Unit 4: forループとリスト

このレッスンでは，乱数を使ってサイコロを投げるシミュレーションを行います。

Application: サイコロ投げ

目標

- **randint()**を使ったサイコロ投げのシミュレーション
- •リストを使って合計の追跡
- •合計の分析

正しく作られた2つのサイコロを投げるとき，2つのサイコロの目の数の合計は2から12までの値になります。どの合計が最も起こりやすいですか。最も起こりにくい合計はいくつですか。サイコロを投げるシミュレーションを行い，11個の合計のそれぞれの頻度を分析するプログラムを作成します。

Teacher Tip: Pythonでのシミュレーションは簡単で高速です。数千個のサイコロを一瞬でシミュレートできます。また，リストを作成するエレガントな方法があります。

```
Python Shell
>>>#Running dice.py
>>>from dice import *
>>>[0]*11
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
>>>
```

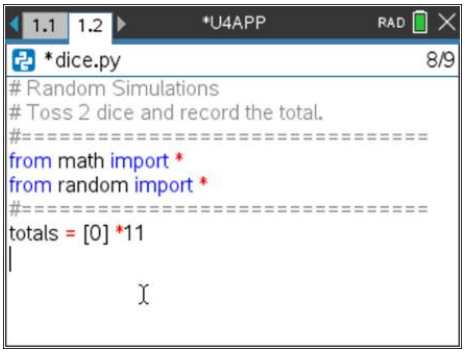
1. 前回のレッスン同様，Random Simulationsテンプレートを使って新規のPythonファイルを開始します。

   11個の合計(2…12)を追跡するには，すべて0の11個の要素のリストを使います。Pythonで簡単に作成する方法は次のとおりです。

   totals = [0] * 11

   これは，11個すべて0の要素のリストを作成することを意味します。前回のレッスンの複製を覚えていますか。

   (Pythonシェルでこの式を試して，機能することを確認してください。)



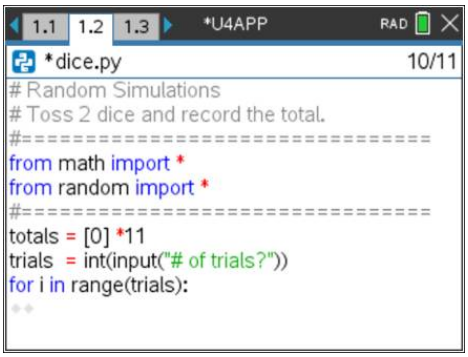
2. inputステートメントを使って，サイコロを投げる回数を尋ねます。

   trials = int(input(“# of trials?”))

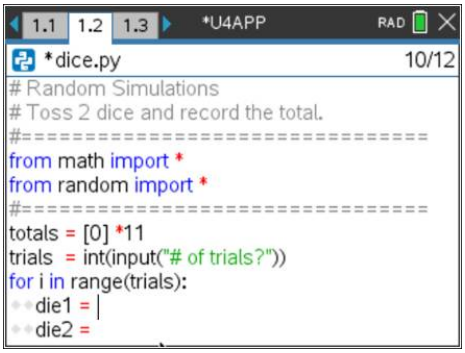
   #と?は，句読点キーにあります。

   つぎに，forループを作成して，各試行(サイコロを投げる)を実行します。

   for i in range(trials):



3. つぎに，ループブロックで，サイコロを投げます。die1とdie2の2つの変数を使います。それぞれに1から6までのランダムな整数を割り当てます。最初に自分で試してから，次のステップに進んでください。



Teacher Tip: die1 = randint(1,6)

4. それぞれに**= randint(1,6)**と書きましたか。

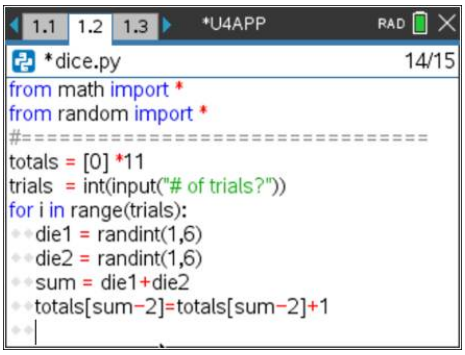
   2つのdieの値を合計します。

   sum = die1 + die2

   合計の範囲は2～12ですが，リストの合計のインデックスの範囲は0～10です。適切なリストインデックスを取得するには，合計から2を引く必要があります。合計のリストの対応する要素に1を追加するには，次のようになります。

   totals[sum - 2] = totals[sum - 2] + 1

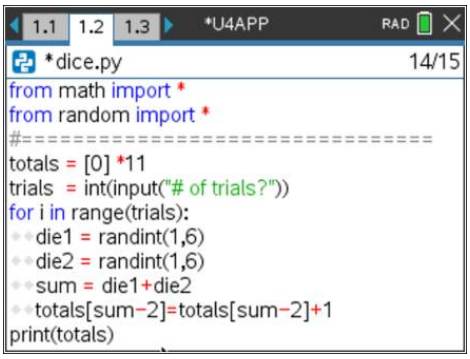
   このステートメントは，c=c+1のようなカウンタとして認識できます。リストの各要素は個別のカウンタです。



5. ループブロックは以上です。試行が完了したら，**totals**のリストを表示します。

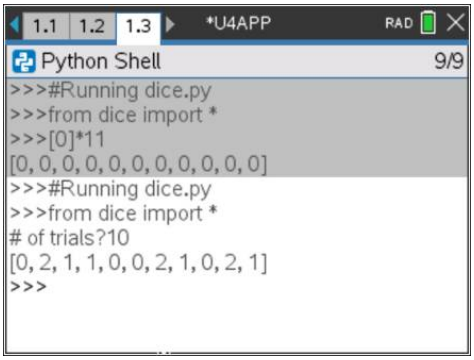
   次の行の先頭までデデント(dedent，インデント解除)して，合計リストを出力します。

   print(totals)

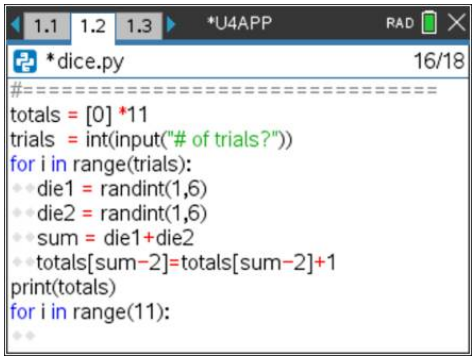


6. プログラムをテストします。プログラムを実行するときは，試行回数として10などの小さな数をまず試してください。リスト内の要素の合計が10であることを確認してください。おそらくこれらは同じ数は得られませんが(ランダムです)，数の合計は試行回数と等しくなるはずです。同じ回数の試行で再実行すると，別のリストが表示されます。

   プログラムが予想どおりに実行されることを確認したら，500，1000，5000など，大きな数の試行を試してください。合計リストのどの要素が最大ですか。最小は?



7. 課題：リスト内の各値を試行回数でわり算するforループを記述します。結果を印刷します。これらの数字の意味は何ですか。

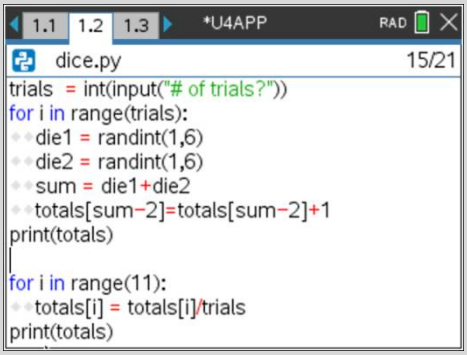


スタートとして，

for i in range(11):

長い小数をより少ない桁数に四捨五入することができます。何が適切でしょうか。
round(n, #places)を使います。

Teacher Tip: コードの残りの部分は，つぎのとおりです。



最後のリスト(小数点以下)は，各合計を2から12まで取得する実験的な確率です。
round(n, #places)関数を導入して，より少ない小数点以下の桁数に四捨五入することができます。
理論上の確率は，次のとおりです。

[1/36, 2/36, 3/36, 4/36, 5/36, 6/36, 5/36, 4/36, 3/36, 2/36, 1/36]