**Pseudocode for the game!**

**Huzaifa Noob!**

### Initialize Variables

- Set `x` to 0

- Set `y` to 0

- Set `moves\_left` to 10

### Define Movement Functions

- Define function `down(x, y)`:

- Return `(x, y + 1)`

- Define function `up(x, y)`:

- Return `(x, y - 1)`

- Define function `left(x, y)`:

- Return `(x - 1, y)`

- Define function `right(x, y)`:

- Return `(x + 1, y)`

### Define Grid Functions

- Define function `create\_grid(x, y)`:

- Set `grid\_size` to 5

- Initialize `grid` as a 5x5 2D array filled with '.'

- If `0 <= y < grid\_size` and `0 <= x < grid\_size`:

- Set `grid[y][x]` to 'P' (current position)

- Set `grid[2][0]` to 'X' (destination)

- Return `grid`

- Define function `print\_grid(grid)`:

- For each `row` in `grid`:

- Print `row` joined by spaces

- Print a newline

### Main Loop

- While `moves\_left` is greater than 0:

- Call `create\_grid(x, y)` and assign the result to `grid`

- Call `print\_grid(grid)`

- If `x == 0` and `y == 2`:

- Print "Congratulations! You found 'X' and won the game!"

- Break out of the loop

- Prompt user for input with "Enter Next Move! (Press 0 for down, 2 for up, 1 for left, and 3 for right, or 9 to exit): "

- Assign the input to `move`

- If `move == 0`:

- Print "Moving Down"

- Call `down(x, y)` and assign the result to `(x, y)`

- Else if `move == 2`:

- Print "Moving Up"

- Call `up(x, y)` and assign the result to `(x, y)`

- Else if `move == 1`:

- Print "Moving Left"

- Call `left(x, y)` and assign the result to `(x, y)`

- Else if `move == 3`:

- Print "Moving Right"

- Call `right(x, y)` and assign the result to `(x, y)`

- Else if `move == 9`:

- Print "Exiting..."

- Break out of the loop

- Else:

- Print "Not a valid key press"

- Continue to the next iteration of the loop

- If `x < 0` or `x > 4` or `y < 0` or `y > 4`:

- Print "Game Over! 'P' went out of bounds."

- Break out of the loop

- Decrement `moves\_left` by 1

- Print "Moves left: `moves\_left`"

- Print five newlines for readability

### End Game Check

- If `moves\_left` is 0 and `x != 0` or `y != 2`:

- Print "Game Over! You didn't find 'X' within 10 moves."

Sure, let's break down and explain the entire code step by step.

### Game Setup

1. \*\*Initialize Starting Position:\*\*

- `x, y = 0, 0`: The player starts at the top-left corner of the grid, which is position (0, 0).

- `moves\_left = 10`: The player has 10 moves to find the secret cell.

### Movement Functions

These functions define how the player moves within the grid.

2. \*\*Move Down:\*\*

```python

def down(x, y):

"""Move down (increase y-coordinate)."""

return x, y + 1

```

- This function increases the `y` coordinate, moving the player down one row.

3. \*\*Move Up:\*\*

```python

def up(x, y):

"""Move up (decrease y-coordinate)."""

return x, y - 1

```

- This function decreases the `y` coordinate, moving the player up one row.

4. \*\*Move Left:\*\*

```python

def left(x, y):

"""Move left (decrease x-coordinate)."""

return x - 1, y

```

- This function decreases the `x` coordinate, moving the player left one column.

5. \*\*Move Right:\*\*

```python

def right(x, y):

"""Move right (increase x-coordinate)."""

return x + 1, y

```

- This function increases the `x` coordinate, moving the player right one column.

### Grid Functions

6. \*\*Create Grid:\*\*

```python

def create\_grid(x, y):

"""Create a 5x5 grid and mark the current position and destination."""

grid\_size = 5

grid = [['.' for \_ in range(grid\_size)] for \_ in range(grid\_size)]

if 0 <= y < grid\_size and 0 <= x < grid\_size:

grid[y][x] = 'P' # Mark current position

grid[4][1] = 'X' # Mark destination

return grid

```

- `grid\_size = 5`: Sets the grid size to 5x5.

- `grid = [['.' for \_ in range(grid\_size)] for \_ in range(grid\_size)]`: Creates a 5x5 grid filled with dots (`.`), representing empty cells.

- `if 0 <= y < grid\_size and 0 <= x < grid\_size`: Checks if the player's position is within the grid boundaries.

- `grid[y][x] = 'P'`: Marks the player's current position with `P`.

- `grid[4][1] = 'X'`: Sets the secret cell (destination) at position (1, 4) (this should actually be random in a real game).

- `return grid`: Returns the created grid.

7. \*\*Print Grid:\*\*

```python

def print\_grid(grid):

"""Print the grid."""

for row in grid:

print(' '.join(row))

print()

```

- This function prints each row of the grid, joining the elements with spaces for better readability.

### Main Game Loop

8. \*\*Game Loop:\*\*

```python

while moves\_left > 0:

```

- The game runs while the player has moves left.

9. \*\*Create and Print Grid:\*\*

```python

grid = create\_grid(x, y)

print\_grid(grid)

```

- The current grid is created and printed.

10. \*\*Check for Win:\*\*

```python

if x == 1 and y == 4:

print("Congratulations! You found 'X' and won the game!")

break

```

- Checks if the player has reached the secret cell. If true, the player wins and the game ends.

11. \*\*Get Player Move:\*\*

```python

move = int(input("Enter Next Move! (Press 0 for down, 2 for up, 1 for left, and 3 for right, or 9 to exit): "))

```

- Prompts the player to enter their next move.

12. \*\*Update Player Position:\*\*

```python

if move == 0:

print("Moving Down")

x, y = down(x, y)

elif move == 2:

print("Moving Up")

x, y = up(x, y)

elif move == 1:

print("Moving Left")

x, y = left(x, y)

elif move == 3:

print("Moving Right")

x, y = right(x, y)

elif move == 9:

print("Exiting...")

break

else:

print("Not a valid key press")

continue

```

- Updates the player's position based on the input move. If the input is invalid, it prompts again.

13. \*\*Check Boundaries:\*\*

```python

if x < 0 or x > 4 or y < 0 or y > 4:

print("Game Over! 'P' went out of bounds.")

break

```

- Checks if the player has moved out of bounds. If true, the game ends.

14. \*\*Decrement Moves:\*\*

```python

moves\_left -= 1

print(f"Moves left: {moves\_left}")

```

- Reduces the number of moves left and prints the remaining moves.

15. \*\*Clear Console:\*\*

```python

print("\n" \* 5)

```

- Adds new lines for readability (simulating a clear console effect).

### End Game Check

16. \*\*Game Over Check:\*\*

```python

if moves\_left == 0 and (x != 0 or y != 2):

print("Game Over! You didn't find 'X' within 10 moves.")

```

- If the player has used all moves and hasn't found the secret cell, the game ends with a loss message.

This code simulates a simple game where the player moves around a grid to find a secret cell within a limited number of moves.