Lecture 4.3 : Random numbers¶

Introduction

- In order to test our code or to run simulations we will often find it useful to generate *random* numbers. Python provides a `random` module which defines a number of useful methods in this regard. We look at a selection of those methods here.

Random numbers

- The `random()` method returns a random floating point number in the interval [0, 1). (This means 0 is included in the interval but 1 is not.) We can use this method as follows:

```
>>> import random
>>> for i in range(5):
...     print(random.random())
...
0.9872566937336238
0.06776613508301477
0.926626463200632
0.9909704332058146
0.3500087707759981
```

- The particular sequence of random numbers generated by the random number generator is determined by the generator's *seed*. Seeding the generator with the same number causes the same random sequence to be produced. The sequence appears random because the next number in the sequence cannot be predicted from previous ones. However the generated sequence is entirely determined by the initial seed supplied to the underlying algorithm. Such generators are therefore referred to as *pseudo random number generators* (PRNGs). For example:

```
>>> random.seed(42)
>>> for i in range(5):
...     print(random.random())
...
0.6394267984578837
0.025010755222666936
0.27502931836911926
0.22321073814882275
0.7364712141640124
>>> random.seed(42)
>>> for i in range(5):
...     print(random.random())
...
0.6394267984578837
0.025010755222666936
0.27502931836911926
0.22321073814882275
0.7364712141640124
```

- If we pass no argument to `seed()` the PRNG is seeded with the current clock value. This provides enough randomness for most purposes.

Other `random` methods

- The `randint(a,b)` method generates a random integer *N* in the range *a* <= *N* <= *b*.

```
>>> random.seed()
>>> random.randint(1,100)
17
>>> random.randint(1,100)
72
>>> random.randint(1,100)
37
```

- The `choice(sequence)` method returns a random element of *sequence*.

```
>>> random.choice([1,2,3,4,5])
2
>>> random.choice([1,2,3,4,5])
5
>>> random.choice([1,2,3,4,5])
1
>>> random.choice('selectacharacter')
'a'
>>> random.choice('selectacharacter')
'c'
>>> random.choice('selectacharacter')
'h'
```

- The `shuffle(sequence)` method shuffles the order of the elements of *sequence* (useful for generating permutations of the elements of a sequence).

```
>>> my_list = [1,2,3,4,5]
>>> random.shuffle(my_list)
>>> my_list
[3, 4, 2, 5, 1]
>>> random.shuffle(my_list)
>>> my_list
[3, 2, 4, 5, 1]
>>> my_string = 'string'
>>> random.shuffle(my_string)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
  File "/usr/lib64/python3.3/random.py", line 265, in shuffle
    x[i], x[j] = x[j], x[i]
TypeError: 'str' object does not support item assignment
>>> my_list = list(my_string)
>>> random.shuffle(my_list)
>>> ''.join(my_list)
'rgntsi'
```

- The `sample(sequence, N)` method returns a new sequence containing *N* randomly selected elements of *sequence*.

```
>>> my_list = [1,2,3,4,5]
>>> random.sample(my_list, 2)
[2, 5]
>>> random.sample(my_list, 2)
[1, 3]
>>> random.sample(my_list, 2)
```

```
[5, 4]
>>> random.sample('string', 3)
['i', 't', 's']
>>> ''.join(random.sample('string', 3))
'nsr'
>>> ''.join(random.sample('string', 3))
'trg'
>>> ''.join(random.sample('string', 3))
'sri'
```