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 Functions

1. Define a function `greet` such that `greet( John )` returns the string "Hello, John, how's it going?"

2. Use `math.log` and `math.exp` to define a function `power(a,k)` that raises a to the power k , where k may not be an integer.

3. Define a function `sharpPulse(x)` which is zero outside the interval [1, 1] , and whose graph on the interval [1, 1] consists of two line segments one joining (1, 0) to (0, 1) the other joining (0, 1) to (1, 0) . Test your code to make sure that it is doing what it should do. Then use `maplotlib` to graph it on the interrva $3 \leq x \leq 3$.

1 2 Loops

1. Compute the sum $S_n = 1 \quad 1/2 + 1/3 \quad 1/4 \pm \cdots \pm 1/n$ for $n = 10$ and $n = 100$. What is the signficance of this sum?

2. Devise a program to compute the function $(s) = 1 + 1/2^s + 1/3^s + \cdots$ using the partial sum $_n(s) = 1 + 1/2^s + 1/3^s + \cdots + 1/n^s$. Compute values of (1.5) , (2) , and (2.5) accurate to within 0.000001. How many terms were necessary in each case.

1 3 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 4 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 *pseudor ndom*. 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 mod 3 = 2 and 18 mod 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 $n = 0, 1, \ldots n \quad 1$ 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 5 Dynamical Systems

A *dyn mic l system* consists of a set S of *st tes* 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 ^{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 *tr jectory* $\{x_n\}$ 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:

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

Exercises A

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 5 1 Investing

Consider a parent who wants to save for a childs eduction. He/she sets an account upon the birth of the child with an initial amount of 1000 dollars. Annaul return on the investment is 4 percent. Each year, 6000 dollars is added to the account. What will the value of the account be in eighteen years.

A problem like this is describe by a dynamical system on the non-negative real numbers, $f :$ $\rightarrow$, where f has the form

$$f(x) = ax + b \tag{3}$$

Exercises B

1. Solve the investment problem above. Your solution should include a printout the value of the account for each year.
2. Let's use the same dynamical system to solve mortgage problem. A family buys a home, taking out a mortgage of 200,000 dollars at four perent annual interest. They make a monthly payment of 1000 dollars How long will it take to pay o the mortgage? How much interest will the family have paid when the mortgage is paid o ? Your solution should contain a month-by-month schedule of the family's mortgage balance.

Remarks

(1) Let $a \in S$ be the initial state of a dynamical systemm and let $f : S \rightarrow S$ be the next-state function. The *orbit* of a is the sequence $a, f(a), f(fa)), f(f(f(a))), \ldots$. One can compute the first n elements of the orbit of f based at a using the folloing code.

```
def orbit(a, n, f):
    x = a
    _orbit = [x]
    for i in range(n):
        x = f(x)
        _orbit.append(x)
    return _orbit
```

(2) In all the examples above, the state space was the set of real numbers, or some suset thereof. But the state space can be anything: strings, grids of symbols as in the game of life, grids of numbers, as in simulations of heat conduction, etc.

(3) L-systems are an example of a dynamical system on a set of strings. L-systems were inventned to describe the growth of plants; they can be used to draw all sorts of figures that have a self-simlar structure. There are two ingredients to an L-system. The first is an *xiom* which is the initial state of the string. In our example, we take the axiom to be "F+F+F+F". One can view such a string as as a set of instructions for drawing a figure. The symbol "F" means "move forward one step." The symbol "+" means "turn through a certain angle θ . We take $\theta = \pi/2$ radians. (**Question:** What figure does the string "F+F+F+F" cause to be drawn?)

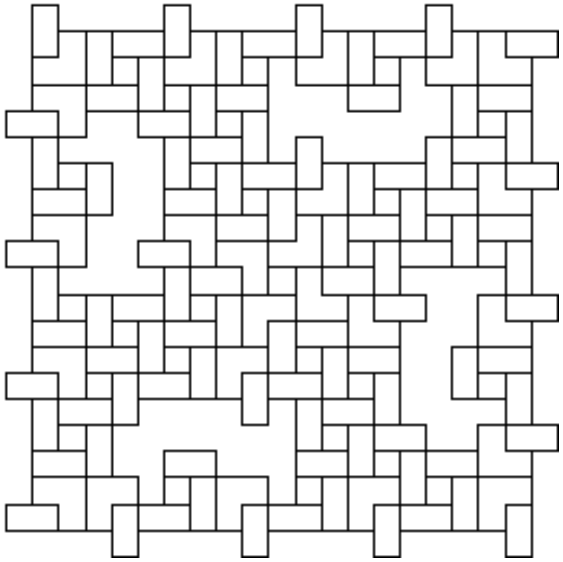
The next ingredient is a set of "rewrite rules." In the present case there is just one:

```
r: F -> FF+F+F+F+FF
```

To apply the rule, we scan through the input string, replacing every occurrence of F by FF+F+F+F+FF. This process can be repeated many times. To summarize, we have a dynamical system

$$r : \text{Strings} \to \text{Strings} \tag{4}$$

Applying r to the axiom a number of times and using the result to draw a figure, one obtains the image displayed below. One reference is L-System User Notes by Paul Bourke.



L-system image

More in accord with Lindenmayer's intentions, one can also produce images like this one:

1 6 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.

Fourier series

In class we used `matplotlib` to graph the Fourier poynomial

$$f(t) = \sin \pi t + \frac{1}{3} \sin 3\pi t + \frac{1}{5} \sin 5\pi t \tag{5}$$

This function is one element of the family of functions

$$f_n(t) = \sin \pi t + \frac{1}{3} \sin 3\pi t + \cdots + \frac{1}{2n+1} \sin(2n+1)\pi t \tag{6}$$

where the sum is taken over odd integers. Thus the function in equation (5) is f_2 . Plot f_3 and f_4 .

Exercise A What is the limit of the graph of f_n as n tends to infinity?

Note Consdier the Fourier series

$$f(t) = a_0 + \sum_{n=1} a_n \cos n\omega t + \sum_{n=1} b_n \sin n\omega t \tag{7}$$

The coefficients a and b are called the *mplitudes*. Let us suppose that $f(t)$ represents the voltage in a circuit. As we know from basic physics, the current in a circuit is governed by Ohm's law, $V = IR$. Moroever, power is given by voltage times current, $P = VI$, so that finally $P = V^2 R$. In other words, power is proportional to the square of the voltage, where the constant of proportionality is the resistance. Returning to the Fourier series $f(t)$, we find that the instantaneous power in mode n is $a_n^2 + b_n^2$. If the series is written entirely in terms of sines (as it may be using a phase shift), then the instaneaous power is simply b_n^2 . In conclusion: *power per mode is given by the square of the Fourier coe cients*. The sequence $\{b_n^2\}$ is called the *power spectrum*

Exercise B Graph (perhaps by hand) the power spectrum of f_4 .

1 7 Noise: white and brown

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 di erent 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?

Note We wil discuss the power spectra of white and brown noise in class and learn how to compute spectra using the `numpy` package.

References for noise

Listen to white noise

Listen to brownian noise

About white noise (Wikipedia)

About brownian noise (Wkipedia)