1. Bibliotecas estándar de Python

MicroPython es compatible con una parte de la biblioteca estándar de Python, aunque con algunas limitaciones. Las bibliotecas que están disponibles y sus métodos principales son:

`os`

- **`chdir(path)`**: Cambia el directorio de trabajo actual.
- **`getcwd()`**: Devuelve el directorio de trabajo actual.
- **`listdir([path])`**: Lista los archivos y directorios en un directorio dado.
- **`mkdir(path)`**: Crea un nuevo directorio.
- **`remove(path)`**: Elimina un archivo.
- **`rename(old, new)`**: Renombra un archivo o directorio.
- **`rmdir(path)`**: Elimina un directorio.
- **`stat(path)`**: Devuelve información sobre un archivo o directorio.
- **`utime([path, time])`**: Cambia la hora de modificación de un archivo o directorio.

`sys`

- **`exit([status])`**: Sale del programa con el código de estado especificado.
- **`getsizeof(object)`**: Devuelve el tamaño en bytes de un objeto.
- **`path`**: Lista de rutas de búsqueda de módulos.
- **`platform`**: Devuelve el nombre de la plataforma (e.g., 'esp32').
- **`version`**: Devuelve la versión de MicroPython.

`time`

- **`sleep(seconds)`**: Suspende la ejecución durante el número de segundos especificado.
- **`time()`**: Devuelve el tiempo actual en segundos desde el 1 de enero de 1970.
- **`ticks_ms()`**: Devuelve el tiempo en milisegundos desde que se inició el dispositivo.
- **`ticks_us()`**: Devuelve el tiempo en microsegundos.
- **`ticks_diff(tick1, tick2)`**: Calcula la diferencia entre dos ticks de tiempo.
- **`ticks_add(tick, delta)`**: Suma un valor de tiempo a un tick de tiempo.

`math`

- **`acos(x)`**: Retorna el arcocoseno de `x`.
- **`asin(x)`**: Retorna el arcoseno de `x`.

- **`atan(x)`**: Retorna el arcotangente de `x`.
- **`atan2(y, x)`**: Retorna el arcotangente de `y/x`, en radianes.
- **`cos(x)`**: Retorna el coseno de `x` (en radianes).
- **`degrees(x)`**: Convierte `x` de radianes a grados.
- **`exp(x)`**: Retorna `e` elevado a la potencia de `x`.
- **`fabs(x)`**: Retorna el valor absoluto de `x`.
- **`floor(x)`**: Redondea `x` hacia abajo al entero más cercano.
- **`fmod(x, y)`**: Retorna el resto de la división de `x` por `y`.
- **`frexp(x)`**: Descompone `x` en su mantisa y exponente.
- **`hypot(x, y)`**: Retorna la hipotenusa de un triángulo con lados `x` e `y`.
- **`isclose(a, b)`**: Comprueba si `a` y `b` son cercanos, útil para comparaciones flotantes.
- **`isfinite(x)`**: Retorna `True` si `x` no es infinito ni NaN.
- **`isinf(x)`**: Retorna `True` si `x` es infinito.
- **`isnan(x)`**: Retorna `True` si `x` es NaN.
- **`log(x, base)`**: Retorna el logaritmo de `x` en la base especificada.
- **`modf(x)`**: Devuelve la parte fraccionaria y la parte entera de `x`.
- **`pow(x, y)`**: Retorna `x` elevado a `y`.
- **`radians(x)`**: Convierte `x` de grados a radianes.
- **`sin(x)`**: Retorna el seno de `x` (en radianes).
- **`sqrt(x)`**: Retorna la raíz cuadrada de `x`.
- **`tan(x)`**: Retorna la tangente de `x` (en radianes).

`random`

- **`seed(a)`**: Inicializa el generador de números aleatorios con la semilla `a`.
- **`getrandbits(k)`**: Retorna un entero con `k` bits aleatorios.
- **`randrange(start, stop[, step])`**: Retorna un número aleatorio en el rango especificado.
- **`randint(a, b)`**: Retorna un número entero aleatorio entre `a` y `b`.
- **`choice(seq)`**: Retorna un elemento aleatorio de la secuencia `seq`.
- **`choices(seq, weights=None, cum_weights=None, k=1)`**: Retorna una lista de elementos aleatorios de `seq`, con pesos opcionales.
- **`shuffle(seq)`**: Mezcla la secuencia `seq` in-place.
- **`sample(seq, k)`**: Retorna una lista de `k` elementos únicos de la secuencia `seq`.

`collections`

- **`Counter(iterable)`**: Crea un contador de elementos.
- **`defaultdict(default_factory)`**: Crea un diccionario con un valor por defecto para claves faltantes.
- **`deque(iterable, maxlen)`**: Crea una lista de doble extremo con una longitud máxima opcional.
- **`OrderedDict()`**: Un diccionario que mantiene el orden de inserción.

- **`namedtuple(typename, field_names, *, rename=False, defaults=None, module='builtins')`**: Crea un tipo de tupla con nombres de campo accesibles.

`re`

- **`compile(pattern, flags=0)`**: Compila una expresión regular.
- **`match(pattern, string, flags=0)`**: Intenta hacer coincidir la expresión regular al principio de la cadena.
- **`search(pattern, string, flags=0)`**: Busca la primera coincidencia de la expresión regular en la cadena.
- **`findall(pattern, string, flags=0)`**: Devuelve todas las coincidencias de la expresión regular en la cadena.
- **`finditer(pattern, string, flags=0)`**: Devuelve un iterador que produce todas las coincidencias de la expresión regular.
- **`sub(pattern, repl, string, count=0, flags=0)`**: Sustituye las coincidencias de la expresión regular en la cadena.
- **`subn(pattern, repl, string, count=0, flags=0)`**: Sustituye y devuelve el número de sustituciones realizadas.
- **`split(pattern, string, maxsplit=0, flags=0)`**: Divide la cadena en partes usando la expresión regular.

`struct`

- **`pack(format, *values)`**: Empaqueta los valores según el formato dado.
- **`unpack(format, string)`**: Desempaqueta los valores de acuerdo con el formato de la cadena.
- **`pack_into(format, buffer, offset, *values)`**: Empaqueta los valores en un buffer a partir de un offset.
- **`unpack_from(format, buffer, offset=0)`**: Desempaqueta los valores del buffer a partir de un offset.

2. Bibliotecas de MicroPython específicas

Estas bibliotecas están diseñadas específicamente para MicroPython y permiten interactuar con el hardware y realizar operaciones de IoT.

`machine`

- **`Pin`**: Para controlar pines GPIO.
- **`Timer`**: Para temporizadores y manejo de interrupciones.

- **`ADC`**: Para la lectura de entradas analógicas.
- **`DAC`**: Para la salida de señales analógicas.
- **`PWM`**: Para la modulación por ancho de pulso.
- **`RTC`**: Para el reloj en tiempo real.
- **`UART`**: Para comunicación serial.
- **`I2C`**: Para comunicación I2C.
- **`SPI`**: Para comunicación SPI.
- **`WDT`**: Para el temporizador de vigilancia.
- **`Freq`**: Para ajustar la frecuencia de la CPU.

`network`

- **`WLAN()`**: Para la configuración y gestión de la red WiFi.
- **`Ethernet()`**: Para la configuración de la red Ethernet (en algunas placas).
- **`connect(ssid, password)`**: Conecta a una red WiFi.
- **`disconnect()`**: Desconecta de la red WiFi.
- **`isconnected()`**: Comprueba si está conectado a la red.
- **`config()`**: Configura parámetros de la red.

`uasyncio`

- **`run(coroutine)`**: Ejecuta una coroutine.
- **`sleep(seconds)`**: Suspende la ejecución de una coroutine.
- **`create_task(coroutine)`**: Crea una tarea para ejecutar una coroutine.
- **`gather(*tasks)`**: Ejecuta múltiples tareas concurrentemente.
- **`wait(tasks)`**: Espera a que todas las tareas se completen.