“**Module 2: Lab 2 - Simulating Computer Processes with Python”**

Harsh Siddhapura

“Department of Information Technology, Arizona State University”

“IFT 510: Principles of Computer Information and Technology”

“Dr. Dinesh Sthapit”

“September 01, 2023”

“Report”

“Binary System in Computing”

“The Importance of the Binary System in Computing”

The binary system is a fundamental concept in computing that underlies the representation and manipulation of data in modern computers. Unlike the decimal system, which uses ten digits (0-9), the binary system uses only two digits: 0 and 1. This seemingly simple system is of paramount importance in computing for several reasons.

- Digital Representation: Computers are inherently digital machines, meaning they process and store data in discrete chunks. The binary system provides a natural way to represent these discrete states. Binary digits, or bits, are the smallest units of data in computing. They can represent two states, often interpreted as "off" (0) and "on" (1), which correspond to the physical states of transistors in a computer's electronic circuits.

- Simplicity of Hardware: Binary representations simplify the design of hardware components in computers. Electronic devices can be built with switches that have two states, making it easier to create logical circuits that perform operations such as addition, subtraction, and more complex functions.

- Data Storage: All data in a computer, whether it's text, images, audio, or software instructions, is ultimately represented in binary. Each piece of data is broken down into a sequence of binary digits. This uniform representation allows different types of data to be stored and processed using the same underlying principles.

- Logical Operations: Binary arithmetic forms the foundation of logical operations in computing. Logical gates, such as AND, OR, and NOT gates, manipulate binary data to perform calculations and make decisions. These gates are the building blocks of complex computations and decision-making processes within computers.

- Efficiency: Binary representations are well-suited for electronic systems. They require minimal resources to represent and process information. This efficiency is critical in designing and manufacturing compact and high-performance computing devices.

- Error Detection and Correction: Binary representations facilitate error detection and correction techniques. Redundancy in binary codes allows computers to identify errors that may occur during data transmission or storage. Techniques like parity bits and checksums rely on binary representations to ensure data integrity.

- Compatibility: The binary system is a universal language in computing. It enables interoperability between different hardware and software systems. Data can be easily exchanged and interpreted across various platforms as long as they adhere to binary encoding standards.

“Simulation of Human-Computer Interaction”

The provided program simulates the human-computer interaction process by demonstrating how a computer takes input from a user, processes it, and provides an output. Let's break down each step:

- Input Collection (`get_key_input`): In this step, the program simulates the user pressing a key. The `get_key_input` function prompts the user to press a key and captures the input. If the input is not a single key, the program requests input again until a single key is provided.

- Processing (`ascii_mapping` and `convert_to_binary`): The program then processes the key input. The `ascii_mapping` function calculates the ASCII value of the input key. The ASCII value is an integer representation of the character according to the ASCII encoding standard. The `convert_to_binary` function takes the ASCII value and converts it to its binary representation (excluding the "0b" prefix that Python's `bin()` function generates).

- Output Display (`display_binary`): Finally, the binary representation is displayed using the `display_binary` function. The program shows the user the binary representation of the ASCII value derived from their key input.
- Main Program (`main`): The `main` function orchestrates the entire process. It calls each step's function in sequence to simulate the interaction. The user provides a key, sees its ASCII value, and then views its binary representation.

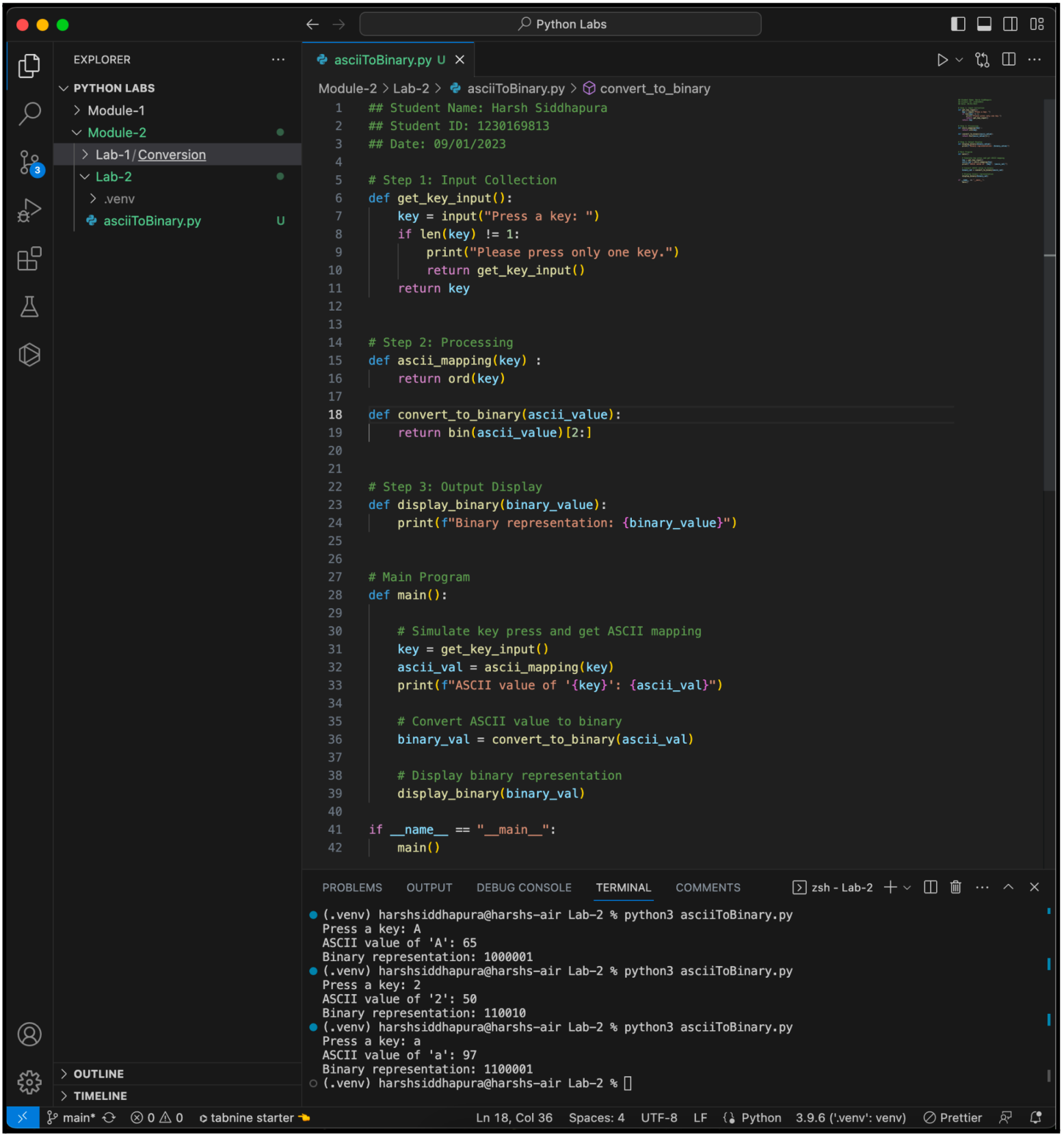
This program demonstrates how computers process user input, manipulate data, and generate output, which is a core aspect of human-computer interaction. The use of the binary system is crucial in the conversion and representation of data at each stage of this interaction.

The binary system's significance in human-computer interaction extends beyond its technical underpinnings. It forms the bridge between human understanding and the digital world, allowing humans to communicate their intentions to computers and enabling computers to respond in a meaningful way. This abstraction enables users to convey complex ideas and instructions using a sequence of just two symbols, 0 and 1, which computers can interpret and act upon.

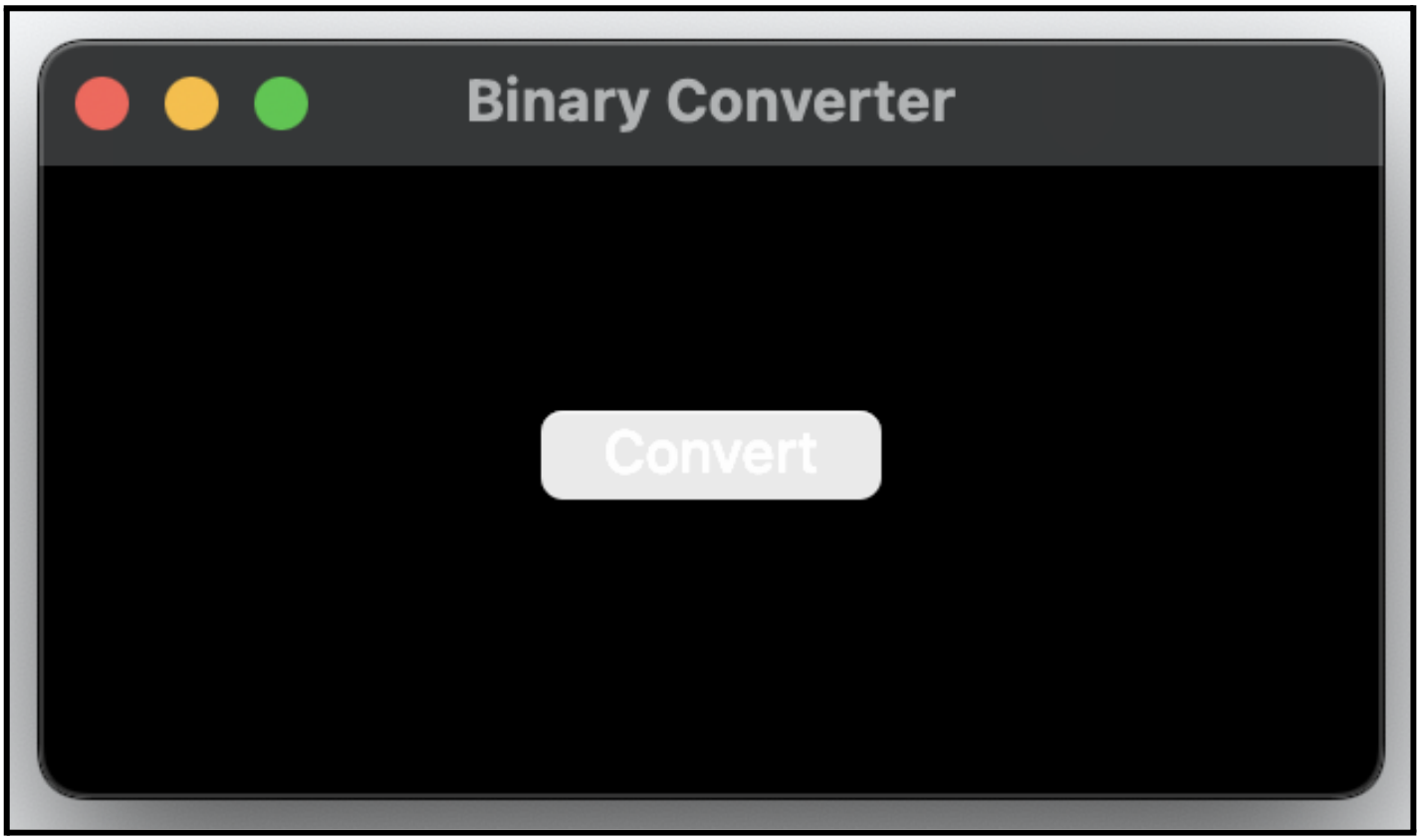
Moreover, the binary system's role in human-computer interaction highlights the remarkable synergy between human creativity and machine processing. The program showcases this synergy by taking a single keystroke, a simple human action, and translating it into a series of binary digits that represent a fundamental building block of computation. This exemplifies how computers, with their binary-based operations, can process vast amounts of data and perform intricate tasks, empowering humans to achieve feats that were once considered unimaginable.

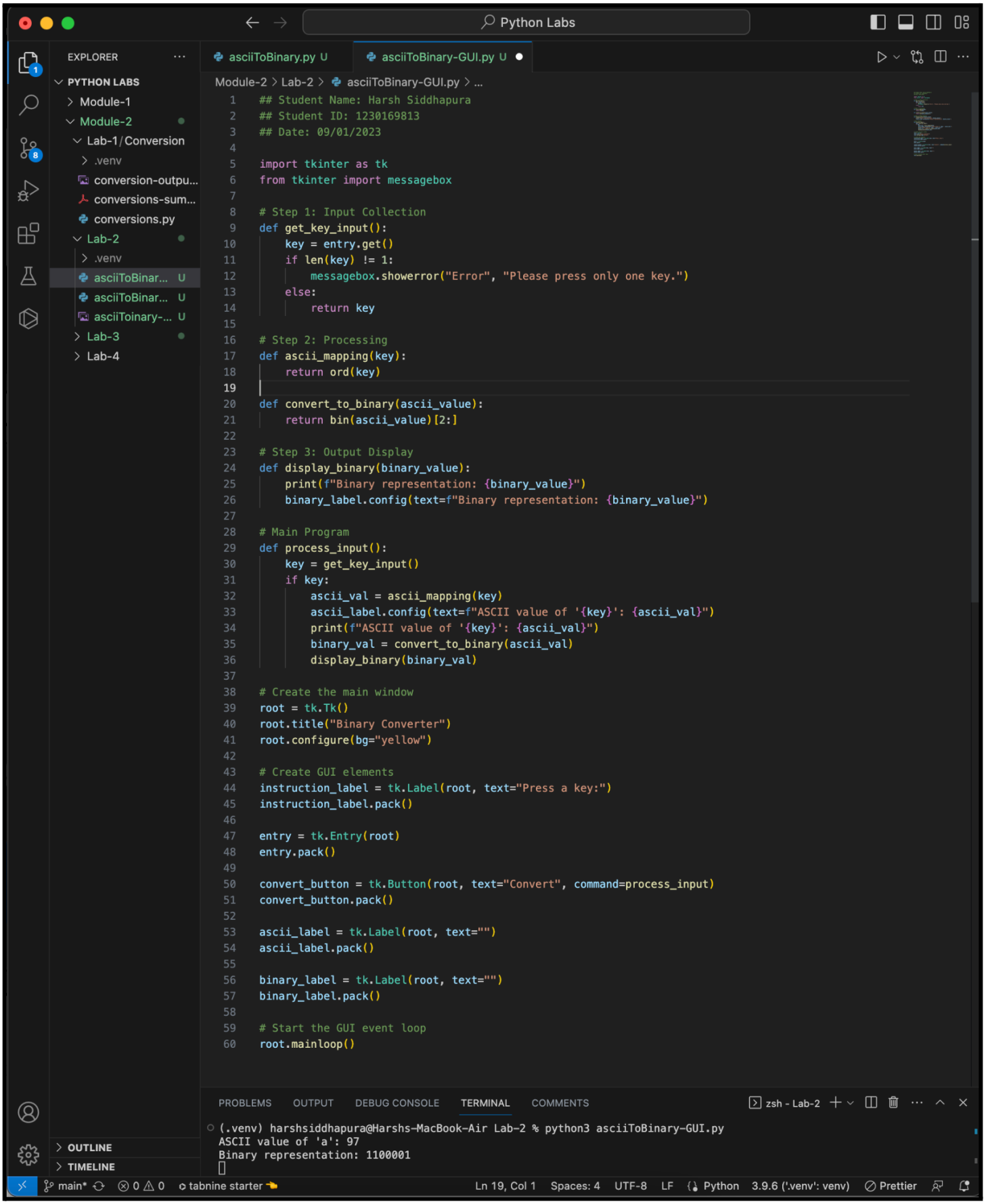
In conclusion, the binary system's significance in computing is profound and pervasive, permeating every facet of modern technology and human-computer interaction. Its role as the foundation of data representation, storage, processing, and communication underscores its indispensability in the digital age. The provided program aptly illustrates how the binary system enables the translation of human input into machine-understandable data, exemplifying the remarkable partnership between human creativity and computational capability.

Without GUI


Python Labs
EXPLORER
PYTHON LABS
Module-1
Module-2
Lab-1 / Conversion
Lab-2
.venv
asciiToBinary.py
OUTLINE
TIMELINE
asciiToBinary.py U
Module-2 > Lab-2 > asciiToBinary.py > convert_to_binary
1 ## Student Name: Harsh Siddhapura
2 ## Student ID: 1230169813
3 ## Date: 09/01/2023
4
5 # Step 1: Input Collection
6 def get_key_input():
7 key = input("Press a key: ")
8 if len(key) != 1:
9 print("Please press only one key.")
10 return get_key_input()
11 return key
12
13
14 # Step 2: Processing
15 def ascii_mapping(key) :
16 return ord(key)
17
18 def convert_to_binary(ascii_value):
19 return bin(ascii_value)[2:]
20
21
22 # Step 3: Output Display
23 def display_binary(binary_value):
24 print(f"Binary representation: {binary_value}")
25
26
27 # Main Program
28 def main():
29
30 # Simulate key press and get ASCII mapping
31 key = get_key_input()
32 ascii_val = ascii_mapping(key)
33 print(f"ASCII value of '{key}': {ascii_val}")
34
35 # Convert ASCII value to binary
36 binary_val = convert_to_binary(ascii_val)
37
38 # Display binary representation
39 display_binary(binary_val)
40
41 if __name__ == "__main__":
42 main()
PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
COMMENTS
zsh - Lab-2
(.venv) harshsiddhapura@harshs-air Lab-2 % python3 asciiToBinary.py
Press a key: A
ASCII value of 'A': 65
Binary representation: 1000001
(.venv) harshsiddhapura@harshs-air Lab-2 % python3 asciiToBinary.py
Press a key: 2
ASCII value of '2': 50
Binary representation: 110010
(.venv) harshsiddhapura@harshs-air Lab-2 % python3 asciiToBinary.py
Press a key: a
ASCII value of 'a': 97
Binary representation: 1100001
(.venv) harshsiddhapura@harshs-air Lab-2 %
main*
0 0
tabnine starter
Ln 18, Col 36
Spaces: 4
UTF-8
LF
Python
3.9.6 ('.venv': venv)
Prettier

With GUI




Python Labs
EXPLORER
PYTHON LABS
Module-1
Module-2
Lab-1 / Conversion
.venv
conversion-outpu...
conversions-sum...
conversions.py
Lab-2
.venv
asciiToBinar... U
asciiToBinar... U
asciiToinary-... U
Lab-3
Lab-4
asciiToBinary.py U
asciiToBinary-GUI.py U
Module-2 > Lab-2 > asciiToBinary-GUI.py > ...
Student Name: Harsh Siddhapura
Student ID: 1230169813
Date: 09/01/2023
import tkinter as tk
from tkinter import messagebox
# Step 1: Input Collection
def get_key_input():
key = entry.get()
if len(key) != 1:
messagebox.showerror("Error", "Please press only one key.")
else:
return key
# Step 2: Processing
def ascii_mapping(key):
return ord(key)
def convert_to_binary(ascii_value):
return bin(ascii_value)[2:]
# Step 3: Output Display
def display_binary(binary_value):
print(f"Binary representation: {binary_value}")
binary_label.config(text=f"Binary representation: {binary_value}")
# Main Program
def process_input():
key = get_key_input()
if key:
ascii_val = ascii_mapping(key)
ascii_label.config(text=f"ASCII value of '{key}': {ascii_val}")
print(f"ASCII value of '{key}': {ascii_val}")
binary_val = convert_to_binary(ascii_val)
display_binary(binary_val)
# Create the main window
root = tk.Tk()
root.title("Binary Converter")
root.configure(bg="yellow")
# Create GUI elements
instruction_label = tk.Label(root, text="Press a key:")
instruction_label.pack()
entry = tk.Entry(root)
entry.pack()
convert_button = tk.Button(root, text="Convert", command=process_input)
convert_button.pack()
ascii_label = tk.Label(root, text="")
ascii_label.pack()
binary_label = tk.Label(root, text="")
binary_label.pack()
# Start the GUI event loop
root.mainloop()
PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
COMMENTS
zsh - Lab-2
(.venv) harshsiddhapura@Harshs-MacBook-Air Lab-2 % python3 asciiToBinary-GUI.py
ASCII value of 'a': 97
Binary representation: 1100001
OUTLINE
TIMELINE
main*
0 0
tabnine starter
Ln 19, Col 1
Spaces: 4
UTF-8
LF
Python
3.9.6 ('.venv': venv)
Prettier