Zadania

Zadanie 1.

Pierwsze zadanie będzie polegało na dokończeniu bardzo podstawowej architektury komunikacji klient – serwer.
Gdy jeden z klientów ruszy prostokątem, jego przemieszczenie będzie widoczne również w oknie drugiego klienta.
Należy uzupełnić kilka fragmentów kodu, tak aby program zaczął działać poprawnie

1.1 W pliku server.py brakuje gniazda . Utwórz gniazdo s=…. i zbinduj je z adrsem serwera i portu

Podpowiedź: Popatrz na slajd nr.17 z prezentacji.

1.2 Korzystając z biblioteki pygame uzupełnij funkcję move (player.py) tak, aby możliwe było poruszanie się prostokątem.

Podpowiedź 1:

Skorzystaj z przykładu znajdującego się pod tym linkiem (strona 133).
https://books.google.pl/books?id=Tw5pDwAAQBAJ&pg=PA133&lpg=PA133&dq=keys%5Bpygame.K_LET&source=bl&ots=ds9wuNAm-M&sig=ACfU3U3p9fIwejrpeXh8boco9SRX1VPddA&hl=pl&sa=X&ved=2ahUKEwiX0Kuji47pAhXusosKHYNLCXoQ6AEwCnoECAgQAQ#v=onepage&q=keys%5Bpygame.K_LET&f=false

Podpowiedź 2:
Zmienna vel określa o ile pikseli ma przesunąć się kwadrat (mający współrzędne x i y)

Uwaga:
Aby uruchomić program należy po pierwsze uruchomić serwer (ma on działać cały czas) i następnie uruchomić dwa razy client.py. Ważne jest pozwolenie Pythonowi na równoległe uruchomienie kilku programów.
Aby to zrobić należy (PyCharm):

1. Kliknąć na przycisk z nazwą programu znajdujący się prawym górnym rogu ekranu (ten z logiem Pythona, obok zielonego trójkąta)

2. Edit Configurations…

3. Zaznaczyć „Allow paralel run” (prawy górny róg)

Zadanie 2

2.1 W pliku obsługa_serwera.py brakuje instrukcji obsługującej klucz z wartością 5 odpowiedzialny za prawidłowe przesyłanie informacji o numerze aktualnej rundy (po uruchomieniu zamiast licznika rund pojawia się od razu napis „Koniec gry").

W celu naprawienia tego błędu, napisz fragment kodu obsługujący key == 5 (linijka 57, obsługa_serwera.py), tak aby po uruchomieniu programu rundy zostały zliczane w prawidłowy sposób.

Podpowiedź:

- Zadanie zrób analogicznie do tego jak obsługiwane są pozostałe klucze
- Jedynym kodem który trzeba napisać jest instrukcja obsługująca klucz 5
- Należy skorzystać ze zmiennej licznik_rund znajdującej się w pliku server/gra.py

2.2 Kolejnym problem, który trzeba obsłużyć jest działanie przycisku odpowiedzialnego za działanie gumki. Przycisk ma działać w ten sposób, że po jego kliknięciu „pędzel" zamieni się w gumkę, a po ponownym kliknięciu w przycisk powróci z powrotem do „trybu pędzla"

Podpowiedź:

- kod należy zapisać w funkcji obsługa_przycisków znajdującej się w pliku client/pasek_dolny (pod za komentowaną instrukcją „if self.gumka_button.klikniecie(*mouse)", która wykorzystując bibliotekę pygame reaguje na kliknięcie w przycisk)
- kolor czarny to (255,255,255), a biały to (0,0,0)
- ponieważ biblioteka pygame po jednym kliknięciu wykrywa zazwyczaj kilka reakcji, polecam dodać funkcje t.sleep(0.3), która zawiesi program po pierwszej reakcji na 0.3 sekundy

2.3 Rozwiąż problem nie działającego czatu (wiadomość po napisaniu nie wyświetla się w oknie czatu). W celu rozwiązania problemu należy dopisać brakujący kod w pliku client/gry.py (zaznaczyliśmy tam przykładowe miejsce, w którym można napisać brakujący kod)

Podpowiedź:

- wzoruj się na kodzie napisanym w linijkach 75-76 odpowiedzialnym za czas
- pamiętaj, że czat przesyłany jest jako key = 2
- wykorzystaj funkcję update_chat z pliku client/chat.py

Zadanie 3

To zadanie polegać będzie na napisaniu w Pythonie kodu tworzącego prostą bazę danych oraz wypełniającego ją danymi. W programie wykorzystaj bibliotekę "**mysql connector**". Załóż że na twoim komputerze znajduje się już serwer przygotowany pod bazę danych (adres: **localhost**, użytkownik: **admin**, hasło: **admin**). Dla ułatwienia zadanie podzielone jest na części.

3.1 Napisz kod tworzący bazę danych o nazwie "**kalamburyzawody**" oraz tworzący w niej dwie tabele: jedną przechowującą hasła (nazwa: **hasla**, pola: **id**, **tresc**, **kategoria**) oraz drugą przechowującą dane graczy biorących udział w zawodach (nazwa: **zawodnicy**, pola: **id**, **imie**, **nazwisko**, **numer**).
Zadbaj o:

- automatyczną inkrementację pól **id** w obydwu tabelach oraz ustawienie tych pól jako klucza głównego
- unikatowość pola **treść** w tabeli **hasla** oraz **nume**r w tabeli **zawodnicy**
- zabezpieczenie odpowiednich pól (**tresc**, **kategoria**, **nazwisko**, **imie**, **numer**) tak aby niemożliwe było pozostawienie ich pustych
- odpowiedni format danych w polach (liczby całkowite dla pól **id** oraz **numer**, tekst dla **pozostałych**)

Podpowiedź:

- pamiętaj aby przed tworzeniem tabel, dołączyć nazwę bazy danych jako argument do wyrażenia **mydb = mysql.connector.connect**

3.2 Napisz kod dodający do bazy danych kilka przykładowych haseł (z co najmniej dwiema różnymi kategoriami) oraz kilku zawodników. W kodzie wykorzystaj dodawanie kilku zestawów danych za jednym razem (osobno dla każdej z tabel).

3.3 Napisz kod który:

- pobierze numery wszystkich zawodników z bazy danych oraz umieści je na lokalnie stworzonej liście (nazwa listy: **numery_zawodnikow**)
- pobierze z bazy danych wszystkie treści haseł o wybranej kategorii (wybierz jedną ze wymyślonych w podpunkcie 3.2) i zapisze je na lokalnie stworzonej liście (nazwa listy: **hasla_z_kategorii**)

Podpowiedź:

- pamiętaj że funkcja **myresult = mycursor.fetchall()** wykorzystana w prezentacji zawsze zwraca listę obiektów tuple nawet jeśli są one jednoelementowe więc konieczne jest odwołanie się do konkretnego elementu