Actividad Programación avanzada en Python

Considere las siguientes listas:

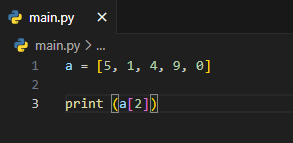


Sin usar el computador indique cuál es el resultado y el tipo de las siguientes expresiones. A continuación, verifique sus respuestas con el computador

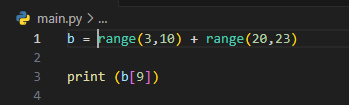
* a[2]
* b[9]
* c[1][2]
* e[0] == e[1]
* len(c)
* len(c[0])
* len(e)
* c[-1]
* c[-1][+1]
* c[2:]+ d[2:]
* a[3:10]
* a[3:10:2]
* d.index(‘jirafa’)
* e[c[0][1]].count(5)
* sorted(a)[2]
* complex(b[0],b[1])

Desarrollo de actividad:

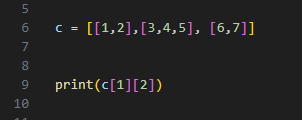
* a[2]: en este código tenemos una lista “a” con “5” cantidad de elementos, y el índice, en este caso [2], nos mostrará el elemento que hay en el tercera posición



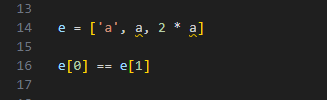
* b[9] : en este código tenemos una lista “b” que concatena con otro y el índice, [9] nos permitirá saber el valor que hay en la posición 10.



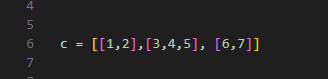
* c[1][2]: aquí hay una lista “c” que contiene listas, donde el valor de [1] te lleva a la posición dentro de las listas y el valor [2] te muestra el valor que hay en esa posición.



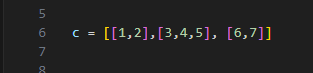
* e[0] == e[1]: en este código podemos comparar los valores que hay en la lista “e”, en sus índices [0] y [1].



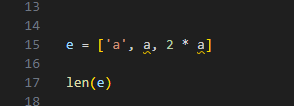
* len(c) : en este código nos mostrará el largo de la lista “c”



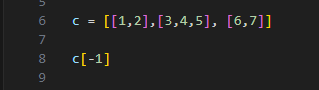
* len (c[0]): este código muestra la longitud de la lista “c”, y [0] nos arrojará la cantidad de elementos que hay, en este caso.



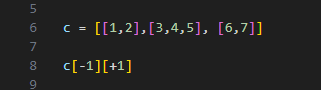
* len(e) : en esta línea nos mostrará el largo de la lista “e”, ya que este contiene 3 elementos, arrojará 3



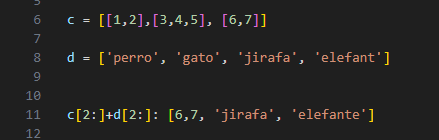
* c[-1] : este código nos arrojará la posición -1 de la lista o tupla que contenga “c”



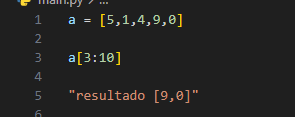
* c[-1][+1]: este código nos lleva a la lista “c” y con [-1] nos vamos a la última lista, luego con [+1] buscaremos la posición 1 dentro la lista.



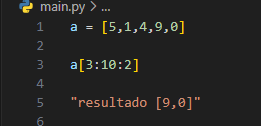
* c [2:]+ d [2:]: este código posee dos listas, una “c” y otra “d”, c[2:] seleccionará los elementos desde el índice 2 en adelante, de igual manera ocurre con d[2]. Al final ambas listas desde el índice dos se unen para mostrarse en una nueva.



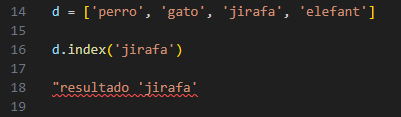
* a[3:10] : este código tiene una lista “a”. “3” indica el índice desde el cual avanzará (incluyendo el valor en ese índice) y el “10” es el límite hasta donde se debe avanzar, pero dado que la lista no contiene tantos valores, este llegará al final de lista y mostrará lo que hay a partir del índice [3]



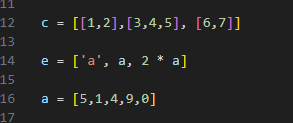
* a[3:10:2] al igual que el código anterior ahora el calor “2” es el que nos permite hacer un salto de “2” dentro de la lista, pero ya que no hay más valores, al final se mostrará los valores que figuran a partir del índice [3]



* d.index(‘jirafa’): este código se aplica a listas para buscar dentro una lista el índice especificado, en este caso ¨jirafa¨ y apenas aparezca el primero, este se mostrará.

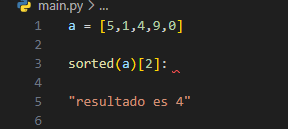


* e[c[0][1]].count(5) : este código primero nos vamos a la lista c[0][1], luego de obtener el resultado lo buscamos en la lista e[2], donde este tiene un valor de 2\*a. Tomamos la lista “a” y como está multiplicado por 2, duplicamos los valor que hay en esa lista. Luego con “count5” contamos las veces que el valor “5” se repite dentro de la lista duplicada.



* sorted(a)[2]: “sorted” toma los valores de la lista “a” y los ordena de manera ascendente

en una nueva lista, y luego con [2], buscamos el valor que hay en el índice solicitado



* complex(b[0],b[1]): “complex” se usa para generar números complejos, este se compone, en este caso, de b[0], que es la parte real y b[1] la cual contiene la parte imaginaria.

