Guía de Ejercicios
Python para I. O.

Investigación Operativa, Universidad de San Andrés

Si encuentran algún error en el documento o hay alguna duda, mandenmé un mail a 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]`, `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:

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

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 3×3 (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) $(x_1, x_2) = (-2, 2)$

1.9. Ejercicio 9

Para los siguientes problemas de optimización encontrar gráficamente el/los punto(s) óptimo(s) y el valor óptimo. Para hacer esto pueden graficar las funciones objetivo en el rango de interés (o sea, que por lo menos contenga el conjunto factible), y visualmente identificar el punto y valor óptimo. Para los casos donde haya más de un óptimo, reportar todos los puntos óptimos que existan. Para los casos en que no existe el óptimo, explicar por qué este es el caso (o sea, si es porque el conjunto factible es nulo, o porque la función objetivo no está acotada).

a)
$$\textbf{min} \quad x^2$$

b)
$$\textbf{min} \quad x^2$$
$$x \geq 2$$

c)
$$\textbf{max} \quad x$$
$$0 \leq x$$

d)
$$\textbf{max} \quad x$$
$$0 \leq x$$
$$x \leq 10$$

e)

$$\begin{aligned} &\textbf{max} && x \\ &x \leq 0 \\ &2 \leq x \end{aligned}$$

f)

$$\begin{aligned} &\textbf{min} && \cos(x) \\ &0 \leq x \\ &x \leq 1 \end{aligned}$$

g)

$$\begin{aligned} &\textbf{min} && \cos(x) \\ &0 \leq x \\ &x \leq 10 \end{aligned}$$

2. Anexo: Soluciones

2.1. Ejercicio 1

```
print("Alo mundo!")
print(2 + 3)
print(2 * 3)
print(2 ** 3)
print(2 / 3)
print(2 % 3)
```

2.2. Ejercicio 2

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

2.3. Ejercicio 3

```
# a)
nums = [1, 32, 53, 14, 55, 36, 27]
suma = 0
for n in nums:
    suma += n
promedio = suma / len(nums)
print(promedio)

# b)
import numpy as np
nums = [1, 32, 53, 14, 55, 36, 27]
print(np.mean(nums))
```

2.4. Ejercicio 4

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

2.5. Ejercicio 5

```python
# a)
import numpy as np
A = [[1, 2, 0], [3, 0, 4], [1, 0, 3]]
A = np.array(A)

# d)
print(A[0][0])
print(A[0,0])

# e)
print(A[:,1])

# f)
print(A[1])
print(A[1,:])

# g)
print(A[:,-1])

# h)
print(A[0:2])
print(A[:,0:2])

# i)
print(A[-1,-1])
print(A[0:2])
print(A[0:2,0])
print(A[0:2,0:2])
```

2.6. Ejercicio 6

```python
# a)
A = [2, 10, 16, 2, 4, 12, 24, 100]
B = [5, 2, 5, 2, 1, 2, 1, 0.5]
res = 0

for i in range(len(A)):
    res += A[i] * B[i]
print(res)

# c)
import numpy as np

A = np.array([2, 10, 16, 2, 4, 12, 24, 100])
B = np.array([5, 2, 5, 2, 1, 2, 1, 0.5])
```

```
print(np.dot(A, B))
```

2.7. Ejercicio 7

```
# a)
import numpy as np
A = np.array([[1, 2, 0], [3, 0, 4], [1, 0, 3]])
B = np.array([[3, 1, 1], [1, 2, 0], [2, 4, 1]])

# d)
C = np.zeros((3,3))
for i in range(3):
    for j in range(3):
        C[i,j] = np.dot(A[i,:], B[:,j])
print(C)

# e)
print(A @ B)
```

2.8. Ejercicio 8

```
def factible(x1, x2):
    return 2*x1 + 3*x2 <= 24 and x1 >= 0 and x2 >= 0

print(factible(1,1))
print(factible(12,8))
print(factible(4,8))
print(factible(-1,0))
print(factible(-2,2))
```

2.9. Ejercicio 9

```
# a)
import numpy as np
import matplotlib.pyplot as plt

x = np.linspace(-5, 5, 200)
plt.plot(x, x**2)
plt.show()

# b)
x = np.linspace(2, 5, 200)
```

```
plt.plot(x, x**2)
plt.show()

# c)
x = np.linspace(0, 5, 200)
plt.plot(x, x)
plt.show()

# d)
x = np.linspace(0, 10, 200)
plt.plot(x, x)
plt.show()

# e)
x = np.linspace(2, 0, 200)
plt.plot(x, x)
plt.show()

# f)
x = np.linspace(0, 1, 200)
plt.plot(x, np.cos(x))
plt.show()

# g)
x = np.linspace(0, 10, 200)
plt.plot(x, np.cos(x))
plt.show()
```