A mini rosseta code example: Fibonacci numbers

The outputs for each language will be visible after running

```
showman execute ./examples/external-code.typ
```

- **Note**: If you're on Windows, the `bash` example will not evaluate.
- `typst` will render for free, independent of `showman execute`.

typst

```
#let fib(n) = {
  if n < 2 {
    n
  } else {
    // Typst memoizes by default :)
    fib(n - 1) + fib(n - 2)
  }
}
#fib(50)
```

```
12586269025
```

python

```
import functools

@functools.lru_cache(maxsize=None)
def fib(n):
    if n < 2:
        return n
    return fib(n-1) + fib(n-2)

fib(50)
```

```
12586269025
```

cpp

```
#include <iostream>
#include <vector>
typedef unsigned long long ulong;
ulong fib(ulong n, std::vector<ulong> &cache) {
    if (n < 2) {
        return n;
    }
    if (cache[n] != -1) {
        return cache[n];
    }
    cache[n] = fib(n-1, cache) + fib(n-2, cache);
    return cache[n];
}

int main() {
    std::vector<ulong> cache(101, -1);
    std::cout << fib(50, cache) << std::endl;
    return 0;
}
```

```
12586269025
```

c

```
#include <stdio.h>

typedef unsigned long long ulong;
unsigned long long fib(ulong n, ulong *cache) {
    if (n < 2) {
        return n;
    }
    // Warning -- this can result in buffer overflow
    if (cache[n] != -1) {
        return cache[n];
    }
    cache[n] = fib(n-1, cache) + fib(n-2, cache);
    return cache[n];
}

int main() {
    ulong cache[101];
    for (ulong i = 0; i < 101; i++) {
        cache[i] = -1;
    }
    printf("%llu\n", fib(50, cache));
    return 0;
}
```

```
12586269025
```

bash

```
fib() {
    local n=$1
    if [ $n -lt 2 ]; then
        echo $n
        return
    fi
    local a=$(fib $((n-1)))
    local b=$(fib $((n-2)))
    echo $((a+b))
}
# Not memoized, so use a much smaller number
fib 10
```

```
<bash is not supported on Windows>
```

js

```
var cache = {};
function fib(n) {
    if (n < 2) {
        return n;
    }
    if (cache[n] !== undefined) {
        return cache[n];
    }
    cache[n] = fib(n-1) + fib(n-2);
    return cache[n];
}
fib(50);
```

```
12586269025
```

r

```
fib <- local({
  memory <- list()
  function(x) {
    valueName <- as.character(x)
    if (!is.null(memory[[valueName]])) return(memory[[valueName]])
    if (x < 2) return(x)
    res <- Recall(x - 1) + Recall(x - 2)
    memory[[valueName]] <<- res # store results
    res
  }
})
print(fib(50))
```

```
[1] 12586269025
```

The execution environment persists across code blocks:

```
print(fib(25))
```

```
[1] 75025
```