الواجب المنزلي 11: محاكاة الروبوتات

مقدمة

في هذا الواجب المنزلي سوف تتمرن على تصميم محاكاة وكتابة برنامج يستخدم الصفوف.

حجم العمل:

رجاء أخبرنا كم من الوقت أمضيت على كل مسألة. نريد أن نكون حذرين بأن لا نحملكم أكثر من طاقتكم بإعطائكم مسائل تأخذ وقتاً أكثر مما نتوقعه.

في هذا الواجب المنزلي هناك عدد صفحات لا بأس به للقراءة والفهم، ولكن معظم المسائل لا تتضمن كتابة عدد كبير من الأسطر البرمجية

التعاون

يمكنك التعاون مع طلاب آخرين، إنما يجب على كل طالب أن يكتب ويسلم واجبه بشكل منفصل. قم بذكر أسماء الطلاب الذين عملت معهم على حل الواجب. من أجل تفاصيل أخرى، قم بمراجعة سياسة التعاون الموجودة في مخطط المساق.

البداية

قم بتحميل وحفظ الملفات في مجلد واحد.

- ps11.py : وهو الملف الهيكلي الذي سوف تقوم بالكتابة فيه.
- ps11_visualize.py : وهي مكتبة أنيقة نقوم بتزويدكم بها (سيتم شرحها لاحقاً).

تنصيب pylab

يجب عليك تنصيب بعض المكتبات لإنشاء مخططات للمسألة 4 وللمسألة 6 من هذا الواجب المنزلي.

تنصيب pylab على الويندوز

1. تحقق من نسخة بايثون على جهازك (في IDLE، اختار Help <– About <– IDLE) وتأكد من أن النسخة التي لديك هي بايثون 2.5
2. قم بتحميل وتنصيب numpy.
3. قم بتحميل وتنصيب Matplotlib.

إذا كان لديك ويندوز vista فمن الممكن أن تحصل على خطأ أثناء تنصيب Matplotlib يقول شيئاً قريباً من "could not create key " يجب عليك تشغيل أو إعادة تشغيل ملف التنصيب بالنقر بالزر اليمين للماوس عليه واختيار "Run as administrator" (تشغيل كمسؤول). (إذا لم تقم بذلك Matplotlib سيعمل ولكن لن يتم السماح لك بالدخول لإلغاء التنصيب حتى تقوم بتشغيل برنامج التنصيب كمسؤول)

تنصيب pylab على mac os x

1. تحقق من نسخة بايثون على جهازك (في IDLE، اختار Help <– About <– IDLE) وتأكد من أن النسخة التي لديك هي بايثون 2.5
2. قم بتحميل وتنصيب python-dateutil و pytz و numpy.
3. قم بتحميل وتنصيب Matplotlib.

استخدام pylab على Athena

إذا كنت تعمل على Athena، قم بإيجاد جهاز يعمل على **Athena Linux** (إنه **ليس Solaris** – أجهزة أرجوانية وعليها كلمة "sun"). في نافذة موجه الأوامر، تأكّد من أن الـ shell هي bash shell، يمكنك التحقق من ذلك بإدخال "echo $SHELL" والتحقق من ظهور سلسلة تحتوي على "bash " (وبالتحديد عدم ظهور csh)؛ إذا لم يظهر ذلك قم بتشغيل bash. ثم قم بتشغيل الأوامر التالية لتشغيل IDLE:

```
$ . ~6.00/arch/share/bin/setup
$ idle &
```

يجب أن إتباع هذه الخطوات في كل مرة تعمل فيها على جهاز Athena.

لمحة عن المحاكاة

"iRobot" هي شركة (تم إنشاؤها من قبل الخريجين والمدرسين في جامعة MIT) تقوم ببيع روبوت الكنس Roomba (قوموا بمشاهدة إحدى فيديوهاتنا لتروا كيف تعمل هذه الروبوتات). روبوتات Roomba تتجول على الأرضية وتقوم بتنظيف المكان الذي تمر عليه. سوف تقوم بتصميم محاكاة لتقدير الوقت الذي سوف تستغرقه مجموعة من روبوتات مثل Roomba لتنظيف أرضية غرفة ما.

النموذج المبسّط التالي لروبوت واحد يتحرك في غرفة مربعة 5×5 يجب أن يعطيك حدساً حول النظام الذي نقوم بمحاكاته.

يبدأ الروبوت في موضع عشوائي في الغرفة وباتجاه حركة عشوائي. التوضيحات التالية تُظهر موضع الروبوت (يُشار إليه بنقطة سوداء) واتجاهه أيضاً (يُشار إليه باتجاه رأس سهم أحمر).

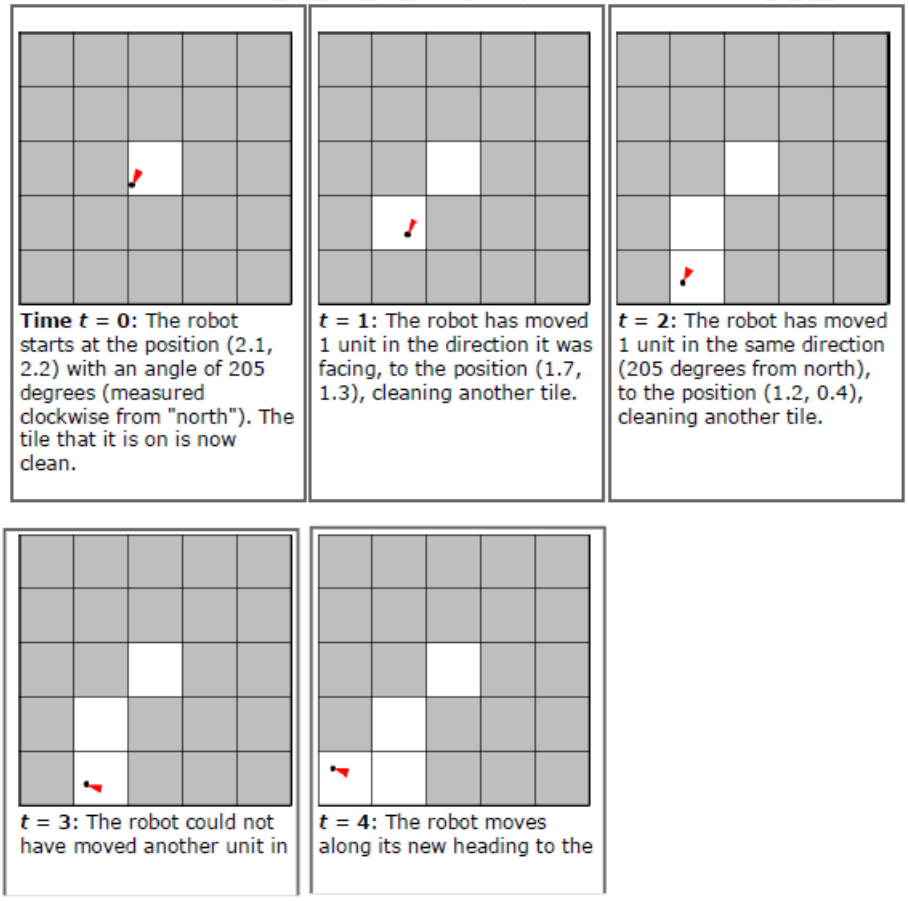
الزمن t = 0: الروبوت يبدأ في الموضع (2.1,2.2) بزاوية تساوي 205 درجة (مقاسة بجهة دوران عقارب الساعة بدءً من "الشمال"). البلاطة التي هو عليها الآن أصبحت نظيفة.

الزمن t = 1: تحرك الروبوت وحدة واحدة في الاتجاه الذي كان عليه إلى هذا الموضع (1.7,1.3) ونظّف بلاطة أخرى.

الزمن t = 2: تحرَّك الروبوت وحدة واحد بالاتجاه نفسه (205 درجة بدءً من "الشمال") إلى الموضع (1.2,0.4) ونظّف بلاطة أخرى.

الزمن t = 3: لم يستطع الروبوت التحرك وحدة أخرى الاتجاه نفسه دون أن يصطدم فقام بإدارة وجهه إلى وجهة عشوائية جديدة، 287 درجة.

الزمن t = 4: تحرَّك الروبوت وفق وجهته الجديدة إلى الموضع (0.3,0.7) ونظّف بلاطة أخرى.



Time *t* = 0: The robot starts at the position (2.1, 2.2) with an angle of 205 degrees (measured clockwise from "north"). The tile that it is on is now clean.

***t* = 1:** The robot has moved 1 unit in the direction it was facing, to the position (1.7, 1.3), cleaning another tile.

***t* = 2:** The robot has moved 1 unit in the same direction (205 degrees from north), to the position (1.2, 0.4), cleaning another tile.

***t* = 3:** The robot could not have moved another unit in

***t* = 4:** The robot moves along its new heading to the

تفاصيل المحاكاة

فيما يلي تفاصيل إضافية عن نموذج المحاكاة. قم بقراءتهم بدقّة.

- **عدة روبوتات.** بشكل عام هناك $0 < N$ روبوت في الغرفة، حيث N مُعطاة. للتبسيط، افترض أنّ الروبوتات هي نقاط وأنّه يمكن أن تمر ببعضها البعض أو تشْغَل نفس النقطة دون أن تتداخل.
- **الغرفة.** الغرفة هي مستطيل عرضه هو عدد صحيح w وارتفاع h، وكلاهما مُعطى. في البداية تكون الأرضية متسخة بالكامل. لا يمكن للروبوت عبور جدران الغرفة. ولا يمكن للروبوت أن ينتقل لنقطة خارج الغرفة.
- **قوانين حركة الروبوت:**
  - كل روبوت له موضع داخل الغرفة. سنمثل الموضع باستخدام الإحداثيات (x, y) والتي هي أعداد حقيقية تحقق الشرط $0 \leq x < w$ و $0 \leq y < h$، في برنامجنا سوف نستخدم instances من الصف *Position* لتخزين هذه الإحداثيات.
  - للروبوت اتجاه حركة. سوف نمثل الاتجاه باستخدام عدد صحيح d يحقق الشرط التالي $0 \leq d < 360$، وهو يعطينا الزاوية بالدرجات.
  - تتحرك كل الروبوتات بنفس السرعة s، وهي مُعطاة وثابتة خلال المحاكاة. في كل خطوة زمنية يتحرك الروبوت في اتجاه الحركة الخاص به بمقدار s وحدة.
  - عندما يصطدم الروبوت بجدار، يختار اتجاهاً جديداً عشوائياً. ويستمر الروبوت في ذلك الاتجاه حتى يصل إلى جدار آخر.
- **البلاطات.** يجب عليك على تتبع أجزاء الأرضية التي تم تنظيفها بواسطة الروبوتات.
  سوف نقسّم مساحة الغرفة إلى بلاطات 1×1 (سيكون هناك w * h بلاطة). عندما يكون الروبوت في أي مكان على البلاطة، سوف نعتبر أنه نظَّف البلاطة بالكامل (كما في الصور الموجودة في الأعلى). بالاتفاق سوف نشير إلى البلاطات باستخدام ثنائيات من الأعداد الصحيحة:

  (0, 0), (0, 1), ..., (0, h-1), (1, 0), (1, 1), ..., (w-1, h-1).

- **الإنهاء.** تنتهي المحاكاة عندما يتم تنظيف جزء محدد من البلاطات.

إذا وجدت أياً من توصيفات المحاكاة السابقة غامضة، فالأمر يعود لك باختيار قرار منطقي حول كيفية تصرّف البرنامج/النموذج وقم بتوثيق القرار في النص البرمجي.

القسم الأول: الصفوف RectangularRoom و BaseRobot.

يجب عليك تصميم صفَّين لتتبع الأجزاء التي تم تنظيفها من الغرفة وموضع (مكان) كل روبوت واتجاهه.

في ps11.py قمنا بتزويدكم بهياكل للصفوف التالية، والتي سوف تقوم بكتابة النص البرمجي الخاص بها في المسألة 1:

RectangularRoom

يمثل المساحة التي سيتم تنظيفها ويقوم بتتبع البلاطات التي تم تنظيفها.

BaseRobot

يخزّن الموضع واتجاه الروبوت.

لقد قمنا بتزويدكم بالنص البرمجي كاملاً للصفوف التالية:

Position

يخزّن احداثيات x وy لروبوت في غرفة ما.

قم بقراءة ps11.py بدقّة قبل أن تبدأ، لكي تفهم النص البرمجي الموجود فيه وماذا يفعل.

المسألة 1

قم بإنهاء كتابة الصفوف RectangularRoom و BaseRobot بكتابة النص البرمجي للـ methods في ps11.py **تبعاً للتوصيفات المُعطاة.**

من أجل الصف RectangularRoom، يجب عليك أن تختار الحقول التي سوف تستخدمها وكيف سيتم تنفيذ العمليات التالية:

- تهيئة الكائن.
- وضع علامة "نظيف" على البلاطة المناسبة عندما يتحرك الروبوت إلى موضع مُعطى.
- تحديد إذا ما كانت البلاطة المعطاة نظيفة.
- تحديد عدد البلاطات في الغرفة.
- تحديد عدد البلاطات التي تم تنظيفها في الغرفة.
- الحصول على موضع عشوائي في الغرفة.

- تحديد إذا ما كان الموضع المعطى في الغرفة.

من أجل الصف BaseRobot، يجب عليك أن تختار الحقول التي سوف تستخدمها وكيف سيتم تنفيذ العمليات التالية:

- تهيئة الكائن.
- الحصول/الوصول على موضع الروبوت.
- الحصول/الوصول على اتجاه الروبوت.
- إعداد/تعيين موضع الروبوت.
- إعداد/تعيين اتجاه الروبوت.

(على الرغم من أنّ هذه المسألة فيها عدة أجزاء، ولكن لا يجب أن تأخذ وقتاً طويلاً للحل ما إن تختار كيفية تمثيل البيانات. في التمثيلات المنطقية سيكون هناك داخل كل method سطر برمجي واحد.)

القسم الثاني: إنشاء واستخدام المُحاكي

المسألة 2

كل روبوت يجب أن يكون له نص برمجي يخبره بكيفية التحرك في الغرفة، والذي سيكون موجوداً في method تُدعى updatePositionAndClean.

المألوف أن نأخذ بعين الاعتبار كتابة الـ methods الخاصة بالروبوت في صفٍّ واحد، لاحقاً في هذا الواجب المنزلي سوف نأخذ بعين الاعتبار كتابة النص البرمجي لروبوتات لهم استراتيجيات حركة بديلة حيث تكون صفوفها مختلفة ولكن لها نفس الواجهة. سيكون للدالّة updatePositionAndClean لهذه الصفوف نصٌّ برمجي مختلف ولكن بقية الأجزاء تبقى كما هي. فسنحتاج لاستخدام الوراثة لتقليص كمية النص البرمجي المكرّر.

لقد قمنا مسبقاً بكتابة النص البرمجي للروبوت في صفَّين: الصف BaseRobot الذي قمتَ بكتابة النص البرمجي له سابقاً (والذي يحتوي على النص البرمجي الأساسي للروبوت)، وصف للروبوت يَرِثُ منه (والذي يحتوي على استراتيجية تحرُّك خاصة به).

اكتب النص البرمجي لـ method الصف Robot التي اسمها updatePositionAndClean لمحاكاة حركة الروبوت بعد خطوة زمنية واحدة (كما هو موضَّح في الأعلى في حركيَّة المحاكاة).

```
class Robot(BaseRobot):
    """
    الـ Robot هو BaseRobot باستراتيجية حركة قياسية.
    في كل خطوة زمنيَّة يحاول الروبوت التحرك باتجاهه الحالي؛ وعندما يصطدم بجدار يقوم باختيار اتجاه جديد عشوائي.
    """
    def updatePositionAndClean(self):
        """
        تُحاكي المرور لخطوة زمنية واحدة.
        تُحرّك الروبوت إلى موضع آخر وتقوم بوضع علامة "نظيف" على البلاطة الذي هو عليها.
        """
```

تلميح: يمكن أن تجد الدالّة getNewPosition في الصف Position مفيدة.

المسألة 3

في هذه المسألة ستكتب برنامجاً يقوم بتنفيذ محاكاة كاملة للروبوت.

في كل تجربة، الهدف هو جمع بيانات حول عدد الخطوات الزمنية التي نحتاجها لتنظيف جزء محدد من الغرفة. قم بكتابة النص البرمجي للدالة التالية:

```
def runSimulation(num_robots, speed, width, height, min_coverage, num_trials, robot_type, visualize):
    """
    تقوم بتنفيذ NUM_TRIALS تجربة للمحاكاة وتعيد قائمة قوائم، قائمة لكل تجربة. وفي كل واحد منها عنصر لكل خطوة زمنية لتلك التجربة، وقيمة للنسبة المئوية للجزء النظيف من الغرفة لتلك الخطوة الزمنية. وكل تجربة تتوقف عندما يكون MIN_COVERAGE للغرفة نظيف.
    يتم تشغيل المحاكاة مع روبوتات عددها NUM_ROBOTS من النمط ROBOT_TYPE ولكل واحد سرعة تساوي SPEED في غرفة أبعادها هي WIDTH x HEIGHT.
    تظهر الصورة عندما يكون قيمة المعامل البولياني VISUALIZE هي True.
    """
```

أول ستة معاملات تعرّف عن نفسها، في الوقت الحالي يجب عليك تمرير *Robot* للمعامل robot_type كالتالي:

avg = runSimulation(10, 1.0, 15, 20, 0.8, 30, Robot, False)

ثم في runSimulation يجب عليك استخدام robot_type(...) بدلاً من Robot(...) في كل مرة تريد فيها إنشاء روبوت. (هذا سيمكننا أن نعدّل المحاكاة بسهولة لتعمل مع روبوتات مختلفة بالنص البرمجي، والتي سوف تصادفها في المسألة 5).

في الوقت الحالي قم بتهيئة المعامل visualize بالقيمة False؛ يمكنك تغييرها إلى True عندما تضيف النص البرمجي للإظهار المرئي في القسم الأخير من هذه المسألة.

يمكنك كتابة الدوال التي تريدها لمساعدتك، ولكن الملف الذي سترسله يجب أن لا يحتوي على نص برمجي خارج هذه الدوال المساعدة والصفوف والدوال المحددة.

لمعلوماتك، هذه أوقات تقريبية لتنظيف الغرفة. هذه الأوقات لروبوت سرعته تساوي 1.0

- روبوت واحد يأخذ وقتاً يساوي 150 تكة لتنظيف غرفة مساحتها 5×5 بالكامل.
- روبوت واحد يأخذ وقتاً يساوي 190 تكة لتنظيف 75% من غرفة مساحتها 10×10.
- روبوت واحد يأخذ وقتاً يساوي 310 تكة لتنظيف 90% من غرفة مساحتها 10×10.
- روبوت واحد يأخذ وقتاً يساوي 3250 تكة لتنظيف غرفة مساحتها 20×20 بالكامل.

(هذه مجرد إرشادات. اعتماداً على التفاصيل الدقيقة للنص البرمجي ربما تحصل على أوقات مختلفة عن الأوقات السابقة).

يجب عليك التحقق من خرج المحاكاة من أجل سرعات غير السرعة 1.0. أضف السطر التالي في بداية الملف ps11.py:

```
import ps11_visualize
```

هذه هي طريقة تشغيل الإظهار المرئي (visualization):

1. في محاكاتك، في بداية التجربة قم بما يلي لتشغيل الرس المتحرك (animation):

```
anim = ps11_visualize.RobotVisualization(num_robots, width, height)
```

2. ثم في كل خطوة زمنية، قم بما يلي لرسم شكل جديد للرسم المتحرك:

```
anim.update(room, robots)
```

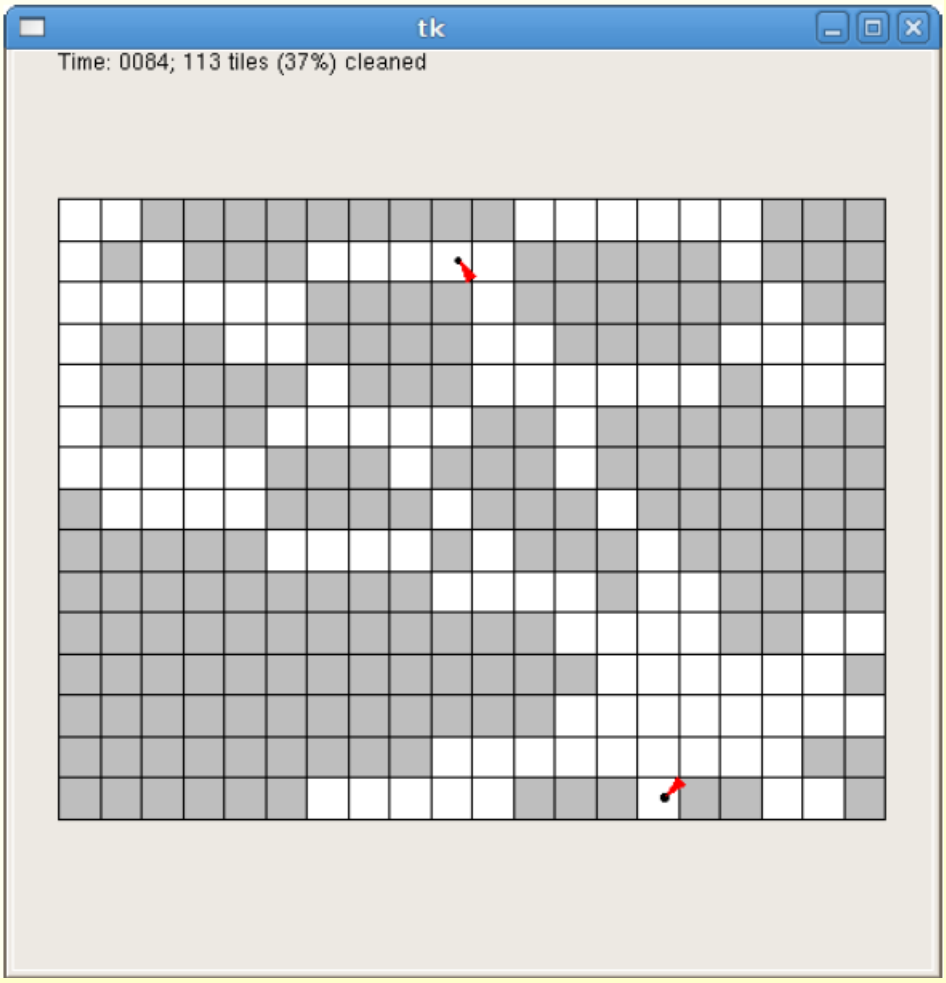
قم بتمرير كائن من النمط RectangularRoom يمثل الحالة الحالية للغرفة وقائمة من الروبوتات الموجودة في الغرفة.

3. عندما تنتهي التجربة، قم باستدعاء الـ method

```
anim.done()
```

لاحظ أنّه عند هذه النقطة، سينتظر البرنامج أن يقوم المستخدم بإغلاق نافذة الرسم المتحرك قبل أن يتابع إلى التجربة التالية.

الرسم المتحرك الناتج سيبدو كالتالي:



الزمن: 0084، تم تنظيف 113 (37%) بلاطة.

النص البرمجي الذي يقوم بالإظهار المرئي سوف يجعل المحاكاة أبطأ لكيلا يتم عرض الرسم المتحرك بسرعة كبيرة (بشكل افتراضي سوف يعرض 5 خطوات زمنية في الثانية). بالطبع سترغب في تجنب تشغيل النص البرمجي للرسم المتحرك إذا كنت تحاول تنفيذ عدة تجارب بنفس الوقت (على سبيل المثال، عندما تقوم بتنفيذ محاكاة كاملة).

من أجل تنقيح برنامجك من الأخطاء، يمكنك جعل المحاكاة أبطأ من السابق. يمكنك القيام بذلك بتغيير استدعاء RobotVisualization كالتالي:

```
anim = ps11_visualize.RobotVisualization(num_robots, width, height, delay)
```

المعامل delay يحدد عدد الثواني التي يجب أن يتوقف البرنامج فيها بين الصور (frames). بشكل افتراضي هو 0.2 (أي 5 صور بالثانية). يمكنك زيادة هذه النسبة لجعل الرسم المتحرك أبطأ.
لا تنسى تهيئة المعامل visualize لـ runSimulation بالقيمة True لرؤية الرسم المتحرك.

المسألة 4

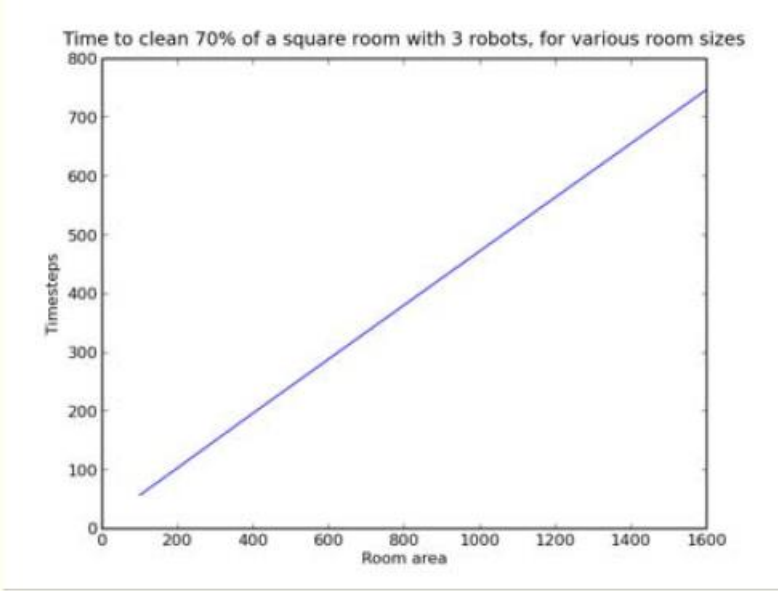
والآن استخدم المحاكاة للإجابة على بعض الأسئلة حول أداء الروبوت.
من أجل الأسئلة التالية اكتب نصاً برمجياً يقوم بتوليد مخطط باستخدام pylab. قم بوضع النص البرمجي في الدوال الهيكلية التالية والموجودة في ps11.py
(showPlot1, showPlot2, showPlot3, and showPlot4, respectively)
لكل مخطط يجب أن يكون هناك عنوان وترقيمات توصيفية على كلا المحورين، ومفتاح (إن أمكن).
افترض أن الروبوتات تتحرك بسرعة 0.1.

1. ما هو الوقت الذي يأخذه روبوت واحد لتنظيف 75% من كل غرفة من الغرف التالية:
   5×5، 10×10، 15×15، 20×20، 25×25؟ أعطي خرجاً هو شكل فيه الرسم البياني لمتوسط الوقت (على المحور y) مقابل مساحة الغرفة.
2. ما هو الوقت المُستغرق لتنظيف 75% من غرفة 25×25 باستخدام كل من 1-10 روبوت؟
   أعطي خرجاً هو شكل فيه الرسم البياني لمتوسط الوقت (على المحور y) مقابل عدد الروبوتات.
3. ما هو الوقت الذي يأخذه روبوتين لتنظيف 75% كل غرفة من الغرف التالية: 20×20، 16×25، 10×40، 8×50، 5×80؟ (لاحظ أن الغرف لها نفس المساحة).
   أعطي خرجاً هو شكل فيه الرسم البياني لمتوسط الوقت (على المحور y) مقابل نسبة العرض إلى الارتفاع.
4. كيف يتغير الوقت المُستغرق لتنظيف غرفة 25×25 عند تغيُّر min_coverage؟
   أعطي خرجاً هو شكل فيه الرسم البياني لمتوسط الوقت (على المحور y) مقابل النسبة للمساحة النظيفة من الغرفة، لكل من 1-5 روبوت. سيكون هناك عدة منحنيات في الرسم البياني.

قم بعدد من التجارب. من أجل المخططات استخدم عدداً كبيراً بما فيه الكفاية من التجارب حتى تعتقد أن الخرج موثوق.

تلميح: من أجل المخططات 1-3 ، يمكن أن يكون من المفيد أن تكتب دالّة مساعدة تقوم بحساب متوسط الطول لكل القوائم في قائمة القوائم المُعادة بواسطة runSimulation. من أجل المخطط 4، يمكن أن تجد الدالّة computeMeans التي قمنا بتزويدك بها مفيدة.

الشكل التالي هو مثال على مخطط جيد:



المسألة 5

تقوم iRobot باختبار تصميم روبوت جديد. الروبوت الجديد المعروض يختلف بأنه يقوم بتغيير الاتجاه بشكل عشوائي **بعد كل خطوة زمنية**، بدلاً من تغيير الاتجاه فقط عند الاصطدام بالجدران. لقد طُلب منك أن تقوم بتصميم محاكاة لتحديد التأثير –إذا كان هناك تأثير- الذي يقوم به هذا التغيير على أوقات تنظيف الغرفة.

اكتب صفاً جديداً RandomWalkRobot **يرث من** BaseRobot (**مثل** Robot) **ولكن قم بكتابة النص البرمجي لاستراتيجية جديدة.** يجب أن يكون لـ RandomWalkRobot نفس واجهة Robot.

قم باختبار الصف الجديد. وقم بتنفيذ تجربة واحدة باستخدام الدالّة RandomWalkRobot الجديدة وقم بمشاهدة الإظهار المرئي للتأكد من أنها تقوم بالأمر بشكل صحيح. عندما تقتنع بها يمكنك استدعاء runSimulation مرةً أخرى بتمرير RandomWalkRobot بدلاً من Robot.

المسألة 6

قم بتوليد مخطط مناسب (من تصميمك) يقارن أداء نوعي الروبوتات. قم بإضافة النص البرمجي إلى showPlot5(). كما هو الحال دائماً، يجب أن يحتوي مخططك على عنوان مناسب وترقيمات للمحاور ومفتاح (إن أمكن).
من ضمن التعليقات في showPlot5، قم بالتعليق بشكل مختصر على كيفية مقارنة نوعي الروبوت.

عملية التسليم:

1. **الحفظ**

قم بحفظ النص البرمجي المعدّل في ملف واحد اسمه ps11.py. تذكر أنه لا يجب أن يكون هناك أي نص برمجي خارج هذه الصفوف والدوال المحددة، إلى جانب الدوال المُساعِدة التي قد تكتبها.

2. **الاختبار**

قم بتشغيل ملفّك لتتأكد أنه لا يحتوي على أخطاء قواعدية. قم بتشغيل الدوال التي تقوم برسم المخططات لتتأكد من أنها تقوم بإعطاء مخططات عندما يتم تشغيلها. قم باختبار runSimulation للتأكد من أنها ما زالت تعمل مع الصفَّين Robot و RandomWalkRobot. (من الطبيعي تخريب النص البرمجي دون قصد أثناء تعديل المعاملات، ولذلك فالاختبار هو أمر ضروري).

3. **التوقيت ومعلومات عن التعاون**

في بداية كل ملف، قم بكتابة تعليق واذكر فيه عدد الساعات -بشكل تقريبي- التي أمضيتها على المسائل في هذا الجزء، واذكر أيضاً أسماء الأشخاص الذين تعاونت معهم. على سبيل المثال:

```
# Problem Set 11
# Name: Jane Lee
# Collaborators: John Doe
# Time: 3:30
#
... اكتب النص البرمجي هنا ...
```