# Preguntas comunes de Python

Responder las preguntas de manera directa utilizando ejemplos.

1. ¿Cuáles son las diferencias en la lista y la tupla?

* Los elementos en la lista se pueden modificar, en cambio en las tuplas no se pueden modificar los elementos.
* Las tuplas requieren menos espacio que las listas.
* Las tuplas se pueden usar como clave en un diccionario, en cambio las listas no.

1. ¿Cómo se puede usar expresiones if else en una sola línea, comúnmente llamadas operaciones ternarias?

La forma de utilizar el operador ternario es la siguiente:

Valor\_verdadero if <condición> else valor\_false

Ejemplo:

valor = 1

resultado = True if valor > 0 else False

1. ¿Para que sirve dir() y help()?

dir() devuelve los métodos y propiedades del objeto evaluado. Ejemplo: dir(lista).

help() muestra el compendio de ayuda sobre el objeto evaluado para poder entender como utilizarlo cuando se escribe un programa.

1. ¿Que son diccionarios?

Son estructuras de datos con características especiales que permiten almacenar cualquier tipo de valor. Para identificar cada elemento se utiliza una clave o llave. Se utilizan los paréntesis de llaves para encerrar los valores separados por coma y se usa los dos puntos para separar la clave y el valor. Ejemplo:

diccionario = {‘edad’ : 12, nombre : ‘Carlos’, ‘cursos’ : [‘java’, ‘sql’, ‘php’]}

1. ¿Que son \*args y \*\*kwargs?, ¿Como se usan?

\*args es una expresión que se utiliza para que una función pueda usar una cantidad indefinida de valores por parámetros. El \* es el que permite que se puedan pasar todos los valores por parámetros que se requieran y la palabra args es representativa. Ejemplo:

def función(\*args):

return args

función(6, 10, True, False, ‘Hola, mundo’)

(6, 10, True, False, ‘Hola mundo’)

Por su parte, la expresión \*\*kwargs es similar ya que permite pasar por parámetro los argumentos junto con su valor. Ejemplo:

def función(\*\*kwargs):

return kwars

función(a=1, b=True, h=50, z=’Hola mundo’)

{‘a’ : 1, ‘h’ : 50, ‘b’ : True, ‘z’ : ‘Hola mundo’}

1. ¿Qué son índices negativos?

Son números negativos que se utilizan en listas, tuplas o cadenas de caracteres para obtener una parte o el elemento que se requiere de acuerdo a la posición donde se encuentre. Se utilizan en el subíndice encerrado entre los corchetes o paréntesis, dependiendo si son listas o tuplas lo que se esta usando. Ejemplo: print(lista[-1]) devolverá el último valor de la lista.

1. ¿Como se puede ordenar aleatoriamente -o desordenar- una lista?

Para ordenar o desordenar una lista aleatoriamente se utiliza la biblioteca random y el comando shuffle. Ejemplo:

#importamos la librería random

Import random

#generamos la lista

lista = []

for i in range(10):

lista.append(i)

#desordenamos la lista aleatoriamente

random.shuffle(lista)

1. ¿Como se puede ordenar una lista?

Utilizando el comando sort(). Ejemplo: lista.sort()

1. Explique o justifique los resultados de A0,A1,A2,A3,A4,A5,A6 ?

* A0 = dict(zip(('a','b','c','d','e'),(1,2,3,4,5)))

El resultado es: {'a': 1, 'b': 2, 'c': 3, 'd': 4, 'e': 5}, lo que se hizo es que se unieron los valores de ambos conjuntos de letras y números, utilizando el comando zip de acuerdo al orden o la posición en la que se encuentra cada uno y al final forman el diccionario resultante.

* A1 = range(10)

Imprime el rango entre 0 a 10 asignado a la variable A1: range(0, 10)

* A2 = sorted([i for i in A1 if i in A0])

Imprime los corchetes vacios: []. Esto sucede porque la expresión busca el valor evaluado en el rango de 0 a 10 en el diccionario A1 y como no lo encuentra imprime los corchetes vacios: []. Si encuentra los valores los muestra de forma ordenada mediante el comando sorted.

* A3 = sorted([A0[s] for s in A0])

Imprime [1, 2, 3, 4, 5], esto se obtiene al buscar en el conjunto A0 los subvalores a la derecha y los ordena.

* A4 = [i for i in A1 if i in A3]

Imprime [1, 2, 3, 4, 5], ya que busca los valores en el rango de 0 a 10 del conjunto A1 que estén en el conjunto A3 y los imprime utilizando la letra i.

* A5 = {i:i\*i for i in A1}

Imprime {0: 0, 1: 1, 2: 4, 3: 9, 4: 16, 5: 25, 6: 36, 7: 49, 8: 64, 9: 81}, este es el resultado de multiplicar cada valor del rango de 0 a 9 en A1 por su mismo y se agrupa con cada valor correspondiente de 0 a 9 en un diccionario.

* A6 = [[i,i\*i] for i in A1]

Imprime el resultado [[0, 0], [1, 1], [2, 4], [3, 9], [4, 16], [5, 25], [6, 36], [7, 49], [8, 64], [9, 81]], que se obtiene de forma similar al anterior solo que en este se obtiene una lista de elementos emparejados en el Arango de 0 a 9 del conjunto A1.

* print(A0,A1,A2,A3,A4,A5,A6)

Imprime cada uno de los valores resultantes para cada conjunto de A1 hasta A6.

1. ¿Cómo se pueden generar números aleatorios enteros y decimales?

Para generar números aleatorios enteros se utiliza la función randint y randrange. Ejemplo:

Print(randint(10, 20))

Print(randgange(10))

Para generar números aleatorios decimales se utiliza la función random y uniform. Ejemplo:

Print(random())

Print(uniform(5, 8))

1. ¿Qué es pickling y unpickling?

Pickling se refiere a la serialización de los archivos en Python para comprimirlos y así poder manipularlos de manera más fácil y segura. Por su parte, unpickling se refiere a la forma inversa, es decir a la deserialización de los archivos comprimidos para poder trabajar con la información que almacenan.

1. ¿Para que sirve la función map, lambda y filter?

La función map aplica a cada valor de una lista la formula o el procedimiento que se declara para obtener el resultado esperado de forma más rápida y sencilla sin tener que recorrer manualmente cada elemento. Ejemplo:

items = [ 1 , 2 , 3 , 4 , 5 ]

conjunto = list ( map ( lambda x : x \*\* 2 , items ))

print(conjunto)

[1, 4, 9, 16, 25]

La función lambda se usa para definir funciones de manera más rápida y fácil, para obtener un resultado esperado. Ejemplo:

def f(x, y, z=1):

return (x+y) \* z

f(5, 6)

11

f(5, 6, 7)

77

#Usando la función lambda es más eficiente el código:

f = lambda x, y, z=1: (x+y) \* z

f(5, 6)

11

f(5, 6, 7)

Finalmente, la función filter se usa para obtener una lista de elementos para los cuales devuelve verdadero. Ejemplo:

numeros = range ( - 5 , 5 )

menores\_cero = list ( filter ( lambda x : x < 0 , numeros ))

print(menores\_cero)

[-5, -4, -3, -2, -1]

1. ¿Qué es list comprehension, set comprehension y dict comprehension?

List comprehension es una herramienta que se usa para crear una lista basada en otra lista de forma fácil y rápida. Ejemplo:

input\_list = [1, 2, 3, 4, 4, 5, 6, 7, 7]

list\_using\_comp = [var for var in input\_list if var % 2 == 0]

print(list\_using\_comp)

[2, 4, 4, 6]

Set comprehension se usa para generar listas de forma más sencilla utilizando llaves {} para encerrar los valores obtenidos. Ejemplo:

lista = [1, 2, 3, 4, 4, 5, 6, 6, 6, 7, 7]

resultado = {var for var in lista if var % 2 == 0}

print(resultado)

{2, 4, 6}

Por su parte la función dict comprehension se usa para crear diccionarios de forma más eficiente. Ejemplo:

input\_list = [1, 2, 3, 4, 5, 6, 7]

dict\_using\_comp = {var: var \*\* 3 for var in input\_list if var % 2 != 0}

print(dict\_using\_comp)

{1: 1, 3: 27, 5: 125, 7: 343}