What is Memoization?

In this lesson, we will learn about a result storing technique used in top-down dynamic programming.

We'll cover the following

- Memoization
  - The way to memoize

We saw in the last chapter what we mean by the top-down approach. It essentially means that we start looking at the problem as a whole, then break it down into smaller and smaller units, and then evaluate those units that produce the answer to our original problem. Recursion acts as a natural way to write a top-down dynamic algorithm. The process of storing evaluated results in the top-down approach is called **memoization**.

Memoization

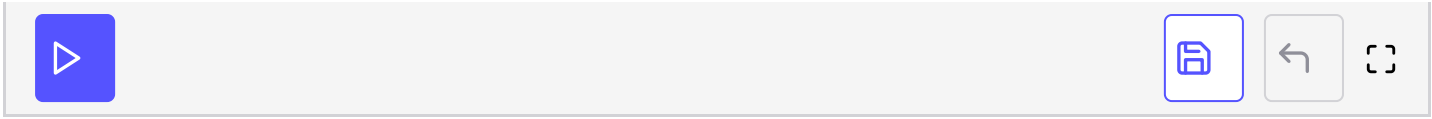
No, we did not make a typo, its memoization, without an r. Memoization is defined as:

> *The act of storing results of costly function call, and retrieving them from the store when needed again to avoid re-evaluation.*

The way to memoize

The most typical way to memoize results is by using **hashtables**. Hashtables are arrays that can be indexed using variable types other than integers. As we have already seen, we may need to store results against strings, floats, and even objects. Hashtables allow us to do all of this quite easily. The only condition for using hashtables is that keys should be unique. In Python, we can use the dictionary data structure as a hash table. The dictionary allows us to store data in the form of key-value pairs in an unordered collection of data values. The code snippet below shows the usage of a dictionary in Python.

```
# You create an empty dictionary using curly brackets
dictionary = {}

key = 1
value = "abcd"

# To add value against a key in the dictionary, do this

dictionary[key] = value
#or
dictionary[2] = "abc"

# keys and values can be anything, from integers to strings to custom objects

dictionary["hello"] = "hi"
dictionary[1.1] = 1

# A custom class
class Dummy:
  def __init__(self, val):
    self.val = val

# A custom class's object
customObject = Dummy(5)
dictionary[customObject] = 5

# To iterate over dictionary do this
for k,v in dictionary.items():
  print (k,":",v)
```



In some cases, you can also use simple lists instead of dictionaries, like in the Fibonacci numbers algorithm. However, hashtables allow an average-case complexity of constant time and are much easier to use. Therefore, hashtables are recommended over lists for memoization.

In the next lesson, we will revisit the Fibonacci numbers algorithm with memoization.