pyRandomdotOrg - A Python Frontend for Random.org

Sean Brewer - sbrewer@students.berry.edu

May 27, 2008

Contents

Chapter 1

Introduction

pyRandomdotOrg is simply a Python interface to the Random.org's random number generating service. It's fairly straightforward and easy to use. If you need "good" random numbers and you need to use Python this is one of the best ways to get "good" random numbers without much hassle, as long as you have an internet connection of course.

Chapter 2

Class and Methods

2.1 Information

The methods and the arguments for the methods are very similar to what is listed at http://www.random.org/clients/http/. There are a few differences and they are described here in very good detail.

2.2 Class

There is only one class in pyRandomdotOrg, the clientlib class. It is what allows you to connect to the Random.org random number generating service.

2.2.1 clientlib

clientlib(clientname, emailaddr)

Argument:	**Type:**	**Information:**
clientname	String	The client name (the name of your program)
emailaddr	String	Your email address. This allows the Random.org admin to contact you if something goes awry with your client.

2.3 Methods

2.3.1 IntegerGeneratorList

IntegerGeneratorList(num,nmin,nmax,col=1,base=10,rnd=”new”)

This method returns a list of random integers from Random.org and returns them as a python list, i.e. [1,2,3,4,5,6].

Argument:	**Type:**	**Information:**
num	Int	The number of integers requested.
nmin	Int	The smallest value allowed for each integer.
nmax	Int	The largest value allowed for each integer.
col	Int	The number of columns in which the integers will be arranged. The integers should be read (or processed) left to right across columns.
base	Int	The base that will be used to print the numbers, i.e., binary (2), octal (8), decimal (10) or hexadecimal (16).
rnd	String	Determines the randomization to use to generate the strings. If new is specified, then a new randomization will created from the truly random bitstream at RANDOM.ORG. This is probably what you want in most cases. If id.identifier is specified, the identifier is used to determine the randomization in a deterministic fashion from a large pool of pregenerated random bits. Because the numbers are produced in a deterministic fashion, specifying an id basically uses RANDOM.ORG as a pseudo-random number generator. The third (date.iso-date) form is similar to the second; it allows the randomization to be based on one of the daily pregenerated files. This form must refer to one of the dates for which files exist, so it must be a day in the past. The date must be in ISO 8601 format (i.e., YYYY-MM-DD).

2.3.2 IntegerGenerator

IntegerGenerator(nmin,nmax,base=10,rnd="new")

This method returns a random integer from Random.org

Argument:	**Type:**	**Information:**
nmin	Int	The smallest value allowed for each integer.
nmax	Int	The largest value allowed for each integer.
base	Int	The base that will be used to print the numbers, i.e., binary (2), octal (8), decimal (10) or hexadecimal (16).
rnd	String	Determines the randomization to use to generate the strings. If new is specified, then a new randomization will created from the truly random bitstream at RANDOM.ORG. This is probably what you want in most cases. If id.identifier is specified, the identifier is used to determine the randomization in a deterministic fashion from a large pool of pregenerated random bits. Because the numbers are produced in a deterministic fashion, specifying an id basically uses RANDOM.ORG as a pseudo-random number generator. The third (date.isodate) form is similar to the second; it allows the randomization to be based on one of the daily pregenerated files. This form must refer to one of the dates for which files exist, so it must be a day in the past. The date must be in ISO 8601 format (i.e., YYYY-MM-DD).

2.3.3 SequenceGenerator

SequenceGenerator(nmin,nmax,rnd="new")

This method returns a randomized sequence of numbers based on a particular interval. The method returns the list of numbers as a python list.

Argument:	**Type:**	**Information:**
nmin	Int	The lower bound of the interval (inclusive).
nmax	Int	The upper bound of the interval (inclusive).
rnd	String	Determines the randomization to use to generate the strings. If new is specified, then a new randomization will created from the truly random bitstream at RANDOM.ORG. This is probably what you want in most cases. If id.identifier is specified, the identifier is used to determine the randomization in a deterministic fashion from a large pool of pregenerated random bits. Because the numbers are produced in a deterministic fashion, specifying an id basically uses RANDOM.ORG as a pseudo-random number generator. The third (date.isodate) form is similar to the second; it allows the randomization to be based on one of the daily pregenerated files. This form must refer to one of the dates for which files exist, so it must be a day in the past. The date must be in ISO 8601 format (i.e., YYYY-MM-DD).

2.3.4 StringGenerator

StringGenerator(num,len,digits=True,upperalpha=True, loweralpha=True,unique=True,rnd="new")

This method gets a list of random strings from Random.org and returns them as a python list.

Argument:	**Type:**	**Information:**
num	Int	The number of strings that you want.
len	Int	The length of the strings. All the strings produced will have the same length. The maximum number is 20 and the minimum is 1
digits	Boolean	Determines whether digits (0-9) are allowed to occur in the strings.
upperalpha	Boolean	Determines whether uppercase alphabetic characters (A-Z) are allowed to occur in the strings.
loweralpha	Boolean	Determines whether lowercase alphabetic characters (a-z) are allowed to occur in the strings.
unique	Boolean	Determines whether the strings picked should be unique (as a series of raffle tickets drawn from a hat) or not (as a series of die rolls). If unique is set to on, then there is the additional constraint that the number of strings requested (num) must be less than or equal to the number of strings that exist with the selected length and characters.
rnd	String	Determines the randomization to use to generate the strings. If new is specified, then a new randomization will created from the truly random bitstream at RANDOM.ORG. This is probably what you want in most cases. If id.identifier is specified, the identifier is used to determine the randomization in a deterministic fashion from a large pool of pregenerated random bits. Because the numbers are produced in a deterministic fashion, specifying an id basically uses RANDOM.ORG as a pseudo-random number generator. The third (date.isodate) form is similar to the second; it allows the randomization to be based on one of the daily pregenerated files. This form must refer to one of the dates for which files exist, so it must be a day in the past. The date must be in ISO 8601 format (i.e., YYYY-MM-DD).

2.3.5 RandomString

RandomString(len,digits=True,upperalpha=True,loweralpha=True, unique=True,rnd="new")

This method gets a random string from Random.org and returns it as a string.

Argument:	**Type:**	**Information:**
len	Int	The length of the strings. All the strings produced will have the same length. The maximum number is 20 and the minimum is 1
digits	Boolean	Determines whether digits (0-9) are allowed to occur in the strings.
upperalpha	Boolean	Determines whether uppercase alphabetic characters (A-Z) are allowed to occur in the strings.
loweralpha	Boolean	Determines whether lowercase alphabetic characters (a-z) are allowed to occur in the strings.
unique	Boolean	Determines whether the strings picked should be unique (as a series of raffle tickets drawn from a hat) or not (as a series of die rolls). If unique is set to on, then there is the additional constraint that the number of strings requested (num) must be less than or equal to the number of strings that exist with the selected length and characters.
rnd	String	Determines the randomization to use to generate the strings. If new is specified, then a new randomization will created from the truly random bitstream at RANDOM.ORG. This is probably what you want in most cases. If id.identifier is specified, the identifier is used to determine the randomization in a deterministic fashion from a large pool of pregenerated random bits. Because the numbers are produced in a deterministic fashion, specifying an id basically uses RANDOM.ORG as a pseudo-random number generator. The third (date.iso-date) form is similar to the second; it allows the randomization to be based on one of the daily pregenerated files. This form must refer to one of the dates for which files exist, so it must be a day in the past. The date must be in ISO 8601 format (i.e., YYYY-MM-DD).

2.3.6 QuotaChecker

QuotaChecker(ipaddr=None)

This method returns the bit quota of the given IP address. If the argument is left empty it will return the bit quota of the network the python program is working from.

Argument:	**Type:**	**Information:**
ipaddr (optional)	String	The IP address that you want to check the quota of. If left blank your IP address will be used

Chapter 3

Examples

pyRandomdotOrg is very straightforward to use. Here are some examples as executed from the Python REPL.

```
>>> import pyRandomdotOrg
>>> rnd = pyRandomdotOrg.clientlib("An Example Client","sbrewer@students.berry.edu")
>>> print rnd.IntegerGeneratorList(5,1,1000000)
[317176, 795730, 842222, 790934, 876738]
>>> print rnd.IntegerGenerator(1,1000000)
237629
>>> print rnd.SequenceGenerator(1,52)
[36, 46, 48, 1, 45, 50, 19, 32, 8, 28, 31, 23, 37, 2, 14, 11, 35, 49, 52, 7, 12, 40, 4, 42, 22,
18, 6, 43, 10, 34, 27, 13, 25, 5, 20, 33, 44, 39, 21, 30, 16, 24, 17, 9, 41, 38, 15, 47, 29, 26, 51, 3]
>>> print rnd.StringGenerator(5,20)
['2BASj44Ugk2douNKGEON', 'YyUOSZqXcQK5hqrZdtxW',
'TPNiOeMrZbMR6g7u1cEQ', 'iRedoDw0h09kpThYVslB', 'INuUq2qMwPgbgUAqujzl']
>>> print rnd.RandomString(5)
cdcuF
>>> print rnd.QuotaChecker()
996902
>>> print rnd.QuotaChecker("6.20.28.79")
1000000
```

Chapter 4

Conclusion/License

If any bugs are found in the library, do not hesitate to send me an e-mail at sbrewer@students.berry.edu. If you know how to fix the problem and can provide a patch that will be even more helpful. This library is licensed under the GPL2, and is therefore free software. Please see the included document (GPL2.txt) for more details. Also, thanks to http://www.random.org/clients/http/, as I used many of the descriptions of the arguments in this document from there. I simply couldn't word them better than that!