Python面向对象语法精讲

嵩 天

实例1：银行ATM等待时间分析

嵩 天

Python面向对象语法精讲

"银行ATM等待时间分析"需求分析

Python123

需求分析

常见的生活场景

- 银行ATM机
- 客户流：排队等待

需求分析

对于特定客户流，用户平均等待时间是多少?

- ATM机：由于不同业务，处理时间符合一定的随机分布
- 客户流：由于不同环境，客户到达符合一定的随机分布
- 平均等待时间：每个客户平均的等待时间

需求分析

简单的假设

- ATM机：服务时间在1到maxtime间随机
- 客户流：n个客户到达时间在1到arrvtime间随机
- 所有时间以整数为单位
- 进一步扩展可以采用随机分布函数

Python面向对象语法精讲

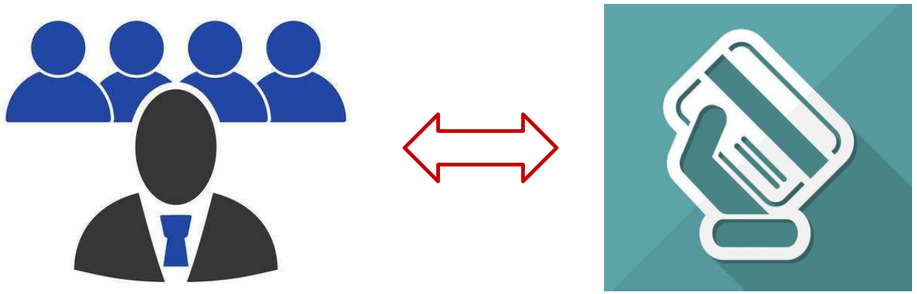
"银行ATM等待时间分析"实例解析

Python123

程序分析

面向对象程序思维

- 封装对象，重点思考对象间交互
- ATM机：ATM类，假设只有1台
- 客户流：Customers类



代码详解

```python
class ATM():
    def __init__(self, maxtime = 5):
        self.t_max = maxtime
    def getServCompleteTime(self, start = 0):
        return start + random.randint(1, self.t_max)

class Customers():
    def __init__(self, n):
        self.count = n
        self.left = n
    def getNextArrvTime(self, start = 0, arrvtime = 10):
        if self.left != 0:
            self.left -= 1
            return start + random.randint(1, arrvtime)
        else:
            return 0
    def isOver(self):
        return True if self.left == 0 else False
```

ATM类

Customers类

代码详解

```
c = Customers(100)
a = ATM()
wait_list = []
wait_time = 0
cur_time = 0
cur_time += c.getNextArrvTime()
wait_list.append(cur_time)
while len(wait_list) != 0 or not c.isOver():
    if wait_list[0] <= cur_time:
        next_time = a.getServCompleteTime(cur_time)
        del wait_list[0]
    else:
        next_time = cur_time + 1

    if not c.isOver() and len(wait_list) == 0:
        next_arrv = c.getNextArrvTime(cur_time)
        wait_list.append(next_arrv)

    if not c.isOver() and wait_list[-1] < next_time:
        next_arrv = c.getNextArrvTime(wait_list[-1])
        wait_list.append(next_arrv)
        while next_arrv < next_time and not c.isOver():
            next_arrv = c.getNextArrvTime(next_arrv)
            wait_list.append(next_arrv)

    for i in wait_list:
        if i <= cur_time:
            wait_time += next_time - cur_time
        elif cur_time < i < next_time:
            wait_time += next_time - i
        else:
            pass
    cur_time = next_time
print(wait_time/c.count)
```

生成2个对象

整体思路：

1. 需要一个全局时间
2. 以ATM每次处理结束的时间为时间驱动事件
3. 需要一个等待队列，维护客户到达时间
4. 时间变化时，驱动等待队列变化

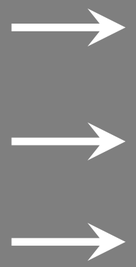
代码详解

```
c = Customers(100)
a = ATM()
wait_list = []
wait_time = 0
cur_time = 0
cur_time += c.getNextArrvTime()
wait_list.append(cur_time)
while len(wait_list) != 0 or not c.isOver():
    if wait_list[0] <= cur_time:
        next_time = a.getServCompleteTime(cur_time)
        del wait_list[0]
    else:
        next_time = cur_time + 1

    if not c.isOver() and len(wait_list) == 0:
        next_arrv = c.getNextArrvTime(cur_time)
        wait_list.append(next_arrv)

    if not c.isOver() and wait_list[-1] < next_time:
        next_arrv = c.getNextArrvTime(wait_list[-1])
        wait_list.append(next_arrv)
        while next_arrv < next_time and not c.isOver():
            next_arrv = c.getNextArrvTime(next_arrv)
            wait_list.append(next_arrv)

    for i in wait_list:
        if i <= cur_time:
            wait_time += next_time - cur_time
        elif cur_time < i < next_time:
            wait_time += next_time - i
        else:
            pass
    cur_time = next_time
print(wait_time/c.count)
```

等待队列

总等待时间

代码详解



```
c = Customers(100)
a = ATM()
wait_list = []
wait_time = 0
cur_time = 0
cur_time += c.getNextArrvTime()
wait_list.append(cur_time)
while len(wait_list) != 0 or not c.isOver():
    if wait_list[0] <= cur_time:
        next_time = a.getServCompleteTime(cur_time)
        del wait_list[0]
    else:
        next_time = cur_time + 1

    if not c.isOver() and len(wait_list) == 0:
        next_arrv = c.getNextArrvTime(cur_time)
        wait_list.append(next_arrv)

    if not c.isOver() and wait_list[-1] < next_time:
        next_arrv = c.getNextArrvTime(wait_list[-1])
        wait_list.append(next_arrv)
        while next_arrv < next_time and not c.isOver():
            next_arrv = c.getNextArrvTime(next_arrv)
            wait_list.append(next_arrv)

    for i in wait_list:
        if i <= cur_time:
            wait_time += next_time - cur_time
        elif cur_time < i < next_time:
            wait_time += next_time - i
        else:
            pass
    cur_time = next_time
print(wait_time/c.count)
```

当前时间 cur_time

下次时间 next_time

代码详解

```
c = Customers(100)
a = ATM()
wait_list = []
wait_time = 0
cur_time = 0
cur_time += c.getNextArrvTime()
wait_list.append(cur_time)
while len(wait_list) != 0 or not c.isOver():
    if wait_list[0] <= cur_time:
        next_time = a.getServCompleteTime(cur_time)
        del wait_list[0]
    else:
        next_time = cur_time + 1

    if not c.isOver() and len(wait_list) == 0:
        next_arrv = c.getNextArrvTime(cur_time)
        wait_list.append(next_arrv)

    if not c.isOver() and wait_list[-1] < next_time:
        next_arrv = c.getNextArrvTime(wait_list[-1])
        wait_list.append(next_arrv)
        while next_arrv < next_time and not c.isOver():
            next_arrv = c.getNextArrvTime(next_arrv)
            wait_list.append(next_arrv)

    for i in wait_list:
        if i <= cur_time:
            wait_time += next_time - cur_time
        elif cur_time < i < next_time:
            wait_time += next_time - i
        else:
            pass
    cur_time = next_time
print(wait_time/c.count)
```

等待队列的维护原理：

队列中保存在next_time时间前即将到达的所有客户的到达时间，以及最多一个超过next_time时间的客户到达时间

该队列始终不空

一旦等待队列为空，则增加一个客户； 队列不空保证时间前进时

代码详解

```
c = Customers(100)
a = ATM()
wait_list = []
wait_time = 0
cur_time = 0
cur_time += c.getNextArrvTime()
wait_list.append(cur_time)
while len(wait_list) != 0 or not c.isOver():
    if wait_list[0] <= cur_time:
        next_time = a.getServCompleteTime(cur_time)
        del wait_list[0]
    else:
        next_time = cur_time + 1

    if not c.isOver() and len(wait_list) == 0:
        next_arrv = c.getNextArrvTime(cur_time)
        wait_list.append(next_arrv)

    if not c.isOver() and wait_list[-1] < next_time:
        next_arrv = c.getNextArrvTime(wait_list[-1])
        wait_list.append(next_arrv)
        while next_arrv < next_time and not c.isOver():
            next_arrv = c.getNextArrvTime(next_arrv)
            wait_list.append(next_arrv)

    for i in wait_list:
        if i <= cur_time:
            wait_time += next_time - cur_time
        elif cur_time < i < next_time:
            wait_time += next_time - i
        else:
            pass
    cur_time = next_time
print(wait_time/c.count)
```

next_time受ATM机操作驱动

当ATM对象处理完当前业务时，时间前进

如果ATM对象不处理业务，则时间前进1

代码详解

```
c = Customers(100)
a = ATM()
wait_list = []
wait_time = 0
cur_time = 0
cur_time += c.getNextArrvTime()
wait_list.append(cur_time)
while len(wait_list) != 0 or not c.isOver():
    if wait_list[0] <= cur_time:
        next_time = a.getServCompleteTime(cur_time)
        del wait_list[0]
    else:
        next_time = cur_time + 1

    if not c.isOver() and len(wait_list) == 0:
        next_arrv = c.getNextArrvTime(cur_time)
        wait_list.append(next_arrv)

    if not c.isOver() and wait_list[-1] < next_time:
        next_arrv = c.getNextArrvTime(wait_list[-1])
        wait_list.append(next_arrv)
        while next_arrv < next_time and not c.isOver():
            next_arrv = c.getNextArrvTime(next_arrv)
            wait_list.append(next_arrv)

    for i in wait_list:
        if i <= cur_time:
            wait_time += next_time - cur_time
        elif cur_time < i < next_time:
            wait_time += next_time - i
        else:
            pass
    cur_time = next_time
print(wait_time/c.count)
```

如果等待队列最后一个元素在next_time内

则在时间更迭过程中，还可能到达客户

需要持续产生客户，直至其到达时间超过next_time

代码详解

```
c = Customers(100)
a = ATM()
wait_list = []
wait_time = 0
cur_time = 0
cur_time += c.getNextArrvTime()
wait_list.append(cur_time)
while len(wait_list) != 0 or not c.isOver():
    if wait_list[0] <= cur_time:
        next_time = a.getServCompleteTime(cur_time)
        del wait_list[0]
    else:
        next_time = cur_time + 1

    if not c.isOver() and len(wait_list) == 0:
        next_arrv = c.getNextArrvTime(cur_time)
        wait_list.append(next_arrv)

    if not c.isOver() and wait_list[-1] < next_time:
        next_arrv = c.getNextArrvTime(wait_list[-1])
        wait_list.append(next_arrv)
        while next_arrv < next_time and not c.isOver():
            next_arrv = c.getNextArrvTime(next_arrv)
            wait_list.append(next_arrv)

    for i in wait_list:
        if i <= cur_time:
            wait_time += next_time - cur_time
        elif cur_time < i < next_time:
            wait_time += next_time - i
        else:
            pass
    cur_time = next_time
print(wait_time/c.count)
```

在每个事件更迭周期，累积wait_time

计算平均等待时间

代码详解

```python
class ATM():
    def __init__(self, maxtime = 5):
        self.t_max = maxtime
    def getServCompleteTime(self, start = 0):
        return start + random.randint(1, self.t_max)

class Customers():
    def __init__(self, n):
        self.count = n
        self.left = n
    def getNextArrvTime(self, start = 0, arrvtime = 10):
        if self.left != 0:
            self.left -= 1
            return start + random.randint(1, arrvtime)
        else:
            return 0
    def isOver(self):
        return True if self.left == 0 else False
```

```python
c = Customers(100)
a = ATM()
wait_list = []
wait_time = 0
cur_time = 0
cur_time += c.getNextArrvTime()
wait_list.append(cur_time)
while len(wait_list) != 0 or not c.isOver():
    if wait_list[0] <= cur_time:
        next_time = a.getServCompleteTime(cur_time)
        del wait_list[0]
    else:
        next_time = cur_time + 1

    if not c.isOver() and len(wait_list) == 0:
        next_arrv = c.getNextArrvTime(cur_time)
        wait_list.append(next_arrv)

    if not c.isOver() and wait_list[-1] < next_time:
        next_arrv = c.getNextArrvTime(wait_list[-1])
        wait_list.append(next_arrv)
        while next_arrv < next_time and not c.isOver():
            next_arrv = c.getNextArrvTime(next_arrv)
            wait_list.append(next_arrv)

    for i in wait_list:
        if i <= cur_time:
            wait_time += next_time - cur_time
        elif cur_time < i < next_time:
            wait_time += next_time - i
        else:
            pass
    cur_time = next_time
print(wait_time/c.count)
```

程序结果

ATM: maxtime = 5

Customers: n = 100, arrvtime = 10

运行结果

0.43

0.44

0.91

……

```
c = Customers(100)
a = ATM()
wait_list = []
wait_time = 0
cur_time = 0
cur_time += c.getNextArrvTime()
wait_list.append(cur_time)
while len(wait_list) != 0 or not c.isOver():
    if wait_list[0] <= cur_time:
        next_time = a.getServCompleteTime(cur_time)
        del wait_list[0]
    else:
        next_time = cur_time + 1

    if not c.isOver() and len(wait_list) == 0:
        next_arrv = c.getNextArrvTime(cur_time)
        wait_list.append(next_arrv)

    if not c.isOver() and wait_list[-1] < next_time:
        next_arrv = c.getNextArrvTime(wait_list[-1])
        wait_list.append(next_arrv)
        while next_arrv < next_time and not c.isOver():
            next_arrv = c.getNextArrvTime(next_arrv)
            wait_list.append(next_arrv)

    for i in wait_list:
        if i <= cur_time:
            wait_time += next_time - cur_time
        elif cur_time < i < next_time:
            wait_time += next_time - i
        else:
            pass
    cur_time = next_time
print(wait_time/c.count)
```

程序结果

ATM: maxtime = 10

Customers: n = 100, arrvtime = 10

运行结果

62.16

36.1

33.93

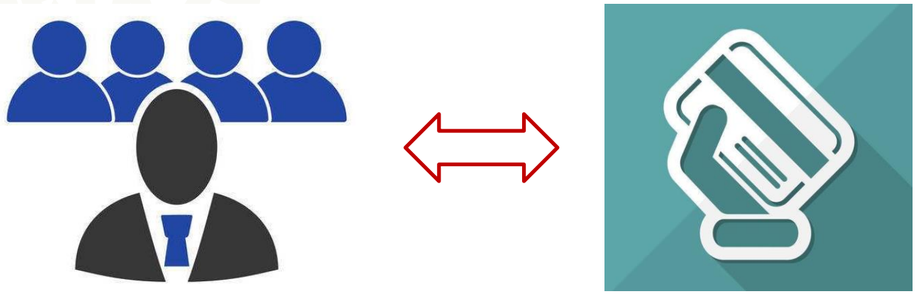
……

```
c = Customers(100)
a = ATM(10)
wait_list = []
wait_time = 0
cur_time = 0
cur_time += c.getNextArrvTime()
wait_list.append(cur_time)
while len(wait_list) != 0 or not c.isOver():
    if wait_list[0] <= cur_time:
        next_time = a.getServCompleteTime(cur_time)
        del wait_list[0]
    else:
        next_time = cur_time + 1

    if not c.isOver() and len(wait_list) == 0:
        next_arrv = c.getNextArrvTime(cur_time)
        wait_list.append(next_arrv)

    if not c.isOver() and wait_list[-1] < next_time:
        next_arrv = c.getNextArrvTime(wait_list[-1])
        wait_list.append(next_arrv)
        while next_arrv < next_time and not c.isOver():
            next_arrv = c.getNextArrvTime(next_arrv)
            wait_list.append(next_arrv)

    for i in wait_list:
        if i <= cur_time:
            wait_time += next_time - cur_time
        elif cur_time < i < next_time:
            wait_time += next_time - i
        else:
            pass
    cur_time = next_time
print(wait_time/c.count)
```

程序扩展

Python不要求全面向对象编程，但可以这样

- ATM机：ATM类，假设只有1台
- 客户流：Customers类
- 休息室：WaitingRoom类



休息室


Python 123
Thank you