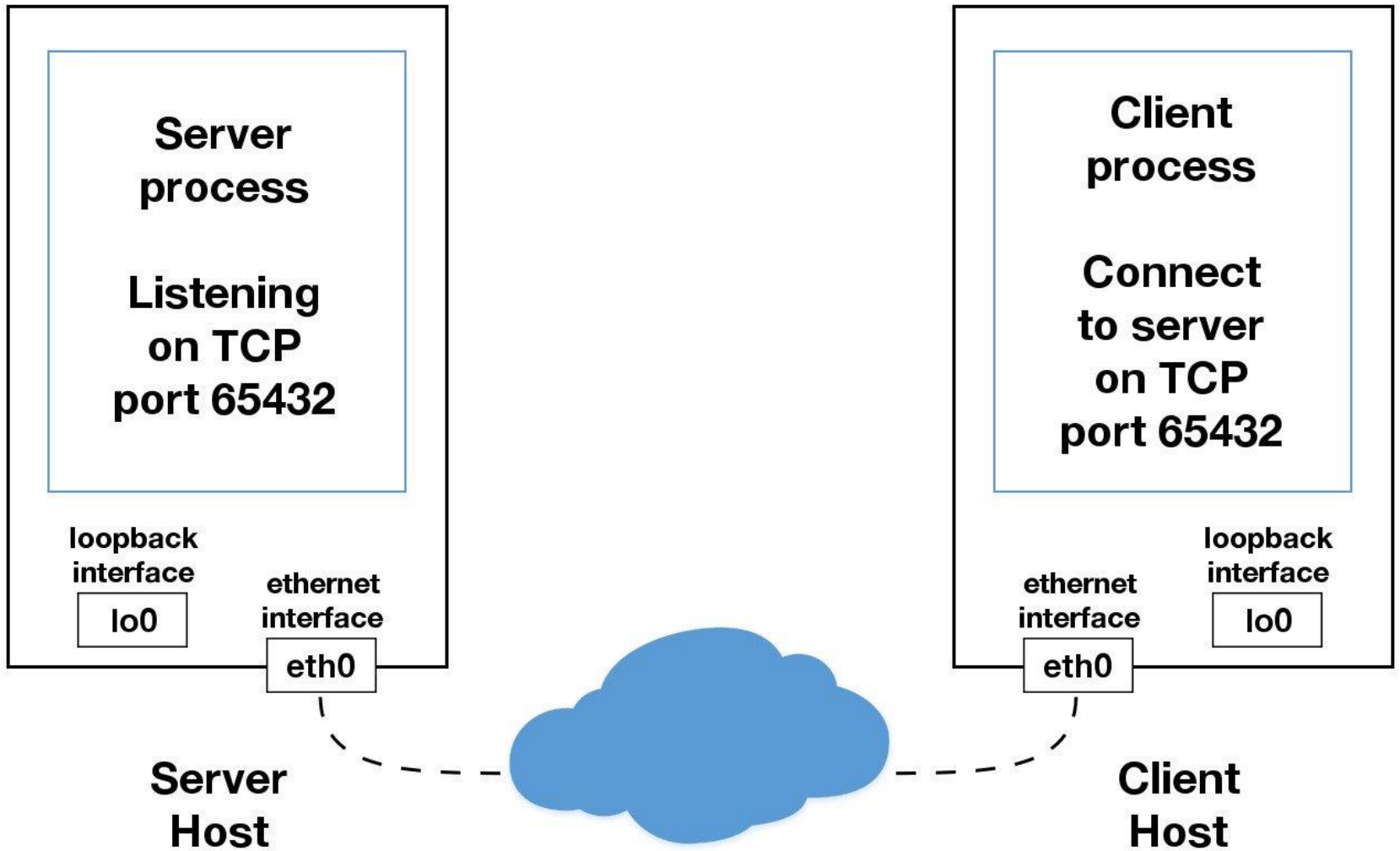
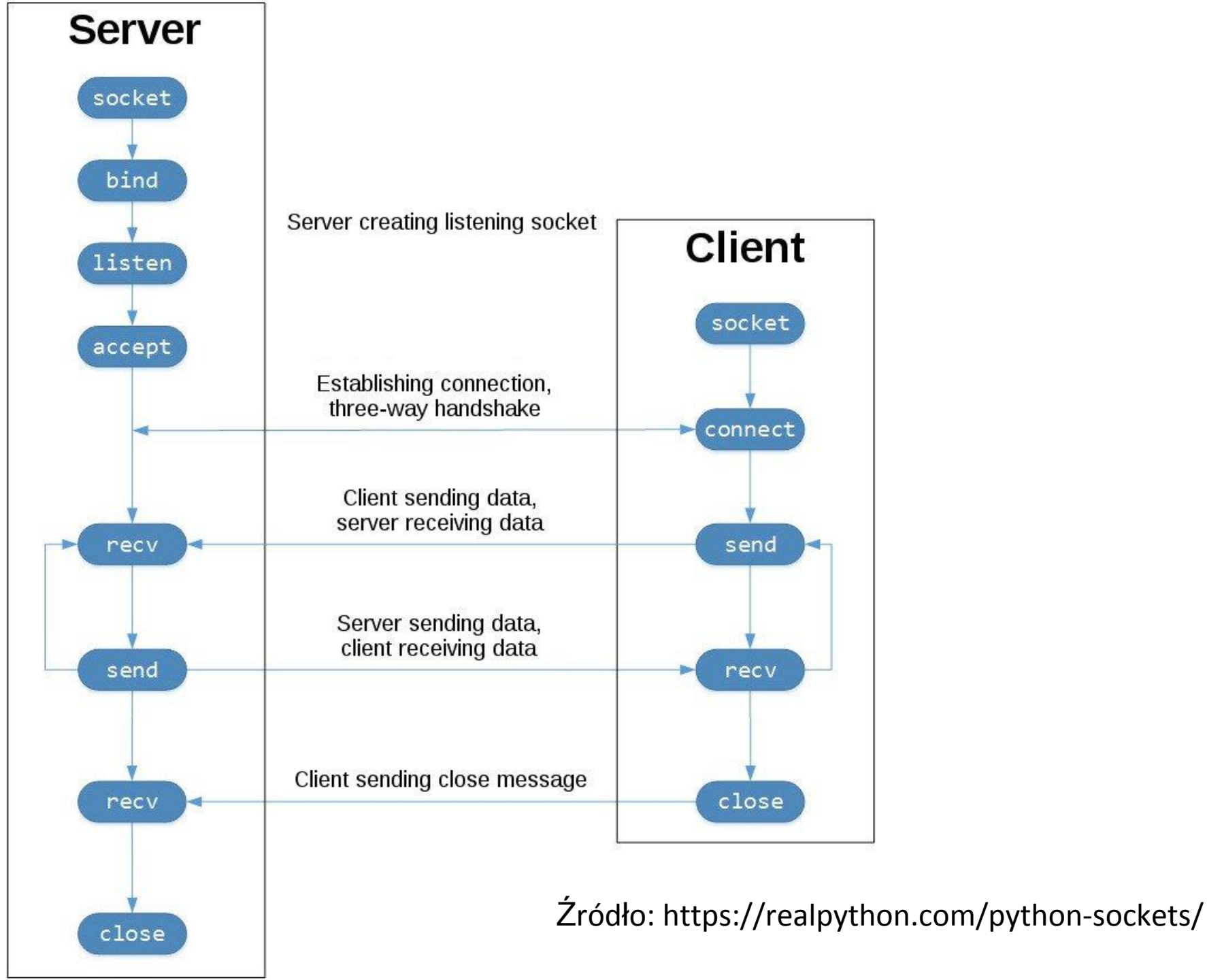
Szachy dla dwóch osób

Koncepcja

Sockety


Server
process
Listening
on TCP
port 65432
loopback
interface
lo0
ethernet
interface
eth0
Server
Host
Client
process
Connect
to server
on TCP
port 65432
loopback
interface
lo0
ethernet
interface
eth0
Client
Host



Źródło: https://realpython.com/python-sockets/

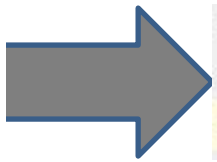
```python
import socket

HOST = '127.0.0.1'  # The server's hostname or IP address
PORT = 65432        # The port used by the server

s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.connect((HOST, PORT))
data = s.recv(1024)
data = data.decode('utf-8')
print(data)
print('teraz wysyłamy!')
s.sendall(bytes('wysyła klient', 'utf-8'))
s.close()
```

```python
import socket

HOST = '127.0.0.1'  # Standard loopback interface address (localhost)
PORT = 65432        # Port to listen on (non-privileged ports are > 1023)

s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
try:
    s.bind((HOST, PORT))
    s.listen()
    conn, addr = s.accept()
    print('Connected by', addr)
    try:
        conn.sendall(bytes('to wysyła serwer', 'utf-8'))
        data = conn.recv(1024)
        data = data.decode('utf-8')
        print(data)
    finally:
        conn.close()
finally:
    s.close()
```

Wyrażenie with

- Zawiera już w sobie obsługę wyjątków
- Samo zamyka zasób po opuszczeniu klauzuli

```
1  f = open('xd.txt', 'w')
2  try:
3      f.write('lubie placki')
4  finally:
5      f.close()
```



```
1  with open('xd.txt', 'w') as file:
2      file.write('lubie placki')
3
```

```python
import socket

HOST = '127.0.0.1'  # Standard loopback interface address (localhost)
PORT = 65432        # Port to listen on (non-privileged ports are > 1023)

s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
try:
    s.bind((HOST, PORT))
    s.listen()
    conn, addr = s.accept()
    print('Connected by', addr)
    try:
        conn.sendall(bytes('to wysyła serwer', 'utf-8'))
        data = conn.recv(1024)
        data = data.decode('utf-8')
        print(data)
    finally:
        conn.close()
finally:
    s.close()
```

Zadanie 1. Przepisać ten kod z użyciem wyrażenia with

```python
import socket

HOST = '127.0.0.1'  # Standard loopback interface address
(localhost)
PORT = 65432        # Port to listen on (non-privileged ports are
> 1023)

with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as s:
    s.bind((HOST, PORT))
    s.listen()
    conn, addr = s.accept()
    print('Connected by', addr)
    with conn:
        conn.sendall(bytes('to wysyła serwer', 'utf-8'))
        data = conn.recv(1024)
        data = data.decode('utf-8')
        print(data)
```

„Specjalne metody”

- Python posiada zestaw wyróżnionych metod, które pełnią szczególne zadania w naszych klasach
- Jedną z takich metod jest konstruktor __init__(self, …). Słowo kluczowe self to po prostu przekazanie instancji klasy do konstruktora. Jest to parametr ukryty, przy tworzeniu obiektu zachowujemy się tak, jakby go nie było.
- Kolejną taką metodą, dość istotną przy debugowaniu jest __repr__(self). Ma ona wygenerować użyteczny opis obiektu.

```
class Figure(ABC):
    def __init__(self, id):
        self.type = ''
        if id < 16:
            self.colour = "white"
        else:
            self.colour = "black"
        self.x_pos, self.y_pos =
moves.number_to_position(id)

    def change_position(self, pos):
        self.x_pos, self.y_pos = pos

    def __repr__(self):
        return self.colour + self.type
```

```
if chessboard[x2] == '':
    chessboard[x1], chessboard[x2] = chessboard[x2],
chessboard[x1]
elif chessboard[x1].colour != chessboard[x2].colour:
    chessboard[x2], chessboard[x1] = chessboard[x1], ''
```

```
class Pawn(Figure):
    def __init__(self, id):
        Figure.__init__(self, id)
        self.type = 'pawn'
        self.first_movement_done = False
```

Wykonanie odpowiedniego ruchu

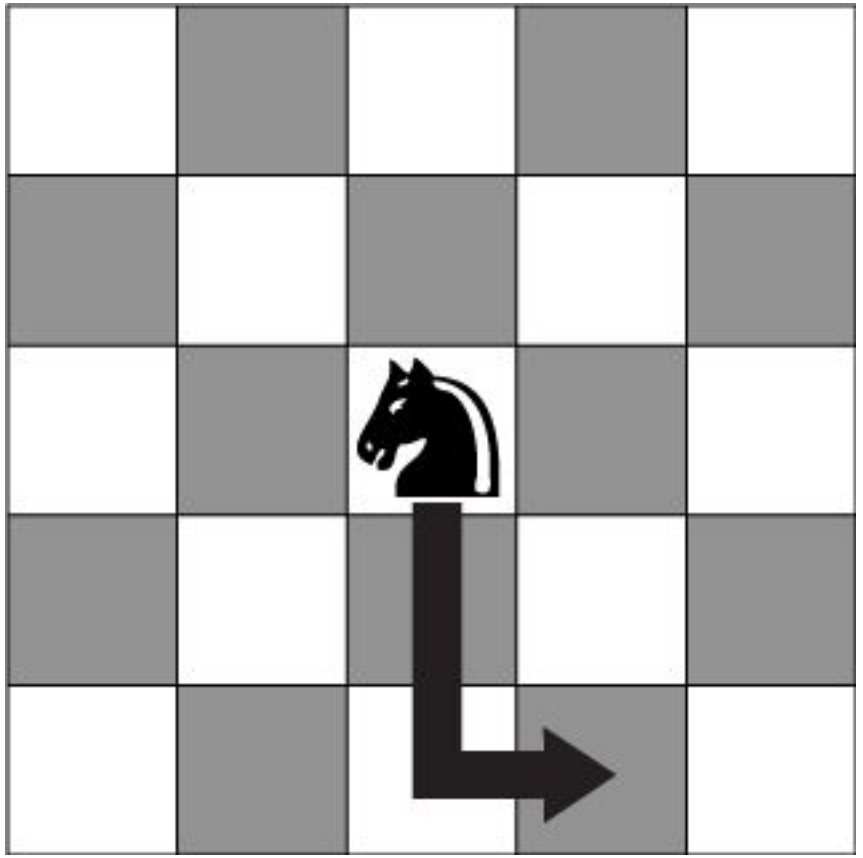
Rozwiązanie problemu

- Napisaliśmy metody dla każdej z klas figur, które zwracają listę pozycji (listę krotek) na które ruch jest możliwy.

- Dla uproszczenia i lepszej czytelności kodu każda figura posiada dwa atrybuty określające jej położenie za szachownicy w postaci numeru wiersza i kolumny na jakiej się znajduje.

Zadanie 2

Skoczek porusza się w kształcie litery L, tak jak na obrazku. Napisz metodę która zwróci listę pozycji na jakie może ruszyć się skoczek. Klasa skoczka posiada atrybuty x i y określające jego położenie na szachownicy.



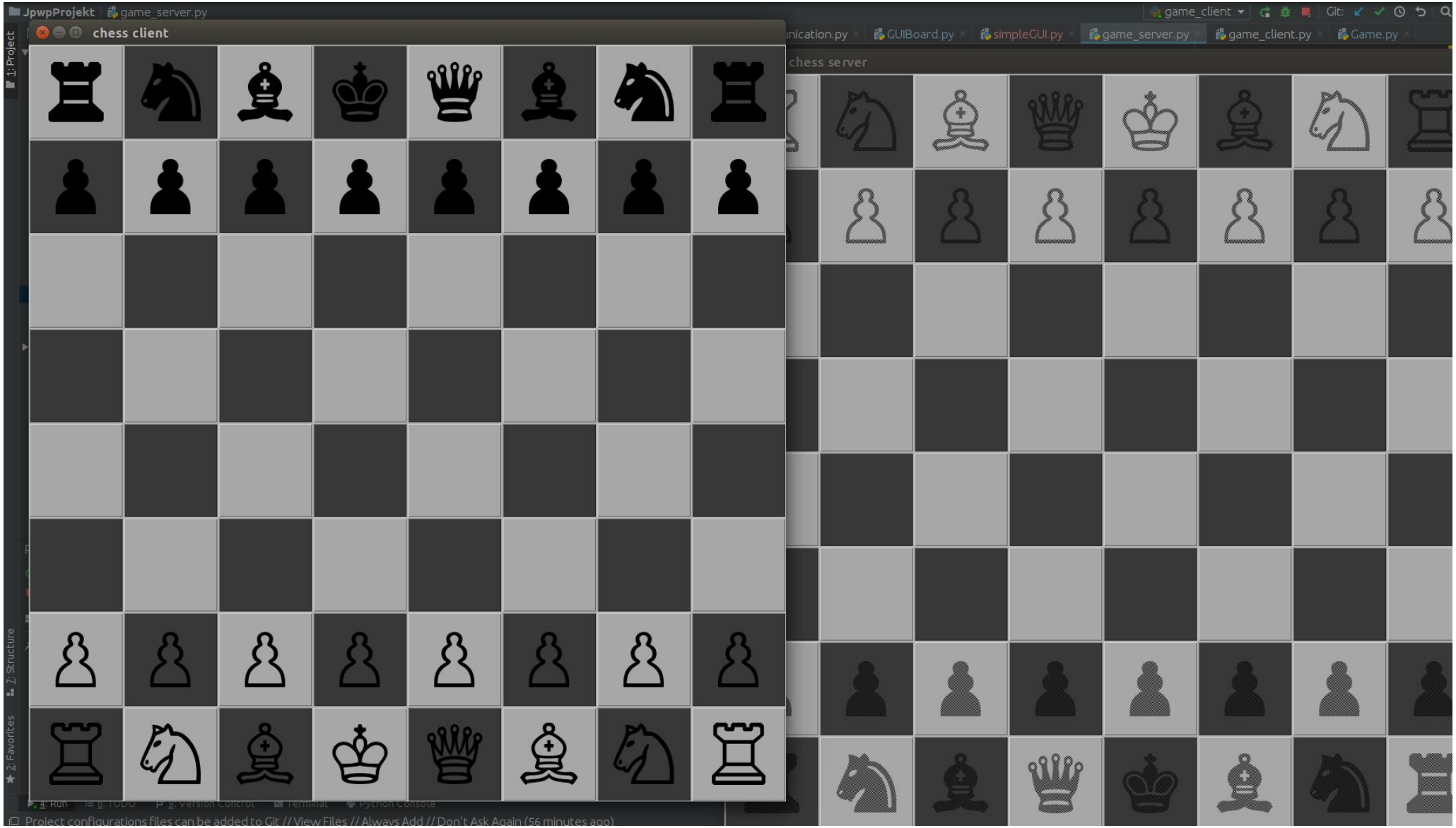
```
def possible_movements(self, chessboard):
    movements_list = []
    for i in [-2, -1, 1, 2]:
        for j in [-2, -1, 1, 2]:
            x = self.x_pos + i
            y = self.y_pos + j
            if abs(i) is not abs(j) and x in range(1, 9) and y in range(1, 9):
                if chessboard[moves.position_to_number((x, y))] == '':
                    # pole jest puste
                    movements_list.append((x, y))
                elif chessboard[moves.position_to_number((x, y))].colour != self.colour:
                    # na polu stoi figura przeciwnika
                    movements_list.append((x, y))
    return movements_list
```

Kod wymiany danych

```
move = str(move_from * 64 + move_to)
```

```
x1 = int(move) // 64
x2 = int(move) % 64
```

GUI


JpwpProjekt
game_server.py
chess client
game_client
Git:
unication.py
GUIBoard.py
simpleGUI.py
game_server.py
game_client.py
Game.py
chess server
1: Project
7: Structure
2: Favorites
Run
TODO
9: Version Control
Terminal
Python Console
Project configurations files can be added to Git // View Files // Always Add // Don't Ask Again (56 minutes ago)

GUI

- Tkinter – moduł podstawowej biblioteki pythona oparty na Tk

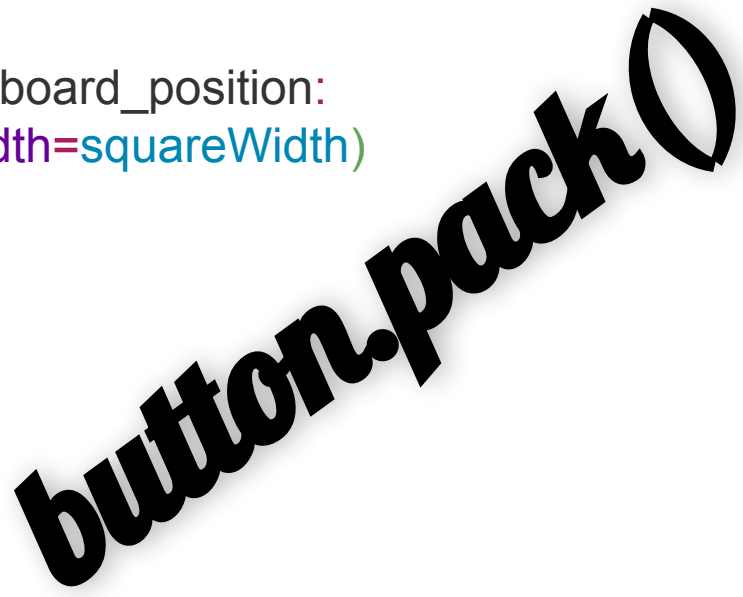
```
from tkinter import *

root = Tk()
root.title('chess')

board = ChessBoard(root)
board.pack()

root.mainloop()
```

```
class ChessBoard:
    def __init__(self, master):
        self.master = master
```

Button

```
button = tkinter.Button(
   self, bg=color, command=lambda position=chessboard_position:
self.button_click(position),height=squareHeight, width=squareWidth)
self.gameButtons.append(button)
```

```
button.rowconfigure(0, weight=1)
button.columnconfigure(0, weight=1)
button.grid_propagate(0)
button.grid(row=r, column=c)
button.grid(sticky="NWSE")
```



```
def refresh_board(self, chessboard):
    if self.SIDE == 'white':
        it = 63
    else:
        it = 0
    for i in chessboard:
        if isinstance(i, Chessboard.King):
            if i.colour == "white":
                self.gameButtons[it].config(image=self.whiteKingImage)
                self.gameButtons[it].image = self.whiteKingImage
            else:
                self.gameButtons[it].config(image=self.blackKingImage)
                self.gameButtons[it].image = self.blackKingImage
```

Zadanie 3

Napisz prosty hello world w GUI (po naciśnięciu przycisku wyświetla się tekst).

https://github.com/kamillo2012/jpwpPrezentacja/

Wątki

Wątki

- Chcemy aby nasz program cały czas sprawdzał, czy przeciwnik wykonał ruch, a jeśli tak, żeby natychmiast zaktualizował planszę.

```
self.receive_thread = Thread(None, self.receive, None, (), {})
self.receive_thread.start()
```

```
def receive(self):
   while True:
      self.chessboard = moves.opponent_movement(self.chessboard,
communication.receive_data(self.conn))
      self.refresh_board(self.chessboard)
      self.change_current_player()
      self.enable_buttons()
```

Zadanie 4

Napisz prosty program, który utworzy kilka wątków. Każdy wątek wywołuje funkcję, która wypisuje tekst, następnie zawiesza działanie na 5s, potem ponownie wypisuje informacje dla użytkownika.

https://github.com/kamillo2012/jpwpPrezentacja/

DZIĘKUJEMY ZA UWAGĘ

Cecylia Borek

Kamil Szczeszek