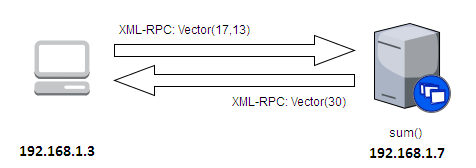
**Tugas 2: Remote Procedure Call**

**Definisi**

RPC adalah memanggil procedure yang ada remote node. Contoh: terdapat sebuah klien yang akan memanggil prosedur (bernama: *sum()*) di suatu server. Setelah server memperoleh panggilan dari client kemudian server mengembalikan hasil penjumlahan kepada client.



Dengan menggunakan RPC, seolah-olah klien memanggil prosedur lokal padahal klien memanggil prosedur (sum) yang berada di server. Klien cukup mengetahui nama prosedur yang akan dipanggil dan memberikan parameter yang sesuai.

Terdapat banyak cara untuk implementasi RPC. Pada tugas ini kita akan mengimplementasikan RPC akan menggunakan XML-RPC (<https://en.wikipedia.org/wiki/XML-RPC>) . Secara ringkas, XML-RPC adalah **“a simple, portable way to make remote procedure calls over HTTP.”**

**Encode (marshall dan unmarshall)**

Pada jaringan, XML-RPC diencode sebagai XML. Jenis RPC lainnya akan menggunakan metode encoding yang berbeda-beda.

<methodCall>

<methodName>**sample.sumAndDifference**</methodName>

<params>

<param><value>**<int>5</int>**</value></param>

<param><value>**<int>3</int>**</value></param>

</params>

</methodCall>

**Tipe Data yang disupport XML-RPC**

int

A signed, 32-bit integer.

string

An ASCII string, which may contain NULL bytes. (Actually, several XML-RPC implementations support Unicode, thanks to the underlying features of XML.)

boolean

Either true or false.

double

A double-precision floating point number. (Accuracy may be limited in some implementations.)

dateTime.iso8601

A date and time. Unfortunately, since XML-RPC forbids the use of timezones, this is very nearly useless.

base64

Raw binary data of any length; encoded using Base64 on the wire. Very useful. (Some implementations don't like to receive zero bytes of data, though.)

array

An one-dimensional array of values. Individual values may be of any type.

struct

A collection of key-value pairs. The keys are strings; the values may be of any type.

**Kegunaan RPC**

RPC dapat dikarakteristikkan sebagai **transaction-oriented communication** dimana:

* Transaksi terdiri dari single-request dan single-response
* Transaksi diinisiasi ketika client mengirim request dan diterminasi oleh respon server

Dengan RPC**, low latency lebih diutamakan dari pada high throughput.**

Contoh RPC:

* Microsoft Exchange Server
* File replication service (FSR) di windows
* Active directory replication
* DCOM
* SQL (querry)

**Python XML-RPC**

XML-RPC Library yang digunakan pada python adalah xmlrpclib (pada python2.x) dan xmlrpc (pada python 3.x). Pastikan gunakan library yang sesuai dengan python yang telah diinstall!

**Server RPC**



RPC server berjalan pada localhost (IP:127.0.0.1) pada port 8000. Server mempunyai 3 buah metode yaitu: pow(x,y), add(x,y) dan mul(x,y).

**Client**

Setelah sever dijaankan, jalankan python secara interaktif. Tuliskan perintah ini satu per satu

>>> import xmlrpc

>>> s = xmlrpc.client.ServerProxy('http://localhost:8000')

>>> print (s.power(2,5))

32



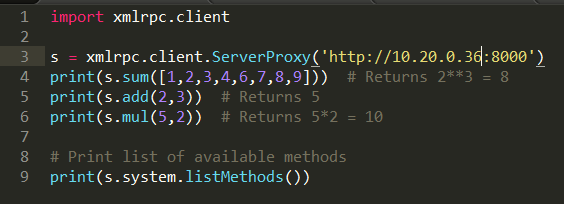
**Tugas**

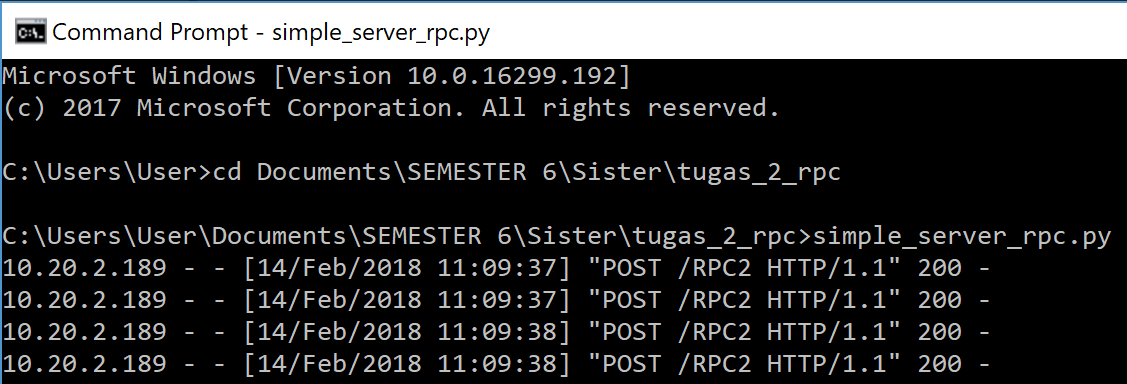
1. Buatlah metode di server yang dapat menjumlah n bilangan.

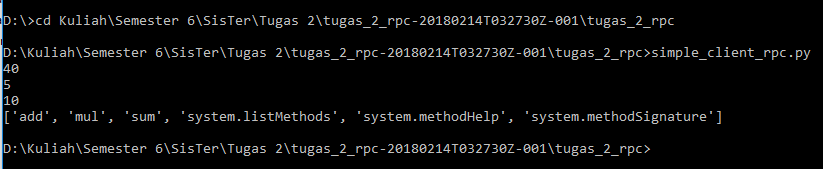
Input: 2,3,7,5,2,4 (jumlah masukan bebas)

Output: 23

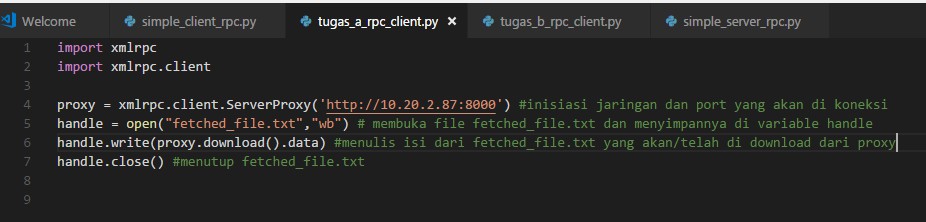


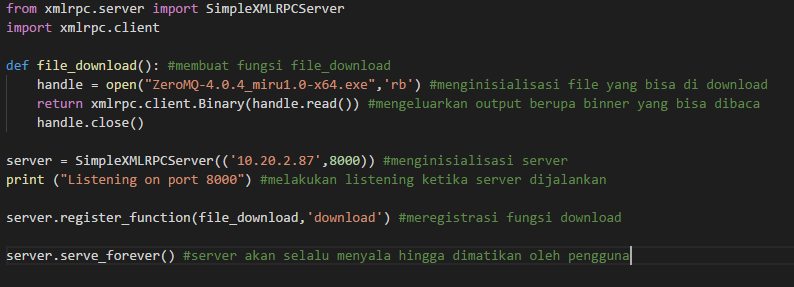




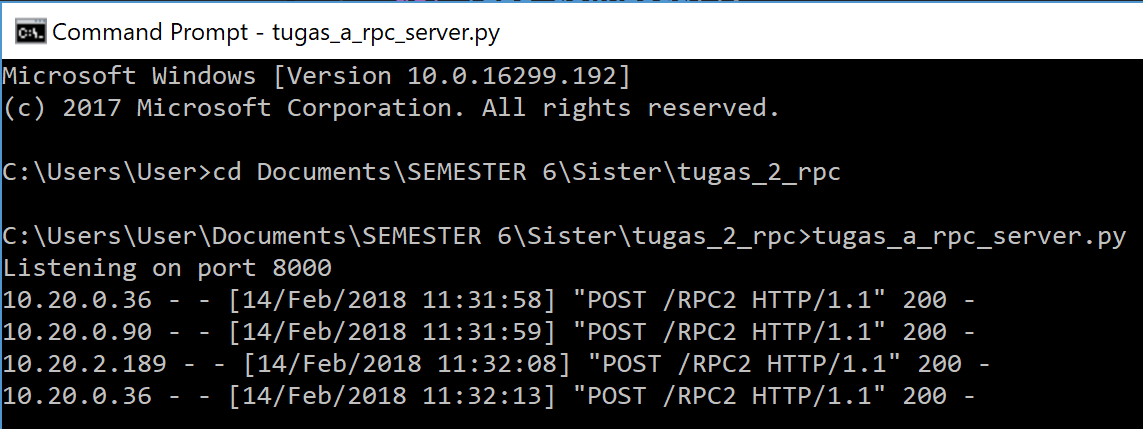


1. Download tugas\_a\_rpc\_client.py dan tugas\_a\_rpc\_server.py
   1. Baca dan beri komentar setiap line pada source code. Apa yang dilakukan oleh program?

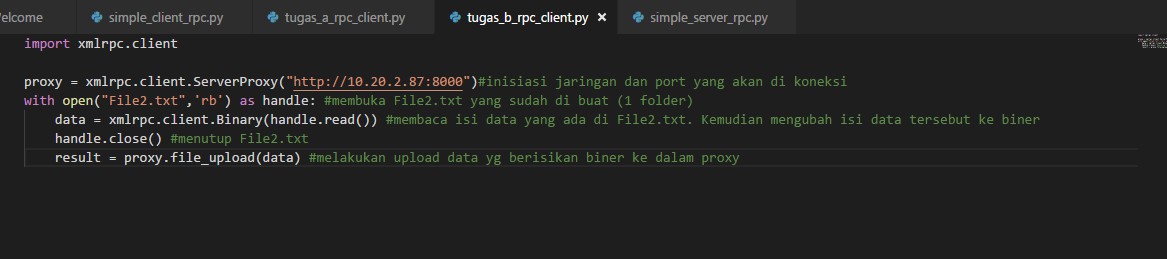
****

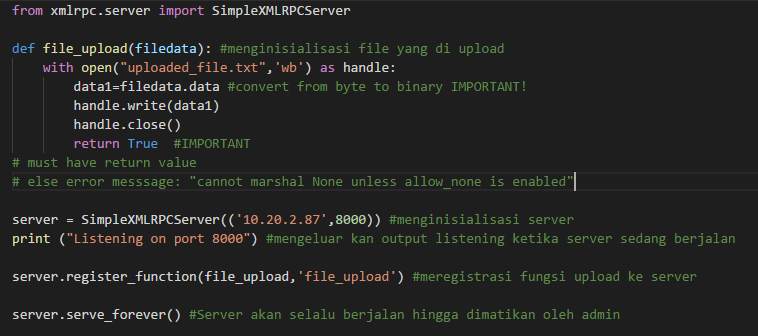


* 1. Jalankan program tersebut pada 2 komputer. Screenshoot hasil kerja program!

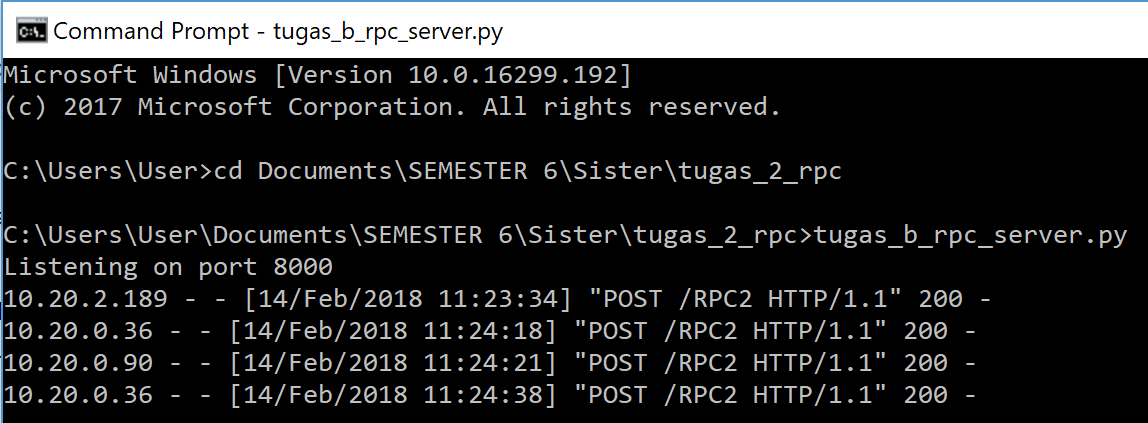


1. Download tugas\_b\_rpc\_client.py dan tugas\_b\_rpc\_server.py
   1. Baca dan beri komentar setiap line pada source code. Apa yang dilakukan oleh program?

****

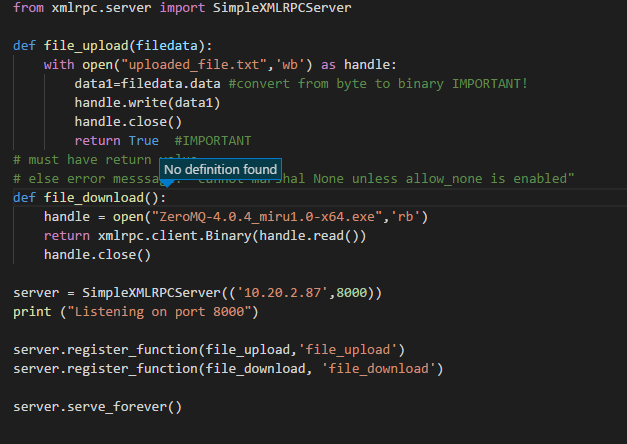


* 1. Jalankan program tersebut pada 2 komputer. Screenshoot hasil kerja program!



1. Buatlah program ftp sederhana menggunakan source code yang telah diberikan

ftp Server



ftp client

