联通	831	644000	
6	5	1300004	四川	自贡	中国联通	813	643000	
7	6	1300005	陕西	西安	中国联通	29	710000	
8	7	1300006	江苏	南京	中国联通	25	210000	
9	8	1300007	陕西	西安	中国联通	29	710000	
10	9	1300008	湖北	武汉	中国联通	27	430000	
11	10	1300009	陕西	西安	中国联通	29	710000	
12	11	1300010	北京	北京	中国联通	10	100000	

```
io_manage.py x config.json x
1      {
2          "person": {
3              "interval": 1,
4              "generate_number": 50,
5              "positive_possibility": 0.1
6          },
7          "point": {
8              "generate_number": 3,
9              "time_cost_range": [0.1, 0.1],
10             "capacity_range": [3, 5],
11             "mixed_sample_amount": 10
12         },
13         "people_directory": "./output",
14         "point_directory": "./output",
15         "telnum_directory": "./quote/phonelocation.csv",
16         "restart": true
17     }
```

启动 main.py 来开始程序，

main 函数先调用 setup 函数导入配置文件

检测 restart 是否为真

若为真，调用 io_manage 的 read 函数，返回 people: list, person: list, point: int，若出错，提示后推出

若为假，带着 config 的参数调用 generator 中 GeneratePerson 类和 GeneratePoint 类的 generate 函数，分别返回由 person.Person 类和 covid_test_point.Point 类组成的列表，设置 start_point 全局变量为 0

初始化后，执行 start_queueing 函数，首先尝试遍历所有 point 测试点，调用 Point.check_queue_long(self)，播报一次各核酸点的长度

然后开始 Person 类的遍历，定义 index = start_point，即从 _person_queue 的 index 位置开始遍历，每隔 0.5s 遍历一次，检查目前过去的时间是否已经超过 interval，若超过，生成一个随机数用于随机选择 _point_queue 中的 Point 对象，在 if 中调用 Point.in_queue 和 check_full

Point.in_queue 会先调用自己的 check_queue_long('in')，该函数会检查自己所属的 Point 类所做核酸名额是否已满，满了会返回 False，并在第一次满时广播自己名额已经耗尽，若没满，会根据附加的字符串 in 还是 out 生成一个 fix 值，用于后续检测排队人数，广播该检测点拥挤程度，随后返回 True

in_queue 在得到 True 返回值后继续执行，否则返回 False。

首先判断所在类的 `start_time` 是否为-1, 是就调用 `new_cost_time` 根据用时范围随机生成一个新的检测用时并重置 `start_time` 为当前时间 (相当于激活了)

随后自身的 `complete_num+=1`, 用于计数
为入队的 `Person` 开始时间和排队状态等等赋值
将入队的 `Person` 加入到自己的 `_queue` 列表中
广播 `xx` 入队 (详见代码)
返回 `True`

`in_queue` 没有返回 `True`, 可能存在所有队列全部名额耗尽的情况, 因此判断时还会调用 `check_full` 遍历所有 `point` 的 `full` 变量是否都为 `True`, 若都为 `True`, 先修改 `main` 文件的 `full` 全局变量为 `True`, 然后播报所有检测点名额均已耗尽, 同样也继续执行下面的语句:

下面调用 `io_manage` 写入保存 `person` 列表 (csv 文件也会同步), 该循环 `break`

写 `for` 循环遍历 `_point_queue`, 执行每一个 `Point` 的 `test` 函数

`test` 函数会先检查 `start_time` 变量是否为-1, 若为-1, 直接返回 (即检查核酸点计时是否激活)

随后检查 `time.time() - self.current_cost_time > self.start_time` 是否为真, 模拟检测是否做完了这个人的核酸, 检查自己的 `_queue` 列表长度是否为 0 (没有人在排队) (若不满足则什么也不做)

若为真, 执行自身的 `out_queue` 函数
`individual: person.Person = self._queue.pop(0)` 从排队列表中拿出一个人, 为其属性状态进行更新 (详见代码) 随后广播采样完毕
将该类放入 `_pending` 列表, 执行 `check_ten_samples`, 即检查人数是否满 `mix` 个 (即满混管数量), (如果附加参数 `True` 即直接清空 `_pending` 队列, 用于后面清空核酸点用)

若为真, 遍历 `_pending` 中的 `Person.positive` 来检查是否有阳性, 随后带着参数执行 `test_complete_broadcast`, 将所有 `_pending` 中的类状态更新并移出列表

随后回到 `test` 函数, 调用 `new_cost_time` 生成一个新的时间

回到 `start_queueing`, 检查全局变量 `full` 是否为真, 是则说明所有名额耗尽, 遍历剩余 `Person` 类, 设置排队状态, 随后执行 `wait_end` 函数 (带着 `index`)

`wait_end` 函数中遍历 `_point_queue`, 执行列表成员的 `clear` 函数
当其排队列表和待核酸结果列表均为空, 在第一次为真时广播该检测点核酸检测结束并返回 1 用于计数

当其排队列表为空, 带核酸结果列表长度且待核酸结果列表长度小于混管数时 (按理说肯定

小于混管数，为了方便阅读），`check_ten_samples(True)`对剩下这几人出具核算结果

当都不满足时，执行 `test` 函数继续该检测点核酸检测

每次 `for` 循环都会保存一次数据

`wait_end` 执行完毕后，保存数据，广播模拟结束

另一种情况是，核酸点名额够用，即 `while` 循环结束，所有人都在排队
`while` 循环结束后，广播所有人都入队，执行 `wait_end` 函数（带着 `index`）等待所有核酸点完成，广播模拟结束。

在此过程中出错（理应只会因为程序被 `ctrl c` 中断才会报错），会打印程序中断，随后打印 `Exception`

此程序全程数据即变即存，但 `io_manage` 优化可能不是很好，但已经满足程序需求

3. 代码清单

main.py

```
1.  import random
2.  import generator
3.  import io_manage
4.  import json
5.  import time
6.  from pycallgraph import PyCallGraph
7.  from pycallgraph.output import GraphvizOutput
8.  from pycallgraph import Config
9.  import person
10.
11.  interval: float
12.  restart: bool = True
13.  start_point: int
14.  _person_queue: list
15.  _point_queue: list
16.  full = False
17.
18.
19.  def setup():
```

```

20.         with open('config.json', 'r') as f:
21.             conf = json.loads(f.read())
22.             generator.time_cost_range = conf['point']['time_cost_range']
23.             generator.capacity_range = conf['point']['capacity_range']
24.             generator.mix = conf['point']['mixed_sample_amount']
25.             generator.tel_csv_pos = conf['telnum_directory']
26.             io_manage.people_directory = conf['people_directory']
27.             io_manage.point_directory = conf['point_directory']
28.             global interval, restart, start_point
29.             interval = conf['person']['interval']
30.             restart = conf['restart']
31.
32.         return (conf['person']['positive_possibility'], conf['person']['generate
        _number']), conf['point']['generate_number']
33.
34.
35.     def check_full():
36.         cnt = 0
37.         for point in _point_queue:
38.             if point.full:
39.                 cnt += 1
40.             if cnt == len(_point_queue):
41.                 global full
42.                 full = True
43.                 print("***所有核酸点试管均耗尽")
44.                 return True
45.         else:
46.             return False
47.
48.
49.     def wait_end(index: int):
50.         ret = 0
51.         while ret != len(_point_queue):
52.             ret = 0
53.             for queue in _point_queue:
54.                 ret += queue.clear()
55.                 io_manage.point_write(_point_queue)
56.                 io_manage.person_write(_person_queue, index)
57.
58.
59.     def start_queueing():
60.         try:
61.             for queue in _point_queue:
62.                 queue.check_queue_long()

```

```
63.         t = time.time()
64.         index = start_point
65.         while index < len(_person_queue):
66.             time.sleep(0.5)
67.
68.             if time.time() - t >= interval:
69.                 t = time.time()
70.                 while True:
71.                     rand = random.randint(0, len(_point_queue) - 1)
72.                     if _point_queue[rand].in_queue(_person_queue[index]) or
        check_full():
73.                         io_manage.person_write(_person_queue, index)
74.                         break
75.
76.                 for point in _point_queue:
77.                     point.test()
78.                     io_manage.point_write(_point_queue)
79.
80.                 if full:
81.                     while index < len(_person_queue):
82.                         _person_queue[index].state = person.Status.error
83.                         index += 1
84.
85.                     wait_end(index)
86.                     io_manage.point_write(_point_queue)
87.                     io_manage.person_write(_person_queue, index)
88.                     print("测试试管耗尽，未排队者未能核酸，模拟结束")
89.                     return
90.
91.                 index += 1
92.
93.         print(f"\n***所有人均已进入核酸队伍，共{len(_person_queue)}人")
94.         wait_end(index)
95.
96.         print("\n 模拟结束! ")
97.
98.     except Exception as e:
99.         print(f"程序中断，资料已保存\n{e}")
100.
101.
102. def main():
103.     # 获取生成参数
104.     global _person_queue, _point_queue, start_point
105.     person_gen, point_gen = setup()
```



```

106.     print("初始化完成")
107.     if restart:
108.         _person_queue = generator.GeneratePerson(person_gen[0], person_gen[1]
109.             ).generate()
110.         _point_queue = generator.GeneratePoint(point_gen).generate()
111.         start_point = 0
112.         print("随机数据已经生成")
113.         start_queueing()
114.     else:
115.         try:
116.             print("等待数据载入, 请不要终止程序")
117.             _person_queue, _point_queue, start_point = io_manage.read()
118.             print("数据已经载入")
119.             except Exception as e:
120.                 print(f"尝试启动 restart 功能启动\n{e}")
121.                 start_queueing()
122.
123.
124. if __name__ == '__main__':
125.     config = Config()
126.     graphviz = GraphvizOutput()
127.     graphviz.output_file = 'graph.png'
128.
129.     with PyCallGraph(output=graphviz, config=config):
130.         main()

```

generator.py

```

1.     import _io
2.     import csv
3.     import random
4.     from quote import get_name
5.     import person
6.     import covid_test_point
7.
8.     tel_csv_pos = ''
9.     time_cost_range = []
10.    capacity_range = []
11.    nums = []
12.    mix: int
13.
14.
15.    def rand_positive(pos):

```

```

16.         val = random.uniform(0, 1)
17.         if val > pos:
18.             return False
19.         else:
20.             return True
21.
22.
23.     def get_tel(reader:list):
24.         rand = random.randint(0, len(reader)-1)
25.         nums.append(rand)
26.         prefix = reader[rand][1]
27.         province = reader[rand][2]
28.         city = reader[rand][3]
29.         tel = prefix + f"{str(random.randint(0, 9999)):0>4}"
30.         return tel, province, city
31.
32.
33.     class GeneratePerson:
34.
35.         def __init__(self, positive_possibility: float, generate_number: int = 1
00):
36.             self.positive_possibility = positive_possibility
37.             self.generate_number = generate_number
38.
39.         def generate(self):
40.             ret = []
41.             with open(fr"{tel_csv_pos}", 'r', encoding='utf-8') as f:
42.                 reader = list(csv.reader(f))
43.                 for i in range(1, self.generate_number + 1):
44.                     id_ = i
45.                     name = get_name.random_chinese_name()
46.                     positive = rand_positive(self.positive_possibility)
47.                     tel, province, city = get_tel(reader)
48.                     while tel in nums:
49.                         tel, province, city = get_tel(reader)
50.                     nums.append(tel)
51.                     ret.append(person.Person(id_, name, positive, tel, province,
city))
52.
53.             return ret
54.
55.
56.     class GeneratePoint:
57.

```

```

58.         def __init__(self, generate_number: int = 10):
59.             self.generate_number = generate_number
60.
61.         def generate(self):
62.             ret = []
63.
64.             for i in range(1, self.generate_number + 1):
65.                 id_ = i
66.                 name = f'Covid-19 test point #{str(i)}'
67.                 time_index = random.uniform(time_cost_range[0], time_cost_range[
1])
68.                 time_cost = [time_index - 3, time_index + 3]
69.                 capacity = random.randint(capacity_range[0], capacity_range[1])
70.
71.                 ret.append(covid_test_point.Point(id_, name, time_cost, capacity,
mix))
72.
73.                 print(f"生成{name}, 试管数量{capacity}")
74.
75.             return ret

```

person.py

```

1.     class Result:
2.         error = "混管异常"
3.         fine = "阴性"
4.         pending = "未完成"
5.
6.
7.     class Status:
8.         error = "检测试管不足, 未能采样"
9.         complete = "核酸结果已出"
10.        pending = "采样完成, 等待结果"
11.        queueing = "排队中"
12.        waiting = "未排队"
13.
14.
15.    class Person:
16.        def __init__(self, id_: int, name: str, positive: bool, tel: str, provin
ce: str, city: str,
17.                        state: int = Status.waiting, queue_time: float = None, queu
eing_time_cost: float = None, checked: bool = False,
18.                        check_time: float = None, check_org: str = None, check_res:
str = Result.pending):
19.            self.id_ = id_

```

```

20.         self.name = name
21.         self.positive = positive
22.         self.tel = tel
23.         self.province = province
24.         self.city = city
25.         self.state = state
26.         self.queue_time = queue_time
27.         self.queueing_time_cost = queueing_time_cost
28.         self.checked = checked
29.         self.check_time = check_time
30.         self.check_org = check_org
31.         self.check_res = check_res
32.         self.strf_queue_time = None
33.         self.strf_check_time = None
34.
35.     def is_positive(self):
36.         return self.positive

```

getname.py 中姓氏均硬编码，不放上去了，此功能是从 GitHub 上下载的
<https://github.com/Donghaopeng/robotframework-RandomName>

covid_test_point.py

```

1.     import time
2.     import random
3.
4.     import person
5.
6.
7.     class Point:
8.
9.         def __init__(self, id_: int, name: str, time_cost: list, capacity: int,
            mix: int, checked_people: int = 0):
10.             self.id_ = id_
11.             self.name = name
12.             self.time_cost = time_cost
13.             self.capacity = capacity
14.             self.mix = mix
15.             self.people_capacity = capacity * mix
16.             self.checked_people = checked_people
17.             self.complete_num = 0
18.             print(f"核酸点: {self.name} 人流量状态: 空闲")
19.

```

```
20.         self.start_time: float = -1
21.         self.current_cost_time: float
22.         self._queue = []
23.         self._pending = []
24.         self.full = False
25.         self.lack = False
26.         self.stop = False
27.
28.     def new_cost_time(self):
29.         self.current_cost_time = random.uniform(self.time_cost[0], self.time
        _cost[1])
30.         self.start_time = time.time()
31.
32.     def clear(self):
33.         if len(self._queue) == 0 and len(self._pending) == 0:
34.             if not self.stop:
35.                 print(f"{self.name} 全部核酸采样完成，采样数量：
        {self.complete_num}")
36.                 self.stop = True
37.                 return 1
38.             elif len(self._pending) < self.mix and len(self._queue) == 0:
39.                 self.check_ten_samples(True)
40.                 return 0
41.             else:
42.                 self.test()
43.                 return 0
44.
45.     def test(self):
46.         if self.start_time == -1:
47.             return
48.         if time.time() - self.current_cost_time > self.start_time and len(se
        lf._queue) != 0:
49.             self.out_queue()
50.             self.new_cost_time()
51.
52.     def in_queue(self, individual: person.Person):
53.         if self.check_queue_long('in'):
54.             if self.start_time == -1:
55.                 self.new_cost_time()
56.                 self.complete_num += 1
57.                 # 计数器
58.                 individual.queue_time = time.time()
59.                 individual.state = person.Status.queueing
60.                 self._queue.append(individual)
```

```

61.         t = time.strftime("%Y-%m-%d %H:%M:%S", time.localtime(individual.
    queue_time))
62.         individual.strf_queue_time = t
63.         # print 广播
64.         print(
65.             f"\n+++ 开 始 排
    队 {individual.name} id{str(individual.id_)}\n{individual.tel} {individual.prov
    ince}"
66.             f"\n 监测点为: {self.name}, 时间: {t}\n 排在第{len(self._queue)}
    个")
67.         return True
68.     else:
69.         return False
70.
71.     def out_queue(self):
72.
73.         t_now = time.time()
74.         individual: person.Person = self._queue.pop(0)
75.         t = time.strftime("%M:%S", time.localtime(t_now - individual.queue_t
    ime))
76.         individual.queueing_time_cost = t
77.         individual.state = person.Status.pending
78.         individual.checked = True
79.         individual.check_time = time.time()
80.         individual.strf_check_time = time.strftime("%Y-%m-%d %H:%M:%S", time.
    localtime(individual.check_time))
81.         individual.check_org = self.name
82.
83.         # print 广播
84.         print(f"\n---采样完成 {individual.name} id{str(individual.id_)}\n 监测
    点为: {self.name}, 用时: {t}\n")
85.         self._pending.append(individual)
86.         # 检查是否满十个样本
87.         self.check_ten_samples()
88.         self.check_queue_long('out')
89.
90.     def check_queue_long(self, _type: str = ''):
91.         fix = 0
92.         if _type == 'in':
93.             fix = -1
94.         elif _type == 'out':
95.             fix = 1
96.         # 检查本核酸点人流量情况
97.         if self.complete_num == self.people_capacity:

```

```

98.         if not self.full:
99.             print(f'{self.name}试管耗尽! ')
100.             self.full = True
101.             return False
102.         else:
103.             # 在此处设置人数剩余 20 的提醒
104.             if self.people_capacity - self.complete_num - 1 <= 50 and not self.lack:
105.                 print(f"\n 核酸点: {self.name} 采样管即将耗尽! (数量小于五根)\n")
106.                 self.lack = True
107.
108.                 l = len(self._queue) # 目前人数
109.                 lp = l + fix # 先前人数
110.                 if l == 9 and lp == 10:
111.                     print(f"\n 核酸点: {self.name} 人流量状态: 空闲\n")
112.                 elif (l == 11 and lp == 10) or (l == 40 and lp == 41):
113.                     print(f"\n 核酸点: {self.name} 人流量状态: 拥挤\n")
114.                 elif l == 40 and lp == 39:
115.                     print(f"\n 核酸点: {self.name} 人流量状态: 饱和\n")
116.
117.                 return True
118.
119.     def check_ten_samples(self, stop: bool = False):
120.         # 等待积累十人
121.         if len(self._pending) == self.mix or stop:
122.             # 判断其中是否有阳性
123.             for i in range(0, len(self._pending)):
124.                 if self._pending[i].positive:
125.                     self.test_complete_broadcast(person.Result.error)
126.                     return
127.
128.             self.test_complete_broadcast(person.Result.fine)
129.
130.     def test_complete_broadcast(self, result: str):
131.         completed = ''
132.
133.         while len(self._pending) > 0:
134.             p = self._pending.pop(0)
135.             p.check_res = result
136.             p.state = person.Status.complete
137.             completed += " " + str(p.id_)
138.             print(f'\n 核酸点: {self.name} 检测结果已出\nid:{completed}\n 结果为{result}\n')

```

io_manage.py

```
1.     import csv
2.     import person
3.     import pickle
4.
5.     people_directory = ''
6.     point_directory = ''
7.
8.
9.     def person_write(_list: list, index: int):
10.         with open(fr"{people_directory}/参加核酸人员.csv", 'w') as f:
11.             head = ['id', '姓名', '是否为阳性', '电话号码', '省', '市', '排队状态',
12.                     '完成采样', '采样完成时间', '核酸检测地点', '核酸检测结果']
13.             data = []
14.             for individual in _list:
15.                 data.append({
16.                     'id': individual.id_, '姓名': individual.name, '是否为阳性': individual.positive, '电话号码': individual.tel,
17.                     '省': individual.province, '市': individual.city,
18.                     '排队状态': individual.state, '排队时间': individual.strf_queue_time, '排队用时': individual.queueing_time_cost,
19.                     '完成采样': individual.checked, '采样完成时间': individual.strf_check_time, '核酸检测地点': individual.check_org,
20.                     '核酸检测结果': individual.check_res
21.                 })
22.
23.             writer = csv.DictWriter(f, head)
24.             writer.writeheader()
25.             writer.writerows(data)
26.
27.             with open(fr"{people_directory}/people.dat", 'wb') as f:
28.                 pickle.dump(_list, f)
29.
30.             with open(fr"{people_directory}/index.dat", 'wb') as f:
31.                 pickle.dump(index, f)
32.
33.
34.     def point_write(_list: list):
35.         with open(fr"{people_directory}/point_data.dat", 'wb') as f:
36.             pickle.dump(_list, f)
37.
```



```

38.
39.  def read():
40.      with open(f"{people_directory}/people.dat", 'rb') as f:
41.          people = pickle.load(f)
42.      with open(f"{point_directory}/point_data.dat", 'rb') as f:
43.          point = pickle.load(f)
44.      with open(f"{people_directory}/index.dat", 'rb') as f:
45.          index = pickle.load(f)
46.      return people, point, index

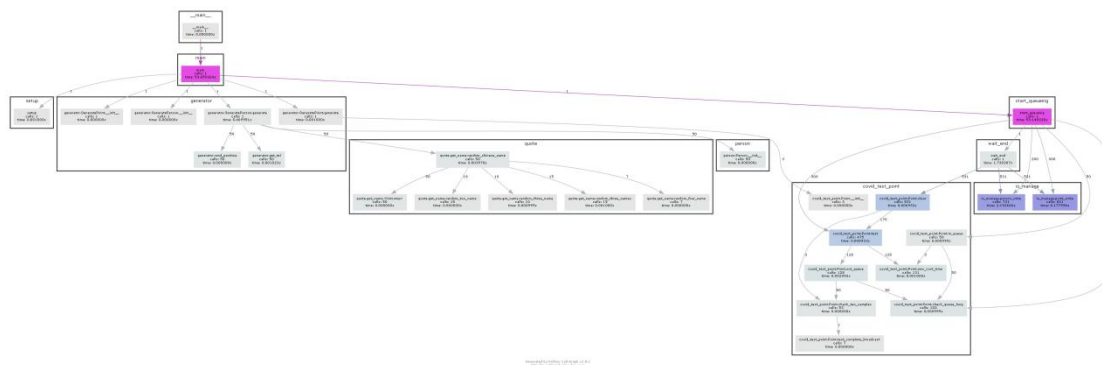
```

4. 运行结果

```

{
  "person": {
    "interval": 1,
    "generate_number": 50,
    "positive_possibility": 0.1
  },
  "point": {
    "generate_number": 3,
    "time_cost_range": [0.1,0.1],
    "capacity_range": [3,5],
    "mixed_sample_amount": 10
  },
  "people_directory": "./output",
  "point_directory": "./output",
  "telnum_directory": "./quote/phonelocation.csv",
  "restart": true
}

```



参加核酸检测人员.csv - Excel(产品激活失败)

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
34																			
35	17	尹偏明	FALSE	15834625376	吉林	白城	核酸检测已出	2022/12/20 19:14	0.01	TRUE	2022/12/20 19:15 Covid-19 test point #3	混管异常							
36	18	祁井除	FALSE	15564810691	山东	泰安	核酸检测已出	2022/12/20 19:15	0.01	TRUE	2022/12/20 19:15 Covid-19 test point #1	混管异常							
37	19	大史雄	TRUE	18670144559	江西	赣州	核酸检测已出	2022/12/20 19:15	0.00	TRUE	2022/12/20 19:15 Covid-19 test point #3	混管异常							
38	20	狄拔	FALSE	15869226412	浙江	丽水	核酸检测已出	2022/12/20 19:15	0.02	TRUE	2022/12/20 19:15 Covid-19 test point #2	混管异常							
41	21	郎同伦	FALSE	17033802653	天津	天津	核酸检测已出	2022/12/20 19:15	0.00	TRUE	2022/12/20 19:15 Covid-19 test point #1	混管异常							
42	22	郭罗汉	FALSE	13197605492	广西	防城港	核酸检测已出	2022/12/20 19:15	0.00	TRUE	2022/12/20 19:15 Covid-19 test point #2	混管异常							
43	23	郭罗超史	FALSE	15199466291	新疆	乌鲁木齐	核酸检测已出	2022/12/20 19:15	0.02	TRUE	2022/12/20 19:15 Covid-19 test point #1	阴性							
44	24	唐合	FALSE	18761622970	江苏	南京	核酸检测已出	2022/12/20 19:15	0.02	TRUE	2022/12/20 19:15 Covid-19 test point #2	混管异常							
45	25	金亮	FALSE	18852320851	江苏	淮安	核酸检测已出	2022/12/20 19:15	0.01	TRUE	2022/12/20 19:15 Covid-19 test point #3	混管异常							
46	26	滕授	TRUE	18927060599	广东	揭阳	核酸检测已出	2022/12/20 19:15	0.01	TRUE	2022/12/20 19:15 Covid-19 test point #2	混管异常							
47	27	谈急	FALSE	18476968910	福建	中山	核酸检测已出	2022/12/20 19:15	0.00	TRUE	2022/12/20 19:15 Covid-19 test point #3	混管异常							
48	28	谢守	FALSE	17067338445	黑龙江	哈尔滨	核酸检测已出	2022/12/20 19:15	0.01	TRUE	2022/12/20 19:15 Covid-19 test point #1	阴性							
49	29	单于放	FALSE	13856573133	安徽	合肥	核酸检测已出	2022/12/20 19:15	0.02	TRUE	2022/12/20 19:15 Covid-19 test point #2	混管异常							
50	30	夹谷授	FALSE	13851570249	江苏	南京	核酸检测已出	2022/12/20 19:15	0.01	TRUE	2022/12/20 19:15 Covid-19 test point #2	混管异常							
51	31	乐正称	FALSE	18960591828	福建	三明	核酸检测已出	2022/12/20 19:15	0.01	TRUE	2022/12/20 19:15 Covid-19 test point #3	混管异常							
52	32	严礼唱	FALSE	17681759718	浙江	温州	核酸检测已出	2022/12/20 19:15	0.00	TRUE	2022/12/20 19:15 Covid-19 test point #3	阴性							

可见，在该阳性概率下，混管异常概率很高，由于用时设置很小，每人排队用时参考性不大，可自行更改。

后面的图片参数和上文不一样

参加核酸检测人员.csv - Excel(产品激活失败)

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
1	id	姓名	是否为阳性	电话号码	省	市	排队状态	排队时间	排队用时	完成采样	采样完成时间	核酸检测地点	核酸检测结果						
2																			
3		1	项家	FALSE	13587279663	浙江	湖州	核酸检测已出	2022/12/21 17:40	0.00	TRUE	2022/12/21 17:40 Covid-19 test point #1	阴性						
4		2	少叔逸秋	FALSE	13297495951	湖南	长沙	核酸检测已出	2022/12/21 17:40	0.02	TRUE	2022/12/21 17:40 Covid-19 test point #1	阴性						
5		3	司空空修	FALSE	15999497057	新疆	奎屯	核酸检测已出	2022/12/21 17:40	0.02	TRUE	2022/12/21 17:40 Covid-19 test point #1	阴性						
6		4	赵皮	FALSE	15670908263	河南	济源	核酸检测已出	2022/12/21 17:40	0.02	TRUE	2022/12/21 17:40 Covid-19 test point #1	阴性						
7		5	鲁圣恩	FALSE	15566923432	辽宁	大连	核酸检测已出	2022/12/21 17:40	0.01	TRUE	2022/12/21 17:40 Covid-19 test point #1	阴性						
8		6	公西桃店	FALSE	15198893145	云南	昆明	检测试管不足, 未能采样			FALSE		未完成						
9		7	洛阳像	FALSE	18467227049	青海	西宁	检测试管不足, 未能采样			FALSE		未完成						
10		8	戴酒妹	FALSE	14753648040	山东	潍坊	检测试管不足, 未能采样			FALSE		未完成						
11		9	郭罗白	TRUE	18076178177	贵州	毕节	检测试管不足, 未能采样			FALSE		未完成						
12		10	士孙克广	FALSE	18891098622	陕西	咸阳	检测试管不足, 未能采样			FALSE		未完成						

这是核酸试管不足情况下的结果

参加核酸检测人员.csv - Excel(产品激活失败)

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
73	36	何口	FALSE	15723867433	广西	百色	采样完成, 等待结果	2022/12/21 17:43	0:03	TRUE	2022/12/21 17:43	Covid-19 test point #1	未完成					
74	37	穆指	FALSE	18105905713	福建	福州	核酸结果已出	2022/12/21 17:43	0:06	TRUE	2022/12/21 17:43	Covid-19 test point #5	混管异常					
76	38	吴对南	FALSE	15974091482	湖南	永州	采样完成, 等待结果	2022/12/21 17:43	0:05	TRUE	2022/12/21 17:43	Covid-19 test point #1	未完成					
78	39	高阳吧惊	FALSE	15698485504	山西	临汾	核酸结果已出	2022/12/21 17:43	0:07	TRUE	2022/12/21 17:43	Covid-19 test point #5	混管异常					
80	40	平青	FALSE	15073644625	湖南	常德	采样完成, 等待结果	2022/12/21 17:43	0:00	TRUE	2022/12/21 17:43	Covid-19 test point #2	未完成					
82	41	敬阳和	FALSE	15332037424	天津	天津	排队中	2022/12/21 17:43		FALSE			未完成					
83	42	明凌草	FALSE	15820654856	广东	佛山	采样完成, 等待结果	2022/12/21 17:43	0:00	TRUE	2022/12/21 17:43	Covid-19 test point #2	未完成					
86	43	宇文逢良	FALSE	17683257454	四川	绵阳	排队中	2022/12/21 17:43		FALSE			未完成					
88	44	姜受	FALSE	15706694951	广东	茂名	排队中	2022/12/21 17:43		FALSE			未完成					
89	45	米达	FALSE	17738450106	四川	凉山	排队中	2022/12/21 17:43		FALSE			未完成					
92	46	唐得	FALSE	13223840893	河南	驻马店	排队中	2022/12/21 17:43		FALSE			未完成					
93	47	宇文端	FALSE	18953608537	山东	潍坊	采样完成, 等待结果	2022/12/21 17:43	0:01	TRUE	2022/12/21 17:43	Covid-19 test point #1	未完成					
95	48	万俊注周	FALSE	13353874051	河南	驻马店	核酸结果已出	2022/12/21 17:43	0:01	TRUE	2022/12/21 17:43	Covid-19 test point #5	混管异常					
96	49	曹岳菲刚	FALSE	15519655106	贵州	遵义	排队中	2022/12/21 17:43		FALSE			未完成					
100	50	汤杰取	FALSE	15069383437	山东	淄博	排队中	2022/12/21 17:43		FALSE			未完成					
101	51	东方祥复	FALSE	13931965693	河北	邢台	排队中	2022/12/21 17:43		FALSE			未完成					
102	52	松树修	FALSE	17873057275	湖南	岳阳	未排队			FALSE			未完成					

这是运行一半中断程序打开 csv 文件的结果

```

main.py
io_manage.py
main.py
config.json
generator.py
get_name.py
covid_test_point.py
person.py
match_0.py

运行: main
C:\Users\86153\AppData\Local\Programs\Python\Python38\python.exe C:\Users\86153\Desktop\市录\大\上学期\高级语言编程与开发\pythonProject\main.py
初始化完成
检测点: Covid-19 test point #1 人流量状态: 空闲
生成Covid-19 test point #1, 试管数3
检测点: Covid-19 test point #2 人流量状态: 空闲
生成Covid-19 test point #2, 试管数5
检测点: Covid-19 test point #3 人流量状态: 空闲
生成Covid-19 test point #3, 试管数4
检测点: Covid-19 test point #4 人流量状态: 空闲
生成Covid-19 test point #4, 试管数5
检测点: Covid-19 test point #5 人流量状态: 空闲
生成Covid-19 test point #5, 试管数4
随机数据已经生成

检测点: Covid-19 test point #1 采样管即将耗尽! (数量小于五瓶)

检测点: Covid-19 test point #2 采样管即将耗尽! (数量小于五瓶)

检测点: Covid-19 test point #3 采样管即将耗尽! (数量小于五瓶)

检测点: Covid-19 test point #4 采样管即将耗尽! (数量小于五瓶)

检测点: Covid-19 test point #5 采样管即将耗尽! (数量小于五瓶)

*** 开始排队 张炳是 1d1
13846815746 开始
检测点为: Covid-19 test point #1, 时间: 2022-12-21 17:49:41
排在第1个
程序中断, 资料已保存
[Errno 13] Permission denied: './output/参加核酸检测人员.csv'
运行已结束, 返回代码0

```

这是错误处理的结果

```
main x
C:\Users\86153\AppData\Local\Programs\Python\Python38\python.exe
初始化完成
核酸点: Covid-19 test point #1 人流量状态: 空闲
生成Covid-19 test point #1,试管数量3
核酸点: Covid-19 test point #2 人流量状态: 空闲
生成Covid-19 test point #2,试管数量4
核酸点: Covid-19 test point #3 人流量状态: 空闲
生成Covid-19 test point #3,试管数量5
核酸点: Covid-19 test point #4 人流量状态: 空闲
生成Covid-19 test point #4,试管数量5
核酸点: Covid-19 test point #5 人流量状态: 空闲
生成Covid-19 test point #5,试管数量3
随机数据已经生成

核酸点: Covid-19 test point #1 采样管即将耗尽! (数量小于五根)

核酸点: Covid-19 test point #2 采样管即将耗尽! (数量小于五根)

核酸点: Covid-19 test point #3 采样管即将耗尽! (数量小于五根)

核酸点: Covid-19 test point #4 采样管即将耗尽! (数量小于五根)

核酸点: Covid-19 test point #5 采样管即将耗尽! (数量小于五根)

+++ 开始排队 仲长油 id1
13110347566 陕西
监测点为: Covid-19 test point #1, 时间: 2022-12-21 17:58:02
排在第1个
```

```
---采样完成 尹方蚬 id99
监测点为: Covid-19 test point #4, 用时: 00:20

核酸点: Covid-19 test point #4 检测结果已出
id: 94 99
结果为阴性

Covid-19 test point #4全部核酸采样完成, 采样数量: 22

---采样完成 第五完 id86
监测点为: Covid-19 test point #2, 用时: 00:35

核酸点: Covid-19 test point #2 检测结果已出
id: 60 70 71 74 77 78 80 85 86
结果为混管异常

Covid-19 test point #2全部核酸采样完成, 采样数量: 19

模拟结束!
```

```
+++开始排队 闻人当兴 id100
18274381204 湖南
监测点为: Covid-19 test point #1, 时间: 2022-12-21 17:51:54
排在第3个

---采样完成 孔很和 id88
监测点为: Covid-19 test point #1, 用时: 00:12

---采样完成 太史理 id81
监测点为: Covid-19 test point #4, 用时: 00:19

---采样完成 贺条 id96
监测点为: Covid-19 test point #5, 用时: 00:04

***所有人均已进入核酸队伍, 共100人

核酸点: Covid-19 test point #3 检测结果已出
id: 92
结果为阴性
```

```
+++ 开始排队 茅尽伙 id42
15039725044 河南
监测点为: Covid-19 test point #3, 时间: 2022-12-21 17:58:44
排在第9个
```

```
+++ 开始排队 谢干运 id43
15826020371 重庆
监测点为: Covid-19 test point #5, 时间: 2022-12-21 17:58:45
排在第6个
```

```
核酸点: Covid-19 test point #4 人流量状态: 拥挤
```

```
+++ 开始排队 拓跋勒 id44
13122360662 上海
监测点为: Covid-19 test point #4, 时间: 2022-12-21 17:58:46
排在第12个
```

```
+++ 开始排队 毕音可 id45
13727828340 广东
监测点为: Covid-19 test point #4, 时间: 2022-12-21 17:58:47
排在第13个
```

```
核酸点: Covid-19 test point #4 检测结果已出
id: 9
结果为阴性
```

```
Covid-19 test point #4 试管耗尽!
***所有核酸点试管均耗尽
Covid-19 test point #1全部核酸采样完成, 采样数量: 2
Covid-19 test point #2全部核酸采样完成, 采样数量: 1
Covid-19 test point #3全部核酸采样完成, 采样数量: 3
Covid-19 test point #4全部核酸采样完成, 采样数量: 2
Covid-19 test point #5全部核酸采样完成, 采样数量: 1
测试试管耗尽, 未排队者未能核酸, 模拟结束
```

这些是程序正在运行的截屏，体现出生成时的显示、模拟即将结束的核酸队列清空（从试管中只有两个样本可以看出）、所有人均已完成核酸的提示、核酸点拥挤状态的广播、核酸点试管耗尽的广播

5. 感想和体会

通过这次大作业，梳理了我对小项目编写的思路，实地动手编写了面向对象编程。在编程的过程中，不断遇到新问题并修改让我在编写程序的同时学习了更多知识（例如数据

结构、不同需求情况下数据的读写选择使用什么格式、类的全局变量和局部变量等等)
在第一版程序做完后、在主体的基础下不断优化程序结构与逻辑,我发现写代码清单这一类工作可以梳理自己代码运行的逻辑,更好地优化自己的代码

这一次作业受益良多,同时向每一位防控疫情的工作者致敬