الواجب المنزلي 4

مقدمة

هذا الواجب المنزلي سيعرّفك على استخدام التقريب المتعاقب وأيضاً سيعرفك على بنى البيانات مثل tuple والقوائم.

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.

عملية التسليم:

سيتم التقييم في هذا الواجب والواجبات اللاحقة بواسطة مهمات اختبارية. برنامج المهمات الاختبارية سيتوقع أن يكون هناك داخل ملفاتك فقط تعريف دوال من دون نص برمجي قابل للتنفيذ خارج تعريف الدوال. **كامل النص البرمجي للاختبار يجب أن يكون داخل دالّة اختبار مناسبة**، والموجودة داخل النموذج؛ على سبيل المثال، النص البرمجي للاختبار لمسألة futureRetire() يجب أن تكون في testFutureRetire() . تأكد من إضافة حالات اختبار لدوال الاختبار هذه بعد كتابة ما هو موجود في النموذج.

التخطيط للمستقبل

الأحداث الأخيرة في سوق الأوراق المالية ربما تبدو بعيدة عنكم، ولكنها تؤكد على الارتياب في المستقبل. الأشخاص الذين كانوا يفكرون بالتقاعد في العام القادم تقريباً سيوجب عليهم ربما إعادة التفكير بهذه الخطط، بما أن قيمة حساباتهم في 410k انخفضت بشكل ملحوظ. على الرغم من أن التقاعد يبدو أنه بعد وقت طويل بالنسبة لك، سوف نقوم باستعراض أفكار بسيطة لتجميع المال. وأثناء ذلك، سنقوم باستعراض استخدام أساليب التقريب المتعاقب، وأيضاً استخدام بعض بنى البيانات البسيطة كالقوائم.

سنبدأ بنموذج بسيط للادخار من أجل التقاعد. كثير من أصحاب الشركات يساهمون في صندوق راتبك التقاعدي بما يعادل 5% من راتبك كموظف، ثم سيضيفون لحسابك دولاراً من أجل كل دولار تدخره

وبحد أقصى يعادل 5% إضافية من راتبك. وبالتالي يمكنك أن تدخر ما قد يصل إلى 15% من راتبك في حساب التقاعد الخاص بك (هناك 10% تأتي كزيادة من صاحب العمل).

يمكننا نمذجة صندوق تقاعد باستخدام بعض المعادلات البسيطة. افتراض أن راتب أول تعيين يُمثل بـ salary ؛ حيث أن النسبة المئوية من راتبك التي تضعها في صندوق التقاعد هي save؛ ونسبة النمو السنوي لصندوق التقاعد هي growthRate. عندئذ فإن صندوقك التقاعدي المُمثل بالقائمة F يجب أن يزداد كالتالي

	صندوق التقاعد
نهاية السنة الأولى	F[0] = salary * save * 0.01
نهاية السنة الثانية	F[1] = F[0] * (1 + 0.01 * growthRate) + salary * save * 0.01
نهاية السنة الثالثة	F[2] = F[1] * (1 + 0.01 * growthRate) + salary * save * 0.01

عندما تكمل هذه العملية، في كل سنة ينمو صندوق التقاعد بسبب المساهمات الجديدة وبسبب ازدياد رأس المال. خلال هذا الواجب المنزلي، سيكون معدل النمو موجب دائماً (هذا غير صحيح في العالم الحقيقي، للأسف!).

المسألة 1

قم بكتابة دالة تُدعى nestEggFixed والتي تأخذ 4 معاملات: راتب والنسبة من الراتب التي تريد ادخارها في حساب استثماري ومعدل النمو السنوي للحساب الاستثماري وعدد سنوات العمل. يجب أن تعيد هذه الدالة قائمة (list) فيها قيمة الحساب الاستثماري في نهاية العام الأول... وآخر قيمة فيها هي حجم الحساب الاستثماري في نهاية العام الحالي.

أكمل النص البرمجي التالي

```
def nestEggFixed (salary, save, growthRate, years):
```

قم بكتابة النص البرمجي في مكان مناسب في النموذج ps4.py . لاختبار الدالّة التي كتبتها قم بتنفيذ حالات الاختبار في دالة الاختبار testNestEggFixed() ، يجب عليك إضافة حالات اختبار إضافية لهذه الدالّة لتعزز اختبار البرنامج.
النموذج الأول بسيط جداً. من الواضح أن سوق الأوراق المالية لا ينمو بمعدل ثابت. فالنموذج الأفضل هو بحساب التغيرات في معدل النمو كل عام.

المسألة 2

اكتب دالّة تدعى nestEggVariable والتي تأخذ 3 معاملات: راتب (salary)، والنسبة من الراتب التي تريد ادخارها (save) وقائمة بمعدلات النمو السنوية على الاستثمارات (growthRates). طول المعامل الأخير يحدد عدد السنوات التي ينوي الموظف العمل خلالها؛ growthRates[0] هو معدل النمو للعام الأول، growthRates[1] هو معدل النمو للعام الثاني، وهكذا . (لاحظ أنّه بسبب أن القيمة المبدئية لصندوق التقاعد تساوي 0 فإن growthRates[0] هو بالحقيقة لا علاقة له). يجب أن تعيد الدالّة قائمة فيها قيم الحساب التقاعدي في نهاية كل عام.

أكمل النص البرمجي التالي:

```
def nestEggVariable(salary, save, growthRates):
```

اكتب النص البرمجي في مكان مناسب في النموذج ps4.py. لاختبار الدالّة التي قمت بكتابتها قم بتنفيذ حالات الاختبار في دالة الاختبار testNestEggVariable() يجب عليك إضافة حالات اختبار إضافية لهذه الدالّة لتعزيز اختبار برنامجك.

بالطبع، عندما تتقاعد أنت تريد أن تكون قادراً على سحب مقدار من المال كل عام لمصاريف المعيشة. يمكنك استخدام برنامجك السابق لتحصل على حجم أموال التقاعد عندما تتوقف عن العمل. والآن نريد أن نقوم بنمذجة الكمية التي يمكنك سحبها لتمضي كل عام بعد التقاعد.

افترض أنّه بعد التقاعد، استمر صندوق التقاعد الخاص بك بالنمو بناء على قائمة من معدلات النمو السنوية للاستثمار (growthRates)، بينما مصاريفك السنوية ثابتة (إذا كانت نسبة التضخم تساوي 0% ألن يكون ذلك جميل؟) وتدعى expenses . لرؤية كيف سيتغير صندوق التقاعد الخاص بك بعد التقاعد، يمكننا استخدام المخطط التالي، حيث ستكون F تمثل حجم صندوق الاستثمار في وقت ما من فترة التقاعد، وسيكون expenses يمثل مقدار المال الذي نقوم بسحبه في العام الأول من أجل مصاريف المعيشة:

	صندوق التقاعد
نهاية السنة الأولى	F[0] = savings * (1 + 0.01 * growthRates[0]) – expenses
نهاية السنة الثانية	F[1] = F[0] * (1 + 0.01 * growthRates[1]) – expenses
نهاية السنة الثالثة	F[2] = F[1] * (1 + 0.01 * growthRates[2]) – expenses

المسألة 3

اكتب دالّة تدعى postRetirement والتي تأخذ ثلاثة معاملات: قيمة مبدئية لكمية المال في صندوق التقاعد الخاص بك (savings) وقائمة من معدلات النمو السنوية خلال فترة التقاعد (growthRates) ونفقاتك السنوية (expenses).

افترض أن الزيادة في مدخرات الحساب الاستثماري يتم حسابها قبل طرح النفقات السنوية (كما هو موضح بالجدول السابق). الدالة التي قمت بكتابتها يجب أن تعيد قائمة فيها حجم المال بعد كل عام من أعوام التقاعد وستعيد أيضاً حساب للنفقات السنوية ونمو صندوق التقاعد. كما في المسألة 2، طول المعامل growthRates يحدد عدد السنوات التي تخطط أن تمضيها متقاعداً.

لاحظ أنه إذا أصبحت ميزانية صندوق التقاعد سالبة يجب أن يستمر طرح النفقات، ومعدل النمو يصبح يمثل معدل الفائدة للدين. (بعبارة أخرى، المعادلات الموجودة في الجدول السابق تبقى صحيحة).

أكمل التعريف

def postRetirement(savings, growthRates, expenses):

اكتب النص البرمجي في مكان مناسب في النموذج ps4.py. لاختبار الدالّة التي قمت بكتابتها قم بتنفيذ حالات الاختبار في دالة الاختبار testPostVariable() يجب عليك إضافة حالات اختبار إضافية لهذه الدالّة لتعزيز اختبار برنامجك.

افترض أنك تريد وضع ميزانية للنفقات السنوية بحيث لا يتبقى أي مال في صندوق التقاعد عند وفاتك (أنت تخطط لاتباع نموذج "كارنيجي" وعدم ترك أي مال لأولادك). إحدى طرق تحديد ميزانية

المصاريف هي بإدخال قيم مختلفة للدالّة ()PostRetirement. يمكنك أتمتة ذلك (جعله أتوماتيكياً) باستخدام فكرة التقريب المتعاقب المُقدمة في المحاضرة.

تذكر أنه في المحاضرة، رأينا فكرة البحث الثنائي. في ذلك المثال كنا نحاول إيجاد الجذر التربيعي لعدد ما، وقمنا بذلك باختيار قيمة كبيرة وقيمة صغيرة نعلم أن الإجابة تقع بينهما، ثم اختبار متوسط القيمة الكبيرة والقيمة الصغيرة لنرى مدى قربه للإجابة الصحيحة (ضمن اختلاف مطلق إيبسلون). وإذا كان قريباً بشكل كافٍ توقفنا؛ إذا لم يكن كذلك، قمنا بتغيير مجال القيم؛ إما بجعل القيمة الصغيرة الجديدة تساوي متوسط القيمة الصغيرة السابقة والقيمة الكبيرة السابقة، وإبقاء القيمة الكبيرة كما هي، أو بجعل القيمة الكبيرة الجديدة تساوي متوسط القيمة الصغيرة السابقة والقيمة الكبيرة السابقة، وإبقاء القيمة الصغيرة كما هي وذلك اعتماداً نتيجة الاختبار. يمكنك استخدام نفس الفكرة هنا.

المسألة 4

اكتب دالّة تدعى findMaxExpenses والتي تأخذ 5 معاملات: راتب (salary)، والنسبة من الراتب التي تريد ادخارها (save) وقائمة بمعدلات النمو السنوية على الاستثمارات أثناء سنوات العمل (preRetireGrowthRates) وقائمة بمعدلات النمو السنوية على الاستثمارات أثناء التقاعد (postRetireGrowthRates) وقيمة لـ إيبسلون (epsilon). كما في المسألة 2 والمسألة 3، فإن طول preRetireGrowthRates وطول postRetireGrowthRates يحدد عدد السنوات التي تخطط للعمل أثنائها وعدد السنوات التي تخطط إمضاءها متقاعداً بالترتيب.

استخدم فكرة البحث الثنائي لإيجاد قيمة لكمية النفقات التي يمكنك سحبها كل عام من صندوق التقاعد الخاص بك، بحيث أنه في نهاية تقاعدك، تكون القيمة المطلقة لكمية المال المتبقية في صندوق التقاعد الخاص بك أصغر من epsilon (لاحظ أنه يمكنك أن تسحب مقداراً من المال أكبر بقليل). ابدأ بمجال من القيم الممكنة للنفقات السنوية بين 0 وقيمة المدخرات عند بداية التقاعد.

(ملاحظة#1: يمكن تحديد ذلك بالاستفادة من حل المسألة 2) الدالّة التي قمت بكتابتها يجب أن تطبع التقدير الحالي لكمية النفقات في كل تكرار خلال البحث الثنائي (ملاحظة#2: يجب أن يستخدم البحث الثنائي الذي قمت ببرمجته حل المسألة 3)، ويجب أن يعيد تقديراً لكمية النفقات التي سيتم سحبها. (ملاحظة#3: الإجابة يجب أن تقع بين الصفر والقيمة الابتدائية للمدخرات + إيبسلون.)

أكمل النص البرمجي التالي

```
def findMaxExpenses(salary, save, preRetireGrowthRates, postRetireGrowthRates, epsilon):
```

اكتب النص البرمجي في مكان مناسب في النموذج ps4.py. لاختبار الدالّة التي قمت بكتابتها وقم بتنفيذ حالات الاختبار في دالة الاختبار testFindMaxExpenses(). يجب عليك إضافة حالات اختبار إضافية لهذه الدالّة لتعزيز اختبار برنامجك.

عملية التسليم:

1. **الحفظ**

قم بحفظ إجابتك في ملف وقم بتسميته باسم كل مسألة، لا تقم بتجاهل هذه الخطوة ولا تسمي الملف بأسماء أخرى.

2. **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   4 (Part I)
# Name: Jane Lee
# Collaborators: John Doe
# Time: 1:30
#
... اكتب النص البرمجي هنا ...
```