Introduction to Programming in Python

Randomness

Dr. Bill Young
Department of Computer Science
University of Texas at Austin

Last updated: June 4, 2021 at 11:05

Random Module

Remember that one of Python's libraries is the `random` module. Several useful functions are defined there.

`randint(a,b)` : generate a random integer between a and b, inclusively.

`randrange(a, b)` : generate a random integer between a and b-1, inclusively.

`random()` : generate a float in the range $(0 \ldots 1)$.

Random Numbers: Examples

```
>>> import random
>>> random.randint(0, 9)       # same as randrange(0, 10)
8
>>> random.randint(0, 9)
3
>>> random.randrange(0, 10)    # same as randint(0, 9)
2
>>> random.randrange(0, 10)
0
>>> random.randrange(0, 10)
3
>>> random.random()
0.689013943338249
>>> random.random()
0.5466061134029843
```

It's often useful to generate random values to test your programs or to perform scientific experiments.

Aside on Import

There are several different ways to use `import`.

```
>>> import random              # imports module, not names
>>> random()
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: 'module' object is not callable
>>> random.random()
0.46714522525882873
>>> from random import random  # import name random
>>> random()
0.9893720304830842
>>> randint(0, 9)              # but not randint
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
NameError: name 'randint' is not defined
>>> from random import *       # import all names
>>> randint(0, 9)
5
```

Rolling Dice

Suppose we have a 6-sided dice. If it's a fair dice it should land equally often on any of the six sides.

So the probabilities *should be*:

roll	odds
1	1/6
2	1/6
3	1/6
4	1/6
5	1/6
6	1/6

Let's see if our random function can act like a fair dice.

Randomness Exercise

Exercise: Generate a series of random integers between 1 and 6 and count how often each appears. See how close each approaches to 1/6 of the total. How do we do this?

Randomness Exercise

Exercise: Generate a series of random integers between 1 and 6 and count how often each appears. See how close each approaches to 1/6 of the total. How do we do this?

In file `rollDice.py`:

```
# Roll a 6-sided dice and see how fair it is.

from random import randint
NUMROLLS = 1000000

def rollDice( ):
    # Create an list of 6 zeros:
    counts = [0] * 6
    for i in range( NUMROLLS ):
        # Record each roll in counts list:
        roll = randint(1, 6)
        counts[roll-1] += 1
    return counts
```


Randomness Exercise

```
def printResults( counts ):
    # Compute and print expected probability:
    oneSixth = 1/6
    print("1/6 =", round(oneSixth, 6))
    # Print header line:
    print( "Roll\tCount\tProbability\tDifference" )
    # See how close the actual values were to the
    # expected probability of 1/6.
    for i in range( 1, 7 ):
        count = counts[i-1]
        prob  = count / NUMROLLS
        diff  = ((prob - oneSixth) / oneSixth ) * 100.0
        print(i, count, prob, round(diff, 2), sep="\t")

def main():
    counts = rollDice()
    printResults( counts )

main()
```

Rolling Dice

Here's the results of one run of the program. Would you expect the results to always look like this? What would you expect to happen if you increase or decrease NUMROLLS?

```
> python rollDice.py
1/6 = 0.166667
Roll    Count   Probability     Difference
1       166963  0.166963        0.18
2       166958  0.166958        0.17
3       166628  0.166628        -0.02
4       166271  0.166271        -0.24
5       166473  0.166473        -0.12
6       166707  0.166707        0.02
```


Exercise: Flipping a Coin

Exercise: How many times do you have to flip a coin to get k heads in a row? How could you write a program to try that?

```
from randomnessTest import *
>>> flipCoin( 5 )
Found sequence of 5 heads at 45 flips!
>>> flipCoin( 2 )
Found sequence of 2 heads at 15 flips!
>>> flipCoin( 10 )
Found sequence of 10 heads at 798 flips!
>>> flipCoin( 15 )
Found sequence of 15 heads at 117725 flips!
>>> flipCoin( 20 )
Found sequence of 20 heads at 28782 flips!
>>> flipCoin( 20 )
Found sequence of 20 heads at 3677435 flips!
>>> flipCoin( 1 )
Found sequence of 1 heads at 4 flips!
>>> flipCoin( 25 )
Found sequence of 25 heads at 62705525 flips!
```


Exercise: Flipping a Coin

In file randomnessTest.py:

```
# Flip a coin until you get k heads in a row and return
# the number of flips required.
TAIL, HEAD = 0, 1

def flipCoin( k ):
    """ Flip a coin until k consecutive heads appear.  Print
        number of flips required? """
    flipCount = 0
    consecutiveHeads = 0
    while True:
        flip = randint(0, 1)
        flipCount += 1
        if flip == HEAD:
            consecutiveHeads += 1
            if consecutiveHeads == k:
                print( "Found sequence of", k, "heads at",
                    flipCount, "flips!")
                return
        else:
            consecutiveHeads = 0
```


Exercise: Guessing Game

Suppose we want to implement a guessing game with the user. Generate a random integer between 1 and 99. Accept guesses from the user until he or she guesses the number or exceeds the allowed number of guesses. It should be possible to guess the number in no more than 7 guesses. Why?

The program should behave as follows:

1. If the user guesses the number, congratulate him or her and stop play.
2. If an illegal guess is entered, say so, but count it as one guess.
3. If the user exceeds 7 guesses, say so and stop play.
4. If the guess is too low/high, say so and allow another guess.

Exercise: Guessing Game

```
> python GuessingGame.py
Enter an integer from 1 to 99: 135
That's an illegal guess.  Try again!
Enter an integer from 1 to 99: 50
Guess is too low. Try again!
Enter an integer from 1 to 99: 75
Guess is too low. Try again!
Enter an integer from 1 to 99: 87
Guess is too high. Try again!
Enter an integer from 1 to 99: 81
Guess is too low. Try again!
Enter an integer from 1 to 99: 84
Guess is too low. Try again!
Enter an integer from 1 to 99: 86
Guess is too high. Try again!
Whoops! You took too many guesses. The answer was 85
```


Exercise: Guessing Game

```
>python GuessingGame.py
Enter an integer from 1 to 99: 50
Guess is too high. Try again!
Enter an integer from 1 to 99: 25
Guess is too high. Try again!
Enter an integer from 1 to 99: 12
Guess is too high. Try again!
Enter an integer from 1 to 99: 6
Guess is too high. Try again!
Enter an integer from 1 to 99: 3
Guess is too high. Try again!
Enter an integer from 1 to 99: 2
Guess is too high. Try again!
Enter an integer from 1 to 99: 1
Congratulations!  You took 7 guesses!
```


Guessing Game Solution

```
import random
GUESSESALLOWED = 7

def main():
    n = random.randint(1, 99)
    guesses = 0
    while guesses < GUESSESALLOWED:
        guess = int( input("Enter an integer 1 to 99: "))
        if guess < 1 or guess > 99:
            print( "That's an illegal guess.  Try again!" )
        elif guess == n:
            print( "Congratulations!  You took", \
                   guesses + 1, "guesses!" )
            return
        elif guess < n:
            print( "Guess is too low. Try again!" )
        else:
            print( "Guess is too high. Try again!" )
        guesses += 1
    print( "Whoops! You took too many guesses.", \
           "The answer was", n )

main()
```