### 

### **Setup and Planning for Your BitLife-style Game in Python**

#### **Step 1: Planning and Structure**

Before starting to code, it’s useful to have a clear structure and plan. Here’s a roadmap to guide you, along with key game components and suggested time frames.

### **Week 1: Basic Concept & Structure (Text-based)**

**Duration**: 5 days

1. **Brainstorm core gameplay**:
   * **Character creation**: Decide how characters will be created (player choice or random).
   * **Game structure**: How will the player interact with the game? Will events happen at certain ages, or will the player have freedom in decision-making?
2. **Write the game flow**:
   * Decide on the general flow of the game. For example:
     1. Create character
     2. Player starts at age 0
     3. Trigger random life events each year
     4. Player makes decisions (affecting stats)
     5. End the game when character dies or reaches a certain age
   * Define the **core stats** that will change based on decisions (e.g., happiness, health, wealth, intelligence).
3. **Create basic functions**:
   * create\_character(): This will handle character creation.
   * advance\_year(): Moves the character one year forward and triggers life events.
   * display\_stats(): Shows the character’s current stats.

### **Week 2: Implement Key Game Systems**

**Duration**: 7 days

1. **Character Creation**:
   * Allow the player to either create a character by entering details (name, gender) or randomly generate a character.
2. **Event System**:
   * Create a list of possible random events. For each event, give multiple player choices.
   * Sample events:
     + "You’re going to school. Do you want to study hard or make friends?"
     + "You have the opportunity to start a job. Will you take it?"
3. **Decision Impact**:
   * For each decision, determine how it affects the character’s stats.
   * Example: If a player chooses to study hard, it might increase intelligence but decrease happiness.
4. **Stat Tracking**:
   * Implement a system that updates and displays character stats as the game progresses.
   * Stats can be as simple as integers for now.

### **Week 3: Polishing and Expanding**

**Duration**: 5 days

1. **Life Events Expansion**:
   * Add more random life events to make the game dynamic and varied.
   * Categorize events (childhood, adulthood, retirement) to ensure that they appear at the right times.
2. **Introduce aging and milestones**:
   * Age the character each turn.
   * Add special events for milestones like turning 18 (voting, driving) or 65 (retirement).
3. **Game End Conditions**:
   * Define the conditions for ending the game, such as death or reaching a specific age.
   * Give feedback to the player on how they lived (e.g., score based on wealth, happiness, etc.).

### **Week 4: Final Touches and Expansion**

**Duration**: 5 days

1. **Refactoring**:
   * Review your code and refactor it for better readability and efficiency.
   * Organize your code into separate functions or even classes (if familiar with object-oriented programming).
2. **Testing**:
   * Test the game to catch bugs and make sure it’s fun.
   * Get feedback from friends or classmates and implement their suggestions.
3. **Optional Features**:
   * **Career system**: Add jobs, promotions, and career paths.
   * **Relationships**: Allow the player to build friendships or start a family.
   * **Health and aging**: Include health events like getting sick or going to the doctor.

### **Development Plan Overview**

| **Week** | **Goals** |
| --- | --- |
| **Week 1** | Plan game structure, create character and basic game loop. |
| **Week 2** | Implement event system, character stats, and basic decisions. |
| **Week 3** | Expand life events, add milestones, and create end game conditions. |
| **Week 4** | Refactor code, polish gameplay, and add optional features (like careers or relationships). |

### **Key Features List**

* **Character Creation**: Player chooses or randomizes character traits.
* **Life Events**: A system for triggering random life events and allowing player decisions.
* **Stats System**: Tracks key character attributes like health, happiness, and wealth.
* **Aging**: Moves the character through different life stages with new challenges and opportunities.
* **End Conditions**: The game ends based on the character’s choices and stats.

This setup will give you a solid foundation, and you can continue expanding with more features or visuals later!

## **Outline for the Game: "BitLife Basecamp Challenge"**

### **1. Introduction**

In the "BitLife Basecamp Challenge," we will develop a **text-based life simulation game** where the player guides a character's life through a series of choices. Each week, we will continue building on this project, applying the knowledge we have learned in class to expand the game.

The game focuses on making decisions that impact various aspects of the character’s life, such as health, career, relationships, and happiness. Every choice has both immediate and long-term consequences, leading to a unique life path for each player.

### **2. Game Concept**

**Title:** *BitLife Basecamp Challenge***Genre:** Life simulation (text-based)  
**Players:** Single-player  
**Platform:** PC (potential console via Python UI)  
**Goal:** Guide a character from birth to death, making decisions that influence health, relationships, career, and life satisfaction. The player's objective can vary depending on personal preferences—whether to achieve career success, a happy family life, or an adventurous existence.

### **3. Gameplay**

#### **Life Stages:**

1. **Birth and Childhood (0-12 years):**
   * Random generation of background (country, family, socio-economic status).
   * Initial choices: hobbies, school performance, family relationships.
   * Impact on early personal development (cognitive and social skills).
2. **Adolescence (13-18 years):**
   * Choices around school, part-time jobs, friendships.
   * Start forming relationships and making early career decisions (whether or not to study).
   * Impact on academic success, social relationships, and early financial choices.
3. **Adulthood (19-65 years):**
   * Career choices, starting a business, studying or working.
   * Family life: starting relationships, marriage, having children.
   * Financial decisions: investments, loans, buying or renting property.
   * Health decisions: exercise, diet, and unhealthy lifestyle choices.
4. **Old Age (65+ years):**
   * Retirement: whether or not to keep working, managing finances, maintaining health.
   * Potential health problems.
   * Reflecting on life choices: happiness, fulfillment, and legacy.

#### **Choices and Consequences**

* **Decisions** in each life stage affect the character’s stats: health, happiness, finances, and relationships.
* **Random events** like accidents, lottery wins, or global events (recessions, pandemics) add unpredictability to the player's journey.
* **Endgame:** Once the character dies, a summary of their life is provided, highlighting major choices and achievements.

### **4. Visual and Functional Aspects**

#### **User Interface (UI):**

* **Text-based interface** displaying the player's choices and their outcomes.
* **Stats panel:** shows the character’s health, happiness, finances, and relationship statuses.
* **Choice menus:** each life stage offers specific options related to age and the character’s circumstances.

#### **Functional Aspects:**

* **Choice logic:** every decision adjusts the character’s stats and influences future options and events.
* **Random event generator:** to introduce unexpected challenges and opportunities.
* **Progress bar:** displaying the passage of time through the character's life.

### **5. Technical Implementation**

* **Programming Language:** Python
* **Frameworks/Tools:**
  + **Pygame** for potential visual UI elements.
  + **Random module** for generating random events.
  + **JSON/CSV** for saving game data, stats, and life progress.

**Key Features:**

1. **Interactive choice menu system:** allowing players to direct the character’s life.
2. **Stat tracking system:** real-time updates to stats like health, happiness, finances, and relationships based on player choices.
3. **Event handler system:** manages random or pre-scripted events such as promotions or accidents.