National University of Singapore
School of Computing
CS1010X: Programming Methodology
Semester II, 2024/2025

Tutorial 10
Memoization & Dynamic Programming

Dynamic Programming and Memoization

1. Consider the following function:

$$f(n) = \begin{cases} n/2 & \text{if } even(n) \\ 3n+1 & \text{if } odd(n) \end{cases}$$

The Collatz conjecture states that for any integer n, the sequence n, $f(n)$, $f(f(n))$, $\cdots$ will eventually reach 1.
We define the Collatz distance for an integer n as the number of steps needed to reach 1.

(a) Write a function `collatz_distance(n)`.

```
>>> collatz_distance(1)
0
>>> collatz_distance(4)
2
>>> collatz_distance(27)
111
```

(b) Write a function `max_collatz_distance(n)`, which computes the maximum Collatz distance of 1, 2, 3, ..., n.

```
>>> max_collatz_distance(6)
8
>>> max_collatz_distance(8)
16
>>> max_collatz_distance(18)
20
```

(c) Give a memoized version of `max_collatz_distance_memo(n)` using memoize as provided in the lecture.

(d) Memoize it without using the function provided in the lecture. You should be able to do better.

Exception Handling

2. The following function accesses a URL on the internet and retrieves its contents:

```
from urllib.request import urlopen
from urllib.parse import urlsplit
from urllib.error import *
def httpget(url):
    parsed = urlsplit(url)
    if not parsed.scheme: #protocol insertion
        url = 'http://' + url
    elif parsed.scheme != 'http':
        raise ValueError("Unknown protocol")
    return urlopen(url).read()
```

(a) Your ability to access a URL on the internet is not guaranteed - it is only on a "best effort" basis. An example of an URL is *http://www.nus.edu.sg/*. Describe qualitatively (no need for exact exceptions/error codes) some of the things that could go wrong.

(b) Why is it a good idea to raise an error instead of simply returning a string `'Not Found'` or an empty string to indicate that the URL is not accessible?

(c) Suppose we are interested in 3 types of errors: user errors, internet errors and all other errors. A user error might be a mistyped URL while a internet error may be a connection problem.

For user errors, we try to catch `URLError` and rethrow[1] it as `ValueError`. For internet error, we try to catch `HTTPError` and rethrow it as custom error type `InternetFail`. For other errors, just rethrow.

Modify `httpget` to accomodate the above error handling.

Note: HTTPError is a subclass of URLError, hence the order is important. We want to handle the more specific error first.

[1] When we catch an exception and raise it again, we are *rethrowing* that error.

(d) Using the above, write a function `download_URLs(URL_filenames)` to download a set of files, where `URL_filenames` is a list of pairs in the following format: `[[URL1, filename1], [URL2, filename2], ...]`. In this instance, the contents of `URL1` should be saved locally as `filename1`.

Naturally, many errors can occur during downloading - if we get `InternetFail` or `ValueError`, we want to ignore those and continue downloading the rest of the list. Otherwise, we rethrow the error.

Hint: To save data to a file, use the following:

```
with open(filename, 'wb') as myFile:
    myFile.write(contents)
```