Estructuras triangulares
(un tema del curso "Álgebra Lineal Numérica")

Egor Maximenko

http://www.egormaximenko.com

Instituto Politécnico Nacional
Escuela Superior de Física y Matemáticas
México

12 de marzo de 2021

Objetivo:

programar ciclos dobles que recorran los pares (j, k) con $0 \leq k \leq j < n$.

Objetivo:

programar ciclos dobles que recorran los pares (j, k) con $0 \leq k \leq j < n$.

Prerrequisitos:

- experiencia de trabajar con matrices triangulares en cursos de álgebra lineal,
- experiencia de programar ciclos anidados,
- la fórmula para la suma de la progresión aritmética.

Ideas similares aparecen también:

- en el cálculo de integrales dobles sobre dominios triangulares,
- en algoritmos triviales del ordenamiento de arreglos (burbuja).

Problema: generar los pares triangulares inferiores

Vamos a escribir una función de un argumento n que genere todos los pares

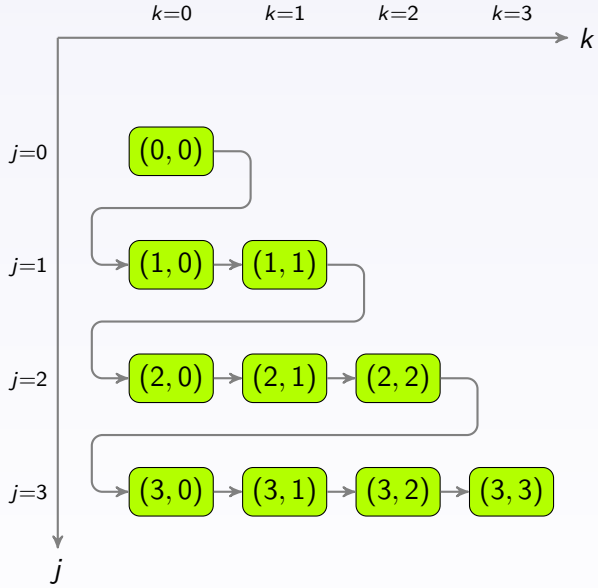
$$(j, k), \qquad j, k \in \mathbb{Z}, \quad 0 \leq k \leq j < n.$$

Por ejemplo, para $n = 3$, la función debe devolver la lista

```
[(0, 0), (1, 0), (1, 1), (2, 0), (2, 1), (2, 2)].
```

Estos pares se pueden representar como las posiciones de los elementos de la parte triangular inferior de una matriz:

$$\begin{array}{lll} (0,0), & & \\ (1,0), & (1,1), & \\ (2,0), & (2,1), & (2,2). \end{array}$$


k=0
k=1
k=2
k=3
k
j=0
j=1
j=2
j=3
j
(0, 0)
(1, 0)
(1, 1)
(2, 0)
(2, 1)
(2, 2)
(3, 0)
(3, 1)
(3, 2)
(3, 3)

Agrupamos pasos elementales en ciclos

```
def triang3_v0():
    r = []
    #
    r.append((0, 0))
    #
    r.append((1, 0))
    r.append((1, 1))
    #
    r.append((2, 0))
    r.append((2, 1))
    r.append((2, 2))
    return r
```

Agrupamos pasos elementales en ciclos

```
def triang3_v0():
    r = []
    #
    r.append((0, 0))
    #
    r.append((1, 0))
    r.append((1, 1))
    #
    r.append((2, 0))
    r.append((2, 1))
    r.append((2, 2))
    return r
```

```
def triang3_v1():
    r = []
    for k in range(1):
        r.append((0, k))
    for k in range(2):
        r.append((1, k))
    for k in range(3):
        r.append((2, k))
    return r
```

Agrupamos pasos elementales en ciclos

```
def triang3_v1():
    r = []
    for k in range(1):
        r.append((0, k))
    for k in range(2):
        r.append((1, k))
    for k in range(3):
        r.append((2, k))
    return r
```

Agrupamos pasos elementales en ciclos

```
def triang3_v1():
    r = []
    for k in range(1):
        r.append((0, k))
    for k in range(2):
        r.append((1, k))
    for k in range(3):
        r.append((2, k))
    return r
```

```
def triang3_v2():
    r = []
    j = 0
    for k in range(j + 1):
        r.append((j, k))
    j = 1
    for k in range(j + 1):
        r.append((j, k))
    j = 2
    for k in range(j + 1):
        r.append((j, k))
    return r
```

Agrupamos pasos elementales en ciclos

```
def triang3_v2():
    r = []
    j = 0
    for k in range(j + 1):
        r.append((j, k))
    j = 1
    for k in range(j + 1):
        r.append((j, k))
    j = 2
    for k in range(j + 1):
        r.append((j, k))
    return r
```

Agrupamos pasos elementales en ciclos

```
def triang3_v2():
    r = []
    j = 0
    for k in range(j + 1):
        r.append((j, k))
    j = 1
    for k in range(j + 1):
        r.append((j, k))
    j = 2
    for k in range(j + 1):
        r.append((j, k))
    return r
```

```
def triang3_v3():
    r = []
    for j in range(3):
        for k in range(j + 1):
            r.append((j, k))
    return r
```

Extendemos al parámetro *n* general

```
def triang3_v3():
    r = []
    for j in range(3):
        for k in range(j + 1):
            r.append((j, k))
    return r
```

Extendemos al parámetro *n* general

```
def triang3_v3():
    r = []
    for j in range(3):
        for k in range(j + 1):
            r.append((j, k))
    return r
```

```
def triang(n):
    r = []
    for j in range(n):
        for k in range(j + 1):
            r.append((j, k))
    return r
```

Solución

```
def triang(n):
    r = []
    for j in range(n):
        for k in range(j + 1):
            r.append((j, k))
    return r
```

Solución

```
def triang(n):
    r = []
    for j in range(n):
        for k in range(j + 1):
            r.append((j, k))
    return r
```

Una solución más pitónica:

```
def ord_pairs(n):
    return [(j, k) for j in range(n) for k in range(j + 1)]
```

Solución

```
def triang(n):
    r = []
    for j in range(n):
        for k in range(j + 1):
            r.append((j, k))
    return r
```

Solución

```
def triang(n):
    r = []
    for j in range(n):
        for k in range(j + 1):
            r.append((j, k))
    return r
```

Un análogo matemático:

$$\{(j,k):\ 0 \le k \le j < n\} = \{(j,k):\quad 0 \le j < n \quad \wedge \quad 0 \le k \le j\}$$

$$= \bigcup_{j=0}^{n-1} \{(j,k):\ 0 \le k \le j\} = \bigcup_{j=0}^{n-1} \bigcup_{k=0}^{j} \{(j,k)\}.$$

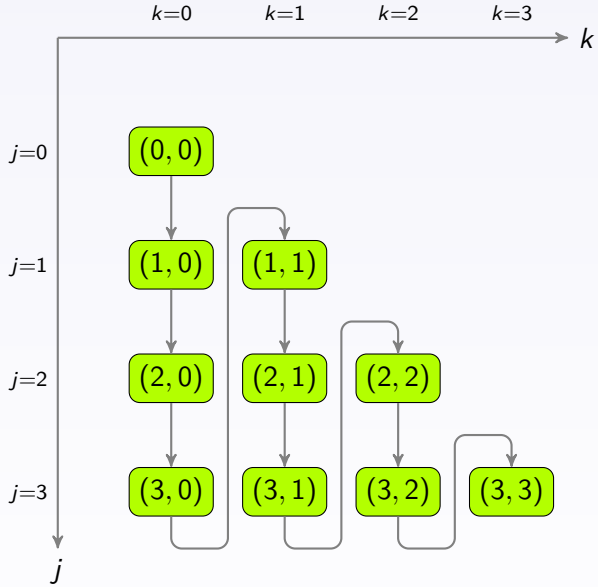
Problema: generar los pares triangulares inferiores en otro orden

Ahora vamos a generar los mismos pares en otro orden:

```
[(0, 0), (1, 0), (2, 0), (1, 1), (2, 1), (2, 2)].
```

Recorremos la parte inferior izquierda de la matriz por columnas.

$$
\begin{array}{lll}
(0,0) & & \\
(1,0) & (1,1) & \\
(2,0) & (2,1) & (2,2).
\end{array}
$$


k=0
k=1
k=2
k=3
k
j=0
j=1
j=2
j=3
j
(0, 0)
(1, 0)
(1, 1)
(2, 0)
(2, 1)
(2, 2)
(3, 0)
(3, 1)
(3, 2)
(3, 3)

Agrupamos pasos elementales en ciclos

```
def triangcol_3_v0():
    r = []
    #
    r.append((0, 0))
    r.append((1, 0))
    r.append((2, 0))
    #
    r.append((1, 1))
    r.append((2, 1))
    #
    r.append((2, 2))
    return r
```

Agrupamos pasos elementales en ciclos

```
def triangcol_3_v0():
    r = []
    #
    r.append((0, 0))
    r.append((1, 0))
    r.append((2, 0))
    #
    r.append((1, 1))
    r.append((2, 1))
    #
    r.append((2, 2))
    return r
```

```
def triangcol3_v1():
    r = []
    for j in range(0, 3):
        r.append((j, 0))
    for j in range(1, 3):
        r.append((j, 1))
    for j in range(2, 3):
        r.append((j, 2))
    return r
```

Agrupamos pasos elementales en ciclos

```
def triangcol3_v1():
    r = []
    for j in range(0, 3):
        r.append((j, 0))
    for j in range(1, 3):
        r.append((j, 1))
    for j in range(2, 3):
        r.append((j, 2))
    return r
```

Agrupamos pasos elementales en ciclos

```
def triangcol3_v1():
    r = []
    for j in range(0, 3):
        r.append((j, 0))
    for j in range(1, 3):
        r.append((j, 1))
    for j in range(2, 3):
        r.append((j, 2))
    return r
```

```
def triangcol3_v2():
    r = []
    k = 0
    for j in range(k, 3):
        r.append((j, k))
    k = 1
    for j in range(k, 3):
        r.append((j, k))
    k = 2
    for j in range(k, 3):
        r.append((j, k))
    return r
```

Agrupamos pasos elementales en ciclos

```
def triangcol3_v2():
    r = []
    k = 0
    for j in range(k, 3):
        r.append((j, k))
    k = 1
    for j in range(k, 3):
        r.append((j, k))
    k = 2
    for j in range(k, 3):
        r.append((j, k))
    return r
```

Agrupamos pasos elementales en ciclos

```
def triangcol3_v2():
    r = []
    k = 0
    for j in range(k, 3):
        r.append((j, k))
    k = 1
    for j in range(k, 3):
        r.append((j, k))
    k = 2
    for j in range(k, 3):
        r.append((j, k))
    return r
```

```
def triangcol3_v3():
    r = []
    for k in range(3):
        for j in range(k, 3):
            r.append((j, k))
    return r
```

Extendemos al parámetro *n* general

```
def triangcol3_v3():
    r = []
    for k in range(3):
        for j in range(k, 3):
            r.append((j, k))
    return r
```

Extendemos al parámetro *n* general

```
def triangcol3_v3():
    r = []
    for k in range(3):
        for j in range(k, 3):
            r.append((j, k))
    return r
```

```
def triangcol(n):
    r = []
    for k in range(n):
        for j in range(k, n):
            r.append((j, k))
    return r
```

Solución

```
def triangcol(n):
    r = []
    for k in range(n):
        for j in range(k, n):
            r.append((j, k))
    return r
```

Solución

```
def triangcol(n):
    r = []
    for k in range(n):
        for j in range(k, n):
            r.append((j, k))
    return r
```

Una solución más pitónica:

```
def ord_pairs_col(n):
    return [(j, k) for k in range(n) for j in range(k, n)]
```

```
def triangcol(n):
    r = []
    for k in range(n):
        for j in range(k, n):
            r.append((j, k))
    return r
```

```
def triangcol(n):
    r = []
    for k in range(n):
        for j in range(k, n):
            r.append((j, k))
    return r
```

Un análogo matemático:

$$\{(j,k)\colon\ 0 \le k \le j < n\} = \{(j,k)\colon\quad 0 \le k < n \quad\wedge\quad k \le j < n\}$$

$$= \bigcup_{k=0}^{n-1} \{(j,k)\colon\ k \le j < n\} = \bigcup_{k=0}^{n-1} \bigcup_{j=k}^{n-1} \{(j,k)\}.$$

Contamos los pares (j, k), $0 \leq k \leq j < n$, sumando una progresión aritmética

Para cada j fijo, el número de elementos (j, k) con $0 \leq k \leq j$ es $j + 1$.

Contamos los pares (j, k), $0 \leq k \leq j < n$, sumando una progresión aritmética

Para cada j fijo, el número de elementos (j, k) con $0 \leq k \leq j$ es $j + 1$.

El número total de elementos es

$$\sum_{j=0}^{n-1}(j+1) \xrightarrow{s=j+1} \sum_{s=1}^{n} s = \frac{n(n+1)}{2}.$$

Contamos los pares (j, k), $0 \leq k \leq j < n$, usando la simetría

$$
\begin{array}{cccc}
(0,0) & (0,1) & (0,2) & (0,3) \\
(1,0) & (1,1) & (1,2) & (1,3) \\
(2,0) & (2,1) & (2,2) & (2,3) \\
(3,0) & (3,1) & (3,2) & (3,3)
\end{array}
$$

Contamos los pares (j, k), $0 \leq k \leq j < n$, usando la simetría

$$
\begin{array}{cccc}
(0,0) & (0,1) & (0,2) & (0,3) \\
(1,0) & (1,1) & (1,2) & (1,3) \\
(2,0) & (2,1) & (2,2) & (2,3) \\
(3,0) & (3,1) & (3,2) & (3,3)
\end{array}
$$

$$C_n \coloneqq \{(j,k) \in \mathbb{Z}^2 \colon \ 0 \leq j, k < n\}$$

Contamos los pares (j, k), $0 \leq k \leq j < n$, usando la simetría

$$
\begin{array}{cccc}
(0,0) & (0,1) & (0,2) & (0,3) \\
(1,0) & (1,1) & (1,2) & (1,3) \\
(2,0) & (2,1) & (2,2) & (2,3) \\
(3,0) & (3,1) & (3,2) & (3,3)
\end{array}
$$

$$
\begin{aligned}
C_n &\coloneqq \{(j,k) \in \mathbb{Z}^2 \colon\ 0 \leq j, k < n\} \\
&= \{0, 1, \ldots, n-1\}^2.
\end{aligned}
$$

Contamos los pares (j, k), $0 \leq k \leq j < n$, usando la simetría

$$
\begin{array}{cccc}
(0,0) & (0,1) & (0,2) & (0,3) \\
(1,0) & (1,1) & (1,2) & (1,3) \\
(2,0) & (2,1) & (2,2) & (2,3) \\
(3,0) & (3,1) & (3,2) & (3,3)
\end{array}
$$

$$
\begin{aligned}
C_n &\coloneqq \{(j,k) \in \mathbb{Z}^2 \colon\ 0 \leq j, k < n\} \\
&= \{0, 1, \ldots, n-1\}^2.
\end{aligned}
$$

$$
\# C_n = n^2.
$$

Contamos los pares (j, k), $0 \leq k \leq j < n$, usando la simetría

$(0,0)$	$(0,1)$	$(0,2)$	$(0,3)$
$(1,0)$	$(1,1)$	$(1,2)$	$(1,3)$
$(2,0)$	$(2,1)$	$(2,2)$	$(2,3)$
$(3,0)$	$(3,1)$	$(3,2)$	$(3,3)$

$$\#\{(j,k) \in C_n\colon \quad j = k\} \; =$$

$$\begin{aligned} C_n &\coloneqq \{(j,k) \in \mathbb{Z}^2\colon \; 0 \leq j, k < n\} \\ &= \{0, 1, \ldots, n-1\}^2. \end{aligned}$$

$$\# C_n = n^2.$$

Contamos los pares (j, k), $0 \leq k \leq j < n$, usando la simetría

$(0,0)$	$(0,1)$	$(0,2)$	$(0,3)$
$(1,0)$	$(1,1)$	$(1,2)$	$(1,3)$
$(2,0)$	$(2,1)$	$(2,2)$	$(2,3)$
$(3,0)$	$(3,1)$	$(3,2)$	$(3,3)$

$$\#\{(j,k) \in C_n \colon \quad j = k\} \;=\; n,$$

$$\begin{aligned} C_n &\coloneqq \{(j,k) \in \mathbb{Z}^2 \colon \; 0 \leq j, k < n\} \\ &= \{0, 1, \ldots, n-1\}^2. \end{aligned}$$

$$\# C_n = n^2.$$

Contamos los pares (j, k), $0 \leq k \leq j < n$, usando la simetría

$(0,0)$	$(0,1)$	$(0,2)$	$(0,3)$
$(1,0)$	$(1,1)$	$(1,2)$	$(1,3)$
$(2,0)$	$(2,1)$	$(2,2)$	$(2,3)$
$(3,0)$	$(3,1)$	$(3,2)$	$(3,3)$

$$\#\{(j,k) \in C_n\colon \quad j = k\} \;=\; n,$$

$$\#\{(j,k) \in C_n\colon \quad j \neq k\} \;=$$

$$C_n \coloneqq \{(j,k) \in \mathbb{Z}^2\colon \; 0 \leq j, k < n\}$$
$$= \{0, 1, \ldots, n-1\}^2.$$

$$\# C_n = n^2.$$

Contamos los pares (j, k), $0 \leq k \leq j < n$, usando la simetría

$(0,0)$	$(0,1)$	$(0,2)$	$(0,3)$
$(1,0)$	$(1,1)$	$(1,2)$	$(1,3)$
$(2,0)$	$(2,1)$	$(2,2)$	$(2,3)$
$(3,0)$	$(3,1)$	$(3,2)$	$(3,3)$

$$\#\{(j,k) \in C_n\colon \quad j = k\} \;=\; n,$$

$$\#\{(j,k) \in C_n\colon \quad j \neq k\} \;=\; n^2 - n,$$

$$\begin{aligned} C_n &\coloneqq \{(j,k) \in \mathbb{Z}^2\colon\ 0 \leq j, k < n\} \\ &= \{0, 1, \ldots, n-1\}^2. \end{aligned}$$

$$\# C_n = n^2.$$

Contamos los pares (j,k), $0 \leq k \leq j < n$, usando la simetría

$(0,0)$	$(0,1)$	$(0,2)$	$(0,3)$
$(1,0)$	$(1,1)$	$(1,2)$	$(1,3)$
$(2,0)$	$(2,1)$	$(2,2)$	$(2,3)$
$(3,0)$	$(3,1)$	$(3,2)$	$(3,3)$

$$C_n \coloneqq \{(j,k) \in \mathbb{Z}^2 \colon\ 0 \leq j,k < n\}$$
$$= \{0,1,\ldots,n-1\}^2.$$

$$\# C_n = n^2.$$

$$\#\{(j,k) \in C_n \colon \quad j = k\} \ = \ n,$$

$$\#\{(j,k) \in C_n \colon \quad j \neq k\} \ = \ n^2 - n,$$

$$\#\{(j,k) \in C_n \colon \quad j < k\} \ =$$

$$\#\{(j,k) \in C_n \colon \quad j > k\} \ =$$

Contamos los pares (j,k), $0 \leq k \leq j < n$, usando la simetría

$(0,0)$	$(0,1)$	$(0,2)$	$(0,3)$
$(1,0)$	$(1,1)$	$(1,2)$	$(1,3)$
$(2,0)$	$(2,1)$	$(2,2)$	$(2,3)$
$(3,0)$	$(3,1)$	$(3,2)$	$(3,3)$

$$\begin{aligned} C_n &\coloneqq \{(j,k) \in \mathbb{Z}^2 \colon\ 0 \leq j,k < n\} \\ &= \{0,1,\ldots,n-1\}^2. \end{aligned}$$

$$\# C_n = n^2.$$

$$\begin{aligned} \#\{(j,k) \in C_n \colon\quad j = k\} &= n, \\ \#\{(j,k) \in C_n \colon\quad j \neq k\} &= n^2 - n, \\ \#\{(j,k) \in C_n \colon\quad j < k\} &= \frac{n^2-n}{2}, \\ \#\{(j,k) \in C_n \colon\quad j > k\} &= \frac{n^2-n}{2}, \end{aligned}$$

Contamos los pares (j, k), $0 \leq k \leq j < n$, usando la simetría

$(0,0)$	$(0,1)$	$(0,2)$	$(0,3)$
$(1,0)$	$(1,1)$	$(1,2)$	$(1,3)$
$(2,0)$	$(2,1)$	$(2,2)$	$(2,3)$
$(3,0)$	$(3,1)$	$(3,2)$	$(3,3)$

$$\begin{aligned} C_n &\coloneqq \{(j,k) \in \mathbb{Z}^2 \colon\ 0 \leq j, k < n\} \\ &= \{0, 1, \ldots, n-1\}^2. \end{aligned}$$

$$\# C_n = n^2.$$

$$\begin{aligned} \#\{(j,k) \in C_n \colon\quad j = k\} &= n, \\ \#\{(j,k) \in C_n \colon\quad j \neq k\} &= n^2 - n, \\ \#\{(j,k) \in C_n \colon\quad j < k\} &= \frac{n^2 - n}{2}, \\ \#\{(j,k) \in C_n \colon\quad j > k\} &= \frac{n^2 - n}{2}, \\ \#\{(j,k) \in C_n \colon\quad j \leq k\} &= \end{aligned}$$

Contamos los pares (j, k), $0 \leq k \leq j < n$, usando la simetría

$(0,0)$	$(0,1)$	$(0,2)$	$(0,3)$
$(1,0)$	$(1,1)$	$(1,2)$	$(1,3)$
$(2,0)$	$(2,1)$	$(2,2)$	$(2,3)$
$(3,0)$	$(3,1)$	$(3,2)$	$(3,3)$

$$\begin{aligned} C_n &\coloneqq \{(j,k) \in \mathbb{Z}^2 \colon\ 0 \leq j, k < n\} \\ &= \{0, 1, \ldots, n-1\}^2. \end{aligned}$$

$$\# C_n = n^2.$$

$$\begin{aligned} \#\{(j,k) \in C_n \colon\ \ j = k\} &= n, \\ \#\{(j,k) \in C_n \colon\ \ j \neq k\} &= n^2 - n, \\ \#\{(j,k) \in C_n \colon\ \ j < k\} &= \frac{n^2 - n}{2}, \\ \#\{(j,k) \in C_n \colon\ \ j > k\} &= \frac{n^2 - n}{2}, \\ \#\{(j,k) \in C_n \colon\ \ j \leq k\} &= \frac{n^2 - n}{2} + n \end{aligned}$$

Contamos los pares (j,k), $0 \le k \le j < n$, usando la simetría

$(0,0)$	$(0,1)$	$(0,2)$	$(0,3)$
$(1,0)$	$(1,1)$	$(1,2)$	$(1,3)$
$(2,0)$	$(2,1)$	$(2,2)$	$(2,3)$
$(3,0)$	$(3,1)$	$(3,2)$	$(3,3)$

$$\begin{aligned} C_n &\coloneqq \{(j,k) \in \mathbb{Z}^2 \colon\ 0 \le j,k < n\} \\ &= \{0,1,\dots,n-1\}^2. \end{aligned}$$

$$\# C_n = n^2.$$

$$\begin{aligned} \#\{(j,k) \in C_n \colon\quad j = k\} &= n, \\ \#\{(j,k) \in C_n \colon\quad j \ne k\} &= n^2 - n, \\ \#\{(j,k) \in C_n \colon\quad j < k\} &= \frac{n^2-n}{2}, \\ \#\{(j,k) \in C_n \colon\quad j > k\} &= \frac{n^2-n}{2}, \\ \#\{(j,k) \in C_n \colon\quad j \le k\} &= \frac{n^2-n}{2} + n \\ &= \frac{n^2+n}{2}. \end{aligned}$$

Creación de matrices triangulares inferiores, ejemplo 1

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 \end{bmatrix}$$

Creación de matrices triangulares inferiores, ejemplo 1

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 \end{bmatrix}$$

```
from numpy import *

def tri_matr_1(n):
    A = zeros((n, n))
    for j in range(n):
        for k in range(j + 1):
            A[j, k] = 1
    return A
```

Creación de matrices triangulares inferiores, ejemplo 2

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 2 & 2 & 0 & 0 \\ 3 & 3 & 3 & 0 \\ 4 & 4 & 4 & 4 \end{bmatrix}$$

Creación de matrices triangulares inferiores, ejemplo 2

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 2 & 2 & 0 & 0 \\ 3 & 3 & 3 & 0 \\ 4 & 4 & 4 & 4 \end{bmatrix}$$

```
def tri_matr_2(n):
    A = zeros((n, n))
    for j in range(n):
        for k in range(j + 1):
            A[j, k] = j + 1
    return A
```

Creación de matrices triangulares inferiores, ejemplo 2

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 2 & 2 & 0 & 0 \\ 3 & 3 & 3 & 0 \\ 4 & 4 & 4 & 4 \end{bmatrix}$$

Creación de matrices triangulares inferiores, ejemplo 2

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 2 & 2 & 0 & 0 \\ 3 & 3 & 3 & 0 \\ 4 & 4 & 4 & 4 \end{bmatrix}$$

Otra solución:

```
def tri_matr_2_row(n):
    A = zeros((n, n))
    for j in range(n):
        A[j, 0 : j + 1] = j + 1
    return A
```

Creación de matrices triangulares inferiores, ejemplo 3

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 2 & 0 & 0 \\ 1 & 2 & 3 & 0 \\ 1 & 2 & 3 & 4 \end{bmatrix}$$

Creación de matrices triangulares inferiores, ejemplo 3

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 2 & 0 & 0 \\ 1 & 2 & 3 & 0 \\ 1 & 2 & 3 & 4 \end{bmatrix}$$

```
def tri_matr_3(n):
    A = zeros((n, n))
    for k in range(n):
        for j in range(k, n):
            A[j, k] = k + 1
    return A
```

Creación de matrices triangulares inferiores, ejemplo 3

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 2 & 0 & 0 \\ 1 & 2 & 3 & 0 \\ 1 & 2 & 3 & 4 \end{bmatrix}$$

Creación de matrices triangulares inferiores, ejemplo 3

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 2 & 0 & 0 \\ 1 & 2 & 3 & 0 \\ 1 & 2 & 3 & 4 \end{bmatrix}$$

Otra solución:

```
def tri_matr_3_col(n):
    A = zeros((n, n))
    for k in range(n):
        A[0 : k, k] = k + 1
    return A
```

Creación de matrices triangulares inferiores, ejemplo 4

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 2 & 3 & 0 & 0 \\ 3 & 4 & 5 & 0 \\ 4 & 5 & 6 & 7 \end{bmatrix}$$

Creación de matrices triangulares inferiores, ejemplo 4

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 2 & 3 & 0 & 0 \\ 3 & 4 & 5 & 0 \\ 4 & 5 & 6 & 7 \end{bmatrix}$$

```
def tri_matr_4(n):
    A = zeros((n, n))
    for j in range(n):
        for k in range(j + 1):
            A[j, k] = j + k + 1
    return A
```

Sacar la parte triangular interior de una matriz

$$\begin{bmatrix} A_{1,1} & A_{1,2} & A_{1,3} & A_{1,4} \\ A_{2,1} & A_{2,2} & A_{2,3} & A_{2,4} \\ A_{3,1} & A_{3,2} & A_{3,3} & A_{3,4} \\ A_{4,1} & A_{4,2} & A_{4,3} & A_{4,4} \end{bmatrix} \mapsto \begin{bmatrix} A_{1,1} & 0 & 0 & 0 \\ A_{2,1} & A_{2,2} & 0 & 0 \\ A_{3,1} & A_{3,2} & A_{3,3} & 0 \\ A_{4,1} & A_{4,2} & A_{4,3} & A_{4,4} \end{bmatrix}.$$

Sacar la parte triangular interior de una matriz

$$\begin{bmatrix} A_{1,1} & A_{1,2} & A_{1,3} & A_{1,4} \\ A_{2,1} & A_{2,2} & A_{2,3} & A_{2,4} \\ A_{3,1} & A_{3,2} & A_{3,3} & A_{3,4} \\ A_{4,1} & A_{4,2} & A_{4,3} & A_{4,4} \end{bmatrix} \mapsto \begin{bmatrix} A_{1,1} & 0 & 0 & 0 \\ A_{2,1} & A_{2,2} & 0 & 0 \\ A_{3,1} & A_{3,2} & A_{3,3} & 0 \\ A_{4,1} & A_{4,2} & A_{4,3} & A_{4,4} \end{bmatrix}.$$

```
def mytril(A):
    n = a.shape[0]; B = zeros((n, n))
    for j in range(n):
        for k in range(j + 1):
            B[j, k] = A[j, k]
    return B
```

Ejercicio.

Hacer cosas similares con estructuras triangulares superiores:

$$\{(j,k):\ 0 \le j \le k < n\}.$$

$$\begin{array}{ccc} (0,0), & (0,1), & (0,2), \\ & (1,1), & (1,2), \\ & & (2,2). \end{array}$$

Ejercicio.

Hacer cosas similares con estructuras triangulares superiores:

$$\{(j,k):\ 0 \leq j \leq k < n\}.$$

$$\begin{array}{ccc} (0,0), & (0,1), & (0,2), \\ & (1,1), & (1,2), \\ & & (2,2). \end{array}$$

Ejercicio.

Construir sus propios ejemplos de matrices que tengan una estructura triangular.