IIT Madras
BSc Degree

[Home](#) [Lesson-3.6](#)

Lesson-3.5

Library

We will look at two more libraries — `math` and `random` — and use them to solve some fascinating problems in mathematics.

`math`

Consider the following sequence:

$$\sqrt{2}, \sqrt{2+\sqrt{2}}, \sqrt{2+\sqrt{2+\sqrt{2}}}, \ \ldots$$

Mathematically, it is known that this sequence converges or approaches a specific value. In other words, this sequence gets closer and closer to a well defined number as more terms are added. This number is called the **limit** of the sequence. What is the limit for the above sequence? Can we use whatever we have learned so far to estimate this value?

```python
import math
x = 0
for n in range(1, 6):
    x = math.sqrt(2 + x)
    print(f'n = {n}, x_n = {x:.3f}')
```

If we execute the above code, we get the following output:

```
n = 1, x_n = 1.414
n = 2, x_n = 1.848
n = 3, x_n = 1.962
n = 4, x_n = 1.990
n = 5, x_n = 1.998
```

`sqrt` is a function in the `math` library that returns the square root of the number that is entered as argument. Representing the output shown above as a table:

n	x_n	**Approximate value**
1	$\sqrt{2}$	1.414
2	$\sqrt{2+\sqrt{2}}$	1.848
3	$\sqrt{2+\sqrt{2+\sqrt{2}}}$	1.962
4	$\sqrt{2+\sqrt{2+\sqrt{2+\sqrt{2}}}}$	1.990
5	$\sqrt{2+\sqrt{2+\sqrt{2+\sqrt{2+\sqrt{2}}}}}$	1.998

Isn't that beautiful? It looks like this sequence — the train of square roots — is approaching the value 2. Let us run the loop for more number of iterations this time:

```
import math
x = 0
for n in range(1, 20):
    x = math.sqrt(2 + x)
print(x)
```

After just 20 iterations, the value is so close to two: `1.9999999999910236`. But we have used trial and error to decide when to terminate the iteration. A better way to do this is to define a tolerance: if the difference between the previous value and the current value in the sequence is less than some predefined value (tolerance), then we terminate the iteration.

```
import math
x_prev, x_curr = 0, math.sqrt(2)
tol, count = 0.00001, 0
while abs(x_curr - x_prev) >= tol:
    x_prev = x_curr
    x_curr = math.sqrt(2 + x_prev)
    count += 1
print(f'Value of x at {tol} tolerance is {x_curr}')
print(f'It took {count} iterations')
```

`random`

How do we toss a coin using Python?

```
import random
print(random.choice('HT'))
```

That is all there is to it! `random` is a library and `choice` is a function defined in it. It accepts any sequence as input and returns an element chosen at random from this sequence. In this case, the input is a string, which is nothing but a sequence of characters.

We know that the probability of obtaining a head on a coin toss is 0.5. This is the theory. Is there a way to see this rule in action? Can we computationally verify if this is indeed the case? For that, we have to set up the following experiment. Toss a coin n times and count the number of heads. Dividing the total number of heads by n will give the empirical probability. As n becomes large, this probability must approach 0.5.

```
import random
n = int(input())
heads = 0
for i in range(n):
    toss = random.choice('HT')
    if toss == 'H':
        heads += 1
print(f'P(H) = {heads / n}')
```

Let us run the above code for different values of n and tabulate our results:

n	$P(H)$
10	0.2
100	0.52
1,000	0.517
10,000	0.5033
100,000	0.49926
1,000,000	0.499983

The value is approaching `0.5` as expected! `random` is quite versatile. Let us now roll a dice!

```
import random
print(random.randint(1, 6))
```

`randint(a, b)` returns a random integer N such that $a \leq N \leq b$. We can do a similar experiment for finding the probability of obtaining a number, say 1, when a dice is thrown.