Il file contiene l'implementazione di una versione testuale (console) del gioco del Blackjack, con alcune funzionalità extra (come il supporto al debug, split e raddoppio). Di seguito una descrizione dettagliata delle parti principali del codice e di ciascuna funzione:

---

1. Impostazioni Iniziali e Modalità Debug

- **Scelta della modalità debug**:
  All'avvio, il programma chiede all'utente se attivare le funzioni di debug. Se l'utente inserisce la stringa `"dev"`, viene impostata la variabile globale `dev = True` (modalità developer) che, tra l'altro, abilita stampe e comportamenti specifici per il testing. In caso contrario, il gioco procede normalmente con `dev = False`.
- **Variabili globali iniziali**:
  Viene definita la lista vuota `mazzo`, che conterrà le carte, e successivamente altre variabili globali (ad es. `fiches` per le fiches del giocatore, `carte_giocatore`, `carte_dealer`, ecc.) che vengono inizializzate e gestite nel ciclo di gioco.

---

2. Composizione del Mazzo

- **`componi_mazzo()`**
  Questa funzione è responsabile della creazione del mazzo (o dei mazzi) da cui verranno pescate le carte. All'interno di essa viene definita la funzione interna `crea_mazzo()`:
  - **Funzione interna `crea_mazzo()`**:
    - Chiede all'utente quanti mazzi utilizzare. Se l'input è vuoto o minore o uguale a 0, imposta il numero di mazzi di default a 2.
    - Per ciascun mazzo, scorre i valori da 1 a 13 (che rappresentano i possibili valori delle carte, dove 1 corrisponde all'Asso e 10–13 agli altri valori) e per ciascun valore aggiunge quattro carte, rappresentate con stringhe formattate come:
      - `"c{x}"`, `"q{x}"`, `"f{x}"`, `"p{x}"`
        Questi prefissi indicano probabilmente i semi (cuori, quadri, fiori, picche).
    - Dopo aver aggiunto le carte per ciascun valore, viene effettuato uno shuffle del mazzo (la chiamata a `random.shuffle(mazzo)` è posta all'interno del ciclo, mescolando progressivamente il mazzo).
    - Se la modalità debug è attiva, il mazzo viene stampato in uscita.
  - Se in modalità debug l'utente sceglie di "comporre il mazzo manualmente" (inserendo nuovamente `"dev"` ad una specifica richiesta), il programma permette di inserire manualmente il numero di carte e i dettagli di ciascuna carta.

---

3. Funzioni di Gestione del Tempo e della Pesca

- **`devskiptimezawarudo()`**
  Questa funzione introduce una breve pausa (0.2 secondi) usando `time.sleep(0.200)` se non si è in modalità debug. In modalità debug viene saltata la pausa per rendere il testing più rapido.

- **`prendicarta()`**
  Funzione che estrae la prima carta dal mazzo globale. Essa:
  - Preleva la carta in posizione 0.
  - La rimuove dalla lista (usando `pop(0)`) e la restituisce.

---

4. Gestione delle Carte del Giocatore e del Dealer

- **`pesca_giocatore(carte)`**
  Gestisce la pesca di una carta per il giocatore:
  - Riceve come argomento la lista (mano) del giocatore e le aggiunge la carta pescata.
  - Stampa la mano attuale.
  - Successivamente scorre la mano: se trova una carta il cui valore (estratto con `int(x[1:])`) è 1 (cioè un Asso), la sposta alla fine della lista. Tale operazione probabilmente serve a gestire il calcolo dei punti, trattando gli Assi in un secondo momento.
- **`pesca_dealer()`**
  Gestisce la pesca delle carte del dealer in base a una variabile globale `dealer_time`:
  - Al primo turno (`dealer_time == 0`): pesca una carta e incrementa `dealer_time`.
  - Al secondo turno (`dealer_time == 1`): stampa la carta scoperta insieme a una "carta coperta", poi pesca una seconda carta.
  - Nei turni successivi: pesca una nuova carta, stampa la mano completa e, come per il giocatore, sposta eventuali Assi (valore 1) in fondo alla lista.

---

5. Calcolo dei Punti e Verifica del Blackjack

- **`calcolopunti(carte)`**
  Calcola il punteggio totale di una mano (lista di carte) seguendo queste regole:
  - Se la stringa della carta ha lunghezza 3 (tipicamente per carte con numero 10, che risultano come "c10", "q10", ecc.), viene aggiunto 10 punti.
  - Altrimenti, se il valore (estratto da `carta[1]`) non è 1, si somma il valore numerico della carta.
  - Se il valore è 1 (Asso), il punteggio aggiunto sarà 11 se il totale attuale è 10 o meno, altrimenti 1.
  - La funzione stampa il punteggio e lo restituisce.
- **`calcolo_BJ(*carte)`**
  Verifica se una data mano (o più mani, grazie al parametro variadico) raggiunge esattamente 21 punti:
  - Scorre ogni carta e somma i punti seguendo una logica simile a `calcolopunti`.
  - Se il punteggio totale è 21, restituisce la stringa `"BJ"`, indicante un Blackjack.

---

6. Gestione dello Split

- **`controllo_split(*carte_mano)`**
  Determina se le mani del giocatore sono “splittabili” (cioè se le due carte iniziali hanno lo stesso valore):
  - Definisce una variabile globale `splittable` (lista di booleani) che conterrà, per ogni mano, un valore True se la coppia di carte è identica (in base a un confronto “personalizzato”) oppure False altrimenti.
  - All’interno viene definita la funzione interna `ciclo_interno(*carte)`:
    - Per ciascuna mano, itera sulle carte e per le prime due carte calcola un valore “semivariabile” usando `int(y[1])` sommato a `len(y)*100`. In questo modo si differenziano anche carte a due cifre (come “10”) da quelle a una cifra.
    - Se i valori calcolati per le due carte coincidono, la mano è considerata splittable.
  - La funzione gestisce il caso in cui il giocatore abbia già splittato, identificato dalla presenza del marker `"multideck"` in `carte_giocatore`.
- **`split_giocatore()`**
  Gestisce l’azione di split per il giocatore:
  - Verifica che il giocatore abbia sufficienti fiches per effettuare una puntata aggiuntiva.
  - Per ogni mano (scorrendo la lista `carte_giocatore` e facendo riferimento alla lista `splittable`), se la mano è splittabile chiede all’utente se vuole splittare.
  - Se l’utente accetta, il gioco esegue le seguenti operazioni:
    - Se non si è già in modalità “multideck”, inserisce il marker `"multideck"` per indicare che ora si gestiscono più mani.
    - Divide la mano in due mani separate: ciascuna ottiene una delle carte originarie e una carta aggiuntiva pescata dal mazzo.
    - Aggiorna la lista delle mani e richiama `controllo_split()` per aggiornare lo stato.
- **`puntate_multiple()`**
  Se il giocatore effettua uno split, questa funzione gestisce l’aggiornamento delle puntate:
  - Per ogni mano aggiuntiva (oltre la prima) viene aggiunta una puntata identica a quella iniziale e viene sottratta dal totale delle fiches.

---

7. Gestione del Turno del Giocatore

- **`gioco_giocatore()`**
  Controlla l’andamento del turno del giocatore per ciascuna mano:
  - Se la mano risulta essere un Blackjack (verificato con `calcolo_BJ`), viene registrato il risultato.
  - Altrimenti, per ciascuna mano viene chiesto ripetutamente se il giocatore vuole:
    - **Raddoppiare (double down)**:
      Se il giocatore ha sufficienti fiches (verificato da `controllo_raddoppio`), può raddoppiare la puntata, ricevere una carta aggiuntiva e poi terminare il turno per quella mano.
    - **Chiedere un’altra carta ("hit")**:
      Se l’utente sceglie di continuare, viene pescata una carta e viene

aggiornato il punteggio.
Se il punteggio raggiunge 21 o supera 21 (sballato), il ciclo si interrompe e il punteggio viene registrato.
        - **Stare (stand)**:
          Se il giocatore decide di non richiedere ulteriori carte, il punteggio corrente viene registrato e il turno per quella mano termina.
    - I punteggi ottenuti (o il marker `"BJ"`) vengono memorizzati in una lista globale `punteggio_giocatore`.

---

8. Gestione del Turno del Dealer

- **`gioco_dealer()`**
  Simula il turno del dealer seguendo le regole tipiche del Blackjack:
    - In un ciclo `while`, il dealer continua a pescare carte finché non raggiunge almeno 17 punti.
    - Durante il ciclo, il punteggio viene calcolato in modo simile a quello del giocatore, con una gestione degli Assi che può valere 1 o 11.
    - Sono previsti casi speciali: ad esempio, se il dealer sballa (punteggio > 21) il punteggio viene settato a 0 oppure se si verifica un Blackjack (punteggio esatto di 21, ma non ottenuto con le sole due carte iniziali).
    - Al termine del turno del dealer, viene chiamata la funzione `confronta_punteggio()` per determinare il vincitore.
- **`confronta_punteggio(giocatore, dealer)`**
  Confronta il punteggio delle mani del giocatore (eventualmente multiple, in caso di split) con quello del dealer:
    - Se il giocatore ha un Blackjack, viene applicato il pagamento tipico (puntata più un bonus pari a 3/2 della puntata).
    - Se il punteggio del giocatore è maggiore di quello del dealer, il giocatore vince e raddoppia la puntata.
    - In caso di pareggio, la puntata viene restituita.
    - Se il dealer vince, la puntata viene persa.
    - Gli esiti vengono stampati e il totale delle fiches aggiornato.

---

9. Funzioni Ausiliarie per le Puntate

- **`getlistpuntata(puntata)`**
  Converte la puntata iniziale (un numero) in una lista, utile per gestire le puntate multiple in caso di split.
  (In modalità debug, la lista viene stampata.)
- **`controllo_raddoppio(puntata)`**
  Controlla se il giocatore ha abbastanza fiches per poter raddoppiare la puntata corrente. Se il totale residuo (fiches – puntata) è negativo, restituisce `False`, altrimenti `True`.

---

10. Ciclo Principale del Gioco

Il corpo principale del programma gestisce il flusso di gioco:

1. **Preparazione**:
   - Viene composto il mazzo tramite `componi_mazzo()`.
   - Il giocatore parte con 1000 fiches e viene spiegato che le puntate devono essere multipli di 10.
2. **Ciclo di Gioco (mainLoop)**:
   - Ad ogni turno viene mostrato il numero di fiches e il numero di carte rimaste nel mazzo.
   - Il giocatore inserisce la puntata (se vuota, viene impostata la puntata minima di 10).
   - Vengono resettate le variabili globali per le mani del giocatore e del dealer.
   - Viene eseguito il "deal iniziale": il giocatore e il dealer ricevono due carte (con la seconda carta del dealer inizialmente nascosta).
   - **Verifica immediata di Blackjack**:
     - Se sia il giocatore che il dealer hanno un Blackjack, si ha un push (pareggio).
     - Se solo il dealer o solo il giocatore ha Blackjack, il risultato viene annunciato e il saldo delle fiches aggiornato.
3. **Turno del Giocatore**:
   - Se non si è verificato un Blackjack immediato, la puntata viene convertita in lista (per gestire eventuali split).
   - Vengono calcolati i punti della mano iniziale e si verifica se è possibile eseguire uno split (chiamata a `controllo_split()`).
   - Se il giocatore decide di splittare, la funzione `split_giocatore()` gestisce la divisione della mano e vengono aggiustate le puntate con `puntate_multiple()`.
   - Il turno del giocatore per ciascuna mano viene gestito da `gioco_giocatore()`, che chiede in loop se il giocatore desidera raddoppiare, prendere un'altra carta o stare.
4. **Turno del Dealer**:
   - Dopo il turno del giocatore, viene eseguito il turno del dealer tramite `gioco_dealer()`, che continua a pescare carte finché non raggiunge almeno 17 (con alcune eccezioni gestite dal codice).
5. **Confronto e Esito**:
   - Viene confrontato il punteggio finale del dealer con quello del giocatore (o dei giocatori, in caso di split) per stabilire l'esito della mano e aggiornare il saldo delle fiches.
6. **Nuova Partita**:
   - Al termine di una mano, viene chiesto all'utente se desidera giocare una nuova partita. Se il mazzo ha meno di 20 carte, viene ricostruito (mescolato nuovamente) chiamando `componi_mazzo()`.

Considerazioni Generali

- **Gestione delle Modalità Speciali**:
  La presenza della modalità debug (`dev`) permette di visualizzare ulteriori informazioni (stati

intermedi, mazzo composto, controllo dei punteggi, ecc.) e di saltare i ritardi introdotti dalla funzione `devskiptimezawarudo()`.

- **Struttura del Codice**:
  Il gioco è organizzato tramite numerose funzioni che gestiscono specifiche operazioni (pescare carte, calcolare punteggi, gestire lo split, ecc.) e un ciclo principale che regola il flusso complessivo della partita. La logica, pur essendo abbastanza "lineare", integra alcune scelte interattive (input dell'utente per puntate, split, raddoppio e richiesta di ulteriori carte).
- **Algoritmi di Valutazione delle Carte**:
  Le funzioni di calcolo dei punti gestiscono in modo particolare il valore degli Assi, decidendo se contarli come 1 o 11 in base al punteggio attuale. Inoltre, per il confronto delle carte nello split, viene usata una tecnica "hacker" (somma di `int(y[1])` e un valore derivante dalla lunghezza della stringa) per distinguere carte come "10" da quelle a cifra singola.

Questa descrizione riassume in modo dettagliato il funzionamento e la logica implementata nel file Python allegato . Se hai ulteriori domande o necessiti di approfondimenti su qualche parte specifica, chiedi pure!