

Unit 4: forループとリスト

Skill Builder 3: ランダムに投げる

このレッスンでは、パターンを調べるため乱数リストを作成します。

目標

- randint(a,b)を使用
- ランダムな整数のリストを作成
- リストのプロパティを分析

はじめに：

リスト(または配列)は、1つの変数に多くの異なる値を格納する便利な方法です。Pythonはリストの内容に非常に柔軟性があり(さまざまなタイプの値を保持できますが、これはまれです)、リストを作成する方法がいくつかあります。

注目に値する式の1つは、要素を3回複製する[0] * 3です。

3 * [1,2,3]は何を生成しますか。電卓で同じ式を実行すると、右図に示されているようにまったく異なる結果が得られます。

Pythonでリストを操作するときは注意が必要です。これは、結果が慣れていない可能性があるためです。

このレッスンでは基本的に固執します。

1. このプロジェクトでは、新規のPythonプログラムを作成するときに、**Type:** のドロップダウンリストからRandom Simulations(ランダムシミュレーション)を選択します。

```
>>>[0]*3
[0, 0, 0]
>>>
```

```
3 * {1,2,3}      {3,6,9}
```

New

Name:

Type:

2. ランダムシミュレーションテンプレートは、**math**(数学)と**random**(乱数)の2つのモジュールを提供します。

Mathモジュールには、sqrt(), trig関数, exp(), log()などの関数があることを確認できます。

randomモジュールには乱数を処理する関数が含まれています。

便利な関数の1つは、aとbの値の間のランダムな整数を返すrandint(a, b)です。引数は、数値、変数、または数値を生成する式にすることができます。

```
*random1.py
# Random Simulations
#-----
from math import *
from random import *
#-----
```

3. まず、空のリストを作成します。

nums=[] (括弧の間に何もありません)

forループを使って、それぞれが0から25の範囲から選択された100個の乱数のリストを作成します(インデックス変数と範囲サイズを指定します)。

loopブロックは次のとおりです。

nums.append(randint(1,25))

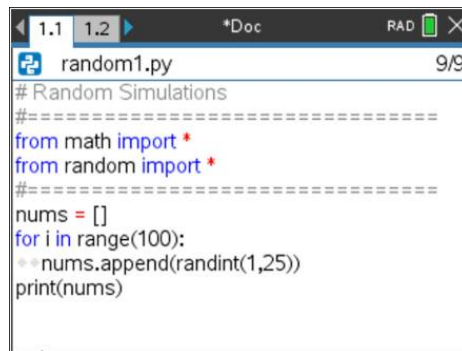
このステートメントの最後にある2つの右括弧に注意してください。一般的な構文エラーです。

```
*random1.py
# Random Simulations
#-----
from math import *
from random import *
#-----
nums = []
for index in range(size):
    *nums.append(randint(1,25))
```

4. プログラムにprintステートメントを追加して(forループの終了後)、数値が作成された後にリストを印刷します。

print(nums)

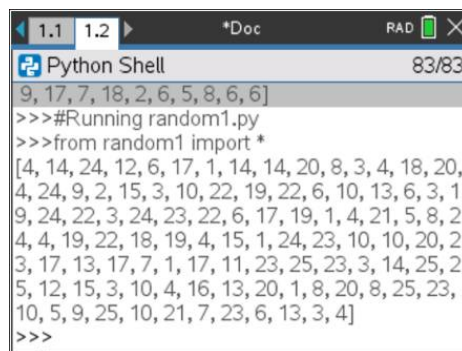
プログラムを実行して、動作することを確認してください。シェルでctrl+Rを押すと、プログラムが再実行され、実行ごとに異なる数値のセットが表示されます。



```
random1.py 9/9
# Random Simulations
#=====
from math import *
from random import *
#=====
nums = []
for i in range(100):
    nums.append(randint(1,25))
print(nums)
```

5. 右図はサンプルの実行例です。

プログラムにコードを追加して、数値の平均を決定します。最初に自分でやってみてください。(前回のレッスンを覚えておきましょう。)



```
Python Shell 83/83
[9, 17, 7, 18, 2, 6, 5, 8, 6, 6]
>>>#Running random1.py
>>>from random1 import *
[4, 14, 24, 12, 6, 17, 1, 14, 14, 20, 8, 3, 4, 18, 20,
4, 24, 9, 2, 15, 3, 10, 22, 19, 22, 6, 10, 13, 6, 3, 1
9, 24, 22, 3, 24, 23, 22, 6, 17, 19, 1, 4, 21, 5, 8, 2
4, 4, 19, 22, 18, 19, 4, 15, 1, 24, 23, 10, 10, 20, 2
3, 17, 13, 17, 7, 1, 17, 11, 23, 25, 23, 3, 14, 25, 2
5, 12, 15, 3, 10, 4, 16, 13, 20, 1, 8, 20, 8, 25, 23,
10, 5, 9, 25, 10, 21, 7, 23, 6, 13, 3, 4]
>>>
```

リストに最小値と最大値を表示します(ヒント: menu > Built-ins > Listsを参照してください)。

要素を並べ替えることができますか。

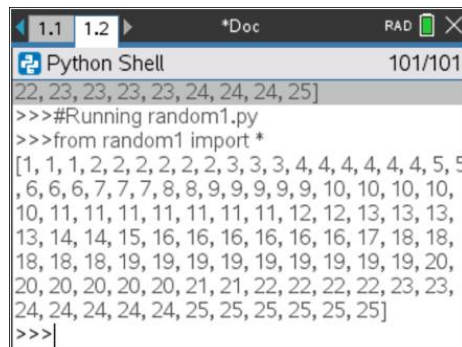
6. 並べ替えについて

Pythonには、リストを並べ替える2つのツールがあります。

nums.sort()は、要素を昇順(値が小さい順)で並べ替えます。

sorted(n)は、ソートされているが元のリストを変更しないリストを返します。

nums2 = sorted(n)を使って、元のリストを保持し、**nums2**という名前の新しいソート済みリストを作成します。



```
Python Shell 101/101
[22, 23, 23, 23, 23, 24, 24, 24, 25]
>>>#Running random1.py
>>>from random1 import *
[1, 1, 1, 2, 2, 2, 2, 2, 2, 2, 3, 3, 3, 4, 4, 4, 4, 4, 5, 5,
6, 6, 6, 6, 7, 7, 7, 8, 8, 8, 8, 8, 8, 8, 8, 9, 9, 9, 9, 10, 10, 10, 10,
10, 11, 11, 11, 11, 11, 11, 11, 12, 12, 13, 13, 13,
13, 14, 14, 15, 15, 16, 16, 16, 16, 16, 16, 17, 18, 18,
18, 18, 18, 19, 19, 19, 19, 19, 19, 19, 19, 19, 20,
20, 20, 20, 20, 21, 21, 22, 22, 22, 22, 23, 23,
24, 24, 24, 24, 25, 25, 25, 25, 25, 25]
>>>
```

選択演習:

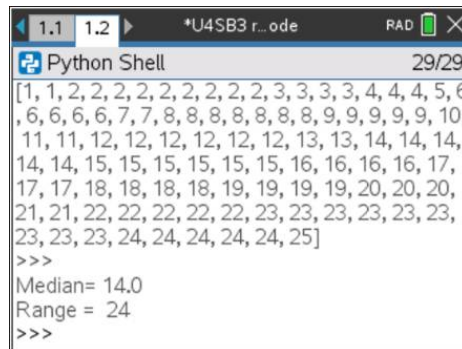
7. リストを並べ替えると、データセットの**median**(中央値)と**range**(最小値と最大値の差)を決定できます。

これらの値を表示するよう、プログラムにステートメントを追加してください。

Note: リストの要素の数が偶数の場合、中央値は2つの中央の数値の平均です。

リストの要素の数は、**menu > Built-ins > Lists**にある**len(<list>)**関数を使って得られます。**len**はlength(長さ)の省略形で、リストの場合、要素の数を返します。

Q1(第1四分位)とQ3(第3四分位)の値も決定できますか。



```
*U4SB3 r...ode 29/29
Python Shell
[1, 1, 2, 2, 2, 2, 2, 2, 2, 2, 3, 3, 3, 4, 4, 4, 4, 5, 6,
6, 6, 6, 6, 7, 7, 7, 8, 8, 8, 8, 8, 8, 8, 8, 9, 9, 9, 9, 10,
11, 11, 12, 12, 12, 12, 12, 12, 13, 13, 14, 14, 14,
14, 14, 15, 15, 15, 15, 15, 15, 16, 16, 16, 16, 17,
17, 17, 18, 18, 18, 18, 19, 19, 19, 19, 20, 20, 20,
21, 21, 22, 22, 22, 22, 22, 23, 23, 23, 23, 23, 23,
23, 23, 24, 24, 24, 24, 25]
>>>
Median= 14.0
Range = 24
>>>
```