1 Basics

Contents

The exercises below are intended to make sure that you have mastered the basics, e.g., writing functions, using loops, working with lists, etc., while doing somthing interesting.

1.1 Working with lists

We give some examples which illlustrate basic operations on lists, then give some exercises which use these operations.

Building lists

```
>>> foo = []                  # empty list
>>> foo.append(2).            # append an element
>>> foo                       # check that it is there
[2]

>>> foo.append(3); foo.append(5)
>>> foo
[2, 3, 5]
```

Using a list as a stack

```
>>> foo.pop()
5
>>> foo
[2, 3]
```

Adding two lists

```
def add(a,b):
    outputList = []
```

```
    for i in range(len(a)):
        outputList.append(a[i] + b[i])
    return outputList

>>> add([1,2,3], [1,1,-1])
[2, 3, 2]

>>> add(["a", "b", "c"], ["=1", "=2", "=3"])
['a=1', 'b=2', 'c=3']
```

Exercises

1. Devise a function `uniquefy` that returns a list without repeats. Thus `uniquefy([1,1,3,3,1,2,3])` returns `[1,3,2]` and `uniquefy(['a', 'b', 'a', 'c', 'c'])` returns `['a', 'b', 'c']`.
2. Devise a function `isUnique` that returns `True` if a list has no repeats, false otherwise.

1.2 Generatng random numbers

There are many uses in computer science and various applied fields for a source of random numbers, e.g., encryption. The best sources are physical. One can, for example, translate the clicks of a Geiger counter placed near a radioactive mineral into a random number generator. For a random number generator based on atmospheric noise, consult random.org.

Another approach is to use some kind of algorithm to generate sequences of numbers that have many of the same statistical properties as do true random number sequences. Because algorithms are deterministic, the sequences they produce are not truly random. We call them *pseudorandom*. We will describe the theory. Your task is to apply the theory to design and test some pseudorandom number generators.

Linear congruential generators

Choose integers a , b , and n . Consider the function

$$f(x) = (ax + b) \bmod n \tag{1}$$

The expression y mod b computes the remainder of y upon division by n . Thus $17 \bmod 3 = 2$ and $18 \bmod 3 = 0$. In Python, we write thes expressions as $17 \ \% \ 3$, etc. One can use f to generate a sequence of numbers as follows. First, choose a *seed* x_0 . Then, assuming that x_n has been computed, let $x_{n+1} = f(x_n)$.

Let's do an example with $f(x) = 5x + 1 \bmod 17$. First, we define f in Python:

```
$ python3

>>> def f(x):
...   return (5*x + 1) % 17
...
```

Then we test it:

```
>>> f(0)
1
>>> f(1)
6
>>> f(6)
14
```

Finally, we use a loop to generate a sequence:

```
>>> a = 0
>>> for i in range(0,17):
...   print(a, end=' ')
...   a = f(a)
...
0 1 6 14 3 16 13 15 8 7 2 11 5 9 12 10 0
```

Exercises

1. Use the function f given above to generate pseudorandom numbers x in the unit interval. That is, x is a floating point number such that $0 \leq x < 1$.
2. Notice that there are no repeats in the sequence $\{x_n\}$ generated above. Does this happen in general? Experiment with various choices of a , b , and n . You may want to simplify and take $b = 0$.
3. Is there a way to automate the task of determining whether the numbers $\{x_n\}$, for $n = 0, 1, \ldots n-1$ has no repeats? That is, by applyig some function to the output instead of eyeballing it.
4. Devise a function that returns a random element of a list.
5. Devise a function `nonsense(n)` that returns a nonsense string of length n . For example, `nonsense(5)` returns the string "axmuh" the first time it is called and "nqwof" the second time. (So it is not a function in the sense used in Mathematics.
6. (Optional) Devise a function that returns a random sentence. The function should take as input several lists of words.

7. Construct a function `randomSuit()` which returns one of the strings "Heart", "Diamond", "Spade", "Club", each with equal probability.

8. (Optional) Construct a funtion `randomSuit(h, d, s, c)` which returns "Heart" with probability h , "Diamon" with proability d , etc. Of course $a + d + s + c = 1$.

9. (Optional, a small project) Construct a function `pokerHand` which returns a random five-card hand.

1.3 Dynamical Systems

A *dynamical system* consists of a set S of *states* and a function $f : S \to S$ that computes a new state from an old one. The set S may the non-negative reals, the set of all reals, the set $\mathbb{R}^n$ of real-valued n -vectors, the letters of the alphabet, the set of all alphanumeric strings, etc. If $x_0 \in S$ is the initial state of the system, then one defineds a *trajectory* $\{x_n\}$ in S via the recursion relation $x_{n+1} = f(x_n)$.

Here is a simple example: S is the set of non-negative reals and $f(x) = kx$, where $k > 0$ This dynamical system models exponential growth (or decay). Here is a slightly more complicated model:

$$f(x) = kx(1 - x) \tag{2}$$

1. Compute the sequence $\{x_j\}$ where $x_0 = 0.5$, $j \leq N$, where $N = 10$, and where k takes on the values $0.5, 1.0, 1.5, \ldots 4.0$. Do you find anything of interest?

2. Repeat the preceding, but graph the results, but with significantly larger valued of N . Do you find anything of interest?

1.4 Plotting things

The most basic way of using `matplotlib` is to construct two lists, the x-values and the y-values. Call these lists `xs` and `ys`. Putting them togehter gives a list of points in the plane. Joing successive pairs of ponts constructs a figure. That is the figure drawn by `plt.plot(xs, ys)`. The `xs` and `ys` can be either lists of numbers or numpy arrays.

1. Let `xs = [0,1,1,0]`, `ys = [0, 0, 1, 1]`. What figure will `plt.plot(xs, ys )` make? Answer using pure thought, then use Python to check your answer.

2. Draw a square using matplotib.

White noise and Brown noise

White noise is produced by uniformly distributed voltage fluctuations. It can be modeled like this:

```
import random as r

ys = []
for i in range(0,100):
    ys.append(r.random())
```

And it can be graphed like this:

```
xs = list(range(100))

plt.plot(xs, ys)
```

Exercise A.. Plot white noise

Brown noise is produced by a different kind of random voltage fluctuation: begin with some initial voltage, say 0. Then add successive uniformly distributed random numbers, say, in the range -0.1 to $+0.1$. We can gernerate brown noise like this:

```
ys = [0]
for i in range(0,99):
    ys.append(ys[i] + 0.1*(r.random() - 0.5))
```

Exercise B. Plot brown noise

Exercise C. (1) Listen to both white and brown noise; (2) investigate physical sources of both types of noise. (3) Which type of noise do you prefer?