

Tarea 6 Sistemas Digitales III

Melanie Aponte

Librería Qi

La librería `qi` es parte del NAOqi framework, el entorno de desarrollo creado por SoftBank Robotics para programar robots como Pepper y NAO. Esta librería permite controlar funciones del robot como movimiento, voz, sensores, cámara, etc.

- Comunicación entre módulos.
- Programación asíncrona y remota.
- Interfaces para sensores, actuadores y reconocimiento de voz.

Ejemplo en Python:

```
import qi

def main(session):
    tts = session.service("ALTextToSpeech")
    tts.say("Hola, soy Pepper")

if __name__ == "__main__":
    connection_url = "tcp://192.168.1.1:9559"
    app = qi.Application(["PepperApp", "--qi-url=" +
        ↪ connection_url])
    app.start()
    main(app.session)
```

Librería argparse

utilizada para crear interfaces de línea de comandos. Permite que tu script acepte argumentos cuando se ejecuta desde la terminal. Es especialmente útil para:

- Automatizar tareas desde la consola.
- Personalizar el comportamiento del script sin modificar el código fuente.
- Validar y documentar argumentos fácilmente.

Ejemplo:

```
import argparse

parser = argparse.ArgumentParser(description="Suma dos
    ↪ n meros")
parser.add_argument("a", type=int, help="Primer n mero")
parser.add_argument("b", type=int, help="Segundo n mero")

args = parser.parse_args()
print("La suma es:", args.a + args.b)
```

En terminal:

```
python suma.py 3 5
# Salida: La suma es: 8
```

Librería sys

`sys` es un módulo incorporado de Python que proporciona acceso a variables y funciones que interactúan directamente con el intérprete de Python. Es muy útil cuando se quiere:

- Manejar argumentos desde la línea de comandos.
- Acceder a información del sistema.
- Controlar la salida estándar (`stdout`) y de errores (`stderr`).
- Terminar un programa manualmente.

Usos:

1. `sys.argv` → Lista de argumentos pasados al script desde la línea de comandos.

En consola de Python:

```
import sys
print(sys.argv)
```

En bash:

```
python ejemplo.py hola mundo
# Salida: ['ejemplo.py', 'hola', 'mundo']
```

2. `sys.exit([n])` → Termina la ejecución del programa. Si se pasa un número distinto de 0, se considera error.

En consola de Python:

```
if len(sys.argv) < 2:
    print("Faltan argumentos")
    sys.exit(1)
```

3. `sys.stdin`, `sys.stdout`, `sys.stderr` → Permiten interactuar con las entradas/salidas estándar.

Por ejemplo:

```
import sys
sys.stdout.write("Hola desde stdout\n")
```

4. `sys.version` → Devuelve la versión actual de Python en uso.

```
print(sys.version)
```

Librería os

`os` es un **módulo estándar de Python** que permite interactuar con el **sistema operativo**. Con él puedes realizar operaciones como:

- Manipular archivos y carpetas.
- Obtener información del sistema.
- Ejecutar comandos del sistema.
- Cambiar el entorno de ejecución.

Usos:

1. `os.getcwd()` → **Devuelve** el directorio de trabajo actual.

```
import os
print(os.getcwd())
```

2. `os.chdir(path)` → Cambia el directorio de trabajo.

```
os.chdir("/ruta/a/otro/directorio")
```

3. `os.listdir(path='.')` → Lista archivos y carpetas del directorio especificado.

```
print(os.listdir())
```

4. `os.mkdir("nueva_carpeta")` y `os.makedirs(ruta/mas/larga)`

```
os.mkdir("proyecto")
os.makedirs("proyecto/datos/resultados")
```

5. `os.remove('archivo.txt')` y `os.rmdir(carpeta)` → Elimina archivos o carpetas vacías.

```
os.remove("documento.txt")
os.rmdir("carpeta_vacía")
```

Librería `almath`

`almath` es una librería matemática especializada que forma parte del framework NAOqi. Fue diseñada por Aldebaran Robotics (ahora SoftBank Robotics) y se utiliza para realizar operaciones matemáticas complejas en robótica, especialmente en cinemática, geometría espacial y álgebra lineal.

Esta librería no es parte del Python estándar. Está incluida en el SDK que se instala junto con el entorno de desarrollo para robots Pepper o NAO.

Funcionalidades:

- Transformaciones homogéneas (matrices 4x4).
- Operaciones con vectores y posiciones 3D.
- Conversión entre rotaciones (matriz, ángulo-eje, quaternion, etc.).
- Cálculos de distancia y orientación entre puntos en el espacio.
- Cinemática inversa y directa.

Ejemplo de uso:

```
import almath

# Crear una posición
pos1 = almath.Position2D(1.0, 2.0, 0.5)

# Crear otra posición
pos2 = almath.Position2D(2.0, 3.0, 1.0)

# Sumar dos posiciones
pos3 = pos1 + pos2

print("Resultado:", pos3.x, pos3.y, pos3.theta)
```

Librería math

La librería math es un módulo estándar en Python que proporciona funciones matemáticas definidas en el estándar de C. Es ideal para cálculos de precisión con números de punto flotante (float) y operaciones matemáticas comunes.

```
>>> import math
>>> math.sqrt(16)      # 4.0
4.0
>>> math.pow(2, 3)     # 8.0
8.0
>>> math.exp(2)        # e^2 ≈ 7.389
7.38905609893065
>>> math.log(10)       # log base e
2.302585092994046
>>> math.log10(1000)
3.0
```

Figura 1: Implementacion en python.

- Funciones trigonométricas.

```
>>> math.sin(math.pi/2)
1.0
>>> math.cos(0)
1.0
>>> math.tan(math.pi/4)
0.9999999999999999
>>> math.degrees(math.pi)
180.0
>>> math.radians(180)
3.141592653589793
```

Figura 2: DImplementacion en python.

- Constantes como `pi`, `e`.

```
>>> math.pi
3.141592653589793
>>> math.e
2.718281828459045
>>> math.tau
6.283185307179586
```

Figura 3: Implementacion en python.

- Funciones de redondeo y valor absoluto.

```
>>> math.ceil(4.3)
5
>>> math.floor(4.7)
4
>>> math.fabs(-5.5)
5.5
>>> math.trunc(7.9)
7
```

Figura 4: Implementacion en python.

- Factoriales y combinatoria

```
>>> math.factorial(5)
120
>>> math.comb(5, 2)
10
>>> math.perm(5, 2)
20
```

Figura 5: Implementacion en python.

Librería motion

motion permite controlar el movimiento de robots NAO o Pepper.

```
import qi

def main(session):
    motion_service = session.service("ALMotion")
    posture_service = session.service("ALRobotPosture")
    posture_service.goToPosture("StandInit", 0.5)
    motion_service.moveTo(0.5, 0.0, 0.0)

if __name__ == "__main__":
    app = qi.Application(["MotionExample", "--qi-url=tcp
    ↪ ://192.168.1.1:9559"])
    app.start()
    main(app.session)
```

Posturas comunes:

- StandInit: Postura de pie inicial
- Stand: Postura de pie lista
- Crouch: Agachado
- Sit, SitRelax: Sentado
- LyingBack, LyingBelly: Acostado

Librería http.client (antes httplib)

Httplib fue una librería estándar en Python 2.x para realizar conexiones HTTP y HTTPS desde cliente (cliente web) hacia servidores. Permitía realizar peticiones como GET, POST, entre otras. httplib era usada en Python 2. En Python 3 se usa `http.client`.

¿Qué hace?

Proporcionaba una API de bajo nivel para trabajar directamente con sockets HTTP, permitiendo:

- Establecer conexiones HTTP/HTTPS.
- Enviar solicitudes manualmente.
- Leer respuestas crudas del servidor.

Ejemplo en Python 3:

```
import http.client
conn = http.client.HTTPSConnection("www.example.com")
conn.request("GET", "/")
response = conn.getresponse()
print(response.status)
print(response.getheaders())
print(response.read().decode())
```

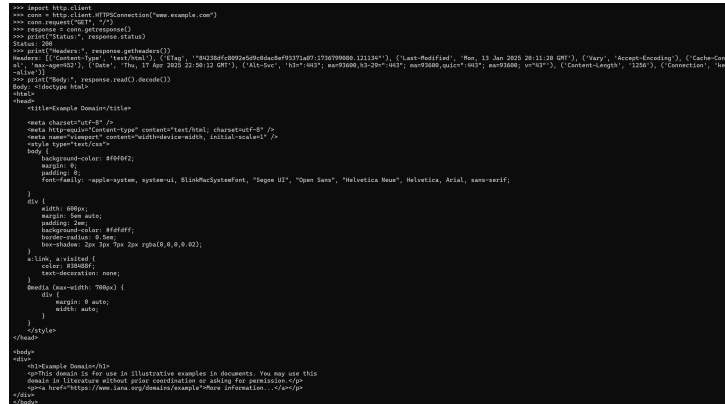


Figura 6: Interfaz en consola para uso de http.client.

Código usado:

```
import http.client

conn = http.client.HTTPSConnection("www.example.com")
conn.request("GET", "/")
response = conn.getresponse()
```

```
print("Status:", response.status)
print("Headers:", response.getheaders())
print("Body:", response.read().decode())
conn.close()
```

Librería json

La librería json es un módulo estándar de Python que permite trabajar con datos en formato JSON (JavaScript Object Notation). Este formato es ampliamente utilizado para almacenar y transferir datos entre aplicaciones web y APIs. El módulo permite codificar (serializar) y decodificar (deserializar) estructuras de datos de Python como diccionarios, listas, etc.

Conversión de Python a JSON:

1. `json.dumps()`: convierte un objeto de Python a una cadena JSON.

```
>>> import json
>>> data = {"nombre": "Derly", "edad": 28}
>>> json_str = json.dumps(data)
>>> print(json_str)
{"nombre": "Derly", "edad": 28}
```

Figura 7: Ejemplo de conversión de objetos de Python a JSON.

2. `json.dump()`: escribe un objeto JSON directamente en un archivo.

```
>>> with open("datos.json", "w") as f:
...     json.dump(data, f)
```

Figura 8: Ejemplo de escritura de JSON en archivo con `json.dump()`.

```
>>> json_str = '{"nombre": "Derly", "edad": 28}'
>>> data = json.loads(json_str)
>>> print(data["nombre"])
Derly
```

Figura 9: Ejemplo de lectura de JSON con `json.loads()`.

Conversión de JSON a Python:

1. `json.loads()`: convierte una cadena JSON a un objeto Python.

```
import json

data = {"nombre": "Pepper", "edad": 5}
json_str = json.dumps(data)
print(json_str)

data2 = json.loads(json_str)
print(data2)
```

1. Secuencia de pasos en Coreographie

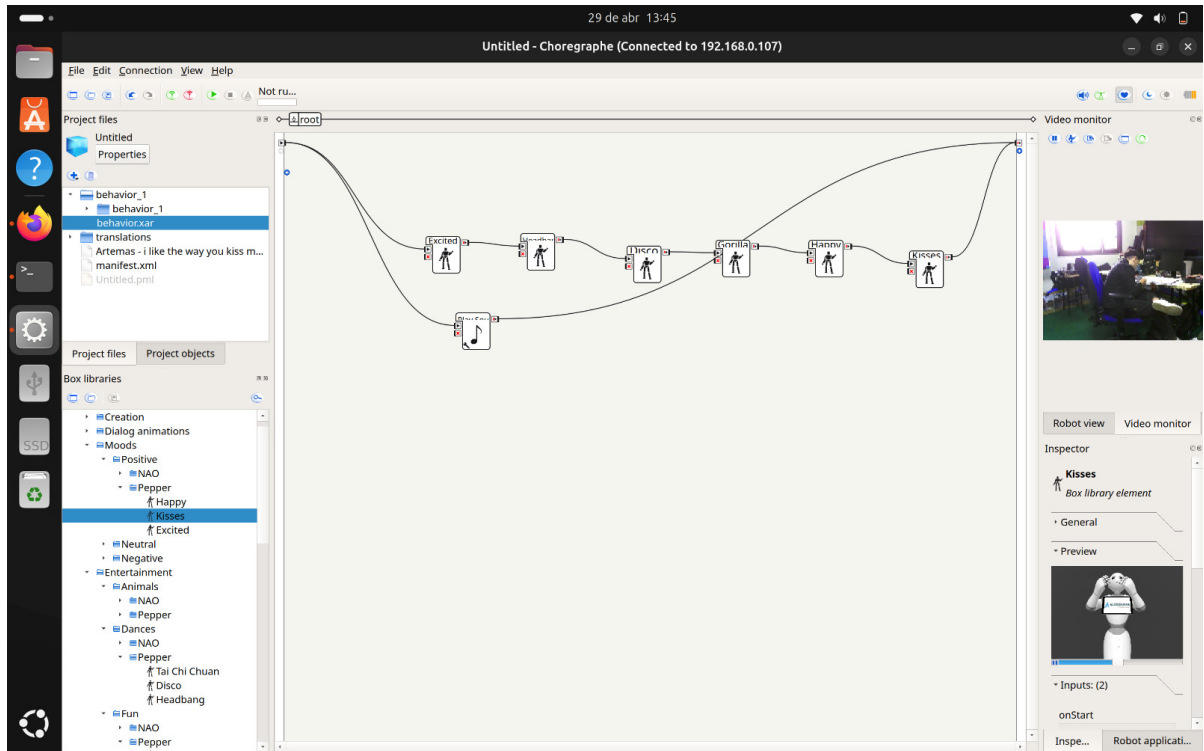
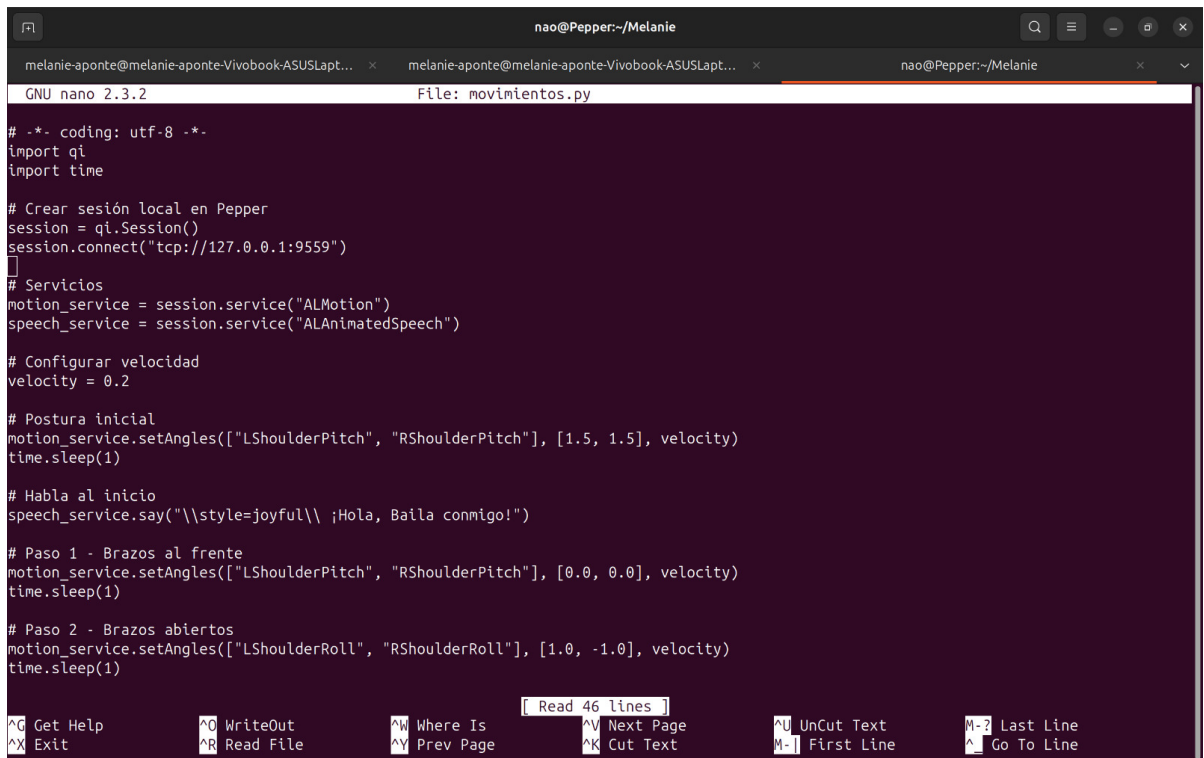


Figura 10: Coreografia en coreograph

2. Movimientos en consola

Para hacer esto por medio de la consola, me conecte a Pepper por medio de ssh `nao@192.168.0.107`, cree una carpeta llamada Melanie y genero un archivo llamado `movimientos.py`, para ejecutarlo lo hice con "Python2 movimientos.py".



```
# -*- coding: utf-8 -*-
import qi
import time

# Crear sesión local en Pepper
session = qi.Session()
session.connect("tcp://127.0.0.1:9559")

# Servicios
motion_service = session.service("ALMotion")
speech_service = session.service("ALAnimatedSpeech")

# Configurar velocidad
velocity = 0.2

# Postura inicial
motion_service.setAngles(["LShoulderPitch", "RShoulderPitch"], [1.5, 1.5], velocity)
time.sleep(1)

# Habla al inicio
speech_service.say("\\style=joyful\\ ¡Hola, Baila conmigo!")

# Paso 1 - Brazos al frente
motion_service.setAngles(["LShoulderPitch", "RShoulderPitch"], [0.0, 0.0], velocity)
time.sleep(1)

# Paso 2 - Brazos abiertos
motion_service.setAngles(["LShoulderRoll", "RShoulderRoll"], [1.0, -1.0], velocity)
time.sleep(1)
```

Figura 11: Código .py

Bibliografía

1. SoftBank Robotics, *Python SDK - NAOqi for Python*, Aldebaran Documentation, [Online]. Available: https://doc.aldebaran.com/2-5/dev/python/intro_python.html. [Accessed: Apr. 2, 2025].
2. SoftBank Robotics, NAOqi Developer Guide - Index, [Online]. Available: https://doc.aldebaran.com/2-5/index_dev_guide.html. [Accessed: Apr. 2, 2025].
3. Python Software Foundation, *argparse — Parser for command-line options, arguments and sub-commands*, Python 3 Documentation, [Online]. Available: <https://docs.python.org/3/library/argparse.html>. [Accessed: Apr. 2, 2025].

4. Python Software Foundation, *sys* — *System-specific parameters and functions*, Python 3 Documentation, [Online]. Available: <https://docs.python.org/3/library/sys.html>. [Accessed: Apr. 2, 2025].
5. Python Software Foundation, *os* — *Miscellaneous operating system interfaces*, Python 3 Documentation, [Online]. Available: <https://docs.python.org/3/library/os.html>. [Accessed: Apr. 16, 2025].
6. SoftBank Robotics, *almath* — *Aldebaran Math Library*, NAOqi Developer Documentation, [Online]. Available: <https://doc.aldebaran.com/2-5/naoqi/motion/control-cartesian.html#almath-overview>. [Accessed: Apr. 16, 2025].
7. Python Software Foundation, *math* — *Mathematical functions*, Python 3 Documentation, [Online]. Available: <https://docs.python.org/3/library/math.html>. [Accessed: Apr. 5, 2025].
8. SoftBank Robotics, *ALMotion* — *NAOqi Motion Module*, Developer Documentation, [Online]. Available: <https://doc.aldebaran.com/2-5/naoqi/motion/index.html>. [Accessed: Apr. 14, 2025].
9. Python Software Foundation, *http.client* — *HTTP protocol client*, Python 3 Documentation, [Online]. Available: <https://docs.python.org/3/library/http.client.html>. [Accessed: Apr. 16, 2025].
10. Python Software Foundation, *json* — *JSON encoder and decoder*, Python 3 Documentation, [Online]. Available: <https://docs.python.org/3/library/json.html>. [Accessed: Apr. 16, 2025].