Random numbers

- What does it mean for a number to be random?
  - A number that is drawn from a set of possible values, each of which is equally probable. More precisely, the outcome is non-predictable.
- Many practical uses:
  - Computer games
  - Simulation of natural phenomena
  - Cryptography
  - Art, Music
  - Etc… any problem where you want to produce an *unpredictable* result!

The `random` module

- Thankfully, you don't have to implement your own RNG

```
import random
```

```
from random import *
```

Some random functions…

```
choice(my_list)
```
chooses 1 element from the sequence my_list

```
choice(['Harris', 'McDonnell', 'Larison'])
```

How would you get a random int from 0 to 9 inclusive?

```
randint(low,hi)
```
chooses a random `int` from low to hi, inclusive

Throw a coin n times …

```
from random import *
def flip_coins(n = 20):
    mapping = ['H', 'T']
    flip_list = [ mapping[randint(0,1)] for flip in range(n) ]
    print(''.join(flip_list))
```

This will do the same…

```
from random import *
def flip_coin2(n = 20):
    flip_list = [ choice(['H', 'T']) for flip in range(n) ]
    print(''.join(flip_list))
```

The above two versions of the program prints the list, how to return the list instead of printing?

Example: random, chr(), ord()

- Given a string, write a function that converts all letters to upper case, keeping others as they are. Note: without using upper(), i.e., write your own upper().
- Example: “abc123” → “ABC123”, “hello” → “HELLO”, “123 456” → “123 456”
- Idea:
  - First check to see if the parameter is a letter ‘a’ – ‘z’. If it is not, return as it is;
  - If it is a lower case letter, compute the distance between this letter and ‘a’, return the letter by adding the distance to ‘A’.
  - Do it for every symbol in the string.

upper_letter()

```
def upper_letter(c):
    """Convert a symbol to upper case if 'a'-'z',
       keep as it is otherwise.
    """

    if c >= 'a' and c <= 'z':  # convert
        dist = ord(c) - ord('a')
        return chr(ord('A') + dist)
    else:
        return c
```

to_upper() and its use

```
def to_upper(s):
    """Convert each letter to upper case if 'a' - 'z',
       keep as it is otherwise. We use map() here.
    """
    return ''.join(list(map(upper_letter, s)))
```

```
Python 3.6.8 |Anaconda custom (64-bi
[GCC 7.3.0] on linux
Type "help", "copyright", "credits"
>>>
 RESTART: /nfs/unixspace/linux/accou
/lectures/11_random/chr_ord_rand.py
>>> to_upper('abc123')
'ABC123'
>>> to_upper('hello')
'HELLO'
>>> to_upper('123 456')
'123 456'
>>>
```

A bit twist

- The same as above, converting to upper case, however, now we want to map each lower case letter to a random upper case letter.
- Example: ‘abc123’ → ‘SGC123’, ‘hello’ → ‘SGEBU’, ‘123 456’ → ‘123 456’
- Idea: everything is the same as the previous case (to_upper()) except when generating the upper case letter, we use a random distance, instead of a fixed one [ord(c) – ord(‘a’)]

Code and execution

```
from random import *

def random_shift(s):
    """Random shift the letters to some upper case if 'a'-'z'
       keep as it is otherwise.
    """
    return ''.join(list(map(rand_upper_letter, s)))

def rand_upper_letter(c):
    """Convert a symbol to upper case if 'a'-'z' in random,
       keep as it is otherwise.
    """

    if c >= 'a' and c <= 'z':  # convert
        dist = randint(0, 25)
        return chr(ord('A') + dist)
    else:
        return c
```

```
Python 3.6.8 |Anaconda custom (64-bit)| (
[GCC 7.3.0] on linux
Type "help", "copyright", "credits" or "li
>>>
 RESTART: /nfs/unixspace/linux/accounts/CC
/lectures/11_random/chr_ord_rand.py
>>> random_shift('abc123')
'SGC123'
>>> random_shift('hello')
'SGEBU'
>>> random_shift('hello')
'OIUOI'
>>> random_shift('abc123')
'CXV123'
>>> random_shift('123 456')
'123 456'
>>>
```

Monte Carlo Methods

- Any method which solves a problem by generating suitable random numbers, and observing the fraction of numbers obeying some property (or properties)
  - The method is useful for obtaining numerical solutions to problems which are too complicated to solve analytically.

http://mathworld.wolfram.com/MonteCarloMethod.html

Monte Carlo in action

How many doubles will you get in n rolls of 2 dice?

the input n is the total number of rolls

```
def count_doubles(n):
    """ inputs a # of dice rolls
        outputs the # of doubles """
    if n == 0:
        return 0       # zero rolls, zero doubles…
    else:
        d1 = choice([1,2,3,4,5,6])
        d2 = choice(range(1,7))

        if d1 != d2:
            return count_doubles(n-1)  # don't count it
        else:
            return 1 + count_doubles(n-1)  # COUNT IT!
```

one roll

where is the doubles check?

Monte Carlo in action

Write the same function with a **list comprehension**

```
def count_doubles(n):
    """ inputs a # of dice rolls
        outputs the # of doubles """
    return sum([choice(range(6)) == choice(range(6)) \
            for x in range(n)])
```

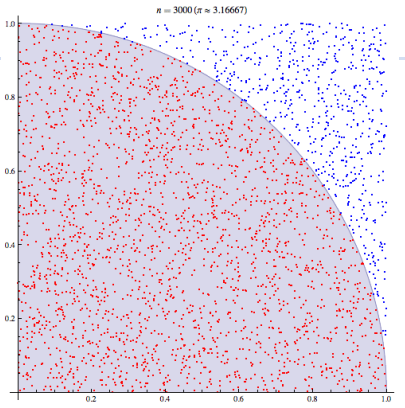
where is the doubles check?

One final example…

- Suppose you wanted to estimate pi

1. Generate two random numbers between -1 and 1.
   – denoted (x,y)
2. Compute the distance of (x,y) from (0,0)
3. Depending on distance, point is either inside or outside of a circle centered at (0,0) with a radius of 1.

Illustration…



Exercise

Design a strategy for estimating pi using random numbers ("dart throws") and this diagram ("dartboard"):

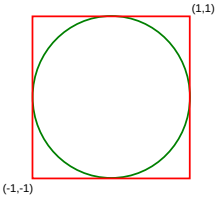
Name(s):

1) Here is a dart-throwing function: What does this return?

```
def throw():
    return [ random.uniform(-1,1),
             random.uniform(-1,1) ]
```

2) Here is a dart-testing function: What does this return?

```
def test(x,y):
    return (x**2 + y**2) < 1.0
```



3) What strategy could use the functions in (1) and (2) to estimate pi?

4) Write a function to implement your strategy: