Python and Strings

Python has some useful string manipulation capabilities “built-in.” Here, we look at a few of these capabilities in the context of manipulating typical network objects or concepts. Some of this material is adapted from the excellent suite of tutorials “Python for Network Engineers” by Kirk Byers.

Simple Input/Output

Input/Output in Python3 is very simple:

- Input: local_string = input('\nEnter some data: ')
- Output: print('data:\t', local_data)

Unicode vs. ASCII

Python3 uses unicode strings (“string objects”) rather than sequences of ASCII bytes (“simple strings”), which were the standard in Python2 and many other programming languages. As a result, Python3 string objects have implicit and explicit methods to “encode” and “decode” between unicode and other formats.

- Type
  - Unicode: local_string = ‘whatever’ … produces type(bb) of “str”
  - ASCII: local_string = b’whatever’ … produces type(bb) of “bytes”
- Indexing:
  - local_string[n] or local_string[-n] … n is zero-based index
  - local_string[start:stop:skip] … [::-1] reverses order
- Concatenation / Splitting
  - concat = ‘x’ + ‘.’ + ‘y’ + ‘.’ + str(55)
  - sep = concat.split(‘.’) … produces [‘x’, ‘y’, ‘55’]
  - concat.strip / .lstrip / .rstrip … removes whitespace
- Conversion:
  - string = bin(255) or hex(255) … produces ‘0b11111111’ or ‘0xff’
  - num = int(‘0b11111111’,2) or int(‘0xff’,16) … produces 255

For example, if a Python3 script has retrieved a sequence of transmitted “text” bytes from a connected socket, the data will be in “simple string” format, and must be converted to the Python3 internal unicode format for further processing using string mechanisms.

```
# Receive ASCII data from the connected socket
rec_data = conn.recv(1024) … rec_data is b’whatever’ and type(rec_data) is “bytes”
# Convert from ASCII to unicode
dum = rec_data.decode('UTF-8') … dum is ’whatever’ and type(dum) is “str”
# Convert from unicode to ASCII
```

```
dum2 = dum.encode('UTF-8') … dum2 is b'whatever' and type(dum2) is "bytes"
dum3 = dum2[::-1] … dum3 is dum2 with bytes reversed (eg. 'abcd' becomes 'dcba')
sent_data = dum3
conn.send(sent_data)
```

Formatting Strings

Python3 can manipulate lists of strings on-the-fly using embedded formatting.

- Repeat: print('-' *40)
- Format:
  - print('a is {a:10} and b is {b:10}'.format(a=var_a, b=var_b))
  - print('positional list: {:<5}.{:^5}.{:^5}.{:>5}]'.format(*ip_octets))

Command-Line Arguments

```
import sys
if len(sys.argv) == 2:
   s_host = sys.argv[1]
if len(sys.argv) == 3:
   s_port = sys.argv[2]
```