Python Interview Problems

Dynamic Programming

Dynamic programming is a technique to optimize algorithms by breaking a problem with overlapping sub-problems into smaller sub-problems and then combining the solutions to the sub-problems to solve the larger problem.

Dynamic Programming

Dynamic programming is both a mathematical optimization method and a computer programming method. It simplifies a complicated problem by breaking it down into simpler sub-problems.
It can be applied to combinatorial and optimization problems such as finding the shortest path between two points or finding the smallest set of objects that satisfies some criteria.

Dynamic Programming Problem Variable

When solving a problem with dynamic programming, we maintain a dynamic set of variables corresponding to each sub-problem that changes in order to find the overall solution. These variables are called problem variables.

Recursive Step in Recursive Function

A recursive function should have a **recursive step** which calls the recursive function with some input that brings it closer to its base case. In the example, the recursive step is the call to `countdown()` with a decremented value.

```
def countdown(value):
  if value <= 0:
```

```python
    print("done")
  else:
    print(value)
    countdown(value-1)  #recursive step
```

What is Recursion

Recursion is a strategy for solving problems by defining the problem in terms of itself. A recursive function consists of two basic parts: the base case and the recursive step.

Fibonacci Recursion

Computing the value of a Fibonacci number can be implemented using recursion. Given an input of index N, the recursive function has two base cases – when the index is zero or 1. The recursive function returns the sum of the index minus 1 and the index minus 2.
The Big-O runtime of the Fibonacci function is O(2^N).

```python
def fibonacci(n):
  if n <= 1:
    return n
  else:
    return fibonacci(n-1) + fibonacci(n-2)
```