MicroPython:

Standardowe biblioteki Python, biblioteki specyficzne dla MicroPython i moduły specyficzne dla Pycom, które są dostępne na urządzeniach Pycom.

Standardowe biblioteki Pythona zostały poddane „mikronizacj”, aby pasowały do filozofii MicroPython. Zapewniają podstawową funkcjonalność tego modułu i mają stanowić bezpośredni zamiennik standardowej biblioteki Pythona.

Niektóre moduły są dostępne pod nazwą **u-name**, a także pod nazwą **non-u-name**. **non-u-name** może zostać zastąpiona przez plik o tej nazwie w ścieżce pakietu. Na przykład import json najpierw wyszuka plik `json.py` lub katalog `json` i załaduje ten pakiet, jeśli zostanie znaleziony. Jeśli nic nie zostanie znalezione, nastąpi powrót do załadowania wbudowanego modułu `ujson`.

Stałe (https://docs.pycom.io/firmwareapi/micropython/sys/)

`sys.argv` - zmienna lista argumentów, z którymi uruchomiono bieżący program.
`sys.byteorder` - kolejność bajtów systemu (“little” or “big”)
`sys.implementation` - obiekt z informacjami o aktualnej implementacji Pythona,
atrybuty w przypadku MicroPythona:
`name` - napis „micropython”
`version` - krotka (major, minor, micro), np. (1, 7, 0)
`sys.maxsize` - maksymalna wartość, jaką natywny typ liczby całkowitej może przechowywać na bieżącej platformie.

```
>>> import sys
>>> sys.byteorder
'little'
>>> sys.argv
[]
>>> sys.implementation
(name='micropython', version=(1, 8, 6))
>>> sys.maxsize
2147483647
```

`sys.modules` - słownik załadowanych modułów
`sys.path` - zmienna lista katalogów do wyszukiwania importowanych modułów
`sys.platform` - platforma, na której działa MicroPython
`sys.stderr` - standardowy strumień błędów.
`sys.stdin` - standardowy strumień wejściowy.
`sys.stdout` - standardowy strumień wyjściowy.
`sys.version` - wersja języka Python, z którą ta implementacja jest zgodna, jako ciąg.
`sys.version_info` - wersja językowa Pythona, z którą jest zgodna ta implementacja, jako krotka int.

```
>>> sys.modules
{}
>>> sys.path
['', '/flash', '/flash/lib']
>>> sys.platform
'FiPy'
```

```
>>> sys.stderr
<io.FileIO 2>
>>> sys.stdin
<io.FileIO 0>
>>> sys.stdout
<io.FileIO 1>
>>> sys.version
'3.4.0'
>>> sys.version_info
(3, 4, 0)
```

UBINASCII

Moduł realizuje konwersje między danymi binarnymi i ich różnymi kodowaniami w postaci ASCII (w obu kierunkach).

`ubinascii.hexlify(data[, sep])`

Konwertuje dane binarne na reprezentację szesnastkową. Zwraca ciąg bajtów.

> **Różnica w stosunku do CPython**
> Jeśli podano dodatkowy argument, `sep`, jest on używany jako separator między wartościami szesnastkowymi.

`ubinascii.unhexlify(data)`

Konwertuje dane szesnastkowe na reprezentację binarną. Zwraca ciąg bajtów. (odwrotność `hexlify`)

`ubinascii.a2b_base64(data)`

Konwertuje dane zakodowane w formacie Base64 na reprezentację binarną. Zwraca ciąg bajtów.

`ubinascii.b2a_base64(data)`

Koduje dane binarne w formacie Base64. Zwraca łańcuch.

UJSON

Moduł pozwala na konwersję pomiędzy obiektami Pythona a formatem danych JSON.

`ujson.dumps (obj)`

Zwróci obj reprezentowane jako ciąg JSON.

`ujson.loads (str)`

Przeanalizuje str JSON i zwróci obiekt. Podnosi błąd ValueError, jeśli łańcuch nie jest poprawnie utworzony.

`ujson.load (fp)`

Przeanalizuje zawartość fp (a .read () - obsługujący obiekt plikopodobny zawierający dokument JSON). Podnosi ValueError, jeśli treść nie jest poprawnie sformułowana.