Számítógépes Hálózatok

3. gyakorlat

Pyton socket, host név feloldás

- Socket csomag használata

```
import socket
```

- gethostname()

```
hostname = socket.gethostname()
```

- gethostbyname()

```
hostname = socket.gethostbyname(‚www.example.org’)
```

- gethostbyname_ex()

```
hostname, aliases, addresses = socket.gethostbyname_ex(host)
```

- gethostbyaddr()

```
hostname, aliases, addrs = socket.gethostbyaddr('157.181.161.79')
```

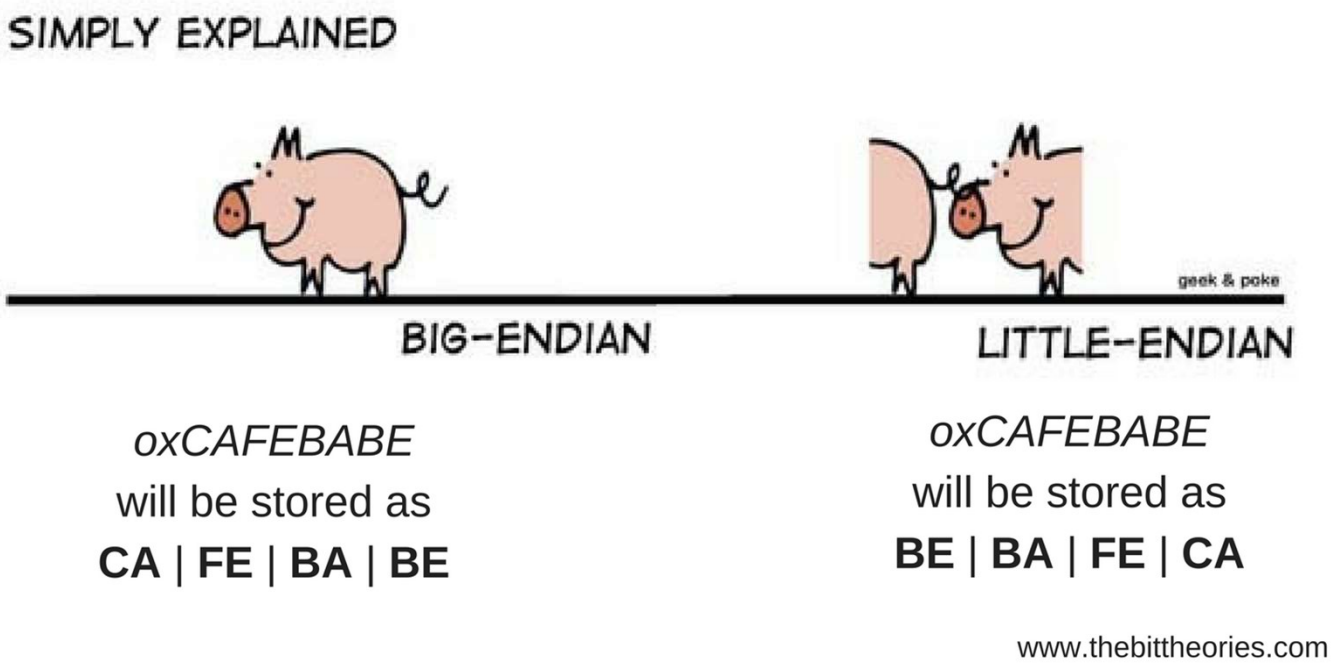
Port számok

- Bizonyos protokollokhoz tartoznak fix portszámok, konstansok (szállítási protokollok)!
- getservbyport()

```
socket.getservbyport(22)
```

- Írassuk ki a 1..100-ig a portokat és a hozzájuk tartozó protokollokat!

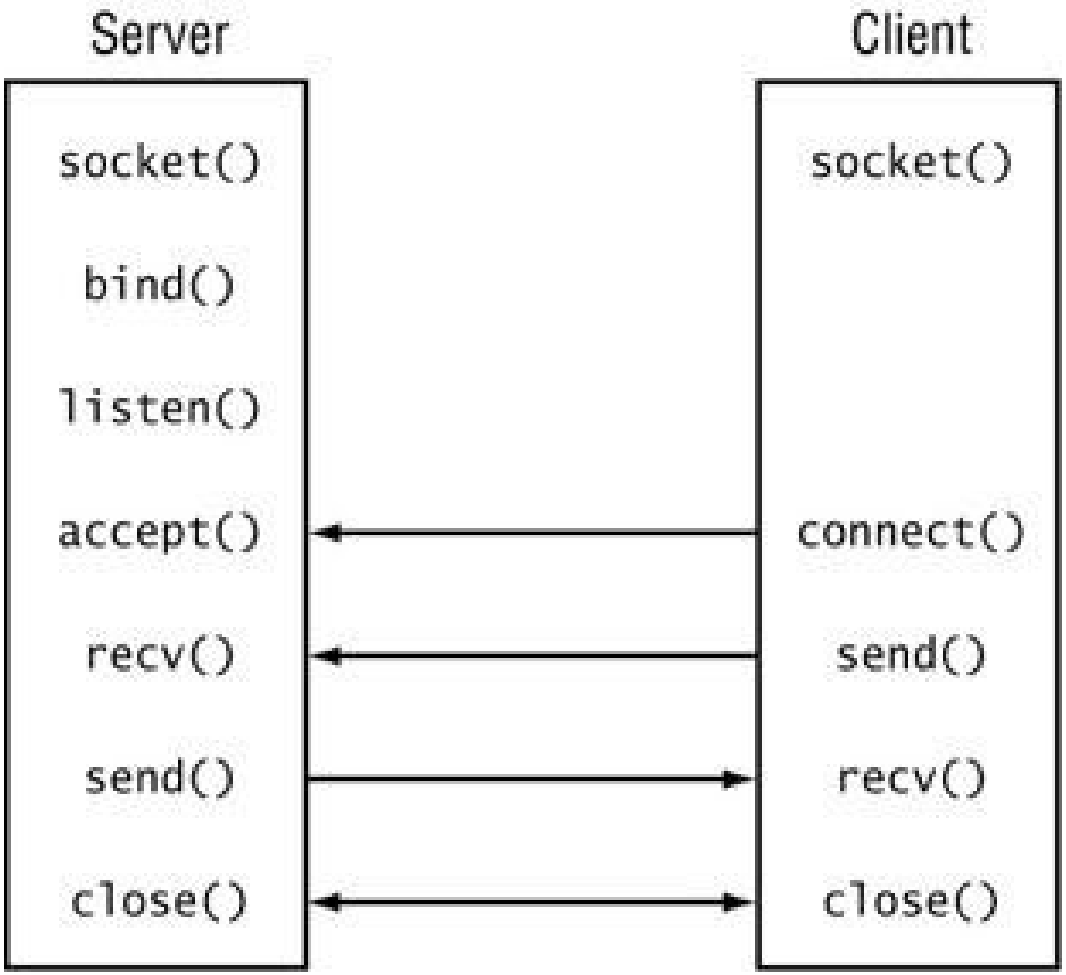
Little endian, big endian



- 16 és 32 bites pozitív számok kódolása
  - htons(), htonl() – host to network short / long
  - ntohs(), ntohl() – network to host short / long

  Saját gép milyen kódolást használ: sys.byteorder

TCP


Server
Client
socket()
bind()
listen()
accept()
recv()
send()
close()
socket()
connect()
send()
recv()
close()

TCP

- socket()

```
sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
```

- bind()

```
server_address = ('localhost', 10000)
sock.bind(server_address)
```

- listen()

```
sock.listen(1)
```

- accept()

```
connection, client_address = sock.accept()
```

TCP

- send(), sendall()

```
connection.sendall(data)                    #python 2.x
```

```
connection.sendall(data.encode())           #python 3.x
```

- recv()

```
data = connection.recv(16)                  #python 2.x
```

```
data = connection.recv(16).decode()         #python 3.x
```

- close()

```
connection.close()
```

- connect()

```
server_address = (‚localhost’, 10000)
sock.connect(server_address)
```

Feladat

- Készítsünk egy egyszerű kliens-server alkalmazást, ahol a kliens elküld egy ‚Hello server' üzenetet, és a szerver pedig válaszol neki egy ‚Hello kliens' üzenettel!

- Változtassuk meg úgy, hogy ne hard codeoljuk a portot, hanem parancssori argumentumként kérjünk be egyet!

Struktúraküldése

- Binárissá alakítjuk az adatot

```
import struct
values = (1, 'ab’.encode(), 2.7)
packer = struct.Struct('I 2s f')                #Int, char[2], float
packed_data = packer.pack(*values)
```

- Visszalakítjuk a kapott üzenetet

```
import struct
unpacker = struct.Struct('I 2s f’)          #struct.calcsize(unpacker)
unpacked_data = unpacker.unpack(data)
```

- megj.: integer 1 – 4 byte, stringként 1 byte, azaz hatékonyabb stringként átküldeni.
- https://docs.python.org/3/library/struct.html

Feladat

Készítsünk egy szerver-kliens alkalmazást, ahol a kliens elküld 2 számot és egy operátort a szervernek, amely kiszámolja és visszaküldi az eredményt. A kliens üzenete legyen struktúra.

Szorgalmi feladat

- Készíts egy szerver-kliens alkalmazást amiben a kliens egy random méretű stringet küld a szervernek, úgy, hogy mielőtt ezt megtenné, elküldi neki a hosszát.
- A szerver pontosan a megkapott mennyiségű adatot fogadjon.

VÉGE