Práctica 2: Sockets en Python

Sistemas Distribuidos

Curso: 2021/2022

Índice

1. Información general de la práctica

- La práctica se puede realizar de forma conjunta con un compañero/a. Ambos integrantes deben pertenecer al mismo grupo de prácticas.
- Se admiten trabajos individuales, pero no de tres estudiantes.
- En el Campus Virtual se encuentra habilitada una tarea para realizar la entrega.
- La fecha de entrega establecida debe consultarse en dicha tarea. Atención, puede ser diferente para cada grupo de prácticas.
- No se admitirán entregas fuera de plazo.
- La entrega debe realizarla solo uno de los integrantes del grupo, en caso de realizar la práctica por parejas.
- La entrega se compone de dos archivos:
  - Primer archivo: documentación del trabajo desarrollado en formato .pdf. El .pdf de la práctica se entrega sin comprimir, y el nombre debe coincidir con NOMBRE_APELLIDO1_P2.pdf o NOMBRE_APELLIDO1_NOMBRE_APELLIDO1_P2.pdf de ambos integrantes si la práctica se realiza en pareja.
  - Segundo archivo: códigos fuente en un archivo comprimido **sin crear subdirectorios**. El comprimido deberá nombrarse como `code.zip`.
- Los ejercicios deben realizarse sobre las plantillas indicadas en cada ejercicio. Éstas pueden encontrarse en el Campus Virtual en la carpeta *"Templates"*.
- Solo es necesario entregar los siguientes códigos fuentes:
  - ex1_client.py
  - ex1_server.py
  - ex2_client.py
  - ex2_server.py
  - ex3_server.py
- En la primera página de la documentación se debe indicar el nombre o los nombres de los integrantes del grupo. La documentación debe recoger las respuestas adecuadamente argumentadas y justificadas, siguiendo las indicaciones de cada ejercicio, así como la bibliografía consultada.

2. Ejercicios

2.1. El método Monte Carlo. Estimación del valor de pi

Mediante el método probabilístico de Monte Carlo se pueden aproximar los valores o soluciones de ecuaciones complejas. Este método, puede utilizarse para calcular el valor aproximado de π (Figura 1).

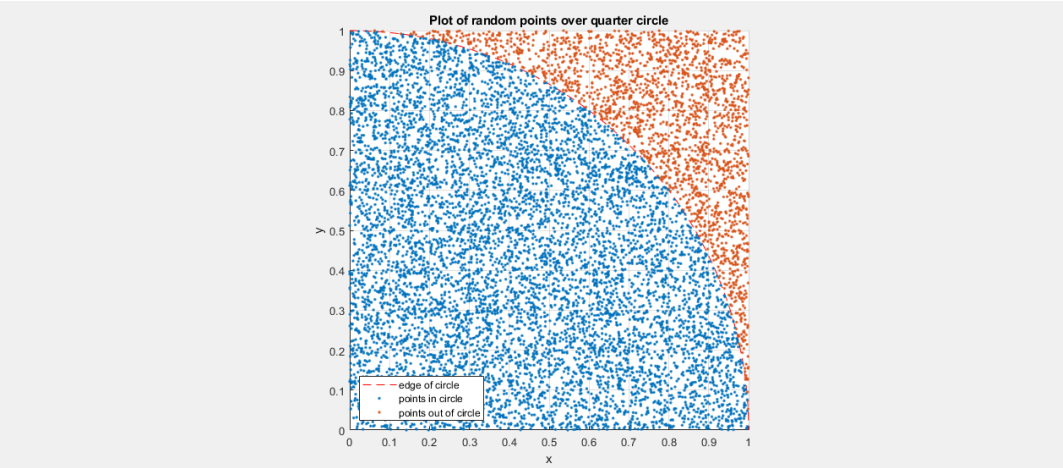


Figura 1: Representación del método de Monte Carlo para el cálculo de π.

Para ello, se genera un número (N) de puntos aleatorios (x, y) en el rango $[0, 1]$; se contabilizan aquellos que son menores estrictos (están por debajo) de la función $f(x) = \sqrt{1 - x^2}$; y, finalmente, el valor de π se aproxima como:

$$\pi \approx 4 \cdot \frac{points\_below}{N}$$

En el siguiente código (`monte_carlo.py`) puede ver una implementación de dicho método en Python:

```python
# Implementation of Monte Carlo method to approximate pi value
import argparse
import math
import random


def f(x):
    return math.sqrt(1 - math.pow(x, 2))


def main(n):
    below_counter = 0
    for i in range(0, n):
        x = random.uniform(0, 1)
        y = random.uniform(0, 1)

        if y < f(x):
            below_counter = below_counter + 1

    pi = 4.0 * float(below_counter) / float(n)
    print("The approximate value of pi with " + str(n) + " random points is " + str(pi))


if __name__ == '__main__':
    parser = argparse.ArgumentParser()
    parser.add_argument('--number', default=100000, help="number of random points to be generated")
    args = parser.parse_args()
```

```
28
29      main(args.number)
```

- Estudie la siguiente sección del libro **Python® Notes for Professionals**:
  - Section 83.1: Hello world in `argparse`.
- Estudie y comprenda el funcionamiento de los ejemplos que puede encontrar en el Campus Virtual en la carpeta *"Exercise 1 examples"*.
- A partir de las plantillas `ex1_client.py` y `ex1_server.py`, implemente un cliente y un servidor en Python que lleven a cabo la siguiente funcionalidad:
  - El cliente recibirá por línea de comandos el puerto remoto del servidor (`--port`, por defecto `1024`), la dirección IP del servidor (`--host`, por defecto `'localhost'`) y el número de puntos aleatorios a generar (`--number`, por defecto `100000`).
  - El servidor recibirá por línea de comandos el puerto local (`--port`, por defecto `1024`) y la dirección IP local (`--host`, por defecto `'localhost'`).
  - Cliente y servidor se comunicarán mediante el protocolo `UDP`.
  - El cliente generará puntos aleatorios en formato `(x, y)`, donde `x` e `y` serán números en coma flotante dentro del rango cerrado $[0, 1]$. Por ejemplo, `''(0.43,0.12)''`.
  - El servidor, recibido un punto en el formato `(x, y)`, extraerá los valores de `x` e `y` y calculará si el punto está por debajo de la función $f(x) = \sqrt{1 - x^2}$.
    - Si el punto está por debajo ($<$) de la función, devolverá al cliente la cadena `''below''`.
    - Si el punto está por encima ($\geq$) de la función, devolverá al cliente la cadena `''above''`.
    - Si el punto está fuera del rango cerrado $[0, 1]$, devolverá al cliente la cadena `''error''`.
  - El cliente, una vez enviados todos los puntos y a partir de las respuestas recibidas por parte del servidor, calculará el valor de π mediante el método de Monte Carlo anteriormente expuesto y lo imprimirá por pantalla.
  - Finalmente, el cliente deberá enviar al servidor la cadena `''exit''`. El servidor, recibida dicha cadena, deberá finalizar su ejecución.
- Discuta y razone en la memoria de la práctica si el flujo de comunicación utilizado es adecuado para `UDP`. Si se utilizará en su lugar `TCP`, ¿cómo modificaría el flujo de comunicación entre el cliente y el servidor?
- **Importante:**
  - Los puntos se enviarán en el formato indicado, por ejemplo, si se generan los números aleatorios `x=0.43` e `y=0.12`, deberá enviarse al servidor la cadena `''(0.43,0.12)''`.
  - Solo se debe enviar un punto por mensaje.
  - Se deben respetar fielmente las palabras clave utilizadas en la comunicación (`''below''`, `''above''`, `''error''` y `''exit''`), incluyendo el uso de minúsculas.

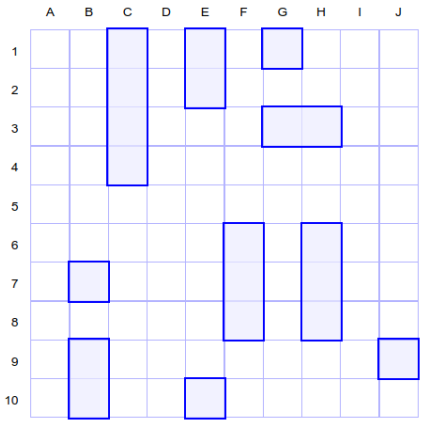
2.2. Transmisión de ficheros con TCP

- Estudie y comprenda el funcionamiento de los ejemplos que puede encontrar en el Campus Virtual en la carpeta *"Exercise 2 examples"*.
- A partir de las plantillas `ex2_client.py` y `ex2_server.py`, implemente un cliente y un servidor en Python que lleven a cabo la siguiente funcionalidad:
  - El cliente recibirá por línea de comandos el puerto remoto del servidor (`--port`, por defecto `1024`), la dirección IP del servidor (`--host`, por defecto `'localhost'`), ruta del fichero a enviar (`--filein`, por defecto `'filein.txt'`) y la ruta del fichero de salida (`--fileout`, por defecto `'fileout.txt'`).
  - El servidor recibirá por línea de comandos el puerto local (`--port`, por defecto `1024`) y la dirección IP local (`--host`, por defecto `'localhost'`).
  - Cliente y servidor se comunicarán mediante el protocolo `TCP`.
  - El cliente leerá el contenido del fichero de texto indicado por línea de comandos y lo transmitirá al servidor. Si el fichero no existe deberá elevar la excepción `OSError`.
  - El servidor, recibido el fichero completo, deberá devolver al cliente:
    - El número de palabras en el fichero que contienen la letra 'a', o 0, si no existe ninguna que cumpla esta condición.
    - La lista de dichas palabras, si existiera alguna.
  - El cliente, recibida esta información, escribirá en el fichero de salida indicado por línea de comandos la lista de palabras, una por línea.
  - Realizada su función, ambos procesos deben finalizar.
- Discuta y razone en la memoria de la práctica el método que ha utilizado para indicar la finalización de la transmisión del fichero en la comunicación. ¿Qué ventajas y desventajas presenta?

2.3. Hundir la flota

En este ejercicio se implementará una simplificación del clásico juego de mesa de *Hundir la flota*. Para ello, a partir de la plantilla `ex3_server.py` implemente un servidor en Python que lleve acabo la siguiente funcionalidad:

- El servidor se encargará de gestionar una partida de hundir la flota para dos jugadores (clientes).
- El servidor recibirá por línea de comandos el puerto local (`--port`, por defecto `1024`) y la dirección IP local (`--host`, por defecto `'localhost'`).
- La comunicación se llevará a cabo mediante `UDP`.
- Al inicio, el servidor permanecerá a la espera de recibir los siguientes mensajes, en el siguiente orden:
  1. El nombre del jugador uno.
  2. El tablero de juego del jugador uno.



(a) Tablero de juego.

```
0 0 1 0 1 0 1 0 0 0
0 0 1 0 1 0 0 0 0 0
0 0 1 0 0 0 1 1 0 0
0 0 1 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 1 0 1 0 0
0 1 0 0 0 1 0 1 0 0
0 0 0 0 0 1 0 1 0 0
0 1 0 0 0 0 0 0 0 1
0 1 0 0 1 0 0 0 0 0
```

(b) Representación del tablero.

Figura 2: Tablero de juego y la representación con la que trabajarán clientes y servidor.

3. El nombre del jugador dos.
4. El tablero de juego del jugador dos.

- Los nombres de los jugadores serán una cadena de texto.
- El tablero de juego consistirá en una matriz de 10x10, donde las celdas que contengan un `0` contendrán agua y las que contengan un `1` contendrán la sección de un navío. En la Figura 2 se muestra un ejemplo tablero y su representación en el formato anteriormente indicado.
- Los jugadores (clientes) enviarán el tablero de juego como una cadena de texto, donde cada fila se separará con un punto y coma (`;`). Por ejemplo, el tablero de la Figura 2b, se transmitirá como la cadena: "0 0 1 0 1 0 1 0 0 0; 0 0 1 0 1 0 0 0 0 0; 0 0 1 0 0 0 1 1 0 0; 0 0 1 0 0 0 0 0 0 0; 0 0 0 0 0 0 0 0 0 0; 0 0 0 0 0 1 0 1 0 0; 0 1 0 0 0 1 0 1 0 0; 0 0 0 0 0 1 0 1 0 0; 0 1 0 0 0 0 0 0 0 1; 0 1 0 0 1 0 0 0 0 0;".
- El primer turno siempre corresponderá al jugador uno.
- Un turno del juego se desarrollará de la siguiente forma:
  1. El servidor enviará el mensaje **"`Turn <id>`"** al jugador al que le corresponda jugar, donde `<id>` será un número entero que se incrementará en el servidor cada vez que pase un turno.
  2. El servidor quedará a la espera del movimiento (coordenadas) del jugador.
  3. Las coordenadas se ajustarán a la representación del tablero de la Figura 2a, donde primero se indicará la columna, mediante una letra mayúscula de la A a la J; seguida de la fila, mediante un número entero del 1 al 10. Por ejemplo, la coordenada "D6" representará la columna de índice 3 y la fila de índice 5 de la matriz.
  4. El movimiento (coordenadas) se enviara en un único mensaje.
  5. Recibido el movimiento del jugador, el servidor comprobará en el tablero del contrincante:
     - Si el jugador ha fallado, es decir, las coordenadas corresponden a una celda que contiene un `0` (agua). En cuyo caso responderá al jugador con la cadena `"Fail"` y pasará a ser el turno del siguiente jugador, comenzando de nuevo el turno de juego (Paso 1).

    - Si el jugador ha acertado, es decir, las coordenadas corresponden a una celda que contiene un `1` (sección de navío). En este caso se deberá comprobar si al contrincante le quedan más navíos a flote:
        - En caso afirmativo enviará la cadena `"Hit"` al jugador y volverá a ser el turno de este mismo jugador, comenzando de nuevo el turno de juego (Paso 1).
        - En caso negativo, es decir, no quedan más navíos a flote, el jugador ha ganado. En este caso, el servidor enviará el mensaje `"You win"` al ganador y el mensaje `"You lost"` al perdedor. Posteriormente finalizará su ejecución, ya que el juego ha finalizado.

- En `ex3_player1.py` y `ex3_player2.py` podrá encontrar la implementación completa de dos jugadores que juegan de forma aleatoria al juego siguiendo el flujo anteriormente descrito. Estos pueden utilizarse con fines de testeo del servidor implementado.

- **Importante:**

    - Las coordenadas se enviarán en el formato indicado en un único mensaje.
    - No es necesario comprobar en el servidor que el tablero o las coordenadas son correctas. Se supondrá que los jugadores son fiables y siempre envian tableros y movimientos válidos, cumpliendo con las reglas de formato y forma.
    - Los scritps de los clientes (`ex3_player1.py` y `ex3_player2.py`) deben ser compatibles con el servidor implementado, sin ser necesaria ninguna modificación en su comportamiento.
    - Se deben respetar fielmente las palabras clave utilizadas en la comunicación (`‘‘Turn <id>’’`, `‘‘Fail’’`, `‘‘Hit’’`, `‘‘You win’’` y `‘‘You lost’’`), incluyendo el uso de mayúsculas y minúsculas.
    - Las funciones `ord()`[1] y `matrix()`[2] podrían resultar de utilidad.

3. Evaluación

La nota de la práctica se divide de la siguiente forma:

- El Método Monte Carlo. Estimación del valor de Pi: 15 %
- Transmisión de ficheros con TCP: 35 %
- Hundir la flota: 50 %

Bibliografía

- GoalKicker (2018). **Python® Notes for Professionals**. Disponible online: https://goalkicker.com/PythonBook/. *Released under Creative Commons BY-SA*.
- NumPy Reference (2022). Disponible online: https://numpy.org/doc/stable/reference/index.html.
- Python 3 Documentation (2022). Disponible online: https://docs.python.org/3/.

[1] https://docs.python.org/3/library/functions.html#ord
[2] https://numpy.org/doc/stable/reference/generated/numpy.matrix.html