Konwersja typu danych

int(x [,base])	konwertuje x na liczbę całkowitą, base określa podstawę, jeśli x jest łańcuchem
float(x)	x na liczbę zmiennoprzecinkową
complex(real [,imag])	tworzy liczbę zespoloną
str(x)	x na łąńcuch
tuple(s)	konwertuje na krotkę
list(s)	Konwertuje na listę
dict(d)	tworzy słownik, d musi być sekwencją krotek (klucz,wartość)
frozenset(s)	Converts s to a frozen set
chr(x)	konwertuje liczbę całkowitą na znak
unichr(x)	konwertuje liczbę całkowitą na znak Unicode
ord(x)	konwertuje pojedynczy znak na jego wartość całkowitą
hex(x)	konwertuje liczbę całkowitą na ciąg szesnastkowy
oct(x)	konwertuje liczbę całkowitą na ciąg ósemkowy

`struct` — interpretuje bajty jako spakowane dane binarne¶

Wykonuje konwersje między wartościami Pythona a strukturami C reprezentowanymi jako obiekty bajtowe Pythona. Może być stosowany w zakresie obsługi danych binarnych przechowywanych w plikach lub z połączeń sieciowych,czy między innymi źródłami. Używa ciągów formatujących jako zwięzłych opisów układu struktur C i zamierzonej konwersji do/z wartości Pythona.

`struct.pack(format, v1, v2, ...)`
Zwraca obiekt bajtów zawierający wartości `v1, v2, ...` spakowane zgodnie z formatem ciągu formatu. Argumenty muszą dokładnie odpowiadać wartościom wymaganym przez `format`.

`struct.unpack(format, bufor)`
rozpakuje z bufora bufora (przypuszczalnie spakowany przez `pack(format, ...)`) zgodnie z formatem łańcucha formatującego. Wynikiem jest `krotka`, nawet jeśli zawiera dokładnie jeden element. Rozmiar bufora w bajtach musi odpowiadać rozmiarowi wymaganemu przez format, co odzwierciedla funkcja `calcsize()`.

`struct.calcsize(format)`
zwraca rozmiar struktury (czyli obiektu bajtów utworzonego przez `pack(format, ...)`) odpowiadającego formatowi ciągu formatującego.

Domyślnie typy są reprezentowane w natywnym formacie maszyny i kolejności bajtów, ale pierwszy znak ciągu formatującego może być użyty do wskazania kolejności bajtów, rozmiaru i wyrównania spakowanych danych:

znak	kolejność bajtów	rozmiar	wyrównanie
@	native	native	native
=	native	standard	none
<	little-endian	standard	none
>	big-endian	standard	none
!	network (= big-endian)	standard	none

Jeśli pierwszy znak nie jest jednym z tych, zakłada się „@”.

Znaki formatujące:

Format	C Type	Python type	Standard size
x	pad byte	no value	
c	char	bytes of length 1	1
b	signed char	integer	1
B	unsigned char	integer	1
?	_Bool	bool	1
h	short	integer	2
H	unsigned short	integer	2
i	int	integer	4
I	unsigned int	integer	4
l	long	integer	4
L	unsigned long	integer	4
q	long long	integer	8
Q	unsigned long long	integer	8
n	ssize_t	integer	
N	size_t	integer	
f	float	float	4
d	double	float	8
s	char[]	bytes	
p	char[]	bytes	
P	void *	integer	

Kolejność bajtów:

W sytuacjach, kiedy liczby całkowite lub jakiekolwiek inne dane zapisywane są przy użyciu wielu (przynajmniej dwóch) bajtów, nie istnieje jeden unikatowy sposób uporządkowania tych bajtów w pamięci lub w czasie transmisji przez dowolne medium i musi być użyta jedna z wielu konwencji ustalająca kolejność bajtów (ang. byte order lub endianness). Jest to analogiczne do zapisu pozycyjnego liczb lub kierunku pisma w różnych językach – ze strony lewej na prawą albo z prawej na lewą.

Big endian

Big endian (spotykane także grubokońcowość) to forma zapisu danych, w której najbardziej znaczący bajt (zwany też górnym bajtem, z ang. high-order byte) umieszczony jest jako pierwszy.

Procesory, które używają formy big endian, to między innymi HP Intel Itanium, SPARC, Motorola 68000, PowerPC 970, IBM System/360, Siemens SIMATIC S7. Jest ona analogiczna do używanego na co dzień przez ludzi sposobu zapisu liczb.

Procesor zapisujący 32-bitowe wartości w pamięci, przykładowo 0x4A3B2C1D pod adresem 100, umieszcza dane, zajmując adresy od 100 do 103 w następującej kolejności:

	100	101	102	103	
...	4A	3B	2C	1D	...

Przykładowe formaty plików, które zawierają dane w formacie big-endian:

- Adobe Photoshop
- JPEG
- MacPaint
- Sun raster files

Little endian

Little endian (spotykane także cienkokońcowość) to forma zapisu danych, w której najmniej znaczący bajt (zwany też dolnym bajtem, z ang. low-order byte) umieszczony jest jako pierwszy. Procesory, które używają formy little endian, to między innymi wszystkie z rodziny x86, DEC VAX. Jest ona odwrotna do używanego na co dzień sposobu zapisu liczb.

Procesor zapisujący 32-bitowe wartości w pamięci, przykładowo 0x4A3B2C1D pod adresem 100, umieszcza dane zajmując adresy od 100 do 103 w następującej kolejności:

	100	101	102	103	
...	1D	2C	3B	4A	...

Przykładowe formaty plików, które zawierają dane w formacie little-endian:

- Windows Bitmap
- GIF
- PC Paintbrush
- RTF by Microsoft