```
#include <stdio.h>
#define tamaño 10

Int main()
{
        char c;
        int x;
        int arreglo[tamano];
        double y;

        printf(“El tamano en bytes de una variable char es: %lu\n”,sizeof(c));
        printf(“El tamano en bits de una variable char es: %lu\n”,(8*sizeof(c)));
        printf(“El tamano en bytes en nibble de una variable char es:%lu\n”(sizeof(c)*2));

        printf(“El tamano en bytes de una variable int es: %lu\n”,sizeof(c));
        printf(“El tamano en bits de una variable int es: %lu\n”,(8*sizeof(c)));
        printf(“El tamano en bytes en nibble de una variable chintar es:%lu\n”(sizeof(c)*2));

        printf(“El tamano en bytes de una variable de un arreglo es: %lu\n”,sizeof(c));
        printf(“El tamano en bits de una variable de un arreglo es: %lu\n”,(8*sizeof(c)));
        printf(“El tamano en bytes en nibble de una variable de un arreglo es:%lu\n”(sizeof(c)*2));

        printf(“El tamano en bytes de una variable double es: %lu\n”,sizeof(c));
        printf(“El tamano en bits de una variable double es: %lu\n”,(8*sizeof(c)));
        printf(“El tamano en bytes en nibble de una variable double es:%lu\n”(sizeof(c)*2));

        return 0;
}
```

Código en Python

```
tamaño = 10


def main():
    c = 'a'
    x = 1
    arreglo = [10] * tamano
    y = 1.2

    print("El tamaño en bytes de una variable char es:", c.__sizeof__())
    print("El tamaño en bits de una variable char es:", (8 * c.__sizeof__()))
    print("El tamaño en bytes en nibble de una variable char es:", (c.__sizeof__() * 2))

    print("El tamaño en bytes de una variable int es:", x.__sizeof__())
    print("El tamaño en bits de una variable int es:", (8 * x.__sizeof__()))
    print("El tamaño en bytes en nibble de una variable int es:", (x.__sizeof__() * 2))

    print("El tamaño en bytes de una variable de un arreglo es:", arreglo.__sizeof__())
    print("El tamaño en bits de una variable de un arreglo es:", (8 * arreglo.__sizeof__()))
    print("El tamaño en bytes en nibble de una variable de un arreglo es:", (arreglo.__sizeof__() *
2))

    print("El tamaño en bytes de una variable double es:", y.__sizeof__())
    print("El tamaño en bits de una variable double es:", (8 * y.__sizeof__()))
    print("El tamaño en bytes en nibble de una variable double es:", (y.__sizeof__() * 2))

    return 0
```

La razón por la cual un código en c y un código en python no da lo mismo es porque c y python son lenjuages de programación con implementaciones diferentes, c es un lenguaje de programación compilado, y Python es lenguaje interpretado.

Otra razón es la representación de números, en c los números enteros se pueden representar usando diferentes tamaños de datos, como int, short int, lo que puede que Python afecte en la precisión y el rango de valores que se pueden almacenar.

Las bibliotecas, Python tiene mas bibliotecas, mientras c no tantas.