Certainly! Let's walk through your Python slot machine code step by step, adding detailed comments to help a complete beginner understand the fundamentals of Python programming.

**🎰 Slot Machine Game in Python: A Beginner's Guide**

This Python program simulates a simple slot machine game. It allows a user to deposit money, place bets on multiple lines, spin the slot machine, and potentially win based on the outcome. The game continues until the user decides to quit.([Medium](https://medium.com/%40eulene/building-a-simple-slot-machine-in-python-58c68df154ba?utm_source=chatgpt.com))

Through this code, you'll learn:

* How to define and use variables
* Working with data structures like dictionaries and lists
* Defining and calling functions
* Using loops (for, while) and conditional statements (if, else)
* Getting user input and displaying output
* Importing and using modules([Infosec Institute](https://www.infosecinstitute.com/resources/secure-coding/python-language-basics-variables-lists-loops-functions-and-conditionals/?utm_source=chatgpt.com))

**📦 Importing Modules**

import random # Importing the 'random' module to use functions that generate random numbers.

* import is used to include external modules in your program.
* The random module provides functions to generate random numbers, which is essential for simulating the randomness of a slot machine.

**🔢 Defining Constants**

MAX\_LINES = 3 # Maximum number of lines a user can bet on.

MAX\_BET = 100 # Maximum amount a user can bet on a single line.

MIN\_BET = 1 # Minimum amount a user can bet on a single line.

ROWS = 3 # Number of rows in the slot machine.

COLS = 3 # Number of columns in the slot machine.

* Constants are variables that are intended to remain unchanged throughout the program.
* By convention, constants are written in uppercase letters.

**🎰 Defining Symbols and Their Values**

symbol\_count = {

"A": 2, # Symbol 'A' appears 2 times.

"B": 4, # Symbol 'B' appears 4 times.

"C": 6, # Symbol 'C' appears 6 times.

"D": 8 # Symbol 'D' appears 8 times.

}

symbol\_value = {

"A": 5, # Symbol 'A' has a value of 5.

"B": 4, # Symbol 'B' has a value of 4.

"C": 3, # Symbol 'C' has a value of 3.

"D": 2 # Symbol 'D' has a value of 2.

}

* Dictionaries (dict) store data in key-value pairs.
* Here, symbol\_count defines how many times each symbol appears, affecting the probability of each symbol.
* symbol\_value assigns a monetary value to each symbol, determining the payout for winning combinations.([W3Schools.com](https://www.w3schools.com/python/python_conditions.asp?utm_source=chatgpt.com))

**🏆 Checking for Winnings**

def check\_winnings(columns, lines, bet, values):

winnings = 0 # Initialize total winnings to zero.

winning\_lines = [] # List to store the lines that resulted in a win.

for line in range(lines): # Iterate over each line the user bet on.

symbol = columns[0][line] # Take the symbol from the first column for the current line.

for column in columns: # Iterate over each column.

symbol\_to\_check = column[line] # Get the symbol in the current column and line.

if symbol != symbol\_to\_check: # If the symbol doesn't match the one from the first column:

break # Exit the inner loop; this line is not a winning line.

else:

# If the loop wasn't broken, all symbols in this line match.

winnings += values[symbol] \* bet # Calculate winnings for this line.

winning\_lines.append(line + 1) # Store the winning line number (1-based index).

return winnings, winning\_lines # Return total winnings and list of winning lines.

* def is used to define a function.
* for loops are used to iterate over sequences.
* The else block after a for loop executes only if the loop wasn't terminated by a break statement.([GeeksforGeeks](https://www.geeksforgeeks.org/loops-in-python/?utm_source=chatgpt.com" \o "Loops in Python – For, While and Nested Loops | GeeksforGeeks), [Python documentation](https://docs.python.org/3/tutorial/controlflow.html?utm_source=chatgpt.com))

**🎰 Spinning the Slot Machine**

def get\_slot\_machine\_spin(rows, cols, symbols):

all\_symbols = [] # List to hold all symbols based on their count.

for symbol, symbol\_count in symbols.items(): # Iterate over each symbol and its count.

for \_ in range(symbol\_count): # Add the symbol to the list 'symbol\_count' times.

all\_symbols.append(symbol)

columns = [] # List to hold the final columns of the slot machine.

for \_ in range(cols): # For each column:

column = [] # Initialize an empty list for the current column.

current\_symbols = all\_symbols[:] # Make a copy of all\_symbols to avoid modifying the original list.

for \_ in range(rows): # For each row in the column:

value = random.choice(current\_symbols) # Randomly select a symbol.

current\_symbols.remove(value) # Remove the selected symbol to prevent duplicates in the same column.

column.append(value) # Add the selected symbol to the current column.

columns.append(column) # Add the completed column to the list of columns.

return columns # Return the final slot machine columns.

* random.choice() selects a random element from a list.
* list[:] creates a shallow copy of the list.
* list.remove(value) removes the first occurrence of value from the list.

**🖨️ Displaying the Slot Machine**

def print\_slot\_machine(columns):

for row in range(len(columns[0])): # Iterate over each row index.

for i, column in enumerate(columns): # Iterate over each column with its index.

if i != len(columns) - 1: # If it's not the last column:

print(column[row], end=" | ") # Print the symbol with a separator.

else:

print(column[row], end="") # Print the symbol without a separator.

print() # Move to the next line after printing all columns for the current row.

* enumerate() allows iteration over a sequence while keeping track of the index.
* print() with end="" prevents automatic newline after printing.

**💰 Handling Deposits**

def deposit():

while True: # Infinite loop until a valid amount is entered.

amount = input("What would you like to deposit? $") # Prompt user for deposit amount.

if amount.isdigit(): # Check if the input is a digit.

amount = int(amount) # Convert the input string to an integer.

if amount > 0: # Check if the amount is greater than zero.

break # Exit the loop if a valid amount is entered.

else:

print("Amount must be greater than 0.") # Inform user of invalid input.

else:

print("Please enter a number.") # Inform user of invalid input.

return amount # Return the valid deposit amount.

* input() reads a line from input.
* str.isdigit() checks if all characters in the string are digits.
* int() converts a string to an integer.

**🔢 Getting Number of Lines to Bet On**

def get\_number\_of\_lines():

while True: # Infinite loop until a valid number of lines is entered.

lines = input("Enter the number of lines to bet on (1-" + str(MAX\_LINES) + ")? ") # Prompt user.

if lines.isdigit(): # Check if the input is a digit.

lines = int(lines) # Convert the input string to an integer.

if 1 <= lines <= MAX\_LINES: # Check if the number is within the valid range.

break # Exit the loop if a valid number is entered.

else:

print("Enter a valid number of lines.") # Inform user of invalid input.

else:

print("Please enter a number.") # Inform user of invalid input.

return lines # Return the valid number of lines.

* Combines user input with validation to ensure correct program behavior.

**💵 Getting Bet Amount**

def get\_bet():

while True: # Infinite loop until a valid bet amount is entered.

amount = input("What would you like to bet on each line? $") # Prompt user.

if amount.isdigit(): # Check if the input is a digit.

amount = int(amount) # Convert the input string to an integer.

if MIN\_BET <= amount <= MAX\_BET: # Check if the amount is within the valid range.

break # Exit the loop if a valid amount is entered.

else:

print(f"Amount must be between {MIN\_BET} - {MAX\_BET}.") # Inform user of invalid input.

else:

print("Please enter a number.") # Inform user of invalid input.

return amount # Return the valid bet amount.

* f"..." is an f-string, allowing expressions inside string literals.

**🎮 Spinning the Slot Machine**

def spin(balance):

lines = get\_number\_of\_lines() # Get the number of lines the user wants to bet on.

while True: # Infinite loop until a valid total bet is entered.

bet = get\_bet() # Get the bet amount per line.

total\_bet = bet \* lines # Calculate the total bet.

if total\_bet > balance: # Check if

::contentReference[oaicite:106]{index=106}