Exercise 1 – readme file:

205892201 Guy Haviv – guyhaviv@mail.tau.ac.il
316294636 Noa Davidovitch – noad1@mail.tau.ac.il

This is an implementation of a Server/Client protocol for the game "nim" as described in the exercise description.

Files & Modules:
2 files only: nim-server.py for server application, nim-client.py for client application.

Run the code:

1. Establish a Server – python3 nim-server.py n_a n_b n_c [,PORT = 6444]
2. Run a client – python3 nim-client.py [,HOSTNAME = localhost,PORT = 6444]

Protocol:
The protocol is based on a two-way **BINARY** communication between client & server in the following message construction (format for struct.pack/unpack) :

Message Protocol					
Client sends / Server receives					
Format : **">ci"** - one char and one integer					
Scenario / Validity	valid		invalid		
Game On	c = 'A'/'B'/'C'	i = any integer	Anything else		
Server sends / Client receives					
Format : **">ciii"** - one char and 3 integers					
Last client move	valid	invalid	integers		
Game init	c = 'i'		i = n_a	i = n_b	i = n_c
Game on	c = 'g'	c = 'x'			
Game over - server wins	c = 's'	c = 't'			
Game over - client wins	c = 'c'	N/A			

Client sends a heap identifier (A/B/C) and amount of die to remove from that heap.
When a Client user inputs 'Q' , the client program quits, and server will follow on data receive error.
Server sends a message tag (char) that stands for the game status (which can rely on the Client's previous move) and the amount of die in each heap.

Errors and edge cases:
This is the program's error handling policy for both server & client:

Error handling		
Error handling	Server	Client
Invalid program-run arguments *	Quit program	Quit program
Failure establishing server	Quit program	N/A
Failure establishing connection	Continue listening	Quit program
Failure sending data	Quit current game	
Failure receiving data	Quit current game	
Invalid input (from connection)	Ignore client input & continue game	

* Client: if *no program arguments* passed – connects to localhost:6444, if *one argument* is passed it will be considered as HOSTNAME , i.e. connects to HOSTNAME:6444.
For 2 arguments or more – will take the *first two* as HOSTNAME:PORT.
Server: program arguments *must* include at least 3 integers for heap sizes, optional 4th argument as PORT .
5 arguments or more , 2 arguments or less - will result in error.
Logically invalid arguments (e.g. a string 'hello' as heap size for server) will result in error.