

```
def pascal_triangle(n)
```

```
    if n == 1
```

```
        return [[1]]
```

```
    end
```

pascal_triangle(4)

Result is being memoized.
Memoized, saving recursive
calls.

```
    result = pascal_triangle(n-1)
```

```
    new_row = [1]
```

```
    previous_row = result[-1]
```

result = [[1], [1, 1], [1, 2, 1]]

```
    (0... previous_row.length-1).each_with_index do |idx|
```

```
        sum = 0
```

```
        sum = result[-1][idx] + result[-1][idx+1]
```

```
        new_row.push(sum)
```

```
    end
```

```
    new_row.push(1)
```

```
    result.push(new_row)
```

```
end
```

result = [[1], [1, 1], [1, 2, 1], [1, 3, 3, 1]]

def pascal_triangle(n)
 if n == 1
 return [[1]]
 end

pascal_triangle(3)

result = pascal_triangle(n-1)

new_row = [1]

previous_row = result[-1]

result = [[1], [1, 1]]

(0... previous_row.length-1).each_with_index do |idx|

sum = 0

sum = result[-1][idx] + result[-1][idx+1]

new_row.push(sum)

end

new_row.push(1)

result.push(new_row)

end

result = [[1], [1, 1], [1, 2, 1]]

~~def~~ pascal-triangle(n)
if n==1
return [[1]]
end

result = pascal-triangle(n-1)

new-row = [1]

previous-row = result[-1]

result = [[1]]

(0... previous-row.length-1).each_with_index do |idx|

sum = 0

sum = result[-1][idx] + result[-1][idx+1]

new-row.push(sum)

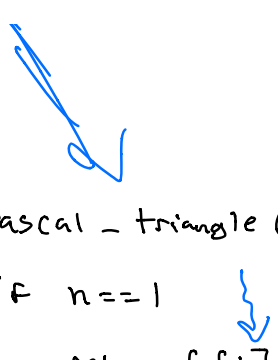
end

new-row.push(1)

result.push(new-row)

result = [[1], [1, 1]]

end



```
def pascal-triangle (n)
  if n == 1
    return [1]
  end
```

pascal-triangle (1)

```
result = pascal-triangle (n-1)
```

```
new-row = [1]
```

```
previous-row = result[-1]
```

```
(0... previous-row.length-1).each_with_index do |idx|
```

```
  sum = 0
```

```
  sum = result[-1][idx] + result[-1][idx+1]
```

```
  new-row.push(sum)
```

```
end
```

```
new-row.push(1)
```

```
result.push(new-row)
```

```
end
```