COMP20270
OOP in Python

Assignment 1

Random

Due Wednesday 17th November (Wk10)

Objective

Python provides a Random class that provides a number of methods that require randomness. The objective for this assignment is to develop your own implementation of some of these methods. You should implement a **MyRandom** class that has **randint, rand, shuffle** and **choice** methods. It should also have a **seed** method that can set the random number generator seed.

You should also implement two subclasses of **MyRandom, MyDie** and **MyCoin,** that provide coin toss and dice functionality.

Properly speaking, random number generators implemented in software are *pseudo* random number generators (PRNGs). Some algorithm variants are described here: https://en.wikipedia.org/wiki/Linear_congruential_generator.

Requirements

1. The main work will be done by a 'private' method **_randint** that will produce a different random integer everytime it is called. This method should implement one of the PRNG algorithms linked above.
2. Most PRNGs have a seed that determines the first of the sequence of random numbers produced and thus the overall sequence can be reproduced. This seed should be an optional parameter on the class constructor (see examples below). If no seed is specified the random number sequence can be seeded from the system clock, e.g. `seed = int(time.time())`.
3. You should provide a **seed(new_seed)** method that will allow the seed to be reset.
4. **rand()** will return a random float between 0 and 1. It will call **_randint.**

5. **randint(bot, top)** will take two integers as argument and return a random integer between these two integers. It should check that the arguments are integers and that **top** is not less than **bot.**
6. **shuffle(list)** will take a list as argument and return a shuffled version of that list. It should do some basic checking on the argument it receives to confirm it is valid for shuffling. A basic shuffle algorithm is described here: https://en.wikipedia.org/wiki/Fisher–Yates_shuffle
7. **choice(list)** selects a single item at random from a list. It should perform some basic error checking.
8. Create two subclasses of **MyRandom (MyDie** and **MyCoin)** that implement dice and coin toss functionality. These will have methods **throw** and **toss** respectively. Instances of these methods are shown in operation in the examples below.

Error Handling:

1. Any data passed to the methods should be subjected to some error checking.

How to proceed:

1. Start by creating a basic version of the **MyRandom**, implementing the **_randint** function initially.
2. Add the other methods in the order you choose. It makes sense to implement **randint** before the other two.
3. Implementing **MyDie** and **MyCoin** should be straightforward once **MyRandom** is completed.
4. Alternatively you could start with **MyDie** and **MyCoin** but as subclasses of the **Random** class in the built-in **random** module.

Submission: This is an individual (not group) project. Submission is through the Brightspace page. Your submission should comprise your notebook only. Clear all outputs in the notebook before saving for submission. You can use markdown cells in the notebook to explain any design decisions you have made.

Appendix:

MyRandom

```
In [264]:
mr1 = MyRandom()
mr1.rand()
Out[309]:
0.06861291054382707

In [265]:
mr1 = MyRandom()
mr1.rand()
Out[265]:
0.367653160923787

In [269]:
mr2 = MyRandom(seed = 11)
mr2.rand()
Out[269]:
0.5900113518772263

In [270]:
mr2 = MyRandom(seed = 11)
mr2.rand()
Out[270]:
0.5900113518772263

In [271]:
for i in range(5):
    print(mr2.rand())

0.8967077145035327
0.8538299987487172
0.376814759031705
0.4812552523785329
0.4701764170753025

In [282]:
for i in range(5):
    print(mr2.randint(9,15))

12
11
13
9
10

In [284]:
mr1.shuffle(['Ace','King','Queen','Jack'])
Out[284]:
['Queen', 'Jack', 'Ace', 'King']
In [297]:
mr1.shuffle('Ace')
The list for shuffling must be a list

In [285]:
mr1.choice(['Ace','King','Queen','Jack'])
Out[285]:
'Ace'

In [286]:
mr1.seed(13)
mr1.choice(['Ace','King','Queen','Jack'])
Out[286]:
'Queen'

In [287]:
mr1.seed(13)
mr1.choice(['Ace','King','Queen','Jack'])
Out[287]:
'Queen'
```

MyCoin and MyDie subclasses

```
In [288]:
mc1 = MyCoin()

In [289]:
mc1.seed(42)
mc1.toss()
Out[289]:
'Tails'
In [290]:
md1 = MyDie()

In [296]:
for i in range(5):
    print(md1.throw(), mc1.toss())

5 Tails
4 Heads
3 Heads
3 Tails
2 Heads
```