

Redes de Comunicaciones I

Práctica 1: Tutorial de Python

<http://docs.python.org/3.7/tutorial/>
<http://getpython3.com/diveintopython3/>

Índice

- Intérprete
- Lenguaje
- Ejemplos
 - Hola mundo
- Apéndice
 - Uso de hilos
 - Construir estructura de bytes
 - Locks

Intérprete Python 3.6

- Normalmente instalado en Linux
- Arranque el intérprete de Python en una línea de comandos
`python3`
- Salga del intérprete de Python
`quit()`
- Ejecutar un script desde línea de comandos
`python3 script.py`

Lenguaje

- Sensible a mayúsculas
- Comentarios de línea con #. Comentarios de bloque con '''
- Delimitar bloques con : y siguientes líneas con tabulador
- \ para caracteres especiales
 - \t tabulador
 - \n retorno de carro
 - \\ barra invertida
- Las líneas de código van terminadas por retorno de carro
 - No hay ;
 - Si hay que partir una línea se puede utilizar \ si no hay delimitadores (,) que lo permitan

Lenguaje

- Mecanismos de control
 - if, elif, else
 - for, in range(inicio[, fin+1[, pasos]]), break, else, continue
 - while, pass
 - try, except, ...
- Comparaciones lógicas similares a C (==, !=...)
- Operadores a nivel de bit similar a C (|,&,...)
- Declaración de variables
 - Sin tipo explícito, al declarar se entiende el tipo utilizado
- Declaración de funciones
 - def func(param1, param2=ValorPorDefecto) :
- Importar funciones y clases
 - from module import function [o *]
 - import module
 - En este segundo caso, las funciones se llamarán *module.function*

Lenguaje

- Tipos de datos
 - Booleanos: True y False
 - Números: enteros (1, 2...), floats (1.0, 2.0)...
int() y float()
 - Operaciones: +, -, *, /, //, **, %
 - Strings: "" o ' ', formato UTF-8 salvo que se indique lo contrario
 - Bytes y byte arrays: string.encode() y ba.decode()
 - Listas: [secuencias ordenadas de valores]
 - Tuplas: (como listas pero inmutables)
 - Conjuntos: {valores sin ordenar}
 - Diccionarios: conjunto de valores con clave
{clave1: valor1, clave2: valor2}

Lenguaje

- Listas (por ejemplo, listas de bytes)
 - `lista[i]`: proporciona el *i*-ésimo elemento, empezando en 0.
 - `lista[-i]`: proporciona el *i*-ésimo elemento, empezando desde atrás en -1
 - `lista[i:j]`: proporciona una lista con los elementos entre *i* y *j*-1.
 - `lista[:j]`: proporciona una lista con los elementos desde el principio hasta *j*-1
 - `lista[i:]`: proporciona una lista con los elementos entre *i* y el final de la lista
 - `lista[:]`: proporciona una lista con todos los elementos

Ejemplos: Hola mundo

```
print('Hola mundo!')
```

- Escribir en un archivo con nombre hola.py
- Ejecutar en línea de comandos
`python3 hola.py`

Uso de hilos


```
import threading
class hilo(threading.Thread):
    def __init__(self): #se pueden añadir más parámetros
                        # inicializa clase madre
                        threading.Thread.__init__(self)
                        #Poner aquí resto de inicializaciones
                        # Usar el objeto self para referirse a variables locales
                        #P.e.:self.mivariable=1

al hilo

    def run(self):
        # Rutina de ejecución del hilo

# Arrancar el hilo
background = hilo() #se pasan los parámetros añadidos en el __init__
background.start()
```

Construir una estructura de bytes usando una lista

- `struct.pack(formato, valor)` permite codificar los valores de enteros. Formatos:
 - 'B': 8 bits
 - '!H': 16 bits en orden de red (big endian)
 - '!I'  32 bits en orden de red (big endian)
- `struct.unpack(formato, string)` para decodificar (ojo: devuelve una lista, normalmente se cogerá el primer elemento)

```
import struct
```

```
cab_udp=[] # Lista de bytes
```

```
cab_udp[0:2]=struct.pack('!H',sourceport)
```

```
...
```

```
bytes_cab_udp=bytes(cab_udp)
```

```
...
```

```
sourceport=struct.unpack('!H',bytes_cab_udp[0:2])[0]
```

Uso de Locks

- Cuando usamos varios hilos tenemos que proteger las variables globales compartidas
- Uso de Lock
- Definición:
 - `from threading import Lock`
 - `mylock = Lock()`
- Uso: Cuando queremos declarar una sección crítica donde modificamos variables compartidas:
 - `with mylock:`
 - `variablecompartida += 1`