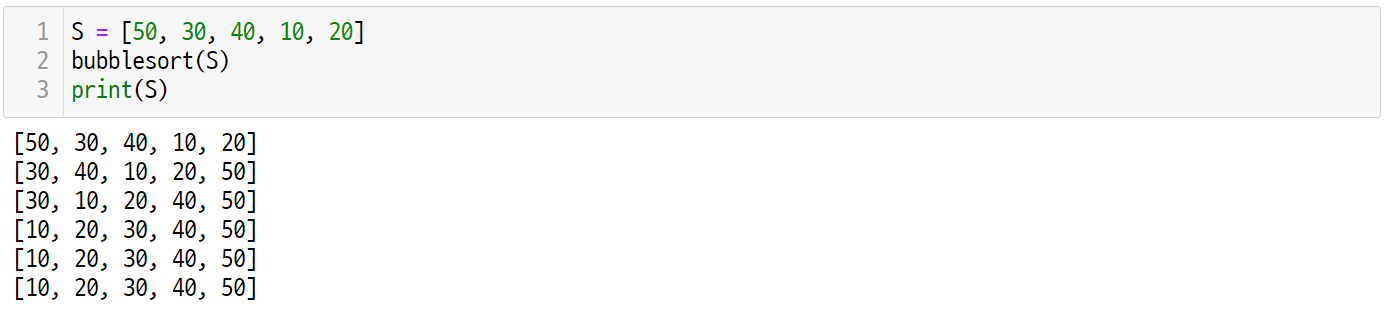
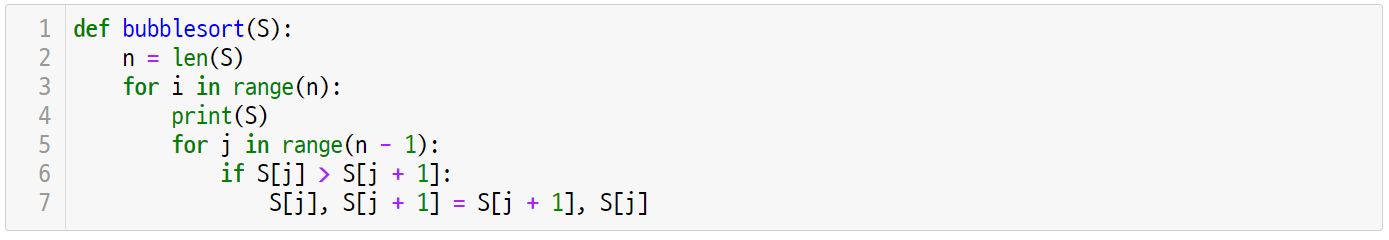
**Codificación y Programación.**

**Quiz Capítulos #5, #6 Y #7.**

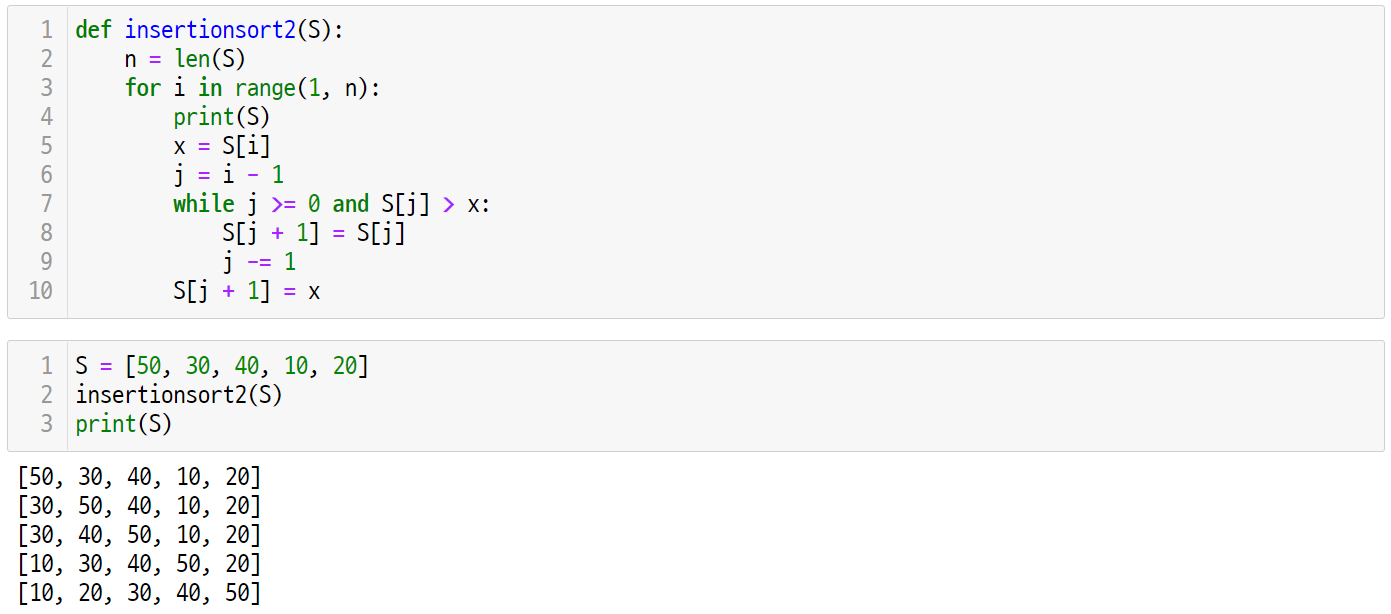
**Q. 05-01.** ¿Cuántas comparaciones se ejecutaron en el siguiente proceso de clasificación por inserción?



R: Una comparación:

if *S*[j] > *S*[j+1]:

**Q. 05-02.** ¿Cuántas comparaciones se ejecutaron en el siguiente proceso de clasificación por inserción?



R: Una comparación:

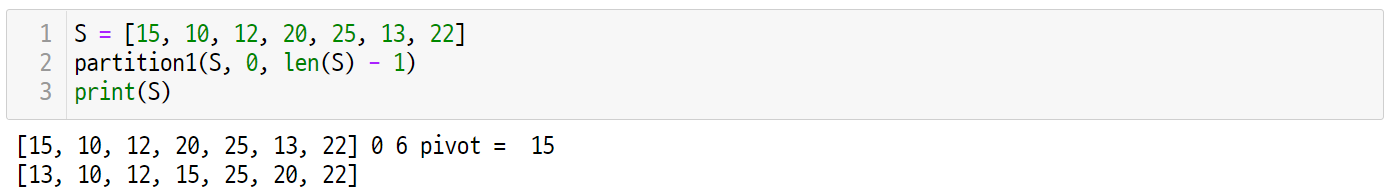
while j >= 0 and *S*[j] > x:

**Q. 05-03.** ¿Cuántas veces se ejecutó la función merge2() en el siguiente proceso de clasificación por fusión?



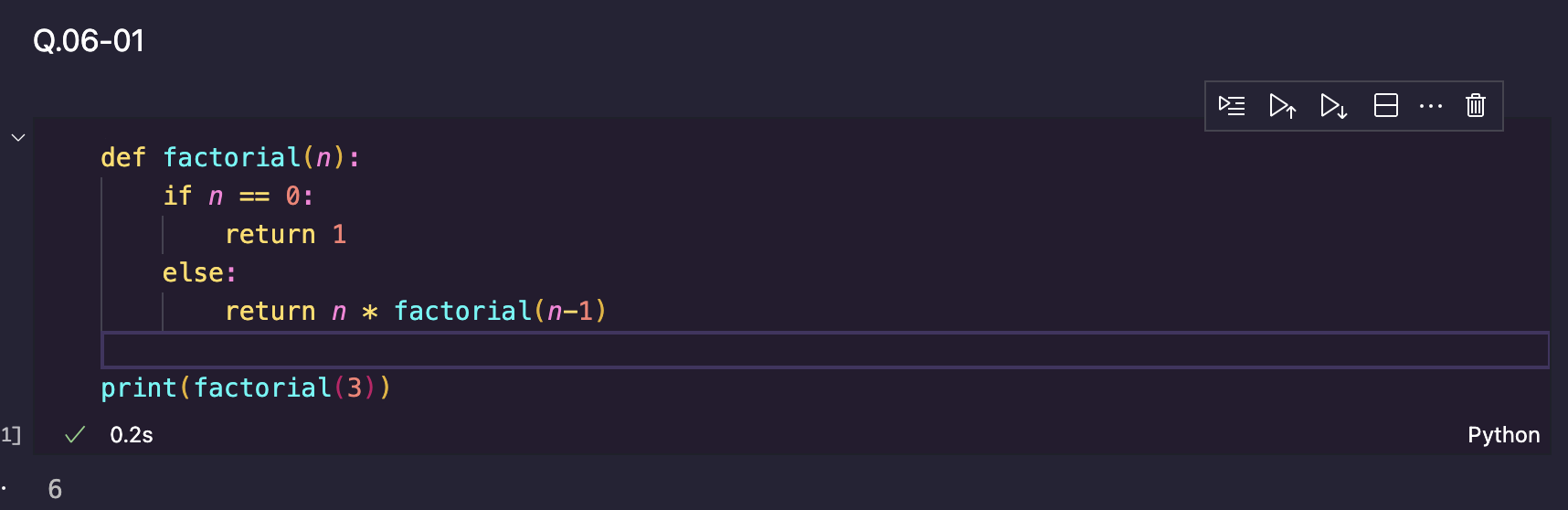
R: La función se ejecutó 0 veces ya que normalmente deberia ejecutarse una vez pero se le resto una por lo tanto arroja un error.

**Q. 05-04.** Dada la lista a continuación, escriba la salida después de ejecutar la función partición1().



R: La salida consiste en que se intercambia los elementos. Se intercambia el pivote con el elemento siguiente al ultimo elemento menor que el pivote.

**Q. 06-01.** Diseñe un algoritmo que halle la función factorial de cualquier número empleando recursividad



Codigo:

def factorial(*n*):

if *n* == 0:

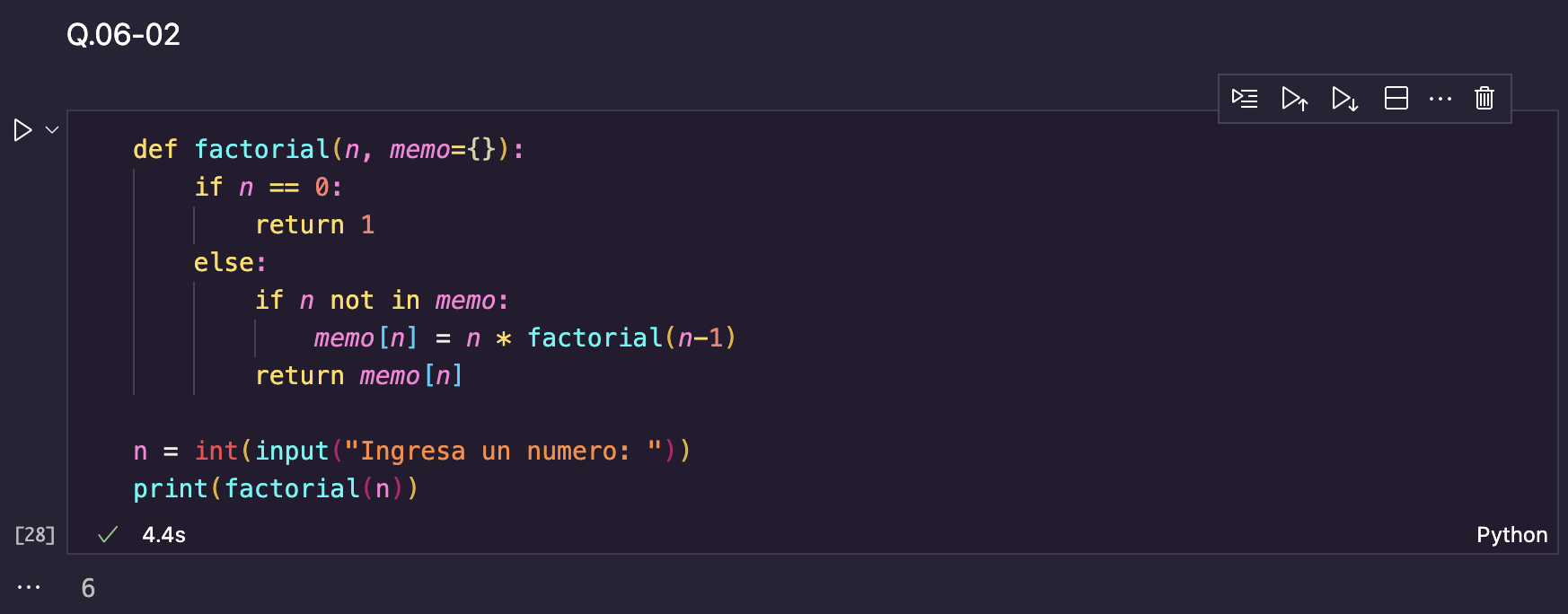
return 1

else:

return *n* \* factorial(*n*-1)

print(factorial(3))

**Q. 06-02.** Diseñe un algoritmo que halle la función factorial de cualquier número empleando memoización



Codigo:

def factorial(*n*, *memo*={}):

if *n* == 0:

return 1

else:

if *n* not in *memo*:

*memo*[*n*] = *n* \* factorial(*n*-1)

return *memo*[*n*]

n = int(input("Ingresa un numero: "))

print(factorial(n))

**Q. 06-03.** Diseñe un algoritmo que codifique la secuencia de Fibonacci empleando recursividad



Codigo:

def fibonacci(*n*):

if *n* == 0 or *n* == 1:

return 1

else:

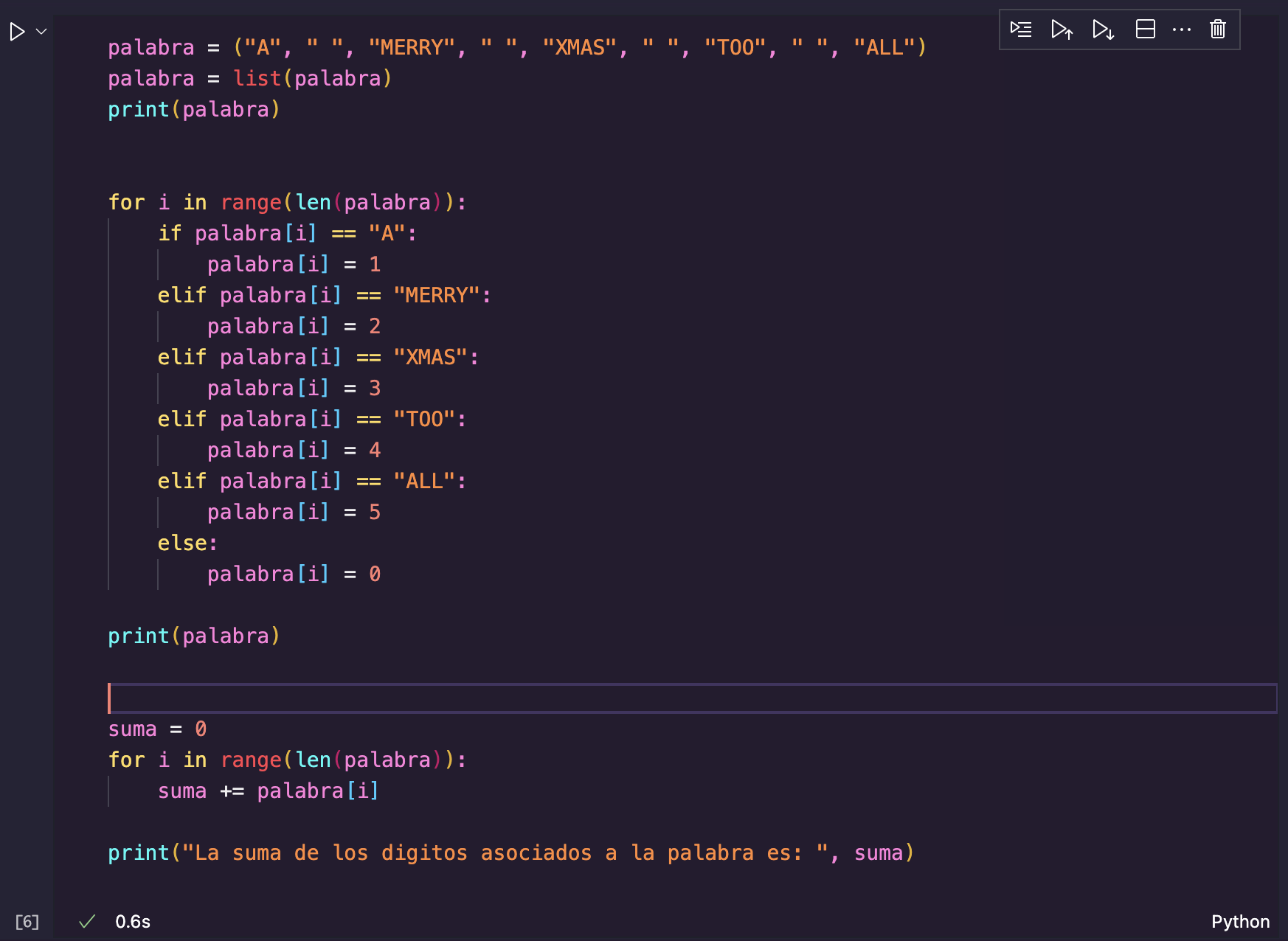
return fibonacci(*n*-1) + fibonacci(*n*-2)

n = int(input("Ingresa un numero: "))

print(fibonacci(n))

**Q. 06-04.** Ejercicio lógico: en cada palabra, reemplace las letras por un número, teniendo en cuenta que para cada palabra separada por un espacio la suma de sus dígitos es un número al cuadrado Encuentra el número representado por cada letra.





Codigo:

palabra = ("A", " ", "MERRY", " ", "XMAS", " ", "TOO", " ", "ALL")

palabra = list(palabra)

print(palabra)

for i in range(len(palabra)):

if palabra[i] == "A":

palabra[i] = 1

elif palabra[i] == "MERRY":

palabra[i] = 2

elif palabra[i] == "XMAS":

palabra[i] = 3

elif palabra[i] == "TOO":

palabra[i] = 4

elif palabra[i] == "ALL":

palabra[i] = 5

else:

palabra[i] = 0

print(palabra)

suma = 0

for i in range(len(palabra)):

suma += palabra[i]

print("La suma de los digitos asociados a la palabra es: ", suma)

* **Código de referencia**

****

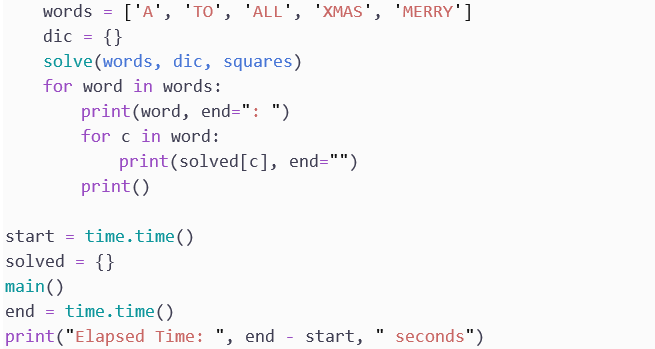
* **Código de referencia**

****

* **Código de referencia**

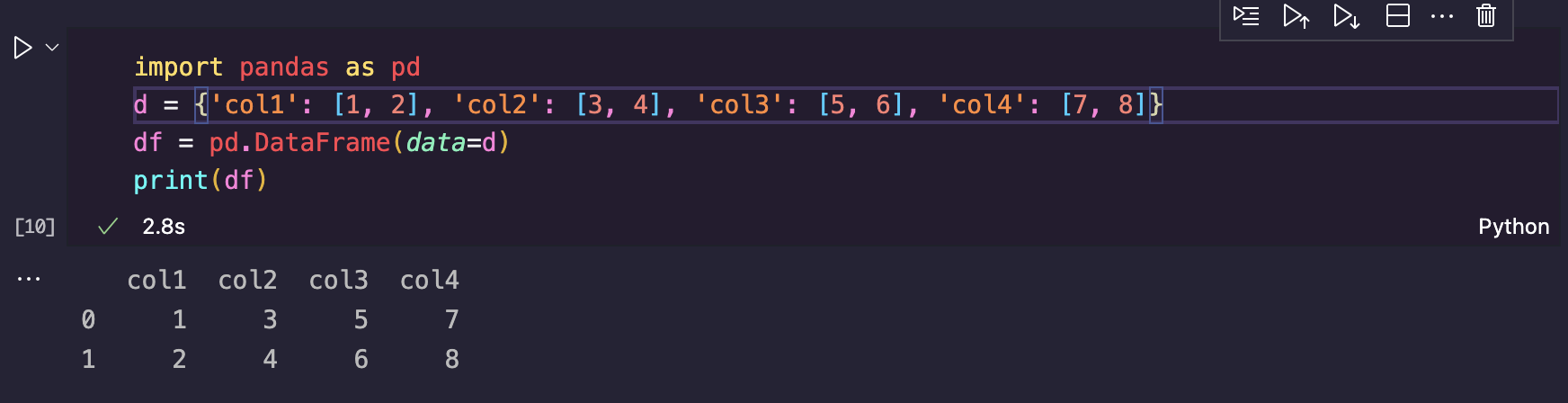
****

* **Código de referencia**

****

**Q. 07-01.** Convierta los siguientes datos del diccionario en un objeto dataframe usando Pandas. (El objeto del marco de datos se denomina df.)





Codigo:

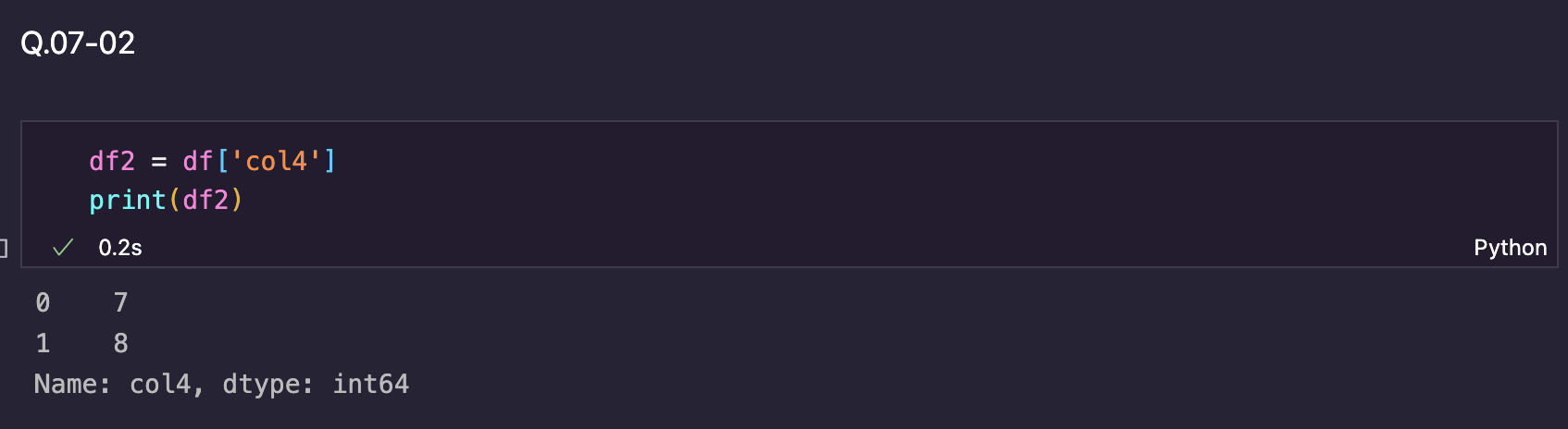
import pandas as pd

d = {'col1': [1, 2], 'col2': [3, 4], 'col3': [5, 6], 'col4': [7, 8]}

df = pd.DataFrame(*data*=d)

print(df)

**Q. 07-02.** Para el dataframe creado en la Pregunta 1, cree un nuevo dataframe que consista sólo en los datos de la columna con el nombre de columna 'col4'. (El dataframe se denomina new\_df.)

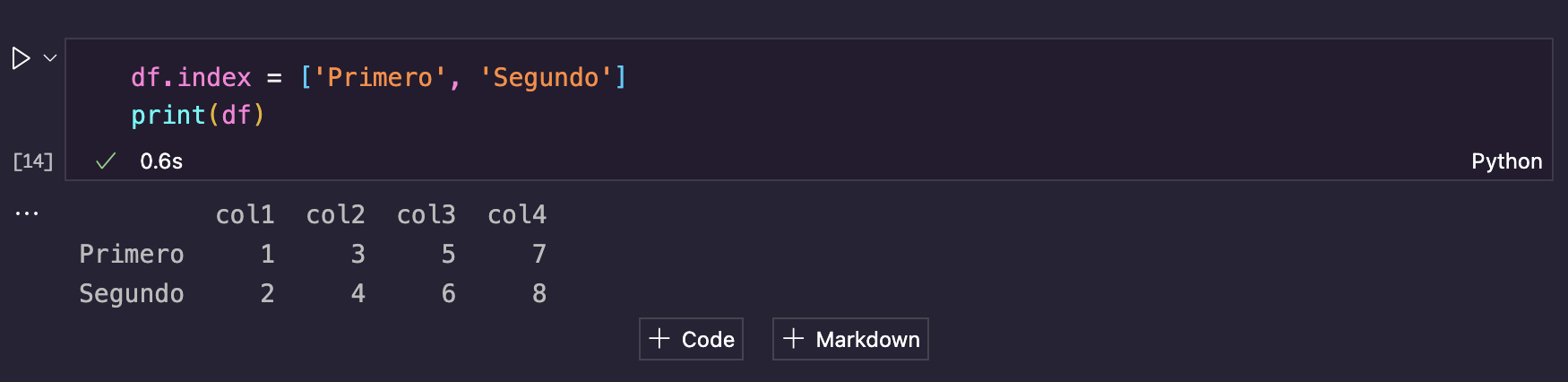


Codigo:

df2 = df['col4']

print(df2)

**Q. 07-03.** El índice del dataframe creado en la Pregunta 1 es 0,1. Escriba un comando para cambiar el nombre de estos índices a primero y segundo.

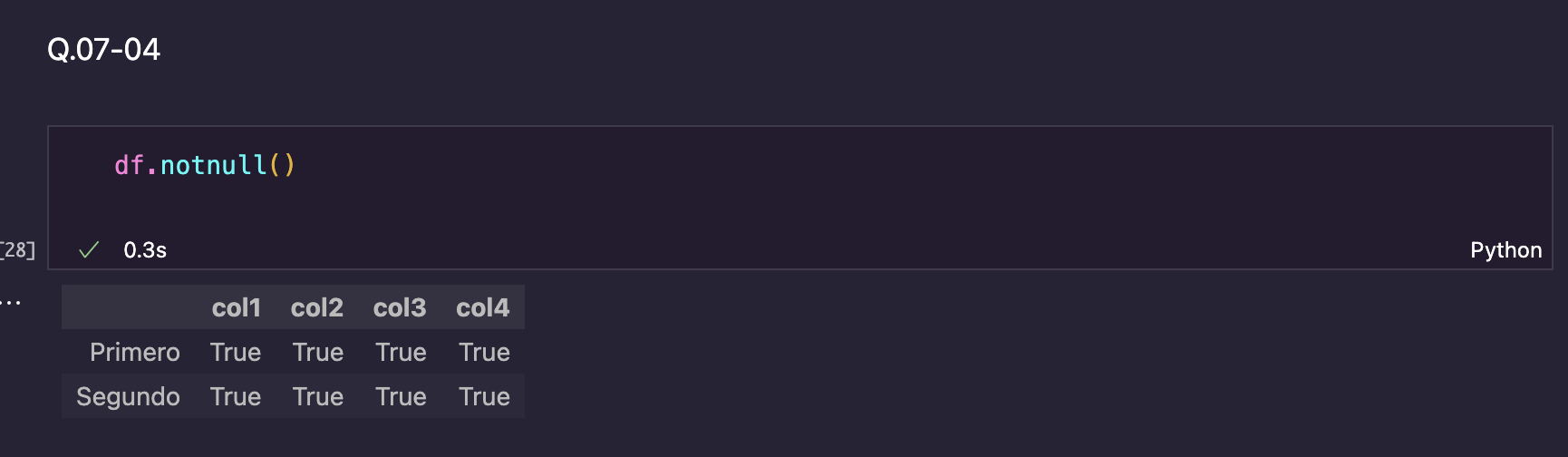


Codigo:

df.index = ['Primero', 'Segundo']

print(df)

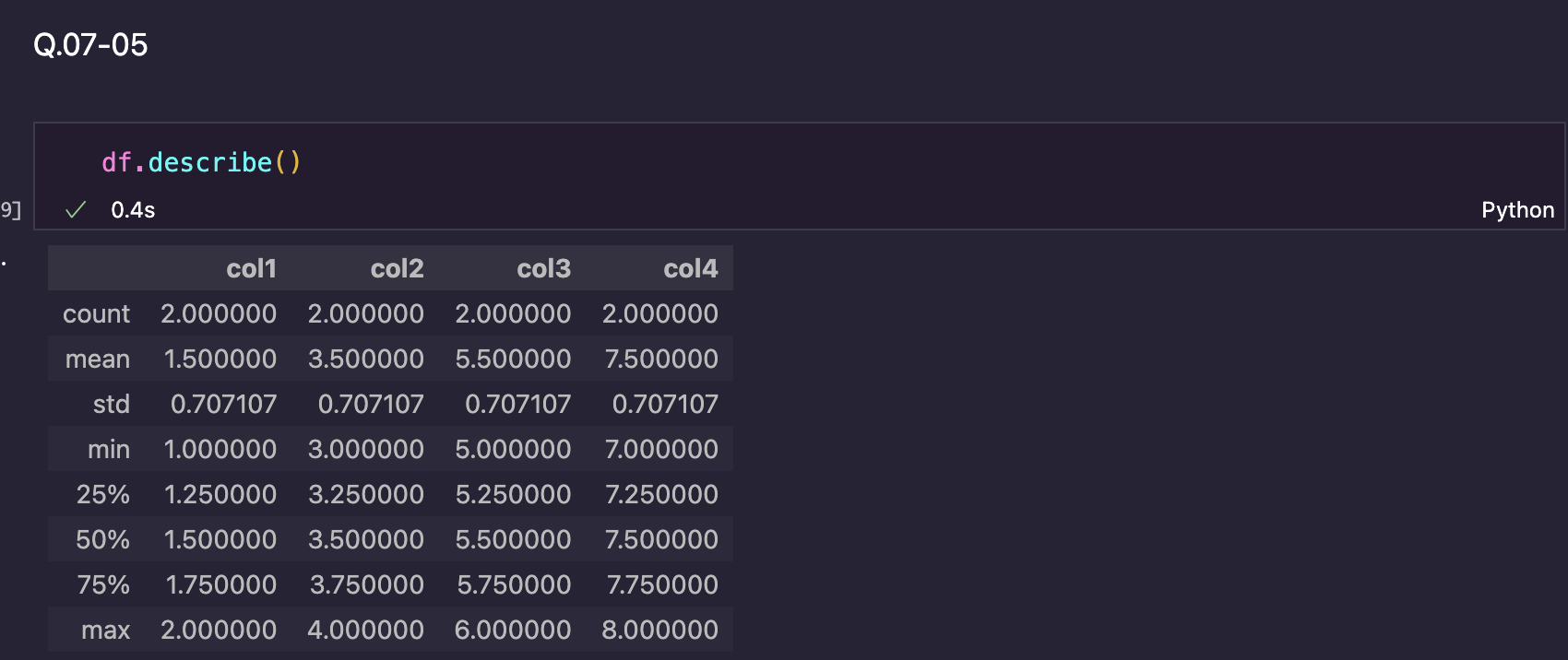
**Q. 07-04.** Escriba un comando para buscar datos faltantes en el dataframe df creado en la Pregunta 1 e imprima el resultado. (Sin embargo, los datos que faltan deben devolverse como verdaderos).



Codigo:

df.notnull()

**Q. 07-05.** Escriba un comando para verificar el resumen de las estadísticas descriptivas (desviación estándar, valor mínimo, moda, etc.) del marco de datos df creado en la pregunta 1 e imprima el resultado.



Codigo:

df.describe()