

# Guía de Ejercicios Python para I. O.

Investigación Operativa, Universidad de San Andrés

Si encuentran algún error en el documento o hay alguna duda, mandenme un mail a [rodriguezf@udesa.edu.ar](mailto:rodriguezf@udesa.edu.ar) y lo revisamos.

## 1. Ejercicios

### 1.1. Ejercicio 1

Escribir un código que imprima en la consola las siguientes frases o el resultado de las operaciones matemáticas:

- a) “Alo mundo!”
- b)  $2 + 3$
- c)  $2 * 3$
- d)  $2^3$
- e)  $\frac{2}{3}$
- f) Resto de la división  $2/3$

### 1.2. Ejercicio 2

Escribir un código que imprima todos los números pares entre 0 y 31, utilizando **for** loops.

### 1.3. Ejercicio 3

Escribir un código que compute el promedio de la lista de números  $[1, 32, 53, 14, 55, 36, 27]$ . Hacerlo de dos maneras distintas:

- a) Mediante for loops
- b) Usando la función `np.mean( )`

## 1.4. Ejercicio 4

Escribir una función que tome como input dos números  $x_1, x_2$  e imprima a la consola la suma de esos dos números.

## 1.5. Ejercicio 5

- a) Importar la librería numpy con el comando “`import numpy as np`”
- b) Considerar la matriz

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

- c) Transformar en un array de numpy con el comando “`A = np.array(A)`”
- d) Corroborar que los comandos `A[0][0]` y `A[0,0]` devuelven el primer elemento en la primera fila, y primera columna.
- e) Corroborar que el comando `A[:,1]` devuelve la segunda columna.
- f) Corroborar que los comandos `A[1]` y `A[1,:]` devuelven la segunda fila.
- g) Corroborar que el comando `A[:, -1]` devuelve la última columna.
- h) Corroborar que `A[0:2]` y `A[:,0:2]` devuelven las primeras dos filas y las primeras dos columnas respectivamente.
- i) ¿Qué devuelven los comandos `A[-1, -1]`, `A[0:2, 0]`, `A[0:2, 0:2]`?

## 1.6. Ejercicio 6

- a) Escribir un for loop que dadas las dos listas:

$$A = [2, 10, 16, 2, 4, 12, 24, 100]$$

$$B = [5, 2, 5, 2, 1, 2, 1, 0,5]$$

sume la multiplicación coordenada a coordenada de todos sus elementos, es decir:

$$A[0] * B[0] + A[1] * B[1] + A[2] * B[2] + \dots = 2 * 5 + 10 * 2 + 16 * 5 \dots$$

**Nota:** Esta operación es llamada producto interno entre dos vectores o listas.

- b) Realizar la cuenta a mano y verificar que el resultado es el mismo que en Python.
- c) Importar la librería numpy y transformar ambas listas en arrays de numpy usando `np.array`
- d) Corroborar que ahora el comando “`np.dot(A, B)`” da el mismo resultado que en (a) y (b).

## 1.7. Ejercicio 7

- a) Considerar las matrices

$$A = \begin{bmatrix} 1 & 2 & 0 \\ 3 & 0 & 4 \\ 1 & 0 & 3 \end{bmatrix}, \quad B = \begin{bmatrix} 3 & 1 & 1 \\ 1 & 2 & 0 \\ 2 & 4 & 1 \end{bmatrix}$$

- b) Transformarlas en arrays de numpy **Definición:** Se define a la multiplicación de matrices  $A * B$  como la matriz que en la coordenada  $i, j$  (fila  $i$ , columna  $j$ ) tiene al producto interno de la fila  $i$  de  $A$  con la columna  $j$  de  $B$ . En este caso la matriz  $A * B$  también es de tamaño  $3 \times 3$  (son 9 operaciones de producto interno).
- c) Realizar a mano la multiplicación de las matrices  $A$  y  $B$  del punto (a).
- d) Hacer un doble **for** loop y utilizar lo aprendido en el ejercicio anterior (el 6) para construir la matriz  $A * B$  en Python. Corroborar que da lo mismo que a mano.
- e) Multiplicar las matrices utilizando el comando abreviado de numpy ‘`A@B`’. Corroborar que también da lo mismo.

## 1.8. Ejercicio 8

Escribir una función que tome como input dos números  $x_1, x_2$ , y determine si están en el conjunto factible definido por las siguientes restricciones lineales. El output debe ser un booleano `True/False`.

$$2x_1 + 3x_2 \leq 24$$

$$x_1 \geq 0$$

$$x_2 \geq 0$$

Usando esta función escribir código que determine si los siguientes puntos están en el conjunto factible.

a)  $(x_1, x_2) = (1, 1)$

b)  $(x_1, x_2) = (12, 8)$

c)  $(x_1, x_2) = (4, 8)$

d)  $(x_1, x_2) = (-1, 0)$

e)  $(x_1, x_2) = (-2, 2)$

## 2. Anexo: Soluciones

### 2.1. Ejercicio 1

```
1 print("Alo mundo!")
2 print(2 + 3)
3 print(2 * 3)
4 print(2 ** 3)
5 print(2 / 3)
6 print(2 % 3)
```

### 2.2. Ejercicio 2

```
1 for i in range(0, 32, 2):
2     print(i)
```

### 2.3. Ejercicio 3

```
1 # a)
2 nums = [1, 32, 53, 14, 55, 36, 27]
3 suma = 0
4 for n in nums:
5     suma += n
6 promedio = suma / len(nums)
7 print(promedio)
8
9 # b)
10 import numpy as np
11 nums = [1, 32, 53, 14, 55, 36, 27]
12 print(np.mean(nums))
```

### 2.4. Ejercicio 4

```
1 def suma(x1, x2):
2     print(x1 + x2)
```

### 2.5. Ejercicio 5

```

1 # a)
2 import numpy as np
3 A = [[1, 2, 0], [3, 0, 4], [1, 0, 3]]
4 A = np.array(A)
5
6 # d)
7 print(A[0][0])
8 print(A[0,0])
9
10 # e)
11 print(A[:,1])
12
13 # f)
14 print(A[1])
15 print(A[1,:])
16
17 # g)
18 print(A[:,-1])
19
20 # h)
21 print(A[0:2])
22 print(A[:,0:2])
23
24 # i)
25 print(A[-1,-1])
26 print(A[0:2,0])
27 print(A[0:2,0:2])

```

## 2.6. Ejercicio 6

```

1 # a)
2 A = [2, 10, 16, 2, 4, 12, 24, 100]
3 B = [5, 2, 5, 2, 1, 2, 1, 0.5]
4 res = 0
5
6 for i in range(len(A)):
7     res += A[i] * B[i]
8 print(res)
9
10 # c)
11 import numpy as np
12
13 A = np.array([2, 10, 16, 2, 4, 12, 24, 100])
14 B = np.array([5, 2, 5, 2, 1, 2, 1, 0.5])
15 print(np.dot(A, B))

```

## 2.7. Ejercicio 7

```
1 # a)
2 import numpy as np
3 A = np.array([[1, 2, 0], [3, 0, 4], [1, 0, 3]])
4 B = np.array([[3, 1, 1], [1, 2, 0], [2, 4, 1]])
5
6 # d)
7 C = np.zeros((3,3))
8 for i in range(3):
9     for j in range(3):
10         C[i,j] = np.dot(A[i,:], B[:,j])
11 print(C)
12
13 # e)
14 print(A @ B)
```

## 2.8. Ejercicio 8

```
1 def factible(x1, x2):
2     return 2*x1 + 3*x2 <= 24 and x1 >= 0 and x2 >= 0
3
4 print(factible(1,1))
5 print(factible(12,8))
6 print(factible(4,8))
7 print(factible(-1,0))
8 print(factible(-2,2))
```