PART 1: FUNDAMENTALS OF GAME ANALYSIS & DESIGN

1. Defining "Game" (The Theorists)

- **Ludwig Wittgenstein:** Believed standard definitions fail because "game" covers too many disparate activities.
- **Roger Caillois:** Defined games by 6 characteristics:
    1. **Fun:** Light-hearted.
    2. **Separate:** Circumscribed in time/place.
    3. **Uncertain:** Unforeseeable outcome.
    4. **Non-productive:** Accomplishes nothing useful.
    5. **Governed by Rules:** Different from real life.
    6. **Fictitious:** Awareness of a different reality.
- **Chris Crawford:** "A closed formal system that subjectively represents a subset of reality."

2. The 5 Major Regions of Games

1. **Board Games:** Playing surface, sectors, movable pieces.
2. **Card Games:** Randomness or matching combinations; "victory value".
3. **Athletic Games:** Physical prowess; precise rules on allowed actions.
4. **Children's Games:** Simple physical play; emphasizes social skills/group role.
5. **Computer Games:** The "current fad"; computer acts as opponent and referee.

3. Key Elements & Components

- **4 Common Factors (Components):**
    - **Representation:** Simplified, subjective reality.
    - **Interaction:** Causes and effects (players exploring the system).
    - **Conflict:** Intrinsic to all games (direct/indirect, violent/non-violent).
    - **Safety:** A safe way to experience reality/danger.
- **The Elemental Tetrad (Jesse Schell):**
    - **Mechanics** (Rules)
    - **Story** (Narrative)
    - **Aesthetics** (Visuals/Sound - *Most Visible*)
    - **Technology** (Medium/Code - *Less Visible*)
    - *Note: None is more important than the others.*

4. The Game Designer

- **Identity:** A game designer is defined by the statement "I am a game designer."
- **Requisites:** Knowledge in Anthropology, Architecture, Business, Psychology, Math, Music, etc.
- **The #1 Skill: Listening**.
    - **Who to listen to (5 Categories):** Team, Audience, Game, Client, Self.
- **The Experience:** "All That's Real Is What You Feel." The experience rises *out* of the game.

5. The Pitch (Selling the Game)

- **Value of Ideas:**
    1. Plain Idea = $0.08 (Dime a dozen).

2. Good Idea + Right Place/Time + Sold Convincingly = **$1,000,000+**.

- **The Interest Curve:** Should include a Hook, Rising Action, and Climax.
- **12 Pitch Tips:**
  1. Get in the Door.
  2. Show You Are Serious.
  3. Be Organized.
  4. Be Passionate!
  5. Assume Their Point of View.
  6. Design the Pitch.
  7. Know All the Details.
  8. Exude Confidence.
  9. Be Flexible.
  10. Rehearse.
  11. Get Them to Own It (Richard Garfield tip).
  12. Follow Up.

6. Psychology & Demographics

- **The 5 Domains of Play (Jason Vanden Berghe):**
  - **Novelty:** Openness to experience.
  - **Challenge:** Conscientiousness.
  - **Stimulation:** Extraversion (social).
  - **Harmony:** Agreeableness.
  - **Threat:** Neuroticism/Negative emotion.
- **Gender Differences:**
  - **Women:** Like mental challenges, elegant solutions, customization, context, and characters that represent *themselves*.
  - **Men:** Generally have more leisure time/money for games; different conflict resolution styles.
- **Kaye Elling's 5 C's (For Girls):** Characterization, Context, Control, Customization, Creativity.

PART 2: PYTHON PROGRAMMING

1. Introduction & Syntax

- **History:** Created by **Guido van Rossum** (late 80s), named after *Monty Python*.
- **Characteristics:** Open-source, interpreted, object-oriented, case-sensitive.
- **Structure:**
  - **No braces {} or semicolons ;.**
  - Uses **indentation** and **newlines** to define blocks.
  - Comments start with #.
- **Variables:**
  - Can use letters, numbers, underscores.
  - **Cannot** start with a number.
  - **Cannot** use reserved words.

2. Input and Output (I/O)

- **Input:**
  - **Python 3:** input() (always returns a string).
  - **Python 2:** raw_input() (for string).
  - *Casting input:* int(input()) or float(input()).
- **Output (print):**
  - **Sep:** Separator between items (default \n).
  - **Triple Quotes ("""):** Used for multi-line strings.
  - **Escape Characters:** \t (tab), \n (newline).
- **String Formatting:**
  - {:>10} (Right align).
  - {:^10} (Center align).
  - {:d} (Integer).

    - {:f} (Float).
    - {:06.2f} (Float, width 6, 2 decimal places).
    - {:+d} (Signed integer).

3. Data Types & Operators

- **Primitive Types:** int, float, complex (uses j), str.
- **Math Functions:**
    - abs(x): Absolute value.
    - ceil(x) / floor(x): Round up / down.
    - pow(x, y): x^y.
    - sqrt(x): Square root.
    - round(x): *Tie-breaker rule:* round(0.5) -> 1.0, round(-0.5) -> -1.0.
- **Type Casting Functions:**
    - int(), float(), str().
    - ord(x): Char to Integer.
    - chr(x): Integer to Char.
- **Operators:**
    - **Arithmetic:** +, -, *, / (float division), // (floor division), % (modulus), ** (exponent).
    - **Logical:** and, or, not.
    - **Relational:** <, >, <=, >=, ==, !=.

4. Conditional Statements

- **Decision Making:** Anticipating conditions during execution.
    - **Syntax:**
        - if expression:
        - elif expression: (Optional, can have multiple).
        - else: (Optional, executes if all else fails).
- **Important Note:** Python **does not** have a switch/case statement. You must use if...elif...else to simulate it.

PART 1: BOOLEAN LOGIC & OPERATORS

(Based on File 117)

1. Boolean Expressions

- In Python, logic expressions don't just return True or False; they return the **value** that was interpreted as True or False[1].

2. The AND Operator

- **Rule:** Evaluates expressions from left to right.
- **If all are True:** It returns the **last** expression[2].
- **If one is False:** It returns the **first** value that evaluates to False[3].
    - *Example:* 1 and 2 $\rightarrow$ returns 2[4].
    - *Example:* 1 and 0 $\rightarrow$ returns 0[5].

3. The OR Operator

- **Rule:** Evaluates expressions from left to right.
- **Returns:** The **first** value that evaluates to True[6].
- **If none are True:** It returns the **last** value[7].
    - *Example:* 1 or 2 $\rightarrow$ returns 1[8].
    - *Example:* 0 or [] $\rightarrow$ returns [] (empty list is treated as false)[9].

4. Common Logic Pitfall

- **Wrong Way:** if a > 2 and b > 2: (This actually checks if a is not zero, AND if b > 2) [10].
- **Correct Way:** You must compare variables separately: if a > 2 and b > 2: [11].

PART 2: LOOPS (For and While)

(Based on Files 117 & 118)

1. The range() Function

Used to generate a sequence of numbers for loops.

- **range(a):** Generates from 0 to a-1 [12].
- **range(a, b):** Generates from a to b-1 [13].
- **range(a, b, c):** Generates from a to b-1, incrementing by c (step value) [14].
    - *Reverse Example:* range(10, 1, -2) $\rightarrow$ 10, 8, 6, 4, 2 [15].

2. Print Formatting in Loops

- **end parameter:** By default, print ends with a newline (\n). You can change this to a space to print on one line: print(a, end=' ') [16161616].
- **sep parameter:** Defines what separates arguments (default is a space) [17].

3. The For Loop

- Used to iterate over a sequence (like a list or string) [18].
- **Enumerate:** for index, val in enumerate(list): allows you to get both the index and the value at the same time [19].

4. The While Loop

- Executes a set of statements as long as a condition remains true [20].
- **Use Cases:** Automating tasks, indefinite iteration, or reducing complexity [21].
- **Input Validation:** Often used to force a user to enter correct data (e.g., repeating the input prompt while the number is incorrect) [22].

5. Loop Control Statements

- **break:** Stops/Exits the loop immediately, even if the condition is still true [23].
- **continue:** Stops the *current* iteration only and jumps to the next iteration [24].
- **else (in loops):** Runs **once** after the loop finishes naturally (when the condition becomes false). **It does not run if the loop was stopped by a break** [25].

6. Nested Loops

- A loop inside another loop (Outer Loop and Inner Loop) [26].
- The inner loop runs fully for **each single iteration** of the outer loop [27].

- *Use Case:* Working with 2D arrays or printing patterns (like multiplication tables)[28].

PART 3: PYTHON LISTS

(Based on File 211)

1. Definition & Characteristics

- **List:** A collection of arbitrary objects enclosed in square brackets []29.
- **Ordered:** The order of elements is maintained30.
- **Mutable:** You can change elements after the list is created31313131.
- **Heterogeneous:** Can contain different data types (int, string, float) mixed together32.
- **Dynamic:** Can contain as many objects as memory allows33.

2. Indexing and Slicing

- **0-based Indexing:** The first item is at index 034.
- **Negative Indexing:** list[-1] accesses the **last** item; list[-2] is the second to last35353535.
- **Slicing:** list[0:2] gets items from index 0 up to (but not including) index 236.

3. Modifying Lists

- **Update:** a[1] = 5 changes the value at index 137.
- **Add (Append):** a.append(10) adds an item to the **end**38.
- **Add (Concatenate):** a += [10, 20] or a = a + [10] combines lists39393939.
- **Remove (Value):** a.remove(3) deletes the specific value 340.
- **Remove (Pop):** a.pop() removes the **last** item41.
- **Delete (Index):** del a[3] deletes the item at a specific index42.

4. List Functions

- len(x): Count items43.
- x.sort(): Sort ascending44.
- x.reverse(): Reverse the order45.
- max(x) / min(x): Get largest/smallest value46464646.
- x.count(5): Count how many times '5' appears47.

5. Nested Lists

- A list can contain other lists48.
- **Accessing:** x[1][0] accesses the first item of the sub-list located at index 1 of the main list49.

ARF ARF ARF

Based on the Python files provided (specifically ITP313 211.pptx regarding Lists) and standard Python rules for the other concepts, here are the answers to your reviewer questions:

1. Assuming we have assigned a list to x `x = [2, 6, 3, 1, 10, 5]` and then we have issued a script `len(x)`, What is the output?

- **6**
    - *Reasoning:* `len()` counts the number of items in the list[1].

2. `numList = [7, 12, 'a', 'b', 5.0]`, given the content of the list what is the value of `numList(3)`?

- **b** (If we assume a typo and you meant `numList[3]`)
- **None of the above** (If strict syntax is applied)
    - *Reasoning:* Strictly speaking, `numList(3)` using parentheses causes a **TypeError** because lists are not callable. However, index 3 accessed correctly via `numList[3]` refers to `'b'`[2].

3. Which of the following is not accepted on List?

(Note: This question contains multiple False statements based on the provided choices. The following are NOT accepted/True for lists):

- **List use parenthesis** (False: Lists use square brackets `[]`[3]).
- **Listed are immutable** (False: Lists are mutable[4]).
- **Lists cannot contain any arbitrary objects** (False: They can contain arbitrary objects[5]).
- **Lists are ordered and unordered** (False: Lists are strictly ordered[6]).

4. `numList = [5, 12, 'a', 'b', 5.0]`, given the content of the list what is the value of `numList[1]`?

- **12**
    - *Reasoning:* Python indexing starts at 0. Index 1 is the second item[777].

5. Program execution is faster when manipulating a tuple than it is for the equivalent list.

- **True**
    - *Reasoning:* Tuples are immutable and generally more memory/speed efficient than lists.

6. `numList = [7, 12, 'a', 'b', 5.0]`, given the content of the list what is the value of `numList[-2]`?

- **b**
    - *Reasoning:* Negative indexing counts from the right. -1 is `5.0`, -2 is `'b'`[8888].

7. The order in which you specify the elements when you define a list is an innate characteristic of that list and is maintained for that list's lifetime.

- **True**
  - *Reasoning:* Lists are ordered collections[9].

8. Given `r = ('up', 'dlsu', 'rtu', 'pup', 'umak', 'tcu')`; using negative sequence create a script to display 'rtu' (use lowercase only)

- **`r[-4]`**
  - *Reasoning:* 'tcu'(-1), 'umak'(-2), 'pup'(-3), 'rtu'(-4).

9. List index by default starts at __

- **0**
  - *Reasoning:* List indexing is zero-based[10].

10. Assuming we have assign a list to x `x = ['a', ['bb', ['ccc', 'ddd'], 'ee', 'ff'], 'g', ['hh', 'ii'], 'j']`, what is the output if we issue a script `(x[0], x[2], x[4])`

- **('a', 'g', 'j')**
  - *Reasoning:* `x[0]` is 'a', `x[2]` is 'g', `x[4]` is 'j'[11].

11. Tuples are immutable.

- **True**
  - *Reasoning:* Unlike lists (which are mutable), tuples cannot be changed after creation.

12. A given object can appear in a list multiple times.

- **True**
  - *Reasoning:* List objects need not be unique[12].

13. `numList = [7, 12, 'a', 'b', 5.0]`, given the content of the list what is the value of `numList[1:2]`?

- **None of the above**
  - *Reasoning:* The slice `[1:2]` starts at index 1 and stops *before* index 2. The result is the list `[12]`. None of the provided options (like `[7, 12]`) match this exactly.

14. Assuming we have assign a list to val val = [1, 2, 3, 4] and then we have issued a script a[2]=5, What would be the content of the list?

(Assuming a was a typo for val)

- **[1, 2, 5, 4]**
  - *Reasoning:* The item at index 2 (which was 3) is replaced by 5[13].

15. A dictionary is define by enclosing a comma-separated list of key-value pairs in curly braces. A colon separates each key from its associated value.

- **True**

16. List can be and cannot be of varying types.

- **True**

- *Reasoning:* Lists can contain elements all of the same type OR elements of varying types[14141414].

17. Given `r = ('up', 'dlsu', 'rtu', 'pup', 'umak', 'tcu')`; using positive sequence create a script to display 'rtu' (use lowercase only)

- **`r[2]`**
  - *Reasoning:* 0='up', 1='dlsu', 2='rtu'.

18. Program execution is faster when manipulating a list than it is for the equivalent tuple

- **False**
  - *Reasoning:* Lists are slower than tuples because of the overhead required to make them mutable.

19. `t = ('foo', 'bar', 'baz', 'qux', 'quux', 'corge')`, issue a script to display 'baz'.

- **`t[2]`**
  - *Reasoning:* Tuples use square brackets `[]` for accessing indices, just like lists.

20. Create a script to create student dictionary.

- **`student = {}`** or **`student = dict()`**

21. `numList = [7, 12, 'a', 'b', 5.0]`, given the content of the list what is the value of `numList[2]`?

- **a**
  - *Reasoning:* Index 0=7, 1=12, 2='a'[15].

22. If the values in the collection are meant to remain constant for the life of the program, using list instead of a tuple guards against accidental modification.

- **False**
  - *Reasoning:* You should use a **Tuple** (immutable), not a list (mutable), to guard against modification.

23. `numList = [7, 12, 'a', 'b', 5.0]`, given the content of the list what is the value of `numList[5]`?

- **None of the above**
  - *Reasoning:* The list has indices 0 to 4. Index 5 raises an `IndexError`.

24. Individual elements in a list can be accessed using an index in parenthesis.

- **False**
  - *Reasoning:* Elements are accessed using square brackets `[]`[16].

25. `numList = [7, 12, 'a', 'b', 5.0]`, given the content of the list what is the value of `numList[-4]`?

- **12**
  - *Reasoning:* -1=5.0, -2='b', -3='a', -4=12.

ARF ARF ARF

1. Assuming we have assigned a list to x `x = [2, 6, 3, 1, 10, 5]` and then we have issued a script `len(x)`, What is the output?

* **6**

    * [cite_start]*Reasoning:* `len()` counts the number of items in the list[cite: 79].

2. `numList = [7, 12, 'a', 'b', 5.0]`, given the content of the list what is the value of `numList(3)`?

* **b** (If we assume a typo and you meant `numList[3]`)

* **None of the above** (If strict syntax is applied)

    * *Reasoning:* Strictly speaking, `numList(3)` using parentheses causes a **TypeError** because lists are not callable. [cite_start]However, index 3 accessed correctly via `numList[3]` refers to `'b'`[cite: 40].

3. Which of the following is not accepted on List?

*(Note: This question contains multiple False statements based on the provided choices. The following are **NOT** accepted/True for lists):*

* [cite_start]**List use parenthesis** (False: Lists use square brackets `[]` [cite: 5]).

* [cite_start]**Listed are immutable** (False: Lists are mutable [cite: 11]).

* [cite_start]**Lists cannot contain any arbitrary objects** (False: They can contain arbitrary objects [cite: 8]).

* [cite_start]**Lists are ordered and unordered** (False: Lists are strictly ordered [cite: 13]).

4. `numList = [5, 12, 'a', 'b', 5.0]`, given the content of the list what is the value of `numList[1]`?

* **12**

    * [cite_start]*Reasoning:* Python indexing starts at 0. Index 1 is the second item[cite: 41, 61].

5. Program execution is faster when manipulating a tuple than it is for the equivalent list.

* **True**

    * *Reasoning:* Tuples are immutable and generally more memory/speed efficient than lists.

6. `numList = [7, 12, 'a', 'b', 5.0]`, given the content of the list what is the value of `numList[-2]`?

* **b**

    * *Reasoning:* Negative indexing counts from the right. [cite_start]-1 is `5.0`, -2 is `'b'`[cite: 52, 55].

7. The order in which you specify the elements when you define a list is an innate characteristic of that list and is maintained for that list's lifetime.

* **True**

    * [cite_start]*Reasoning:* Lists are ordered collections[cite: 15].

8. Given `r = ('up', 'dlsu', 'rtu', 'pup', 'umak', 'tcu')`; using negative sequence create a script to display 'rtu' (use lowercase only)

* **`r[-4]`**

    * *Reasoning:* 'tcu'(-1), 'umak'(-2), 'pup'(-3), 'rtu'(-4).

9. List index by default starts at __

* **0**

    * [cite_start]*Reasoning:* List indexing is zero-based[cite: 41].

10. Assuming we have assign a list to x `x = ['a', ['bb', ['ccc', 'ddd'], 'ee', 'ff'], 'g', ['hh', 'ii'], 'j']`, what is the output if we issue a script `(x[0], x[2], x[4])`

* **('a', 'g', 'j')**

    * [cite_start]*Reasoning:* `x[0]` is 'a', `x[2]` is 'g', `x[4]` is 'j' [cite: 102-103].

11. Tuples are immutable.

* **True**

    * *Reasoning:* Unlike lists (which are mutable), tuples cannot be changed after creation.

12. A given object can appear in a list multiple times.

* **True**

    * [cite_start]*Reasoning:* List objects need not be unique[cite: 35].

13. `numList = [7, 12, 'a', 'b', 5.0]`, given the content of the list what is the value of `numList[1:2]`?

* **None of the above**

    * *Reasoning:* The slice `[1:2]` starts at index 1 and stops *before* index 2. The result is the list `[12]`. None of the provided options (like `[7, 12]`) match this exactly.

14. Assuming we have assign a list to val `val = [1, 2, 3, 4]` and then we have issued a script `a[2]=5`, What would be the content of the list?

(Assuming `a` was a typo for `val`)

* **[1, 2, 5, 4]**

    * [cite_start]*Reasoning:* The item at index 2 (which was 3) is replaced by 5[cite: 68].

**15. A dictionary is define by enclosing a comma-separated list of key-value pairs in

curly braces. A colon separates each key from its associated value.**

* **True**

16. List can be and cannot be of varying types.

* **True**

    * [cite_start]*Reasoning:* Lists can contain elements all of the same type OR elements of varying types[cite: 26, 30].

17. Given `r = ('up', 'dlsu', 'rtu', 'pup', 'umak', 'tcu')`; using positive sequence create a script to display 'rtu' (use lowercase only)

* **`r[2]`**

    * *Reasoning:* 0='up', 1='dlsu', 2='rtu'.

18. Program execution is faster when manipulating a list than it is for the equivalent tuple

* **False**

    * *Reasoning:* Lists are slower than tuples because of the overhead required to make them mutable.

19. `t = ('foo', 'bar', 'baz', 'qux', 'quux', 'corge')`, issue a script to display 'baz'.

* **`t[2]`**

    * *Reasoning:* Tuples use square brackets `[]` for accessing indices, just like lists.

20. Create a script to create student dictionary.

* **`student = {}`** or **`student = dict()`**

21. `numList = [7, 12, 'a', 'b', 5.0]`, given the content of the list what is the value of `numList[2]`?

* **a**

    * [cite_start]*Reasoning:* Index 0=7, 1=12, 2='a'[cite: 48].

22. If the values in the collection are meant to remain constant for the life of the program, using list instead of a tuple guards against accidental modification.

* **False**

    * *Reasoning:* You should use a **Tuple** (immutable), not a list (mutable), to guard against modification.

23. `numList = [7, 12, 'a', 'b', 5.0]`, given the content of the list what is the value of `numList[5]`?

* **None of the above**

    * *Reasoning:* The list has indices 0 to 4. Index 5 raises an `IndexError`.

24. Individual elements in a list can be accessed using an index in parenthesis.

* **False**

    * [cite_start]*Reasoning:* Elements are accessed using square brackets `[]`[cite: 40].

25. `numList = [7, 12, 'a', 'b', 5.0]`, given the content of the list what is the value of `numList[-4]`?

* **12**

    * *Reasoning:* -1=5.0, -2='b', -3='a', -4=12.