010 Python Bytes, Bytearray

Bytes, Bytearray

Python supports a range of types to store sequences. There are six sequence types: strings, byte sequences (bytes objects), byte arrays (bytearray objects), lists, tuples, and range objects.

Strings contain Unicode characters. Their literals are written in single or double quotes : 'python', "data". Bytes and bytearray objects contain single bytes – the former is immutable while the latter is a mutable sequence. Bytes objects can be constructed the constructor, bytes(), and from literals; use a b prefix with normal string syntax: b'python'. To construct byte arrays, use the bytearray() function.

Bytes literals

```
bytesliteral   ::=  bytesprefix(shortbytes | longbytes)
bytesprefix    ::=  "b" | "B" | "br" | "Br" | "bR" | "BR"
shortbytes     ::=  "'" shortbytesitem* "'" | '"'
shortbytesitem* '"'
longbytes      ::=  "'''" longbytesitem* "'''" | '"""'
longbytesitem* '"""'
shortbytesitem ::=  shortbyteschar | bytesescapeseq
longbytesitem  ::=  longbyteschar | bytesescapeseq
shortbyteschar ::=  <any ASCII character except "\" or newline
or the quote>
longbyteschar  ::=  <any ASCII character except "\">
bytesescapeseq ::=  "\" <any ASCII character>
```

bytes() and bytearray() functions

bytes() function:

Return a new "bytes" object, which is an immutable sequence of small integers in the range 0 <= x < 256, print as ASCII characters when displayed. bytes is an

immutable version of bytearray – it has the same non-mutating methods and the same indexing and slicing behavior.

Syntax:

```
bytes([source[, encoding[, errors]]])
```

bytearray() function :

Return a new array of bytes. The bytearray type is a mutable sequence of integers in the range 0 <= x < 256. It has most of the usual methods of mutable sequences, described in Mutable Sequence Types, as well as most methods that the bytes type has, see Bytes and Byte Array Methods.

Syntax:

```
bytearray([source[, encoding[, errors]]])
```

The optional source parameter can be used to initialize the array in a few different ways:

- If it is a string, you must also give the encoding (and optionally, errors) parameters; bytearray() then converts the string to bytes using str.encode().
- If it is an integer, the array will have that size and will be initialized with null bytes.
- If it is an object conforming to the buffer interface, a read-only buffer of the object will be used to initialize the bytes array.
- If it is iterable, it must be an iterable of integers in the range 0 <= x < 256, which are used as the initial contents of the array.

Without an argument, an array of size 0 is created.

Create a bytes object in Python

Example-1 :

Code :

```
>>> x = b"Bytes objects are immutable sequences of single bytes"
>>> print(x)
```

```
b'Bytes objects are immutable sequences of single bytes'
>>>
```

Example-2:

Code:

```
#triple single or double quotes allows multiple lines
x = b'''Python Tutorial,
Javascript Tutorial,
MySQL Tutorial'''
print(x)
```

Output:

```
b'Python Tutorial,\nJavascript Tutorial,\nMySQL Tutorial'
```

Example-3 :

Code :

```
#created from a iterable of ints, string, bytes or buffer objects.
x = bytes('Python, bytes', 'utf8')
print(x)
```

Output:

```
b'Python, bytes'
```

Convert bytes to string

Example-1:

Code:

```
#create a bytes object

x = b'El ni\xc3\xb1o come camar\xc3\xb3n'

print(x)
```

Output:

```
b'El ni\xc3\xb1o come camar\xc3\xb3n'
```

Example-2:

Code:

```
# create a string using the decode() method of bytes.

#This method takes an encoding argument, such as UTF-8, and optionally
an errors argument.

x = b'El ni\xc3\xb1o come camar\xc3\xb3n'

s = x.decode()

print(type(s))

print(s)
```

Output:

```

El niño come camarón
```

Example-3:

Code:

```
#create a bytes object encoded using 'cp855'

x = b'\xd8\xe1\xb7\xeb\xa8\xe5 \xd2\xb7\xe1'

print(x)

#return a string using decode 'cp855'

y = x.decode('cp855')

print(y)
```

Output:

```
b'\xd8\xe1\xb7\xeb\xa8\xe5 \xd2\xb7\xe1'
привет мир
```

Convert hex string to bytes

Example-1:

Code :

```
#create a string with hexadecimal data

x = '45678c6c56f205876f72c64'

print(x)
```

Output:

```
45678c6c56f205876f72c64
```

Example-2:

Code :

```
#this class method returns a bytes object, decoding the given string
object.

#the string must contain two hexadecimal digits per byte.

x = '45678c6c56f205876f72c64'

y = bytes.fromhex(x)
```

Output:

```
b'.\xf0\xf1\xf2'
```

Numeric code representing a character of a bytes object in Python

Example-1:

Code:

```
#return an integer representing the Unicode code point of that character.

x = ord(b'm')

print(x)
```

Output:

```
109
```

Example-2:

Code:

```
#create a bytes object

y = b'Python bytes'

#generates a list of codes from the characters of bytes

z = list(y)

print(z)
```

Output:

```
[80, 121, 116, 104, 111, 110, 32, 98, 121, 116, 101, 115]
```

Define a mapping table characters for use with a bytes object in Python

Example-1:

Code:

```
#create a str

x = b'Python mapping table characters'

print(x)
```

Output:

```
b'Python mapping table characters'
```

Example-2:

Code:

```
b_table = bytes.maketrans(b'abcdef', b'uvwxyz')

print(type(b_table))

print(b_table)
```

Output:

```
<class 'bytes'>

b'\x00\x01\x02\x03\x04\x05\x06\x07\x08\t\n\x0b\x0c\r\x0e\x0f\x10
\x11\x12\x13\x14\x15\x16\x17\x18\x19\x1a\x1b\x1c\x1d\x1e\x1f
!"#$%
&\'()*+,-
./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\\]^_`uvwxyzghijkl
mnopqrstuvwxyz{|}~\x7f\x80\x81\x82\x83\x84\x85\x86\x87\x88\x89\x
8a\x8b\x8c\x8d\x8e\x8f\x
90\x91\x92\x93\x94\x95\x96\x97\x98\x99\x9a\x9b\x9c\x9d\x9e\x9f\x
a0\xa1\xa2\xa3\xa4\xa5\xa6\xa7\xa8\xa9\xaa\xab\xac\xad\xae\xaf\x
b0\xb1\
xb2\xb3\xb4\xb5\xb6\xb7\xb8\xb9\xba\xbb\xbc\xbd\xbe\xbf\xc0\xc1\
xc2\xc3\xc4\xc5\xc6\xc7\xc8\xc9\xca\xcb\xcc\xcd\xce\xcf\xd0\xd1\
xd2\xd3
\xd4\xd5\xd6\xd7\xd8\xd9\xda\xdb\xdc\xdd\xde\xdf\xe0\xe1\xe2\xe3
\xe4\xe5\xe6\xe7\xe8\xe9\xea\xeb\xec\xed\xee\xef\xf0\xf1\xf2\xf3
\xf4\xf
5\xf6\xf7\xf8\xf9\xfa\xfb\xfc\xfd\xfe\xff'
```

Example-3:

Code:

```
b_table = bytes.maketrans(b'abcdef', b'uvwxyz')

str = 'Write a Python function to find a distinct pair of numbers
whose product is odd from a sequence of integer values.'

b_new = str.translate(b_table)

print(b_new)
```

Output:

```
Writy u Python zunwtion to zinx u xistinwt puir oz numvyrs whosy
proxuwt is oxx zrom u syquynwy oz intygyr vuluys.
```

Convert bytes to hex in Python

```
>>> import binascii

>>> binascii.hexlify("Python".encode("utf8"))

b'507974686f6e'

>>> binascii.unhexlify(_).decode("utf8")

'Python'

>>>
```

How to get the character from the numeric code in bytes objects in Python

```
>>> #this method return a single character based on the integer value.

>>> x = chr(60)
```

```
>>> print(x)

<

>>> x = chr(50)

>>> print(x)

2

>>> #create a list with integers in the range 0 through 255

>>> y = [70, 111, 106, 94, 101, 100, 22, 95, 105, 22, 91, 87, 125,
135]

>>> print(y)

[70, 111, 106, 94, 101, 100, 22, 95, 105, 22, 91, 87, 125, 135]

>>> #create a bytes object from a list of integers in the range 0
through 255.

>>> z = bytes(y)

>>> print(z)

b'Foj^ed\x16_i\x16[W}\x87'

>>>
```

Determine the length of a bytes object in Python

```
>>> #create a string

>>> x = "Python, Bytes"

>>> print(x)

Python, Bytes

>>> #know the length of the string using the len() function

>>> print(len(x))

13
```

```
>>> #create a bytes object
>>> y = bytes(x, "utf8")
>>> print(y)
b'Python, Bytes'
>>> #know the length of the bytes object using the len() function
>>> print(len(y))
13
>>>
```

Use the operators + and * with bytes objects in Python

```
>>> #create a bytes object
>>> x = b"byte 213"
>>> print(x)
b'byte 213'
>>> #The * operator allow repeat the characters of a bytes object
>>> print(x * 5)
b'byte 213byte 213byte 213byte 213byte 213'
>>> #create two bytes objects.
>>> x1 = bytes([70, 111, 106, 94, 101, 100, 22, 95, 105, 22, 91, 87,
125, 135])
>>> x2 = b"Python"
>>> #The + operator allow create a new bytes object joining two or
more bytes.
>>> x = x1 + x2
>>> print(x)
```

```
b'Foj^ed\x16_i\x16[W}\x87Python'

>>> #create a bytes object combining operators

>>> x = b"Python" + b"Bytes" * 3 + b"$"

>>> print(x)

b'PythonBytesBytesBytes$'

>>>
```

How to get a byte from a bytes object in Python?

```
>>> y = [80, 121, 116, 104, 111, 110, 32, 105, 115, 32, 101, 97, 115,
121]

>>> print(y)

[80, 121, 116, 104, 111, 110, 32, 105, 115, 32, 101, 97, 115, 121]

>>> #create a bytes object

>>> x1 = bytes([70, 111, 106, 94, 101, 100, 22, 95, 105, 22, 91, 87,
125, 135])

>>> print(x1)

b'Foj^ed\x16_i\x16[W}\x87'

>>> #is similar to the handling of lists, the index is defined in
brackets

>>> x = y[3]

>>> print(x)

104

>>> print(chr(x))

h

>>> #can also use negative indices to get a byte from bytes object
```

```
>>> x = [-8]
>>> print(x)
[-8]
>>> x = y[-8]
>>> print(x)
110
>>> print(chr(x))
n
>>>
```

Create a bytearray object in Python

```
>>> #create a bytearray from a bytes object
>>> x = bytearray(b"Python Bytes")
>>> print(x)
bytearray(b'Python Bytes')
>>> #create a bytearray from a string defining the standard of coding
>>> x = bytearray("Python Bytes", "utf8")
>>> print(x)
bytearray(b'Python Bytes')
>>> #create a bytearray from a list of integers in the range 0 through
255
>>> x = bytearray([94, 91, 101, 125, 111, 35, 120, 101, 115, 101,
200])
>>> print(x)
bytearray(b'^[e}o#xese\xc8')
```

```
>>>
```

Difference between bytes and bytearray object in Python

```
>>> #bytearray objects are a mutable counterpart to bytes objects
>>> x = bytearray("Python bytearray", "utf8")
>>> print(x)
bytearray(b'Python bytearray')
>>> #can remove items from the bytes
>>> del x[11:15]
>>> print(x)
bytearray(b'Python bytey')
>>> #can add items from the bytes
>>> x[11:15] = b" object"
>>> print(x)
bytearray(b'Python byte object')
>>> #can use the methods of mutable type iterable objects as the lists
>>> x.append(45)
>>> print(x)
bytearray(b'Python byte object-')
>>>
```

Convert a bytes to bytearray

```
>>> #create a bytes object from a list of integers in the range 0
through 255

>>> x = bytes([105, 100, 107, 112, 132, 118, 107, 112, 200])

>>> print(x)

b'idkp\x84vkp\xc8'

>>> #generates a new array of bytes from a bytes object

>>> x1 = bytearray(x)

>>> print(x1)

bytearray(b'idkp\x84vkp\xc8')

>>>
```

Slice of a bytes object in Python

```
>>> #create a bytes object

>>> x = b"Python slice"

>>> print(x)

b'Python slice'

>>> #b[start:stop] the start index is inclusive and the end index is
exclusive.

>>> x1 = x[2:6]

>>> print(x1)

b'thon'

>>> #if the start index isn't defined, is starts from the beginning

>>> x1 = x[-5:]

>>> print(x1)

b'slice'
```

```
>>> #if the end index isn't defined, it goes until the end
>>> x1 = x[:4]
>>> print(x1)
b'Pyth'
>>> #if neither is defined, returns the full bytes object
>>> x1 = x[:]
>>> print(x1)
b'Python slice'
>>>
```

Difference between bytes and string object

```
>>> # bytes objects are immutable sequences of integers, each value in
the sequence
>>> # string objects are immutable sequences of unicode characters.
>>> x = "Python String"
>>> y = b"Python String"
>>> print(x)
Python String
>>> print(y)
b'Python String'
>>> # Found in unicode representation of characters but not in ascii
>>> x = "Python"
>>> y = bytes("Python", "utf8")
>>> print(x)
```

```
Python
>>> print(y)
b'Python'
>>>
```