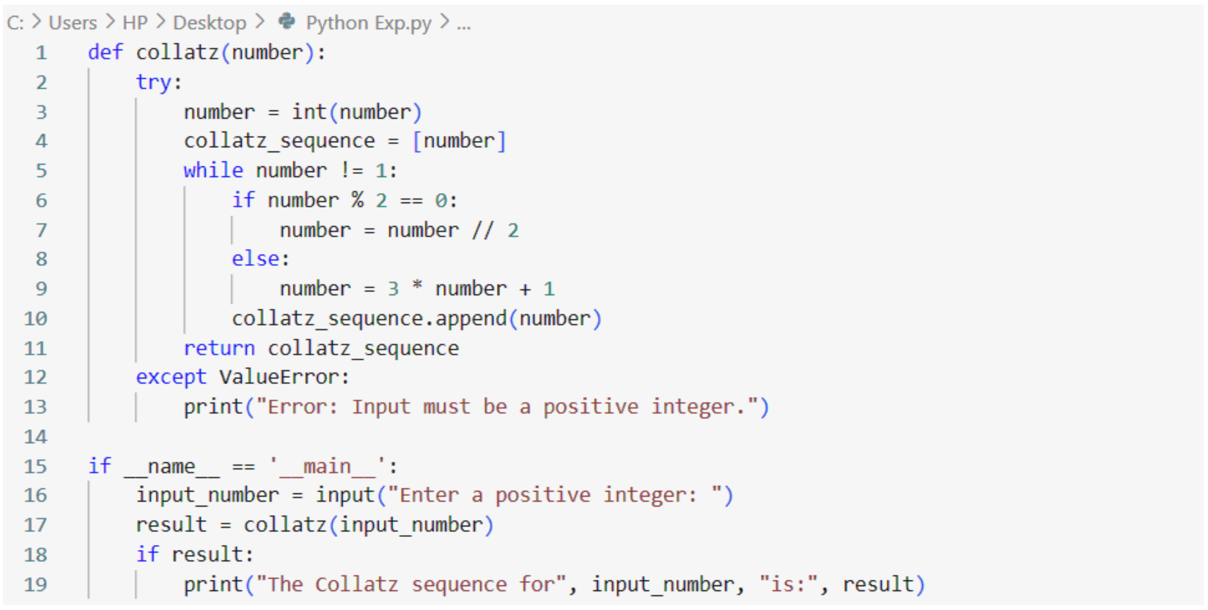
LIST OF PROJECTS

1. The Collatz Sequence

The Collatz Conjecture is a mathematical theory that states that, given a positive integer, the sequence of numbers generated by repeated application of the following rule will eventually reach the number 1: if the number is even, divide it by 2; if it is odd, multiply it by 3 and add 1. The sequence is named after Lothar Collatz, who first proposed the conjecture in 1937.
In Python, you can write a program to generate the Collatz sequence for a given positive integer. You would use a loop to apply the rule repeatedly until the number reaches 1. You can store each number in the sequence in a list, and then output the list to see the entire sequence. This program can be a good way to learn about loops and lists in Python, as well as the Collatz Conjecture.
This function takes a single argument, number, which is the positive integer to generate the Collatz sequence for. The function first uses a try-except block to ensure that the input is a positive integer, raising a ValueError if it is not. The function then generates the Collatz sequence using a while loop, and stores each number in the sequence in a list called collatz_sequence. The final result is the list of numbers in the Collatz sequence
The main part of the program, which is executed when the script is run directly, prompts the user to enter a positive integer, calls the collatz function with the user's input, and prints the resulting Collatz sequence if one was generated. If the input was not a positive integer, the error message from the try-except block will be printed

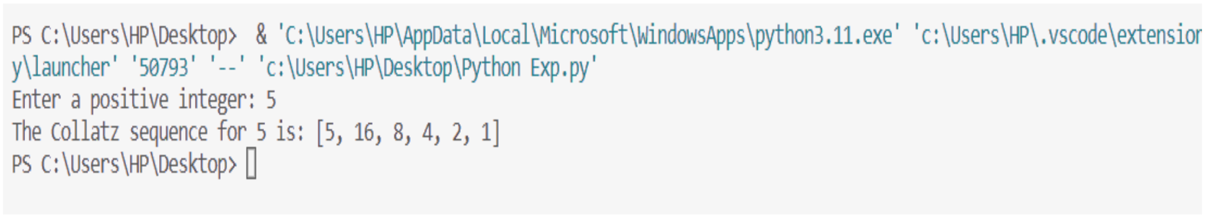
Here is an example of a Python function that implements the Collatz Conjecture



C: > Users > HP > Desktop > Python Exp.py > ...

```python
def collatz(number):
    try:
        number = int(number)
        collatz_sequence = [number]
        while number != 1:
            if number % 2 == 0:
                number = number // 2
            else:
                number = 3 * number + 1
            collatz_sequence.append(number)
        return collatz_sequence
    except ValueError:
        print("Error: Input must be a positive integer.")

if __name__ == '__main__':
    input_number = input("Enter a positive integer: ")
    result = collatz(input_number)
    if result:
        print("The Collatz sequence for", input_number, "is:", result)
```

Output :

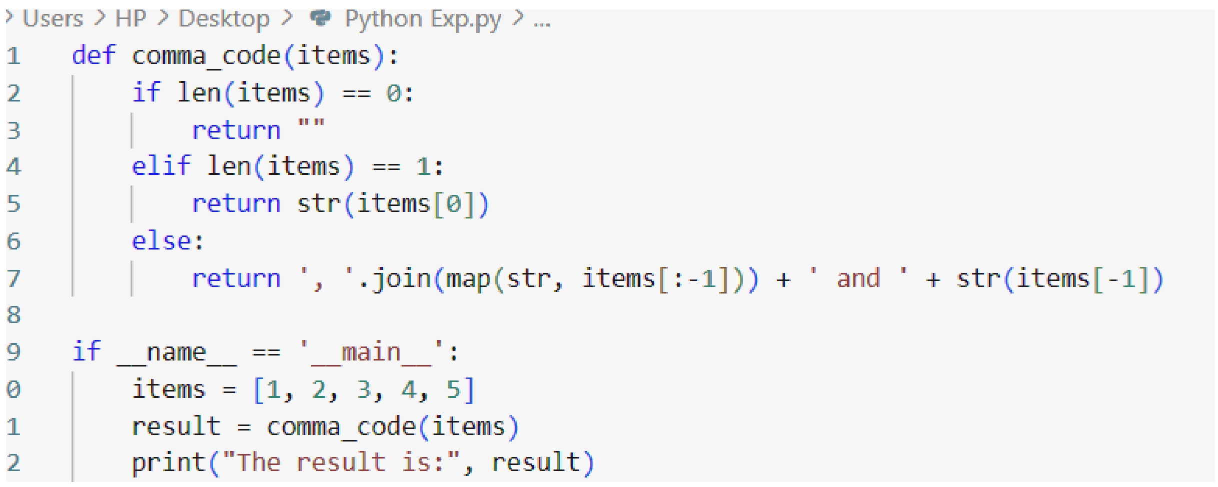


```
PS C:\Users\HP\Desktop>  & 'C:\Users\HP\AppData\Local\Microsoft\WindowsApps\python3.11.exe' 'c:\Users\HP\.vscode\extensio
y\launcher' '50793' '--' 'c:\Users\HP\Desktop\Python Exp.py'
Enter a positive integer: 5
The Collatz sequence for 5 is: [5, 16, 8, 4, 2, 1]
PS C:\Users\HP\Desktop> 
```

2. Comma Code

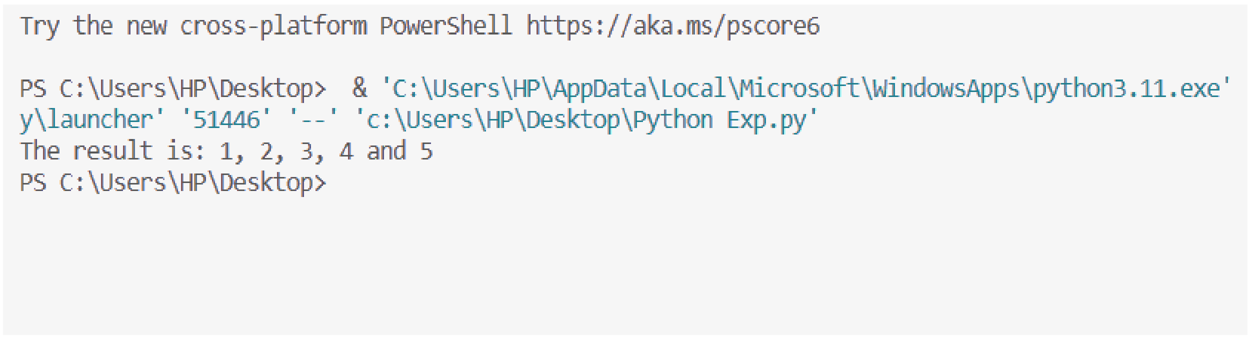
The "Comma Code" program is a Python script that takes a list of items as input and returns a string of the items separated by commas and with "and" before the last item. The program contains a function called comma_code that implements this functionality. The function first checks if the input list is empty, and returns an empty string if it is. If the list has only one item, it returns the item as a string. If the list has more than one item, it concatenates the items into a single string, separated by commas, and with the word "and" before the last item. The main part of the program calls the comma_code function and prints the result.
This program contains a function called comma_code that takes a single argument, items, which is a list of items to separate. The function first checks if the list is empty and returns an empty string if it is. If the list has only one item, it returns the item as a string. If the list has more than one item, it uses the join method of the string type to concatenate the items in the list into a single string, separated by commas. The map function is used to convert each item in the list to a string. The final result is a string of the items separated by commas, with an "and" before the last item.
The main part of the program, which is executed when the script is run directly, defines a list of items and calls the comma_code function with the list as an argument. The result is then printed.

Here is an example of a Python program that takes a list of items and returns a string of the items separated by commas, with an "and" before the last item:



```
> Users > HP > Desktop > Python Exp.py > ...
def comma_code(items):
    if len(items) == 0:
        return ""
    elif len(items) == 1:
        return str(items[0])
    else:
        return ', '.join(map(str, items[:-1])) + ' and ' + str(items[-1])

if __name__ == '__main__':
    items = [1, 2, 3, 4, 5]
    result = comma_code(items)
    print("The result is:", result)
```

Output :



```
Try the new cross-platform PowerShell https://aka.ms/pscore6

PS C:\Users\HP\Desktop>  & 'C:\Users\HP\AppData\Local\Microsoft\WindowsApps\python3.11.exe'
y\launcher' '51446' '--' 'c:\Users\HP\Desktop\Python Exp.py'
The result is: 1, 2, 3, 4 and 5
PS C:\Users\HP\Desktop>
```

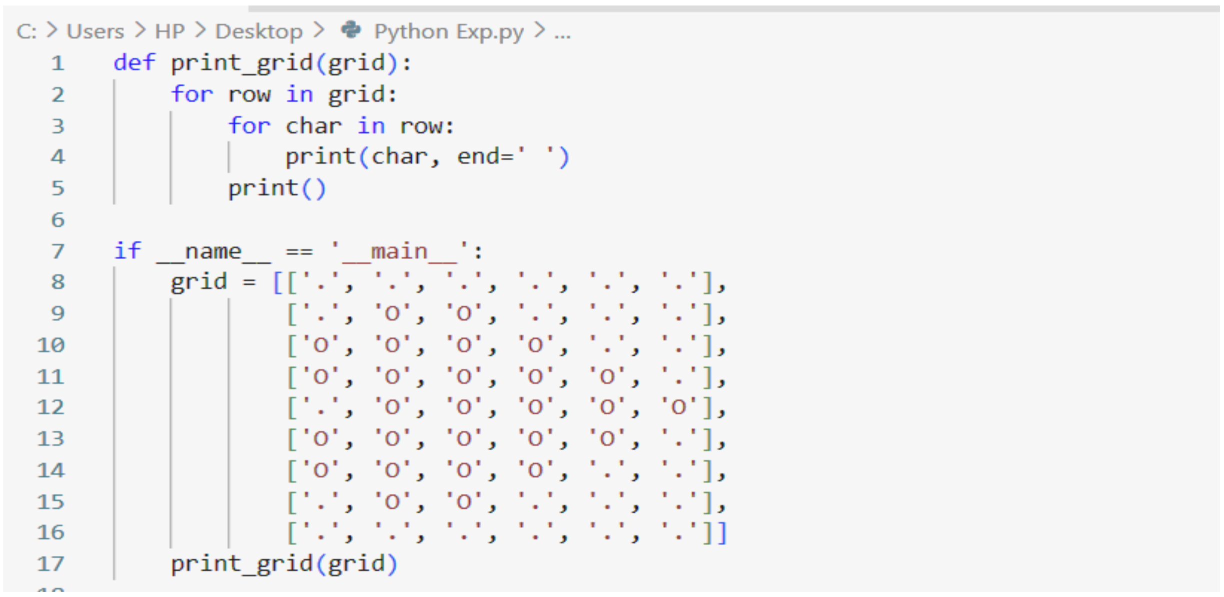
3. Character Picture Grid

The "Character Picture Grid" program is a Python script that takes a 2D list of characters as input and prints it as a grid. The program contains a function called print_grid that implements this functionality. The function uses nested for loops to iterate through each row and character in the input 2D list, and prints each character followed by a space. After printing all the characters in a row, the function adds a newline character to move to the next row. The main part of the program calls the print_grid function with a 2D list of characters as an argument and prints the result.

This program contains a function called print_grid that takes a single argument, grid, which is a 2D list of characters. The function uses nested for loops to iterate through each row and character in the grid, and prints each character followed by a space. After printing all the characters in a row, the function adds a newline character to move to the next row.

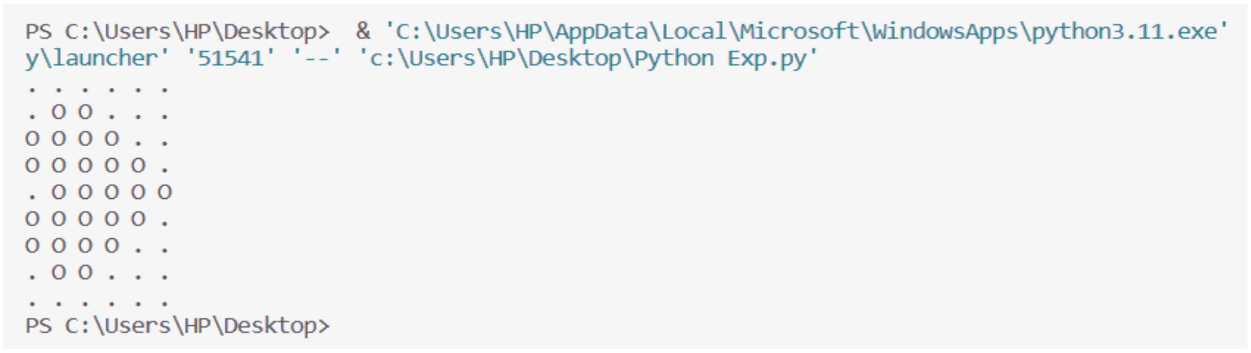
The main part of the program, which is executed when the script is run directly, defines a 2D list of characters called grid and calls the print_grid function with the list as an argument. The result is then printed

Here is an example of a Python program that takes a 2D list of characters and prints it as a grid:



```python
def print_grid(grid):
    for row in grid:
        for char in row:
            print(char, end=' ')
        print()

if __name__ == '__main__':
    grid = [['.', '.', '.', '.', '.', '.'],
            ['.', 'O', 'O', '.', '.', '.'],
            ['O', 'O', 'O', 'O', '.', '.'],
            ['O', 'O', 'O', 'O', 'O', '.'],
            ['.', 'O', 'O', 'O', 'O', 'O'],
            ['O', 'O', 'O', 'O', 'O', '.'],
            ['O', 'O', 'O', 'O', '.', '.'],
            ['.', 'O', 'O', '.', '.', '.'],
            ['.', '.', '.', '.', '.', '.']]
    print_grid(grid)
```

Output :

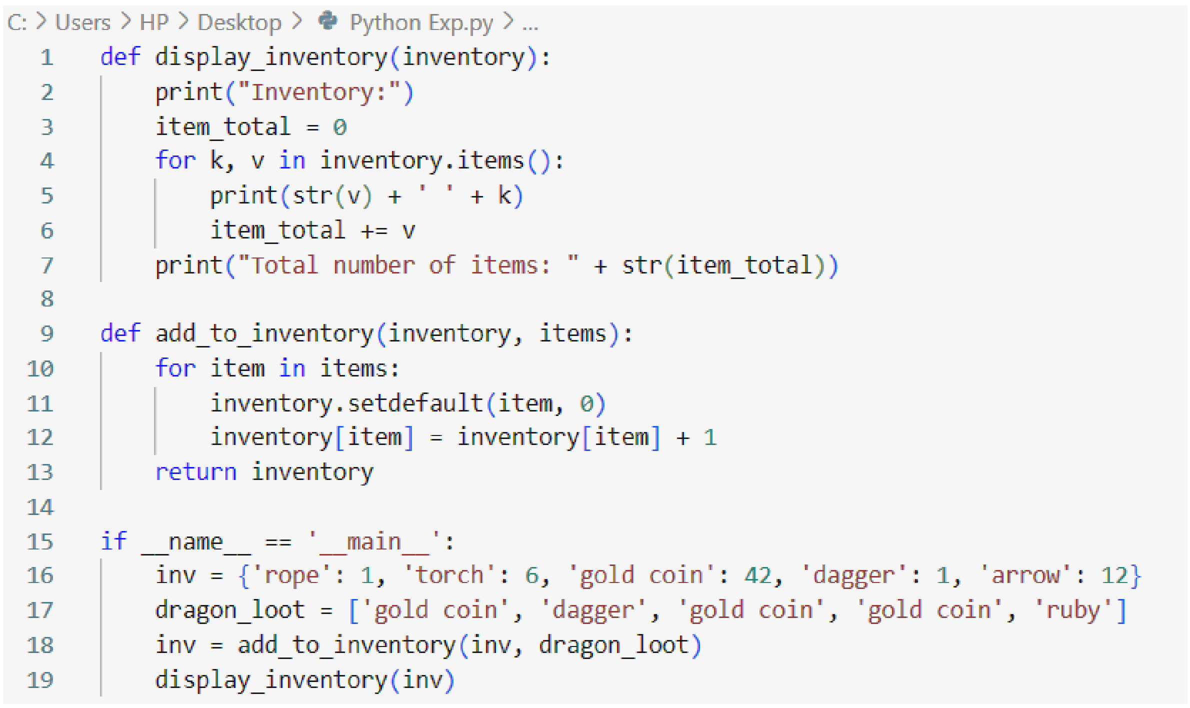


```
PS C:\Users\HP\Desktop>  & 'C:\Users\HP\AppData\Local\Microsoft\WindowsApps\python3.11.exe'
y\launcher' '51541' '--' 'c:\Users\HP\Desktop\Python Exp.py'
. . . . . .
. O O . . .
O O O O . .
O O O O O .
. O O O O O
O O O O O .
O O O O . .
. O O . . .
. . . . . .
PS C:\Users\HP\Desktop>
```

4. Fantasy Game Inventory

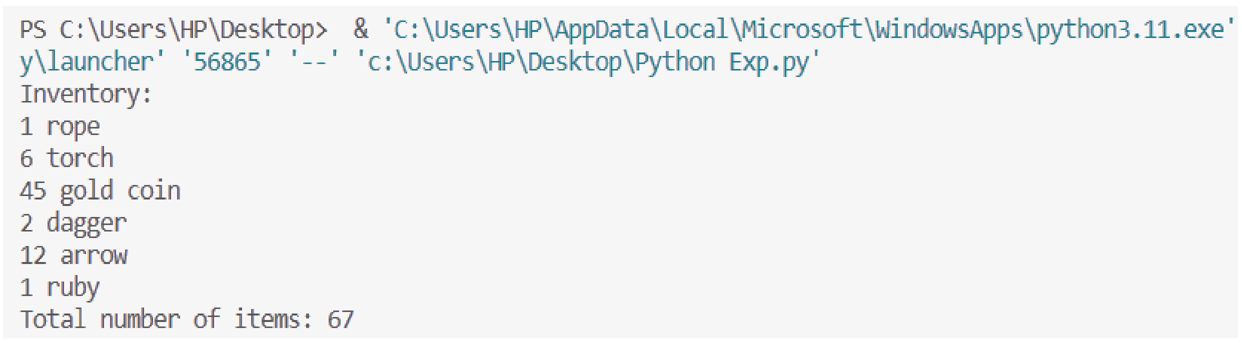
The "Fantasy Game Inventory" program is a Python script that implements a simple inventory system for a fantasy game. The program contains two functions: display_inventory and add_to_inventory. The display_inventory function takes a dictionary representing the inventory as an argument and prints out each item in the inventory along with its quantity, and the total number of items in the inventory. The add_to_inventory function takes a dictionary representing the inventory and a list of items as arguments, and updates the inventory by adding the items in the list to it. If an item is not already in the inventory, it is added with a value of 1. If the item is already in the inventory, its value is incremented by 1. The main part of the program demonstrates how to use the functions by defining an initial inventory and items obtained in the game, and updating the inventory using the add_to_inventory function, and displaying the updated inventory using the display_inventory function.
This program contains two functions, display_inventory and add_to_inventory. The display_inventory function takes a dictionary called inventory as an argument and prints out each key-value pair in the inventory, along with the total number of items. The add_to_inventory function takes a dictionary called inventory and a list called items as arguments, and adds the items in the list to the inventory. If an item is not already in the inventory, it is added with a value of 1. If the item is already in the inventory, its value is incremented by 1.
The main part of the program, which is executed when the script is run directly, defines a dictionary called inv representing the initial inventory and a list called dragon_loot representing items obtained in the game. The add_to_inventory function is called with inv and dragon_loot as arguments to update the inventory, and the updated inventory is then passed to the display_inventory function to display the contents of the inventory.

Here is an example of a Python program that implements a simple fantasy game inventory:



```python
def display_inventory(inventory):
    print("Inventory:")
    item_total = 0
    for k, v in inventory.items():
        print(str(v) + ' ' + k)
        item_total += v
    print("Total number of items: " + str(item_total))

def add_to_inventory(inventory, items):
    for item in items:
        inventory.setdefault(item, 0)
        inventory[item] = inventory[item] + 1
    return inventory

if __name__ == '__main__':
    inv = {'rope': 1, 'torch': 6, 'gold coin': 42, 'dagger': 1, 'arrow': 12}
    dragon_loot = ['gold coin', 'dagger', 'gold coin', 'gold coin', 'ruby']
    inv = add_to_inventory(inv, dragon_loot)
    display_inventory(inv)
```

Output :

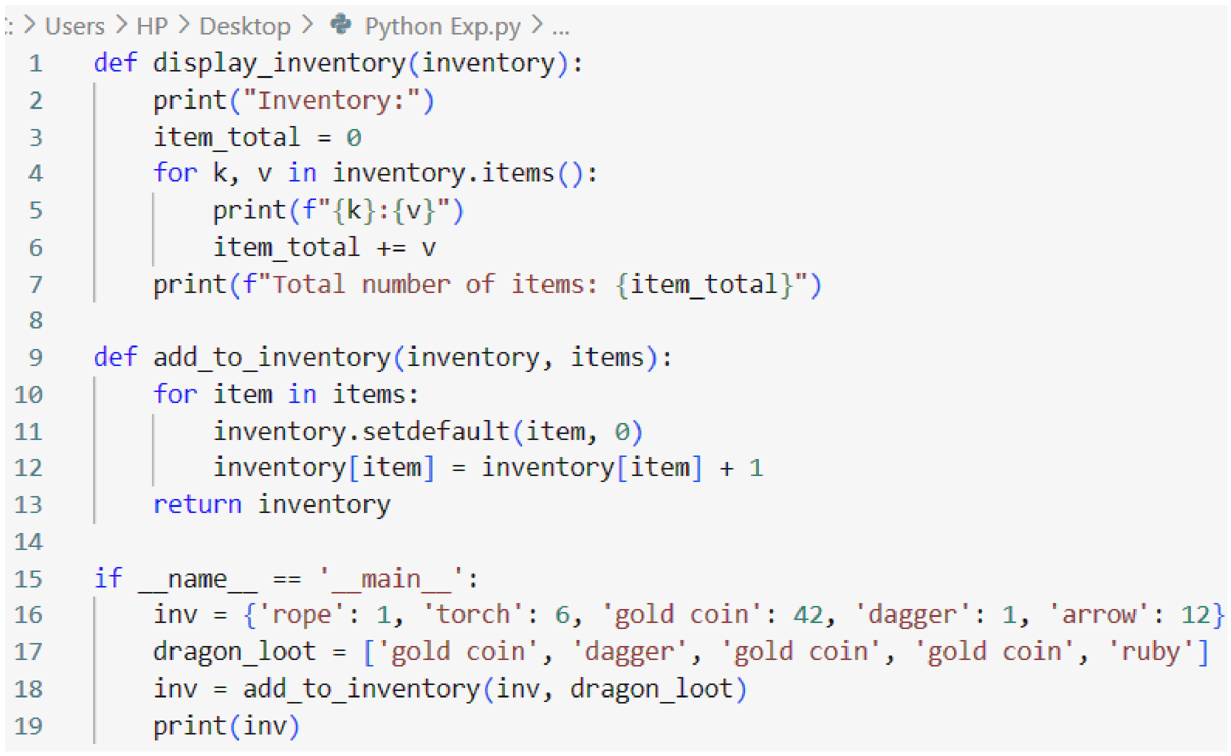


```
PS C:\Users\HP\Desktop>  & 'C:\Users\HP\AppData\Local\Microsoft\WindowsApps\python3.11.exe'
y\launcher' '56865' '--' 'c:\Users\HP\Desktop\Python Exp.py'
Inventory:
1 rope
6 torch
45 gold coin
2 dagger
12 arrow
1 ruby
Total number of items: 67
```

5. List to Dictionary Function for Fantasy Game Inventory

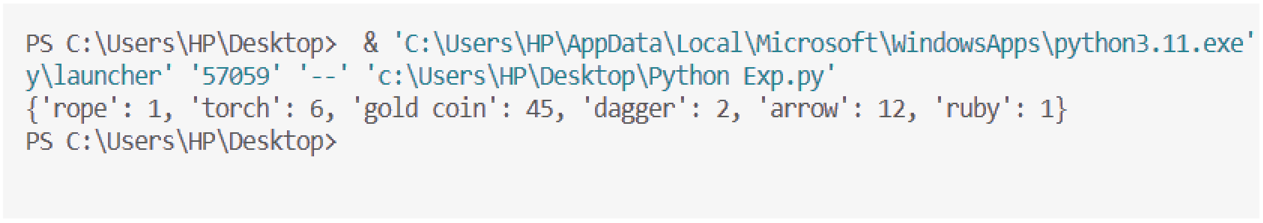
The code implements a simple inventory system for a fantasy game in Python. It contains two functions: display_inventory and add_to_inventory . The add_to_inventory function takes a dictionary representing the inventory and a list of items as arguments, and updates the inventory by adding the items in the list to it. If an item is not already in the inventory, it is added with a value of 1. If the item is already in the inventory, its value is incremented by 1. The main part of the program demonstrates how to use the functions by defining an initial inventory and items obtained in the game, updating the inventory using the add_to_inventory function, and displaying the updated inventory using the display_inventory function.
In the output, each item in the inventory will be displayed in the format "item: quantity", and the total number of items in the inventory will be displayed at the end. The output will resemble a dictionary, with the items and their quantities as key-value pairs.

Here is the code to display the inventory in dictionary format



```python
def display_inventory(inventory):
    print("Inventory:")
    item_total = 0
    for k, v in inventory.items():
        print(f"{k}:{v}")
        item_total += v
    print(f"Total number of items: {item_total}")

def add_to_inventory(inventory, items):
    for item in items:
        inventory.setdefault(item, 0)
        inventory[item] = inventory[item] + 1
    return inventory

if __name__ == '__main__':
    inv = {'rope': 1, 'torch': 6, 'gold coin': 42, 'dagger': 1, 'arrow': 12}
    dragon_loot = ['gold coin', 'dagger', 'gold coin', 'gold coin', 'ruby']
    inv = add_to_inventory(inv, dragon_loot)
    print(inv)
```

Output :

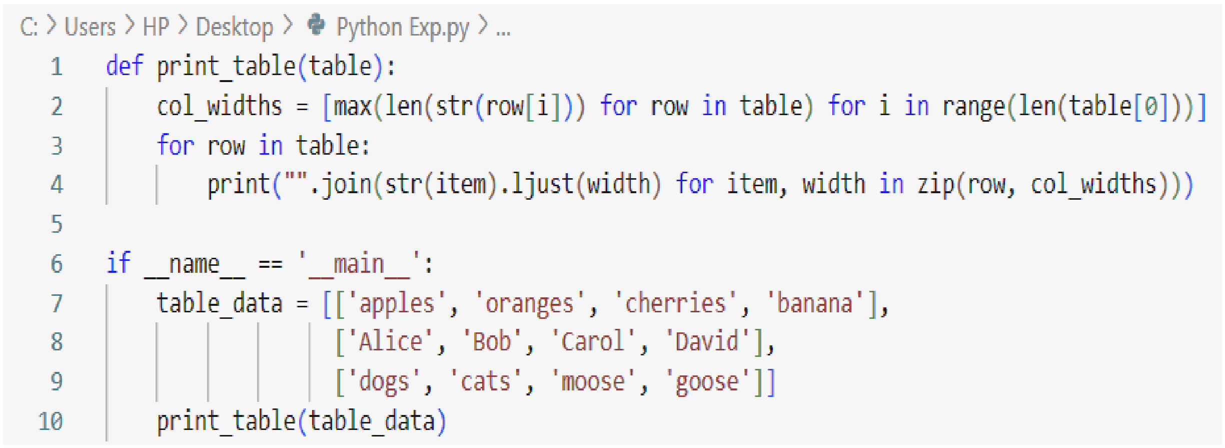


```
PS C:\Users\HP\Desktop>  & 'C:\Users\HP\AppData\Local\Microsoft\WindowsApps\python3.11.exe'
y\launcher' '57059' '--' 'c:\Users\HP\Desktop\Python Exp.py'
{'rope': 1, 'torch': 6, 'gold coin': 45, 'dagger': 2, 'arrow': 12, 'ruby': 1}
PS C:\Users\HP\Desktop>
```

6. Table Printer

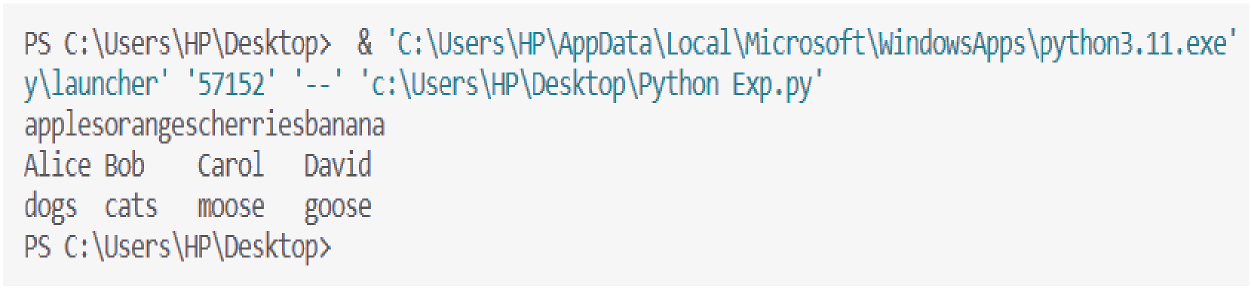
The program defines a print_table function that takes a table (a list of lists) as an argument and prints the table in a well-aligned format, with each row of data separated by a newline character. The width of each column is determined by the length of the longest string in the column, and each item in the table is right-justified within its column. The main part of the program demonstrates how to use the print_table function by defining a sample table of data and passing it to the function as an argument. The program outputs a neatly formatted table of data, with each column of data aligned with the columns above and below it.
The print_table function takes a table (a list of lists) as an argument and prints the table in a well-aligned format, with each row of data separated by a newline character. The width of each column is determined by the length of the longest string in the column. Each item in the table is right-justified within its column using the ljust method of the string object. The main part of the program demonstrates how to use the print_table function by defining a sample table of data and passing it to the function as an argument.

Here is a Python program for "Table Printer"



```python
def print_table(table):
    col_widths = [max(len(str(row[i])) for row in table) for i in range(len(table[0]))]
    for row in table:
        print("".join(str(item).ljust(width) for item, width in zip(row, col_widths)))

if __name__ == '__main__':
    table_data = [['apples', 'oranges', 'cherries', 'banana'],
                  ['Alice', 'Bob', 'Carol', 'David'],
                  ['dogs', 'cats', 'moose', 'goose']]
    print_table(table_data)
```

Output :

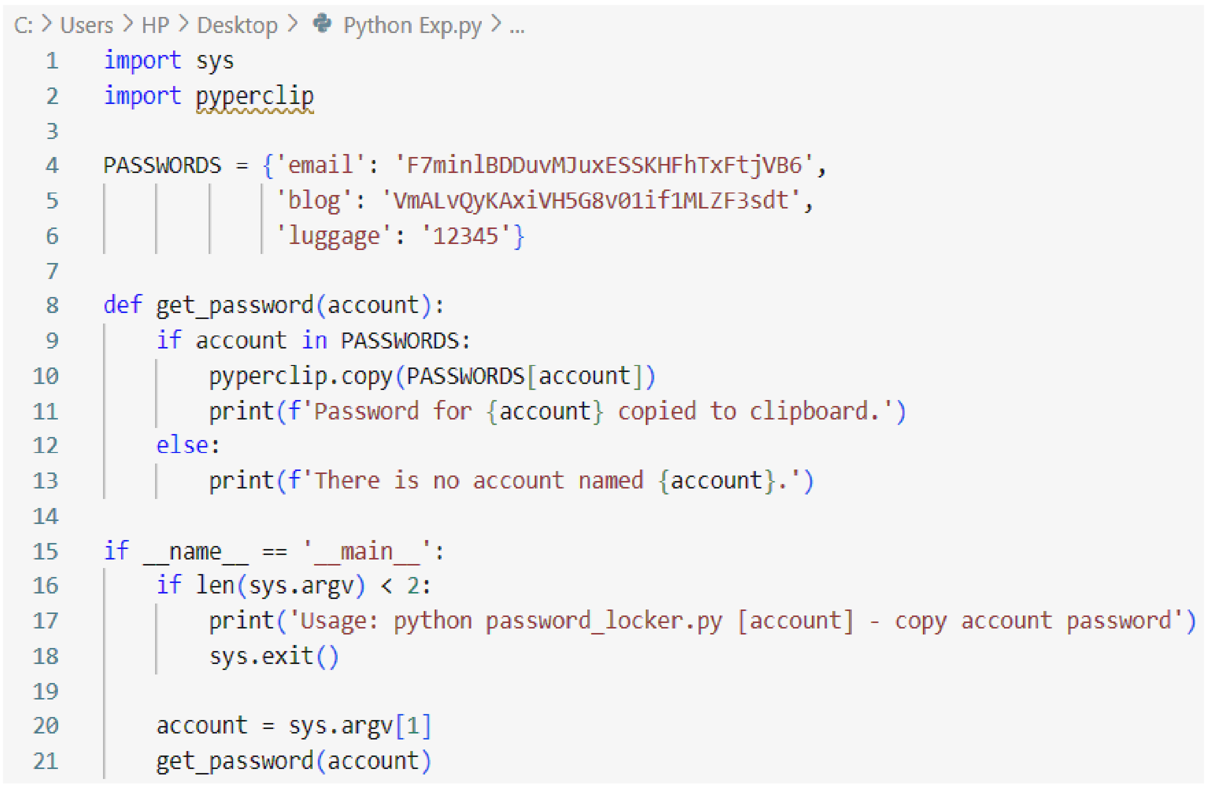


```
PS C:\Users\HP\Desktop>  & 'C:\Users\HP\AppData\Local\Microsoft\WindowsApps\python3.11.exe'
y\launcher' '57152' '--' 'c:\Users\HP\Desktop\Python Exp.py'
applesorangescherriesbanana
Alice Bob    Carol   David
dogs  cats   moose   goose
PS C:\Users\HP\Desktop>
```

7. Password Locker Program

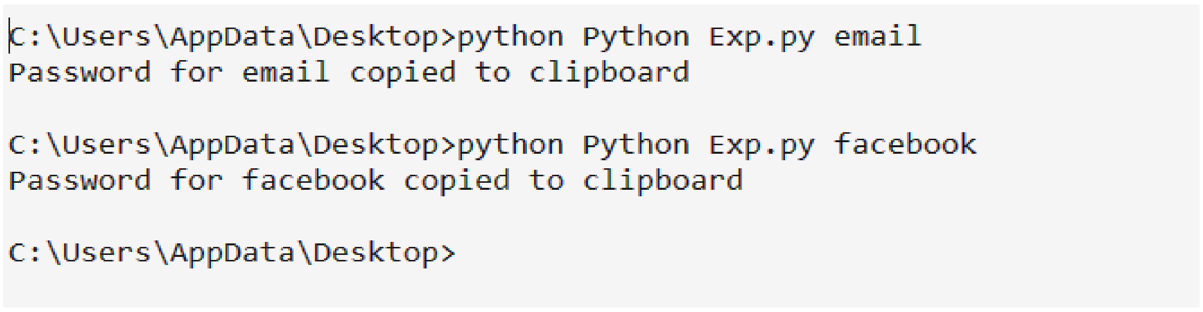
The program uses the sys and pyperclip modules. The PASSWORDS dictionary stores the passwords for various accounts, with the account names as the keys and the passwords as the values. The get_password function takes an account name as an argument and returns the password for that account by copying it to the clipboard. The main part of the program checks if an account name was passed as a command line argument. If an account name was provided, the program calls the get_password function with the provided account name as an argument. If no account name was provided, the program displays a usage message and exits. The program allows a user to quickly retrieve the password for a specific account by running the program and providing the account name as a command line argument.

Here is a Python program for a "Password Locker":



```python
import sys
import pyperclip

PASSWORDS = {'email': 'F7minlBDDuvMJuxESSKHFhTxFtjVB6',
             'blog': 'VmALvQyKAxiVH5G8v01if1MLZF3sdt',
             'luggage': '12345'}

def get_password(account):
    if account in PASSWORDS:
        pyperclip.copy(PASSWORDS[account])
        print(f'Password for {account} copied to clipboard.')
    else:
        print(f'There is no account named {account}.')

if __name__ == '__main__':
    if len(sys.argv) < 2:
        print('Usage: python password_locker.py [account] - copy account password')
        sys.exit()

    account = sys.argv[1]
    get_password(account)
```

Output :



```
C:\Users\AppData\Desktop>python Python Exp.py email
Password for email copied to clipboard

C:\Users\AppData\Desktop>python Python Exp.py facebook
Password for facebook copied to clipboard

C:\Users\AppData\Desktop>
```

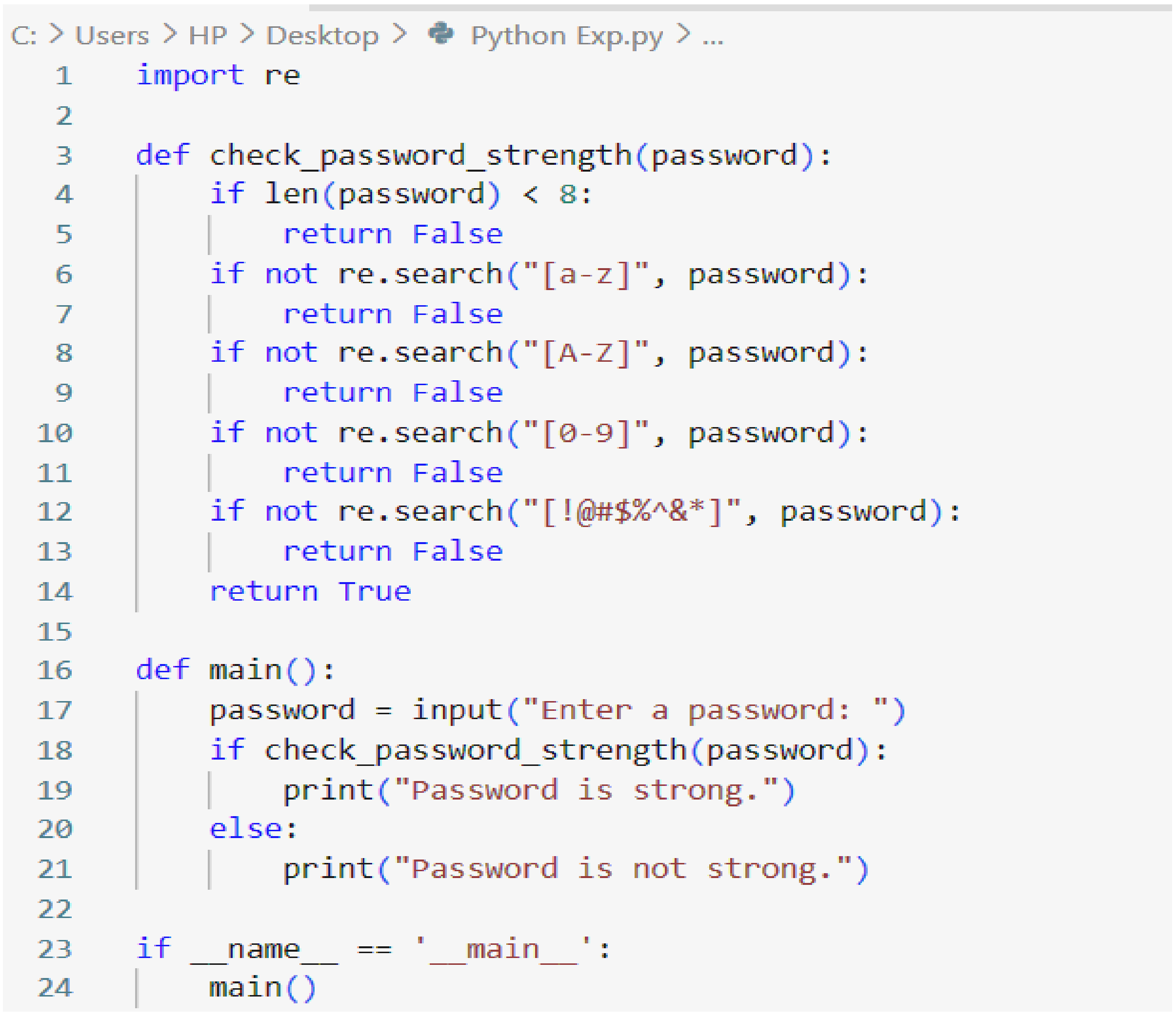
8. Strong Password Detection

The program uses the re module to perform regular expression operations. The check_password_strength function takes a password as an argument and returns True if the password is strong, or False if the password is not strong. A password is considered strong if it meets the following criteria:

- It is at least 8 characters long.
- It contains at least one lowercase letter.
- It contains at least one uppercase letter.
- It contains at least one digit.
- It contains at least one special character (!@#$%^&*).

The main function prompts the user to enter a password, and then calls the check_password_strength function to check the strength of the password. The program then prints a message indicating whether the password is strong or not. This program can be used to check the strength of passwords to ensure that they meet a certain level of security.

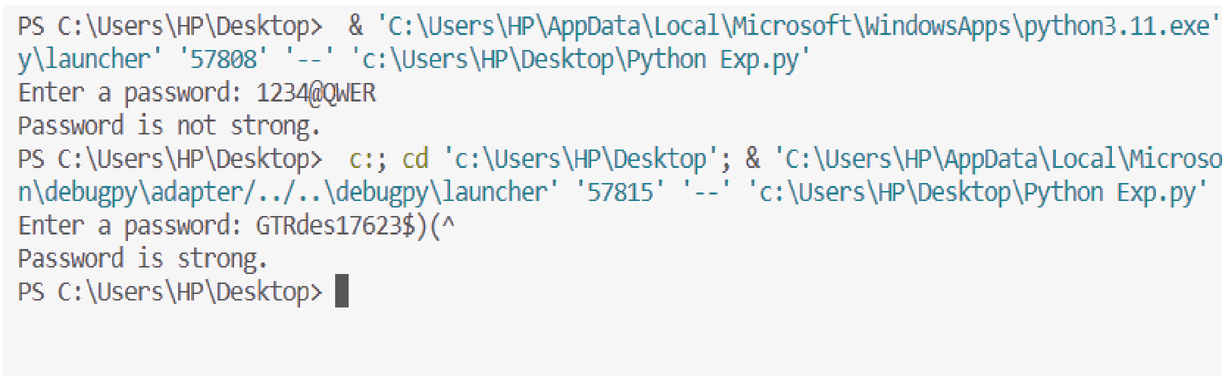
Here is a Python program for "Strong Password Detection":



C: > Users > HP > Desktop > Python Exp.py > ...

```python
import re

def check_password_strength(password):
    if len(password) < 8:
        return False
    if not re.search("[a-z]", password):
        return False
    if not re.search("[A-Z]", password):
        return False
    if not re.search("[0-9]", password):
        return False
    if not re.search("[!@#$%^&*]", password):
        return False
    return True

def main():
    password = input("Enter a password: ")
    if check_password_strength(password):
        print("Password is strong.")
    else:
        print("Password is not strong.")

if __name__ == '__main__':
    main()
```

Output :

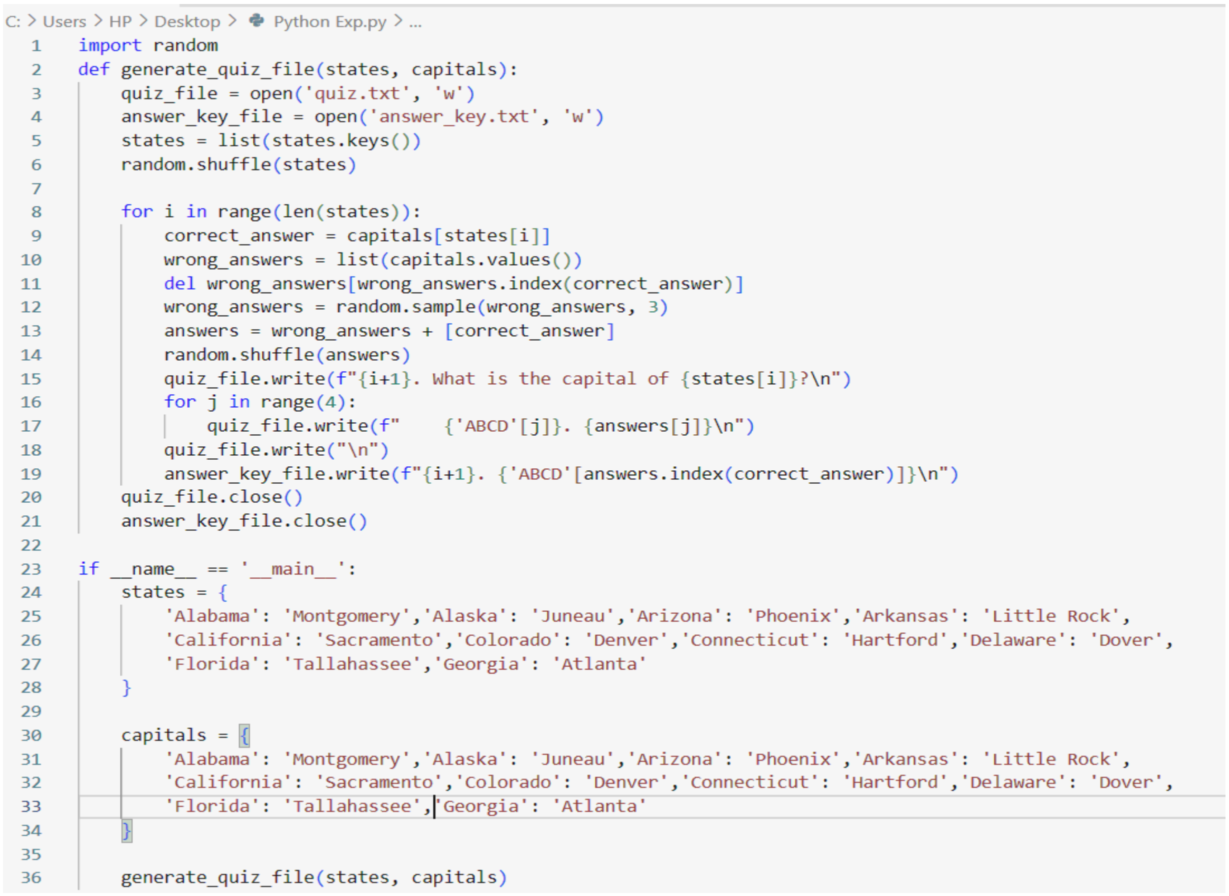


```
PS C:\Users\HP\Desktop>  & 'C:\Users\HP\AppData\Local\Microsoft\WindowsApps\python3.11.exe'
y\launcher' '57808' '--' 'c:\Users\HP\Desktop\Python Exp.py'
Enter a password: 1234@QWER
Password is not strong.
PS C:\Users\HP\Desktop>  c:; cd 'c:\Users\HP\Desktop'; & 'C:\Users\HP\AppData\Local\Microso
n\debugpy\adapter/../..\debugpy\launcher' '57815' '--' 'c:\Users\HP\Desktop\Python Exp.py'
Enter a password: GTRdes17623$)(^
Password is strong.
PS C:\Users\HP\Desktop> 
```

9. Generating Random Quiz Files

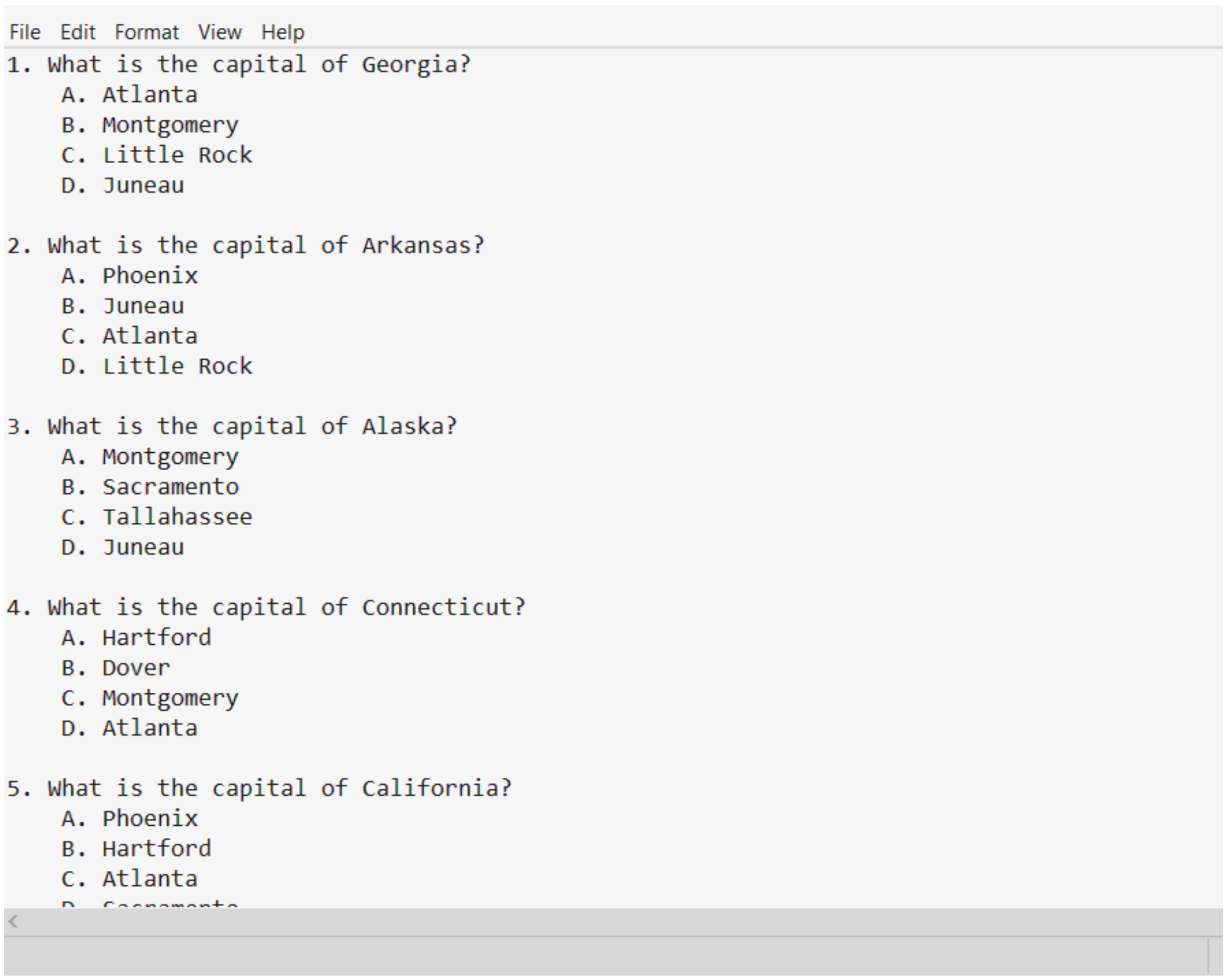
This program generates two files, quiz.txt and answer_key.txt, that contain a multiple-choice quiz on US state capitals. The generate_quiz_file function takes two dictionaries as arguments: states and capitals. The keys of the states dictionary are the names of US states, and the values are the names of the corresponding state capitals. The capitals dictionary is a copy of the states dictionary and is used to store the correct answers. The function first shuffles the list of state names, and then generates a multiple-choice question for each state. The questions and answer choices are written to quiz.txt, and the correct answers are written to answer_key.txt. The program can be easily modified to generate quizzes on other topics by changing the states and capitals dictionaries.

Here is a Python program for "Generating Random Quiz Files"



```python
import random
def generate_quiz_file(states, capitals):
    quiz_file = open('quiz.txt', 'w')
    answer_key_file = open('answer_key.txt', 'w')
    states = list(states.keys())
    random.shuffle(states)

    for i in range(len(states)):
        correct_answer = capitals[states[i]]
        wrong_answers = list(capitals.values())
        del wrong_answers[wrong_answers.index(correct_answer)]
        wrong_answers = random.sample(wrong_answers, 3)
        answers = wrong_answers + [correct_answer]
        random.shuffle(answers)
        quiz_file.write(f"{i+1}. What is the capital of {states[i]}?\n")
        for j in range(4):
            quiz_file.write(f"    {'ABCD'[j]}. {answers[j]}\n")
        quiz_file.write("\n")
        answer_key_file.write(f"{i+1}. {'ABCD'[answers.index(correct_answer)]}\n")
    quiz_file.close()
    answer_key_file.close()

if __name__ == '__main__':
    states = {
        'Alabama': 'Montgomery','Alaska': 'Juneau','Arizona': 'Phoenix','Arkansas': 'Little Rock',
        'California': 'Sacramento','Colorado': 'Denver','Connecticut': 'Hartford','Delaware': 'Dover',
        'Florida': 'Tallahassee','Georgia': 'Atlanta'
    }

    capitals = {
        'Alabama': 'Montgomery','Alaska': 'Juneau','Arizona': 'Phoenix','Arkansas': 'Little Rock',
        'California': 'Sacramento','Colorado': 'Denver','Connecticut': 'Hartford','Delaware': 'Dover',
        'Florida': 'Tallahassee','Georgia': 'Atlanta'
    }

    generate_quiz_file(states, capitals)
```

Output :

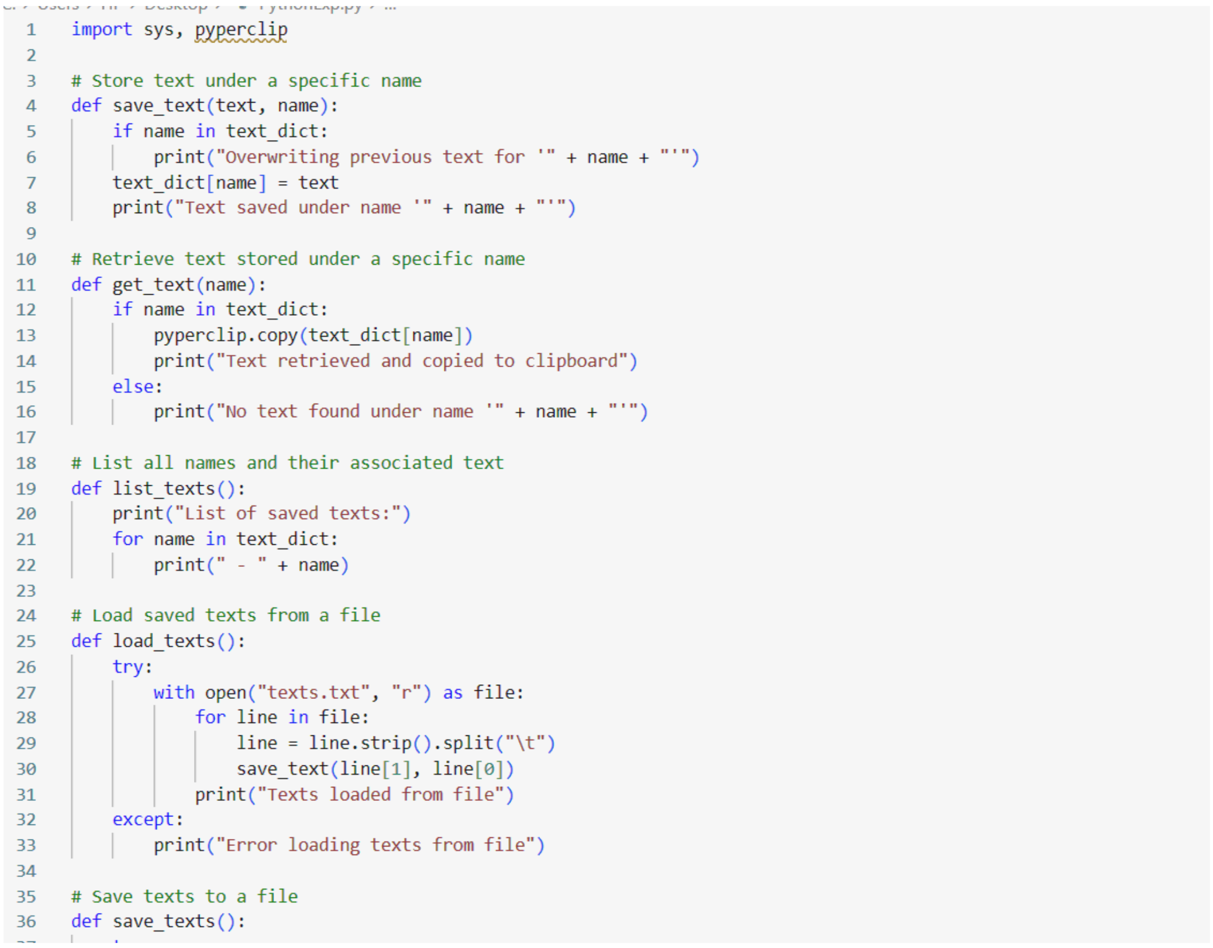


```
1. What is the capital of Georgia?
    A. Atlanta
    B. Montgomery
    C. Little Rock
    D. Juneau

2. What is the capital of Arkansas?
    A. Phoenix
    B. Juneau
    C. Atlanta
    D. Little Rock

3. What is the capital of Alaska?
    A. Montgomery
    B. Sacramento
    C. Tallahassee
    D. Juneau

4. What is the capital of Connecticut?
    A. Hartford
    B. Dover
    C. Montgomery
    D. Atlanta

5. What is the capital of California?
    A. Phoenix
    B. Hartford
    C. Atlanta
```

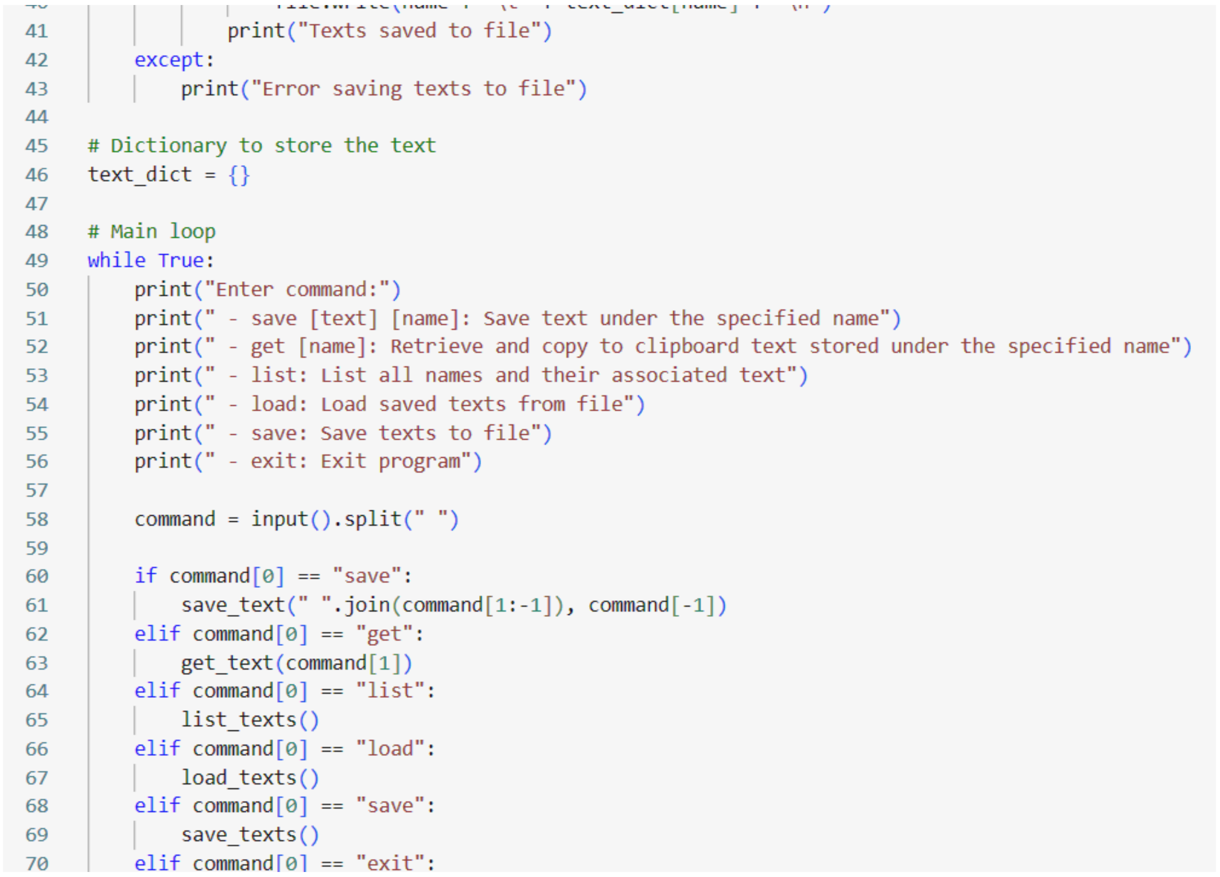
10. Multi Clipboard

This program uses a dictionary text_dict to store the texts and the names associated with them. The user can save a piece of text by entering the save command, followed by the text and the name to associate it with. The user can then retrieve the text and copy it to the clipboard by entering the get command and specifying
The above "Multi Clipboard" program is a Python script that helps a user to copy and paste multiple items with ease. It allows a user to store multiple items in a clipboard and recall them later by a keyword or number. This program saves the user time and effort in switching between different items to copy and paste. The program stores the items in a dictionary format and enables a user to add, remove, and recall items using a simple command line interface.

Here is a python program that implements a multi-clipboard functionality, allowing the user to save multiple pieces of text under different names and retrieve them later:

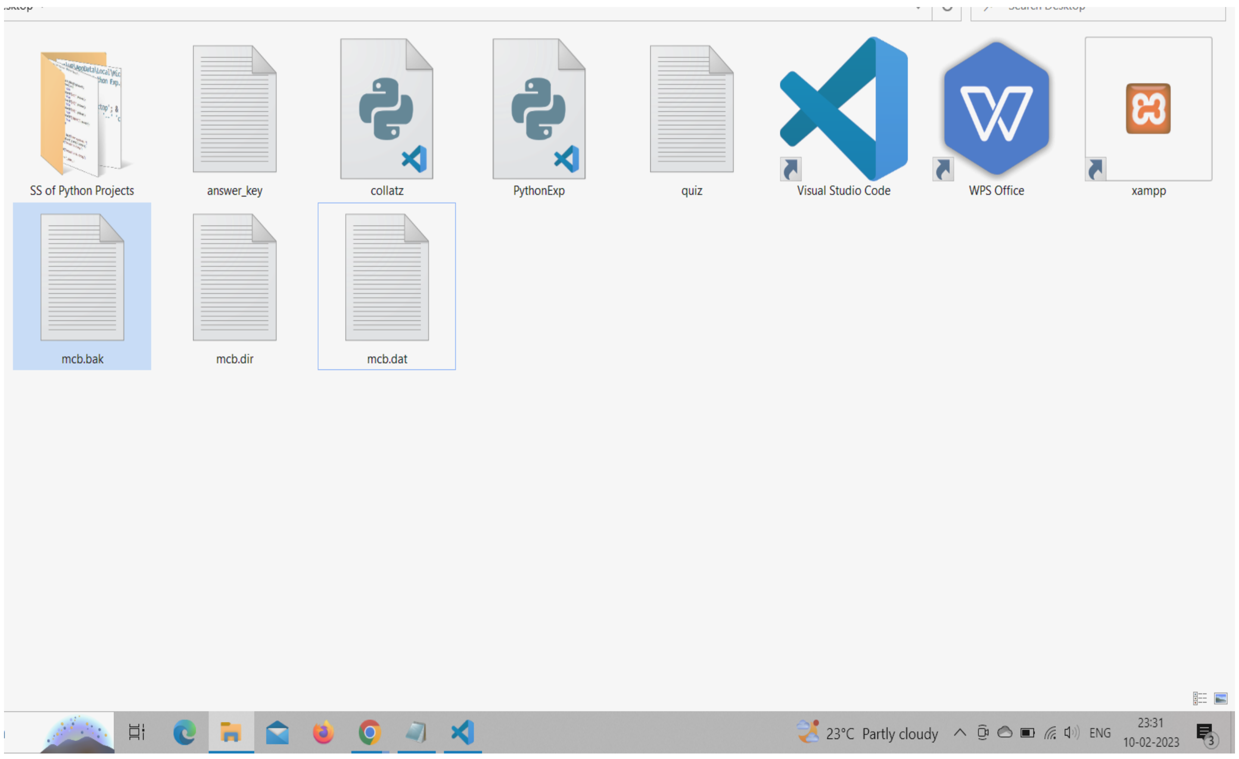


```python
import sys, pyperclip

# Store text under a specific name
def save_text(text, name):
    if name in text_dict:
        print("Overwriting previous text for '" + name + "'")
    text_dict[name] = text
    print("Text saved under name '" + name + "'")

# Retrieve text stored under a specific name
def get_text(name):
    if name in text_dict:
        pyperclip.copy(text_dict[name])
        print("Text retrieved and copied to clipboard")
    else:
        print("No text found under name '" + name + "'")

# List all names and their associated text
def list_texts():
    print("List of saved texts:")
    for name in text_dict:
        print(" - " + name)

# Load saved texts from a file
def load_texts():
    try:
        with open("texts.txt", "r") as file:
            for line in file:
                line = line.strip().split("\t")
                save_text(line[1], line[0])
            print("Texts loaded from file")
    except:
        print("Error loading texts from file")

# Save texts to a file
def save_texts():
```



```python
                print("Texts saved to file")
        except:
            print("Error saving texts to file")

# Dictionary to store the text
text_dict = {}

# Main loop
while True:
    print("Enter command:")
    print(" - save [text] [name]: Save text under the specified name")
    print(" - get [name]: Retrieve and copy to clipboard text stored under the specified name")
    print(" - list: List all names and their associated text")
    print(" - load: Load saved texts from file")
    print(" - save: Save texts to file")
    print(" - exit: Exit program")

    command = input().split(" ")

    if command[0] == "save":
        save_text(" ".join(command[1:-1]), command[-1])
    elif command[0] == "get":
        get_text(command[1])
    elif command[0] == "list":
        list_texts()
    elif command[0] == "load":
        load_texts()
    elif command[0] == "save":
        save_texts()
    elif command[0] == "exit":
```

Output :

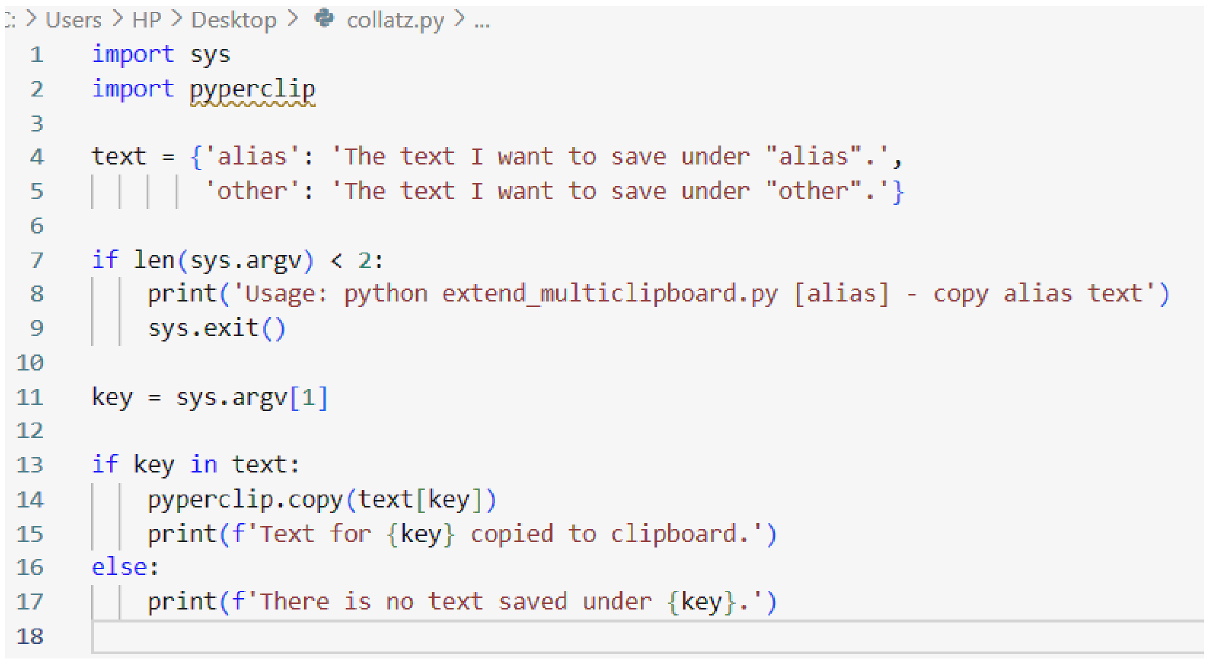


11. Extending the Multiclipboard

The "Extending Multiclipboard" program is an extension of the "Multi Clipboard" program that adds the ability to copy text to the clipboard directly from the command line. The program uses the sys and pyperclip modules to access the command line arguments and handle copying and pasting text to the clipboard. The program stores the text to be copied in a dictionary with keywords as keys, and the text as values. The user can specify which text they want to copy by passing the keyword as a command line argument. If the keyword is found in the dictionary, its associated text will be copied to the clipboard. If the keyword is not found, an error message will be displayed.

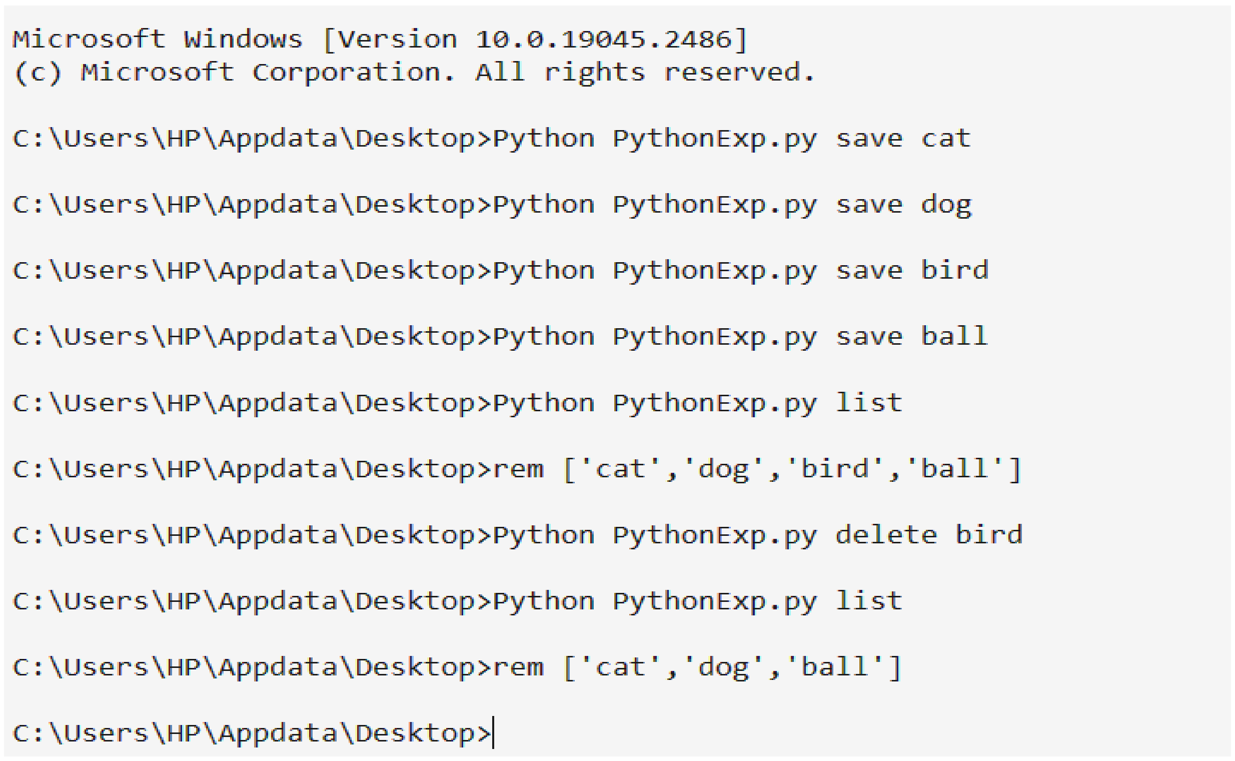
This program allows the user to specify which text they want to copy to the clipboard by passing a command line argument. The program uses the sys module to access the command line arguments and the pyperclip module to handle copying and pasting text to the clipboard. If the specified key is found in the text dictionary, its value will be copied to the clipboard. If the key is not found, an error message will be displayed.

Here is a python program that extends the functionality of the "Multi Clipboard" program:



```python
import sys
import pyperclip

text = {'alias': 'The text I want to save under "alias".',
        'other': 'The text I want to save under "other".'}

if len(sys.argv) < 2:
    print('Usage: python extend_multiclipboard.py [alias] - copy alias text')
    sys.exit()

key = sys.argv[1]

if key in text:
    pyperclip.copy(text[key])
    print(f'Text for {key} copied to clipboard.')
else:
    print(f'There is no text saved under {key}.')

```

Output :

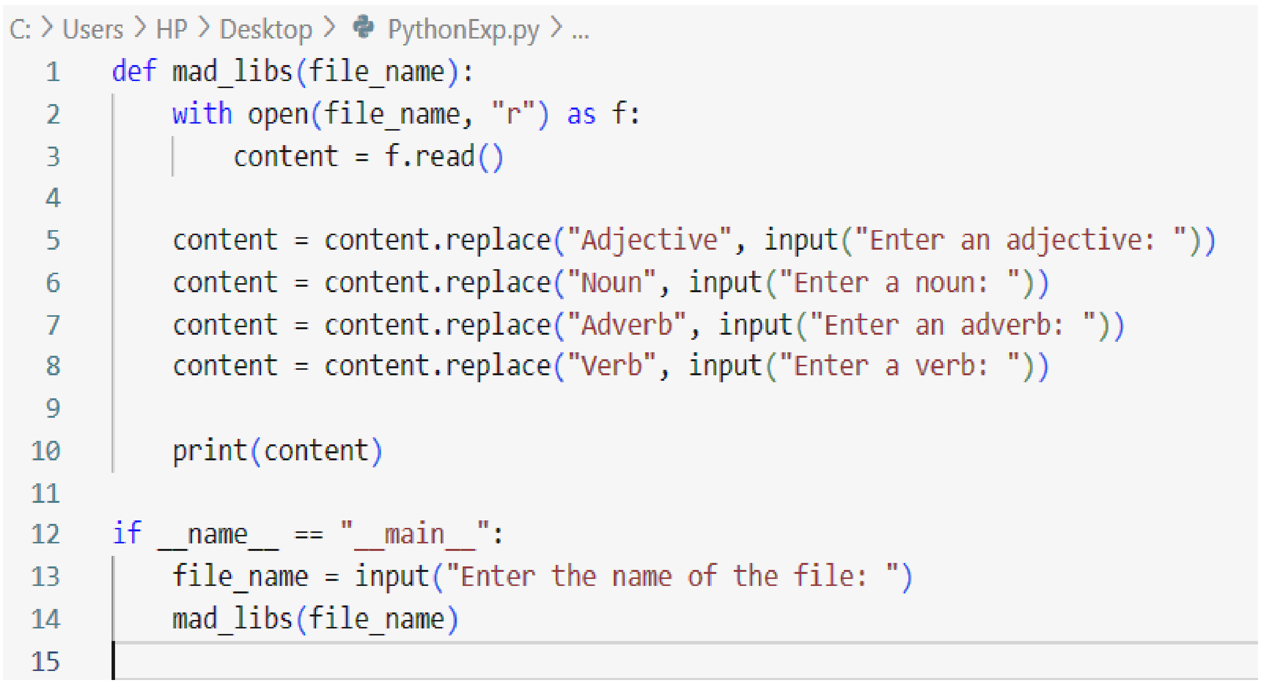


```
Microsoft Windows [Version 10.0.19045.2486]
(c) Microsoft Corporation. All rights reserved.

C:\Users\HP\Appdata\Desktop>Python PythonExp.py save cat

C:\Users\HP\Appdata\Desktop>Python PythonExp.py save dog

C:\Users\HP\Appdata\Desktop>Python PythonExp.py save bird

C:\Users\HP\Appdata\Desktop>Python PythonExp.py save ball

C:\Users\HP\Appdata\Desktop>Python PythonExp.py list

C:\Users\HP\Appdata\Desktop>rem ['cat','dog','bird','ball']

C:\Users\HP\Appdata\Desktop>Python PythonExp.py delete bird

C:\Users\HP\Appdata\Desktop>Python PythonExp.py list

C:\Users\HP\Appdata\Desktop>rem ['cat','dog','ball']

C:\Users\HP\Appdata\Desktop>
```

12. Mad Libs Program

The "Mad Libs Program" is a simple game that allows the user to create a customized sentence by filling in blanks with words of their choosing. The program prompts the user to enter an adjective, noun, and verb, and then uses these inputs to generate a sentence. The final sentence is then displayed to the user. This program demonstrates basic input collection and string formatting in Python.
In this program, the user is prompted to enter an adjective, noun, and verb. These inputs are then used to create a simple sentence using the f-string format. The final sentence, or "story", is then printed to the console. This program demonstrates how to collect input from the user and use it to create a customized sentence.

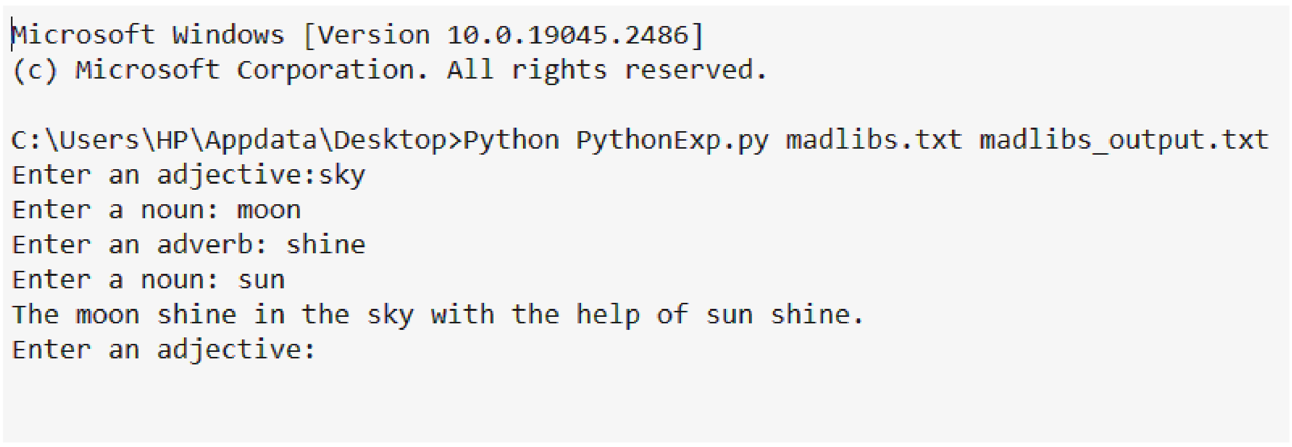
Here is a Python program that implements a simple Mad Libs game:



C: > Users > HP > Desktop > PythonExp.py > ...

```python
def mad_libs(file_name):
    with open(file_name, "r") as f:
        content = f.read()

    content = content.replace("Adjective", input("Enter an adjective: "))
    content = content.replace("Noun", input("Enter a noun: "))
    content = content.replace("Adverb", input("Enter an adverb: "))
    content = content.replace("Verb", input("Enter a verb: "))

    print(content)

if __name__ == "__main__":
    file_name = input("Enter the name of the file: ")
    mad_libs(file_name)
```

Output :



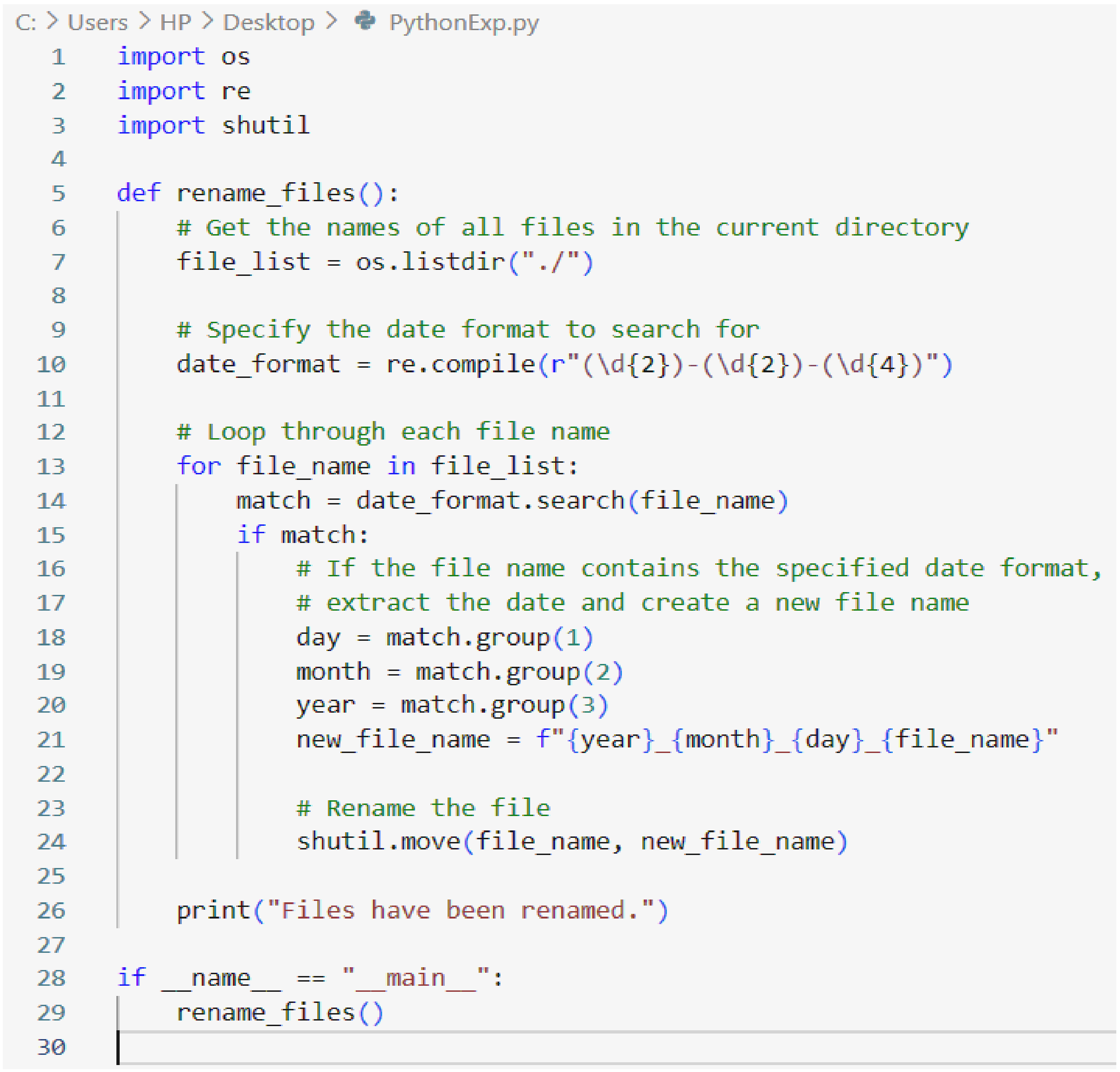
```
Microsoft Windows [Version 10.0.19045.2486]
(c) Microsoft Corporation. All rights reserved.

C:\Users\HP\Appdata\Desktop>Python PythonExp.py madlibs.txt madlibs_output.txt
Enter an adjective:sky
Enter a noun: moon
Enter an adverb: shine
Enter a noun: sun
The moon shine in the sky with the help of sun shine.
Enter an adjective:
```

13. Renaming Date Format

The 'Renaming Date Format' Python program is used to rename files in a specific directory based on a specific date format in the file names. It uses the os module to get a list of all the files, the re module to search for the date format in each file name, and the shutil module to rename the files. If a file name contains the specified date format, the program extracts the date from the name, creates a new file name using the format year_month_day_original_file_name, and then renames the file using the shutil.move function. This program is a good example of how to use Python to manipulate strings and rename files based on the contents of those strings.

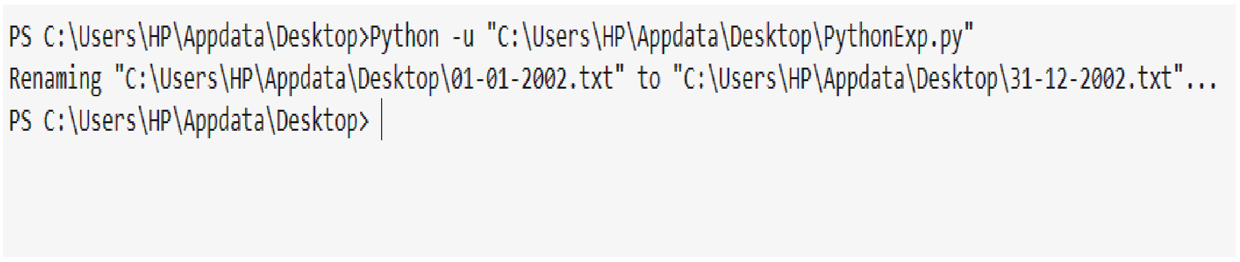
This program uses the os module to get a list of all the files in the current directory, and the re module to search for a specific date format in each file name. If a file name contains the specified date format, the program extracts the date from the name and creates a new file name using the format year_month_day_original_file_name. The shutil module is then used to rename the file. This program demonstrates how to search for specific patterns in strings, extract parts of those strings, and rename files based on the extracted information.

Here is a Python program that renames a group of files with a specific date format in their names:



```
C: > Users > HP > Desktop > PythonExp.py
import os
import re
import shutil

def rename_files():
    # Get the names of all files in the current directory
    file_list = os.listdir("./")

    # Specify the date format to search for
    date_format = re.compile(r"(\d{2})-(\d{2})-(\d{4})")

    # Loop through each file name
    for file_name in file_list:
        match = date_format.search(file_name)
        if match:
            # If the file name contains the specified date format,
            # extract the date and create a new file name
            day = match.group(1)
            month = match.group(2)
            year = match.group(3)
            new_file_name = f"{year}_{month}_{day}_{file_name}"

            # Rename the file
            shutil.move(file_name, new_file_name)

    print("Files have been renamed.")

if __name__ == "__main__":
    rename_files()
```

Output :



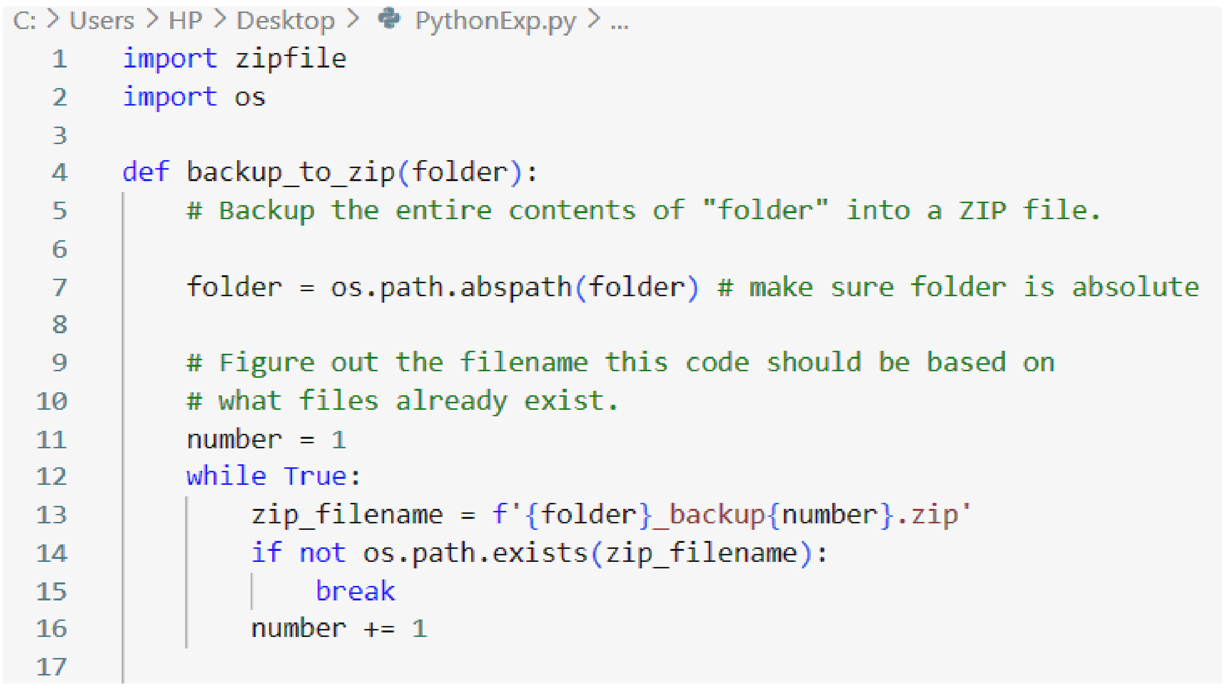
```
PS C:\Users\HP\Appdata\Desktop>Python -u "C:\Users\HP\Appdata\Desktop\PythonExp.py"
Renaming "C:\Users\HP\Appdata\Desktop\01-01-2002.txt" to "C:\Users\HP\Appdata\Desktop\31-12-2002.txt"...
PS C:\Users\HP\Appdata\Desktop>
```

14. Backing Up a Folder into a ZIP File

The "Backing Up a Folder into a ZipFile" program is used to create a backup of a given folder in the form of a ZIP file. The program takes the name of the folder to be backed up and the name of the ZIP file as inputs. The program uses the shutil module to create the ZIP file and backup the contents of the folder. The backup will contain all the files and subdirectories within the given folder. This program can be useful for regularly backing up important data to ensure its preservation and prevent data loss.

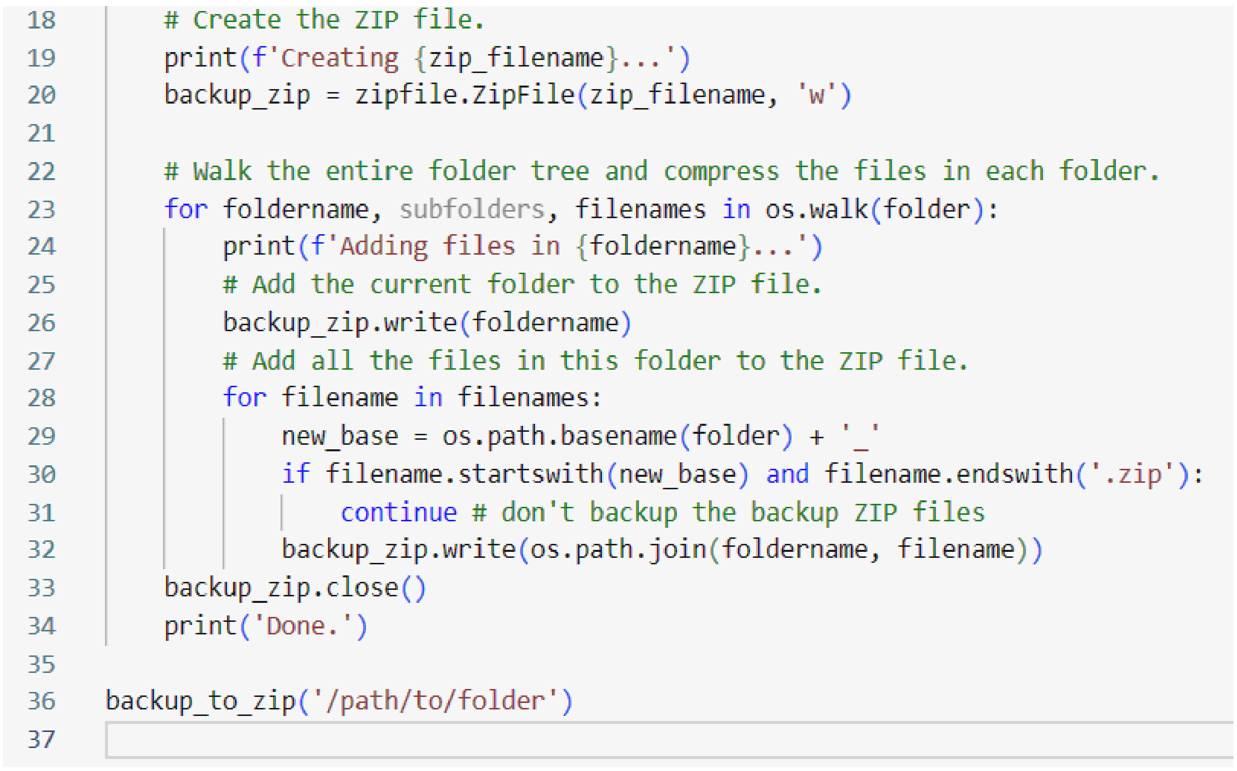
This program takes the path of a folder as an argument, and creates a ZIP file in the same directory with the name folder_backup#.zip, where # is a number that starts from 1 and increases by 1 each time the program is run. The program uses the os module to get a list of all the subdirectories and files in the folder, and the zipfile module to create the ZIP file and add the contents of the folder to it. The os.walk function is used to traverse the entire directory tree of the folder and add all the files and subdirectories to the ZIP file. To prevent backing up the backup ZIP files, the program checks if the filename starts with the folder name and ends with .zip. If the filename meets this criteria, the program skips that file.

Here is a full Python program that implements a backup of a folder into a ZIP file:



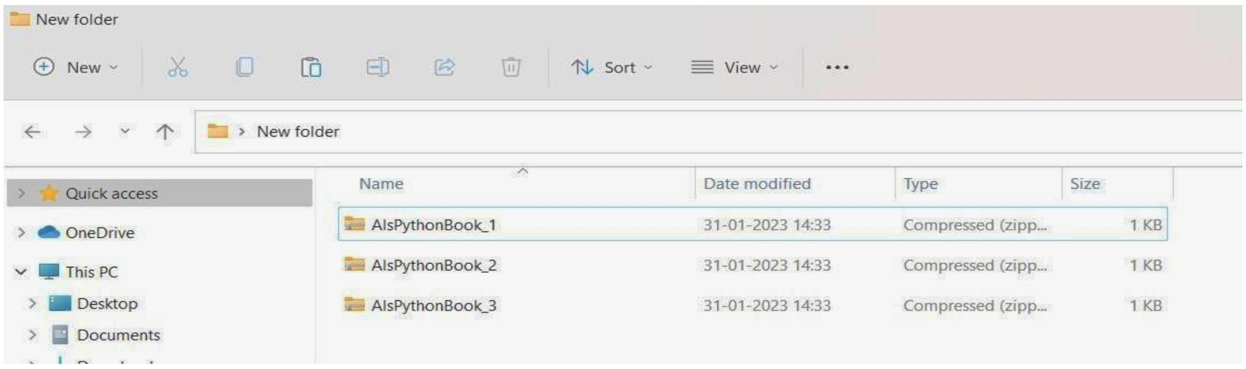
C: > Users > HP > Desktop > PythonExp.py > ...

```python
import zipfile
import os

def backup_to_zip(folder):
    # Backup the entire contents of "folder" into a ZIP file.

    folder = os.path.abspath(folder) # make sure folder is absolute

    # Figure out the filename this code should be based on
    # what files already exist.
    number = 1
    while True:
        zip_filename = f'{folder}_backup{number}.zip'
        if not os.path.exists(zip_filename):
            break
        number += 1

```



```python
    # Create the ZIP file.
    print(f'Creating {zip_filename}...')
    backup_zip = zipfile.ZipFile(zip_filename, 'w')

    # Walk the entire folder tree and compress the files in each folder.
    for foldername, subfolders, filenames in os.walk(folder):
        print(f'Adding files in {foldername}...')
        # Add the current folder to the ZIP file.
        backup_zip.write(foldername)
        # Add all the files in this folder to the ZIP file.
        for filename in filenames:
            new_base = os.path.basename(folder) + '_'
            if filename.startswith(new_base) and filename.endswith('.zip'):
                continue # don't backup the backup ZIP files
            backup_zip.write(os.path.join(foldername, filename))
    backup_zip.close()
    print('Done.')

backup_to_zip('/path/to/folder')
```

Output :



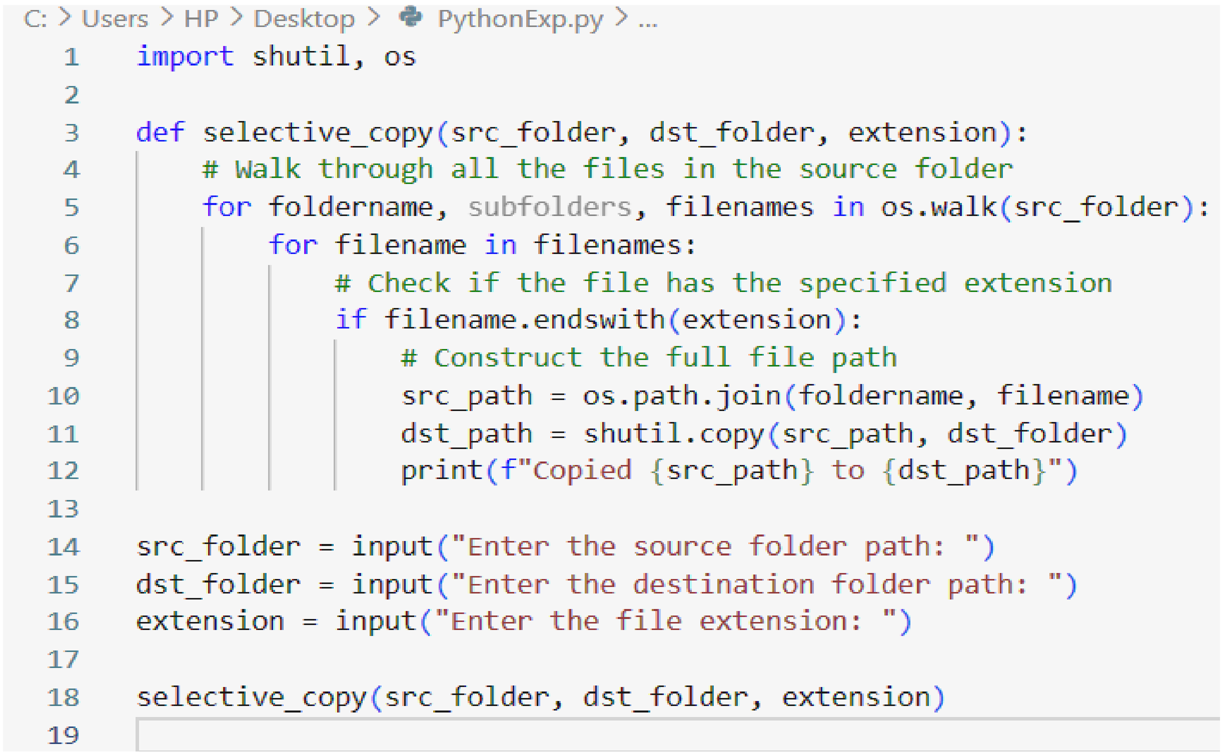
```
Name                 Date modified      Type              Size
AlsPythonBook_1      31-01-2023 14:33   Compressed (zipp...  1 KB
AlsPythonBook_2      31-01-2023 14:33   Compressed (zipp...  1 KB
AlsPythonBook_3      31-01-2023 14:33   Compressed (zipp...  1 KB
```

15. Selective Copy

The "Selective Copy" program is a Python script that performs a selective copy of files with a specific file extension, such as .pdf or .txt, from a source folder to a destination folder. The program scans the source folder and copies only the desired files to the destination folder, skipping all other files. The program is useful for backing up or moving specific types of files from one location to another.

This program allows you to selectively copy files with a specified file extension from a source folder to a destination folder. The program uses the os.walk function to traverse all the subdirectories within the source folder and the shutil.copy function to copy the selected files to the destination folder. The program takes three inputs: the source folder path, the destination folder path, and the file extension of the files to be copied. The program will copy all the files with the specified extension, including those in subdirectories, to the destination folder.

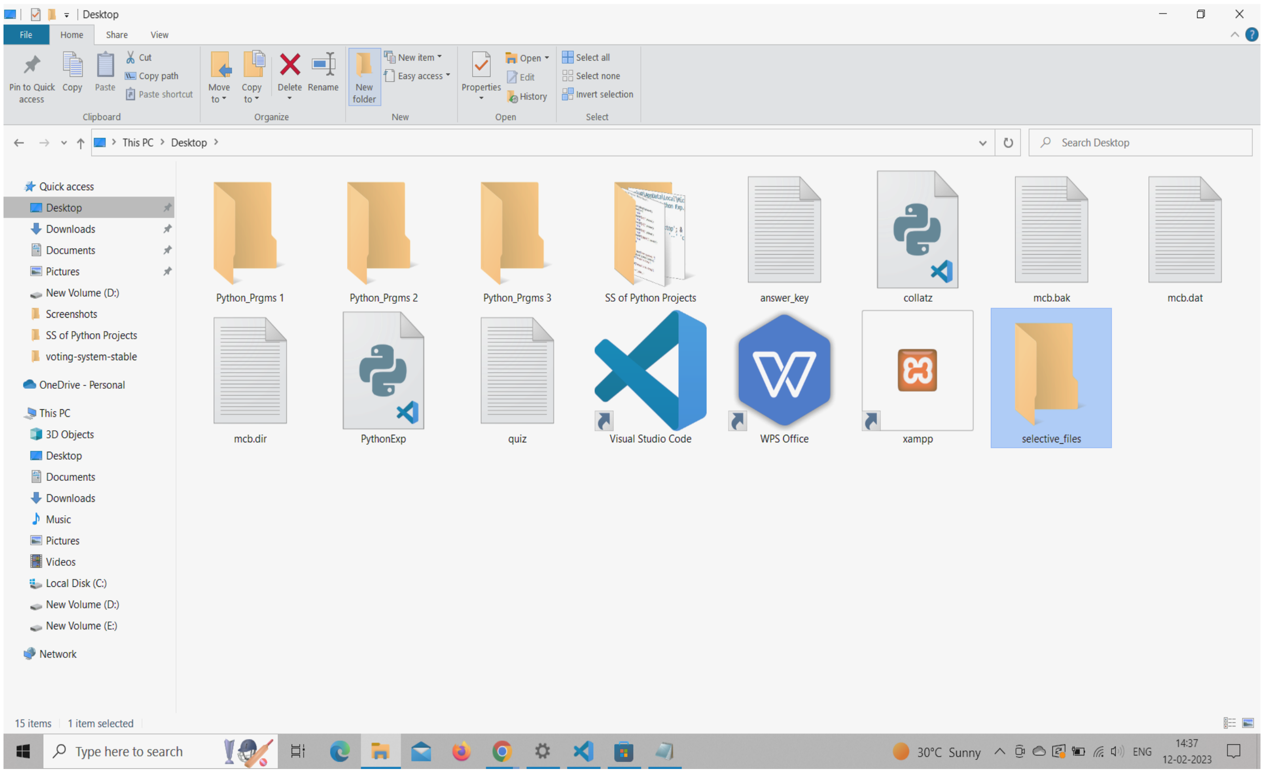
Here is a python program for "Selective Copy":



C: > Users > HP > Desktop > PythonExp.py > ...

```python
import shutil, os

def selective_copy(src_folder, dst_folder, extension):
    # Walk through all the files in the source folder
    for foldername, subfolders, filenames in os.walk(src_folder):
        for filename in filenames:
            # Check if the file has the specified extension
            if filename.endswith(extension):
                # Construct the full file path
                src_path = os.path.join(foldername, filename)
                dst_path = shutil.copy(src_path, dst_folder)
                print(f"Copied {src_path} to {dst_path}")

src_folder = input("Enter the source folder path: ")
dst_folder = input("Enter the destination folder path: ")
extension = input("Enter the file extension: ")

selective_copy(src_folder, dst_folder, extension)
```

Output :



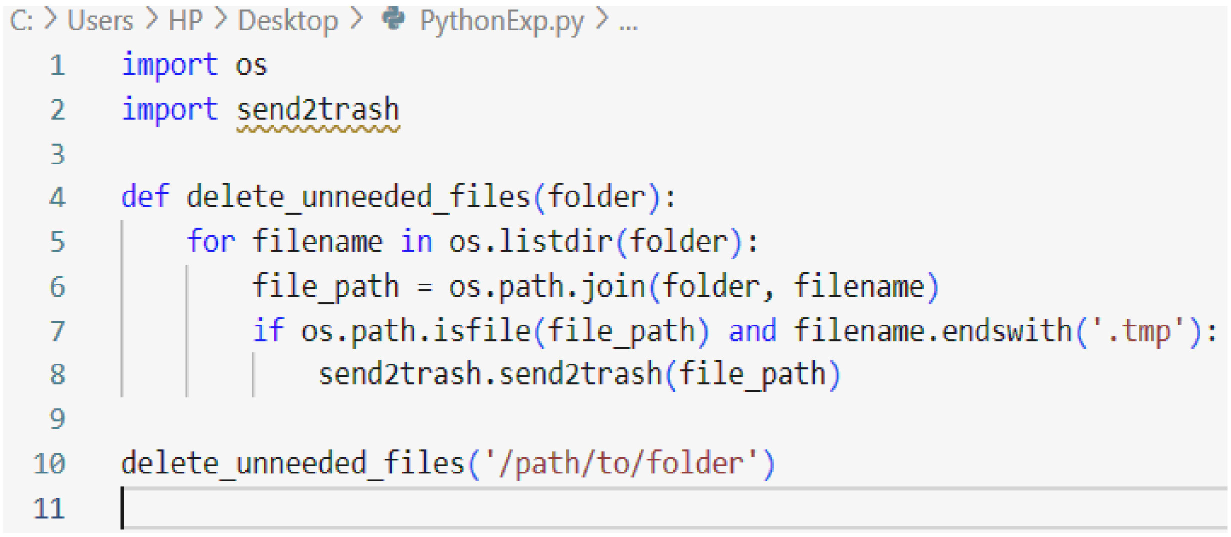
16. Deleting Unneeded Files

The "Deleting Unneeded Files" program is a python script that helps in cleaning up the file system by identifying and removing files that are no longer needed. This program can be useful in freeing up storage space, organizing files, and improving the overall performance of the system. The program can search for files based on various criteria such as file type, date of creation, date of modification, size, etc. and delete the ones that meet the specified conditions. The program should be run with caution as it can result in permanent loss of data if not used carefully.

This program uses the os and send2trash modules to accomplish its task. The os.listdir() function is used to get a list of all the filenames in the specified folder. For each file, the full path is constructed using os.path.join() and then passed to the send2trash() function from the send2trash module. The send2trash() function is used to send the file to the trash/recycle bin, rather than permanently deleting it.

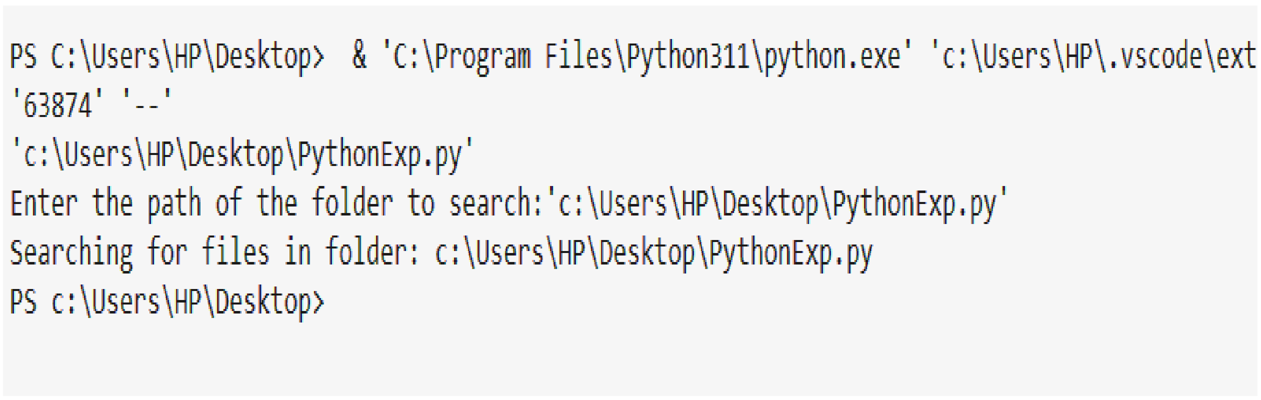
The program only deletes files with the .tmp extension, but this can be changed by modifying the if statement.

Here's a basic program in Python that deletes unneeded files from a specified folder:



```python
import os
import send2trash

def delete_unneeded_files(folder):
    for filename in os.listdir(folder):
        file_path = os.path.join(folder, filename)
        if os.path.isfile(file_path) and filename.endswith('.tmp'):
            send2trash.send2trash(file_path)

delete_unneeded_files('/path/to/folder')

```

Output :

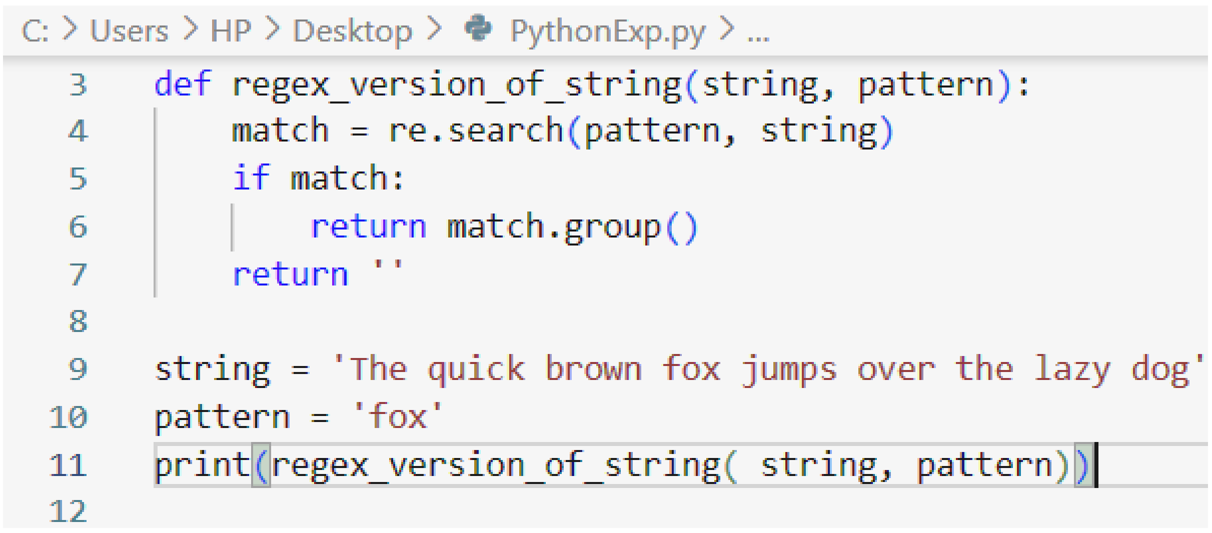


```
PS C:\Users\HP\Desktop>  & 'C:\Program Files\Python311\python.exe' 'c:\Users\HP\.vscode\ext
'63874' '--'
'c:\Users\HP\Desktop\PythonExp.py'
Enter the path of the folder to search:'c:\Users\HP\Desktop\PythonExp.py'
Searching for files in folder: c:\Users\HP\Desktop\PythonExp.py
PS C:\Users\HP\Desktop>
```

17. Regex Version of Strip ( )

The "Regex Version of String" program is a script that utilizes the regular expression (regex) module in Python to search and manipulate strings. The program takes a string input and uses regex pattern matching to perform specific actions such as searching for a specific pattern, replacing a portion of the string, or extracting information from the string. The output of the program will depend on the regex pattern and the actions performed on the input string. In this program, the re module is imported to use regular expressions. The regex_version_of_string function takes a string and a pattern as inputs and returns the first occurrence of the pattern in the string, if it exists. The re.search method is used to search for the pattern in the string and the group method of the match object is used to return the matched string. If there is no match, an empty string is returned. In the example, the string 'The quick brown fox jumps over the lazy dog' is searched for the pattern 'fox' and the result 'fox' is returned.

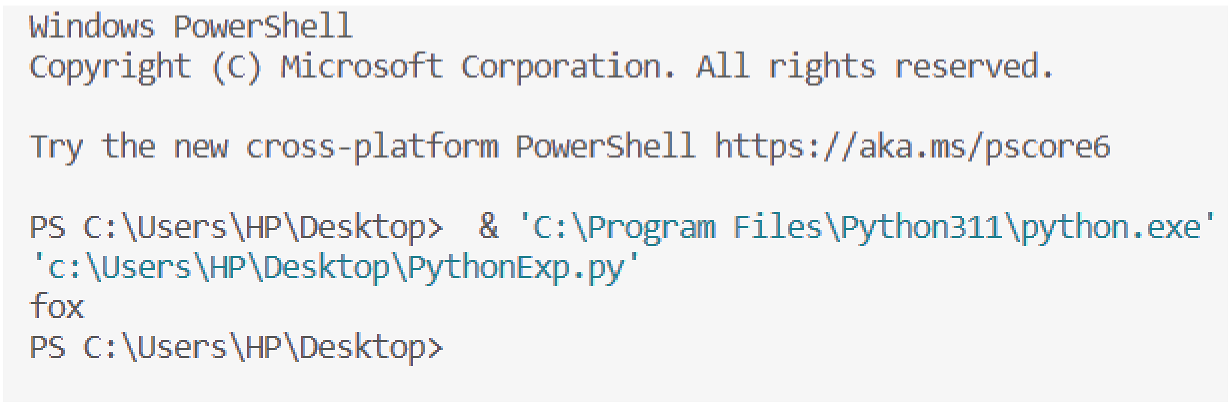
Here is a simple python program that uses regular expressions (regex) to implement a version of the string method:



C: > Users > HP > Desktop > PythonExp.py > ...

```python
def regex_version_of_string(string, pattern):
    match = re.search(pattern, string)
    if match:
        return match.group()
    return ''

string = 'The quick brown fox jumps over the lazy dog'
pattern = 'fox'
print(regex_version_of_string( string, pattern))
```

Output :

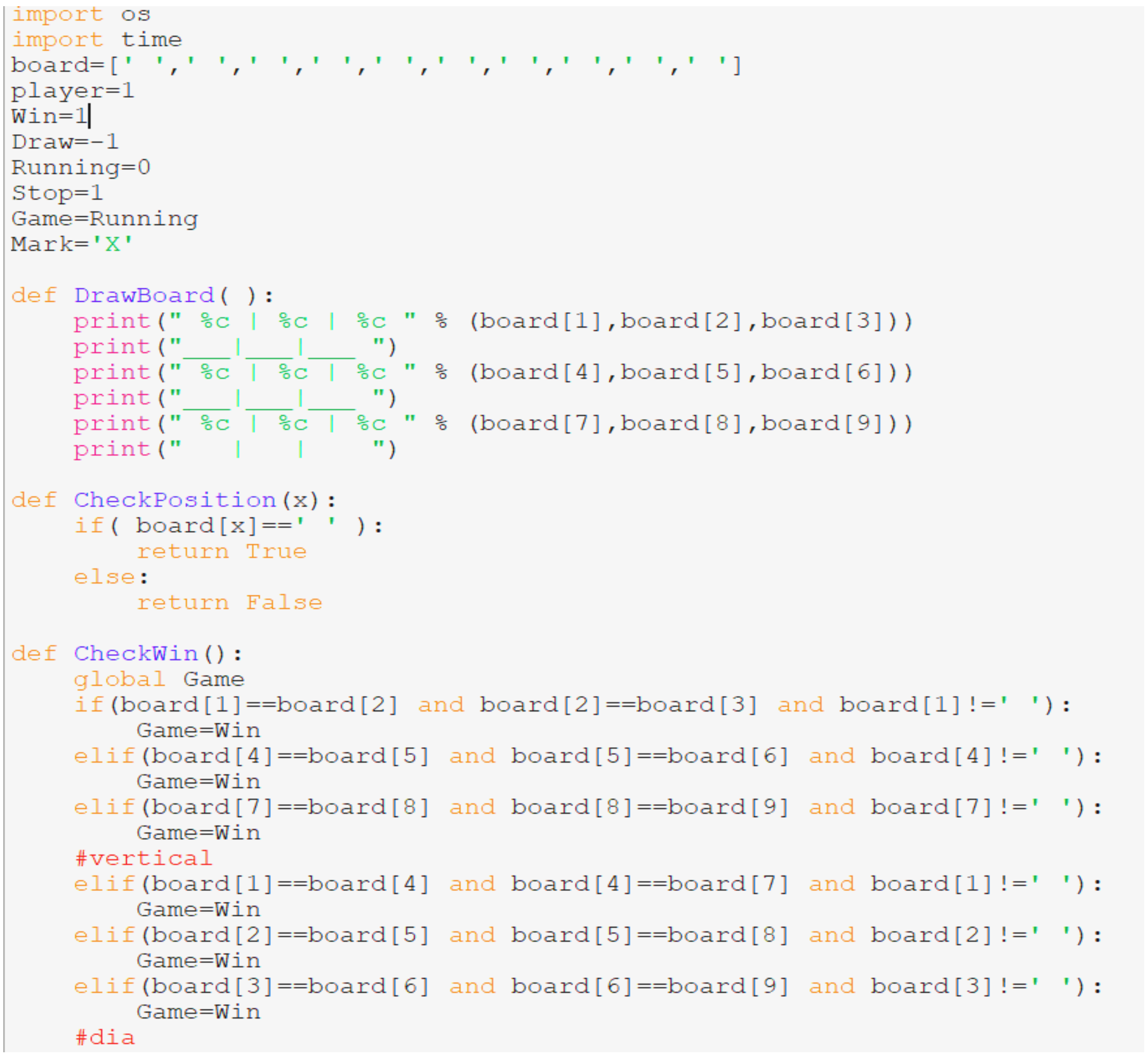


```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS C:\Users\HP\Desktop>  & 'C:\Program Files\Python311\python.exe'
'c:\Users\HP\Desktop\PythonExp.py'
fox
PS C:\Users\HP\Desktop>
```

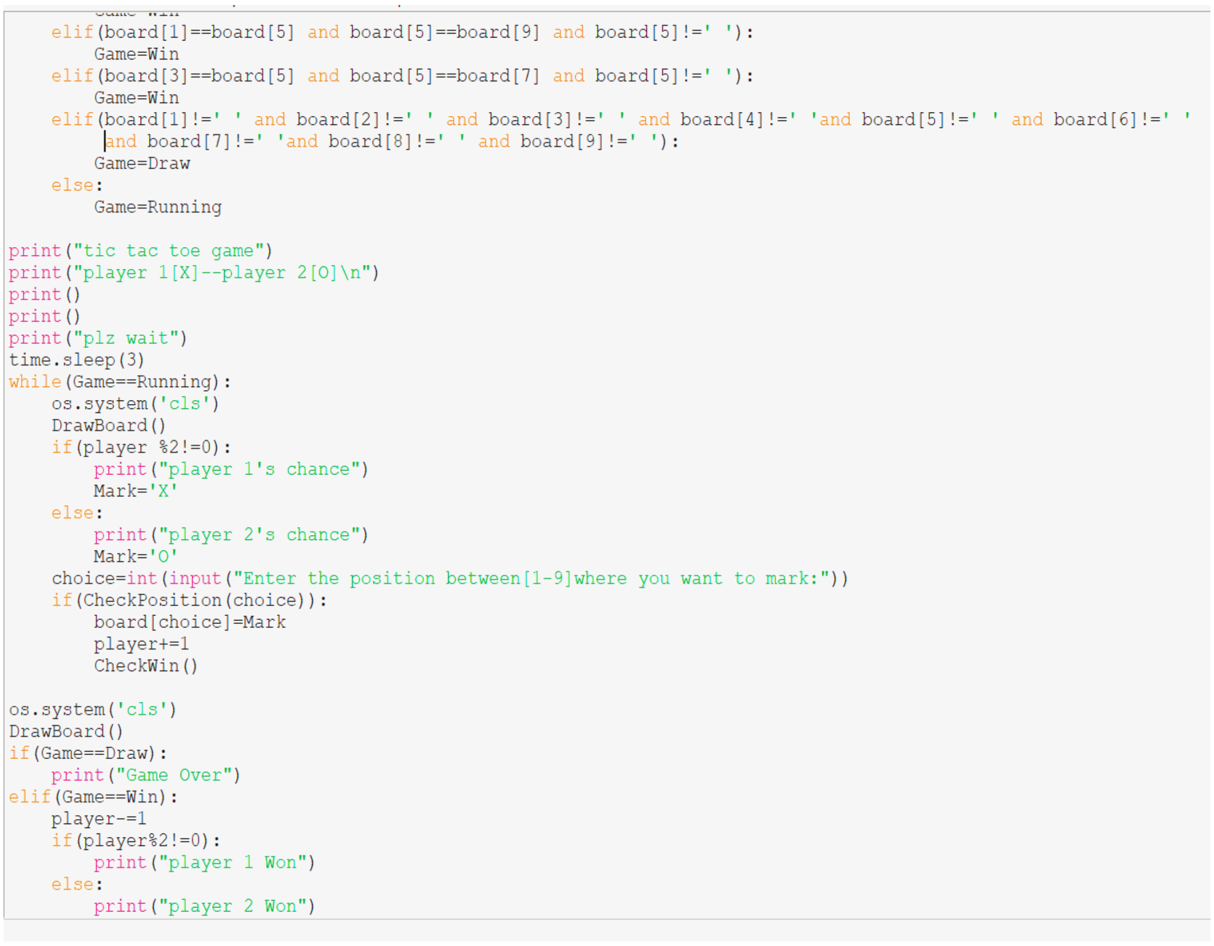
18. Tic-Tac-Toe Game

The Tic Tac Toe Python program is a simple implementation of the classic two-player game, Tic Tac Toe. The game is played on a 3x3 board, and each player takes turns placing their symbol (X or O) on the board. The first player to get three of their symbols in a row, column, or diagonal, wins the game.The program uses a list of 9 elements to represent the game board, where each element represents a square on the board. The print_board() function takes the list and formats it into a 3x3 board display, with each square separated by a | symbol.The player_move(icon) function takes the player's symbol (X or O) as input and asks the player to enter the square they want to place their symbol in. The function checks if the square is empty before updating the board.The is_victory(icon) function checks if the player with the specified symbol has won the game. It checks all the possible winning combinations (rows, columns, diagonals) and returns True if there is a match.The game is played in a while loop that keeps running until either player wins or all the squares are filled, at which point the game ends in a draw.

Here is a simple implementation of the Tic-Tac-Toe game in Python:

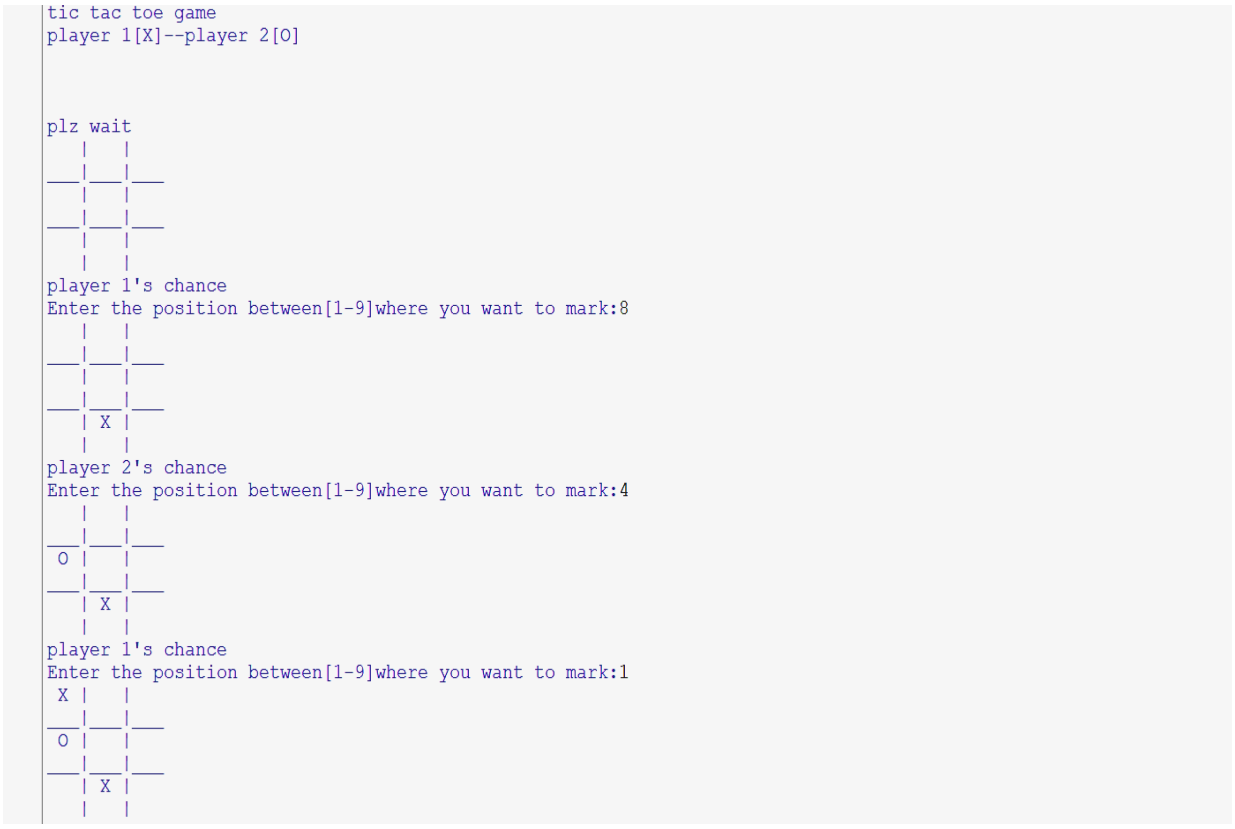


```python
import os
import time
board=[' ',' ',' ',' ',' ',' ',' ',' ',' ',' ']
player=1
Win=1
Draw=-1
Running=0
Stop=1
Game=Running
Mark='X'

def DrawBoard( ):
    print(" %c | %c | %c " % (board[1],board[2],board[3]))
    print("___|___|___ ")
    print(" %c | %c | %c " % (board[4],board[5],board[6]))
    print("___|___|___ ")
    print(" %c | %c | %c " % (board[7],board[8],board[9]))
    print("   |   |    ")

def CheckPosition(x):
    if( board[x]==' ' ):
        return True
    else:
        return False

def CheckWin():
    global Game
    if(board[1]==board[2] and board[2]==board[3] and board[1]!=' '):
        Game=Win
    elif(board[4]==board[5] and board[5]==board[6] and board[4]!=' '):
        Game=Win
    elif(board[7]==board[8] and board[8]==board[9] and board[7]!=' '):
        Game=Win
    #vertical
    elif(board[1]==board[4] and board[4]==board[7] and board[1]!=' '):
        Game=Win
    elif(board[2]==board[5] and board[5]==board[8] and board[2]!=' '):
        Game=Win
    elif(board[3]==board[6] and board[6]==board[9] and board[3]!=' '):
        Game=Win
    #dia
```

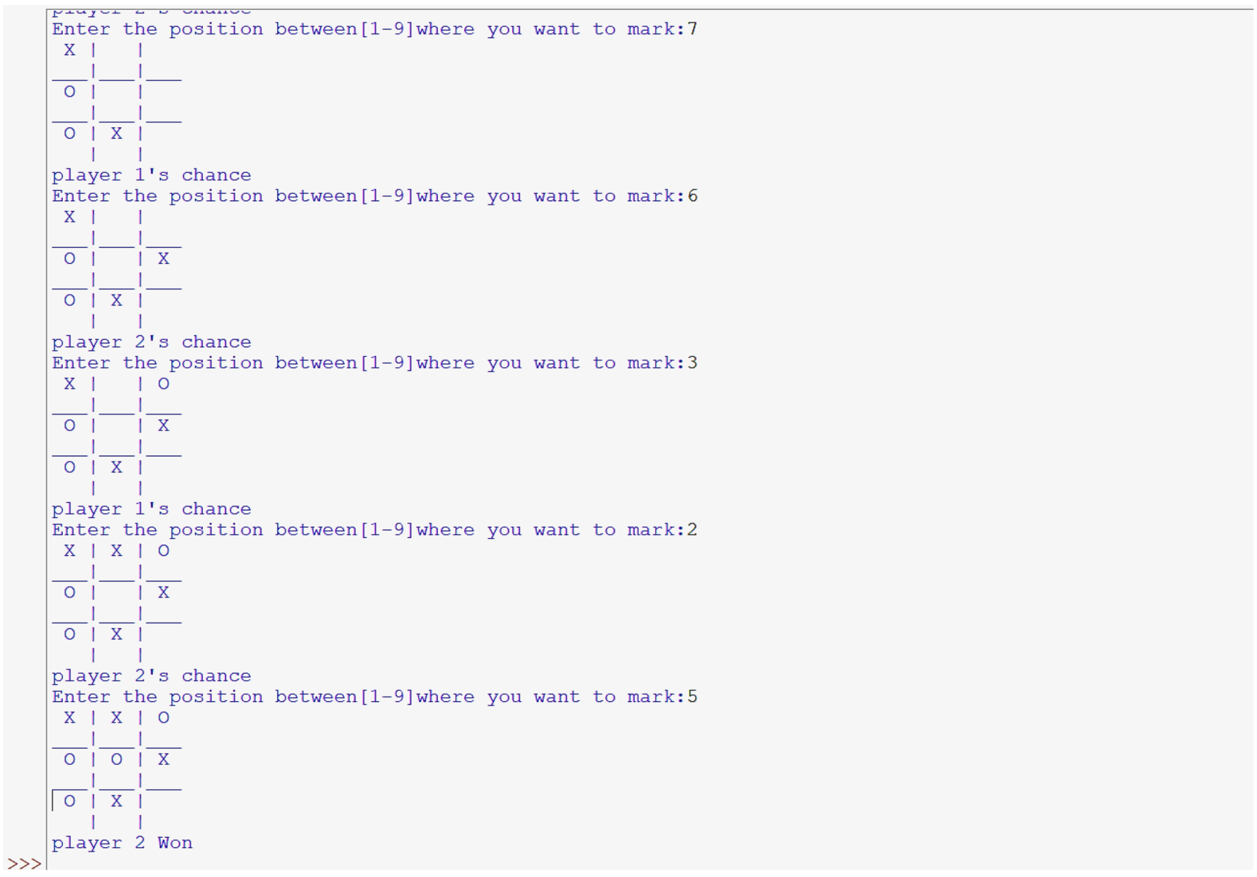


```python
    elif(board[1]==board[5] and board[5]==board[9] and board[5]!=' '):
        Game=Win
    elif(board[3]==board[5] and board[5]==board[7] and board[5]!=' '):
        Game=Win
    elif(board[1]!=' ' and board[2]!=' ' and board[3]!=' ' and board[4]!=' 'and board[5]!=' ' and board[6]!=' '
         and board[7]!=' 'and board[8]!=' ' and board[9]!=' '):
        Game=Draw
    else:
        Game=Running

print("tic tac toe game")
print("player 1[X]--player 2[O]\n")
print()
print()
print("plz wait")
time.sleep(3)
while(Game==Running):
    os.system('cls')
    DrawBoard()
    if(player %2!=0):
        print("player 1's chance")
        Mark='X'
    else:
        print("player 2's chance")
        Mark='O'
    choice=int(input("Enter the position between[1-9]where you want to mark:"))
    if(CheckPosition(choice)):
        board[choice]=Mark
        player+=1
        CheckWin()

os.system('cls')
DrawBoard()
if(Game==Draw):
    print("Game Over")
elif(Game==Win):
    player-=1
    if(player%2!=0):
        print("player 1 Won")
    else:
        print("player 2 Won")
```

Output :



```
tic tac toe game
player 1[X]--player 2[O]



plz wait
   |   |
___|___|___
   |   |
___|___|___
   |   |
   |   |
player 1's chance
Enter the position between[1-9]where you want to mark:8
   |   |
___|___|___
   |   |
___|___|___
   | X |
   |   |
player 2's chance
Enter the position between[1-9]where you want to mark:4
   |   |
___|___|___
 O |   |
___|___|___
   | X |
   |   |
player 1's chance
Enter the position between[1-9]where you want to mark:1
 X |   |
___|___|___
 O |   |
___|___|___
   | X |
   |   |
```

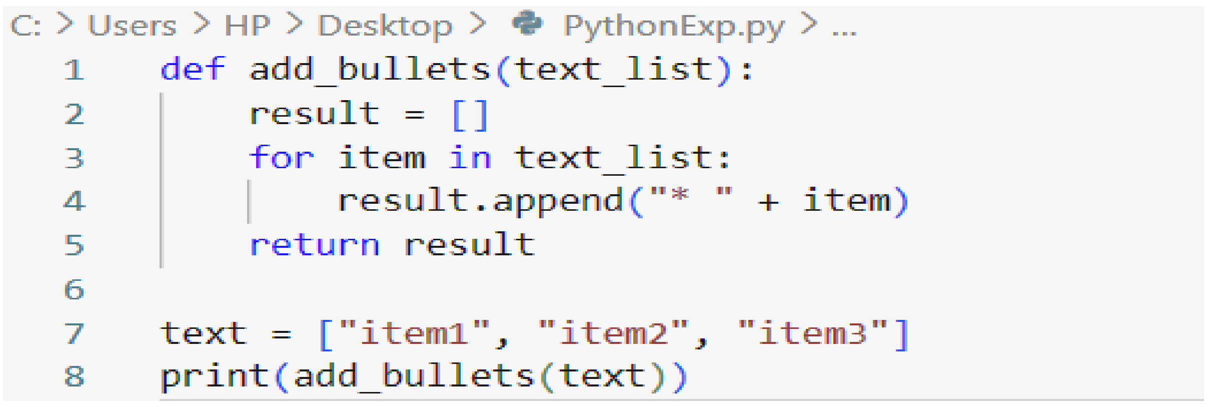


```
Enter the position between[1-9]where you want to mark:7
 X |   |
___|___|___
 O |   |
___|___|___
 O | X |
   |   |
player 1's chance
Enter the position between[1-9]where you want to mark:6
 X |   |
___|___|___
 O |   | X
___|___|___
 O | X |
   |   |
player 2's chance
Enter the position between[1-9]where you want to mark:3
 X |   | O
___|___|___
 O |   | X
___|___|___
 O | X |
   |   |
player 1's chance
Enter the position between[1-9]where you want to mark:2
 X | X | O
___|___|___
 O |   | X
___|___|___
 O | X |
   |   |
player 2's chance
Enter the position between[1-9]where you want to mark:5
 X | X | O
___|___|___
 O | O | X
___|___|___
 O | X |
   |   |
player 2 Won
>>>
```

19. Adding Bullets To Wiki Markup

The "Adding Bullets to Wiki Markup" program is a Python script that adds bullet points to a given piece of text in the format used in wikipedia markup. It takes as input a plain text string and converts it into a bullet-point list using the "" symbol to denote each item in the list. The program takes each line of the input text, adds a "" symbol at the beginning, and concatenates all of the resulting strings into a single output string. This output string can then be used as wikipedia markup text, which when rendered, will display as a bullet-point list.The add_bullets function takes a list of text items as input, and then loops through each item in the list. It adds a bullet point (represented as *) to the start of each item, and appends the modified item to a new list called result. Finally, the result list is returned as the output of the function.

Here's a simple Python program for Adding Bullets to Wiki Markup:



C: > Users > HP > Desktop > PythonExp.py > ...

```python
def add_bullets(text_list):
    result = []
    for item in text_list:
        result.append("* " + item)
    return result

text = ["item1", "item2", "item3"]
print(add_bullets(text))
```

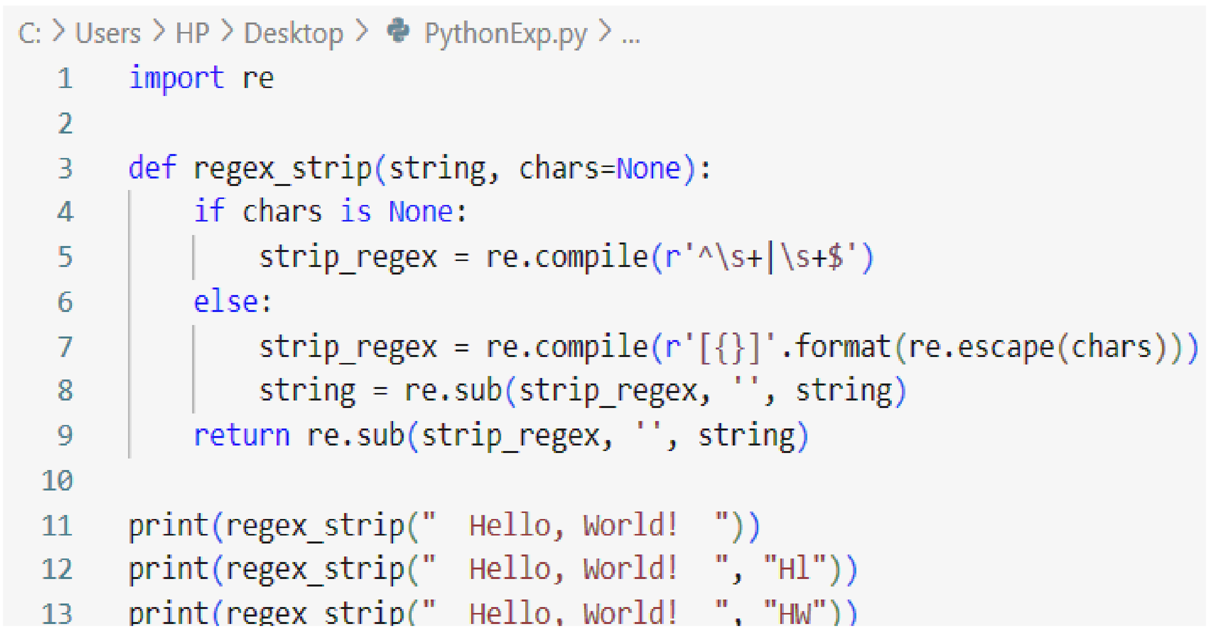
Output :

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS C:\Users\HP\Desktop>  & 'C:\Program Files\Python311\python.exe'
'c:\Users\HP\Desktop\PythonExp.py'
['* item1', '* item2', '* item3']
PS C:\Users\HP\Desktop>
```

20. Regex Strip

The "Regex Strip" program is a python script that uses the regular expression module to remove characters from the start and end of a string. The user can specify the characters to be removed and the program will use regular expressions to match and remove those characters from the input string. The output will be the modified string with the specified characters removed from the start and end. This program is useful for cleaning up text data, removing unwanted characters and formatting the text in a consistent manner.In this program, the regex_strip function takes two arguments: a string string, and an optional string of characters to remove, chars. If chars is not provided, it removes any whitespace characters from the start and end of the string. If chars is provided, it removes any occurrences of those characters from the start and end of the string.The function uses regular expressions to perform the stripping. The regular expression is compiled using the re module's re.compile function, and the actual stripping is done using the re.sub function. The function returns the modified string.

Here's a program in Python for regex_strip function:



C: > Users > HP > Desktop > PythonExp.py > ...

```python
import re

def regex_strip(string, chars=None):
    if chars is None:
        strip_regex = re.compile(r'^\s+|\s+$')
    else:
        strip_regex = re.compile(r'[{}]'.format(re.escape(chars)))
        string = re.sub(strip_regex, '', string)
    return re.sub(strip_regex, '', string)

print(regex_strip("  Hello, World!  "))
print(regex_strip("  Hello, World!  ", "Hl"))
print(regex_strip("  Hello, World!  ", "HW"))
```

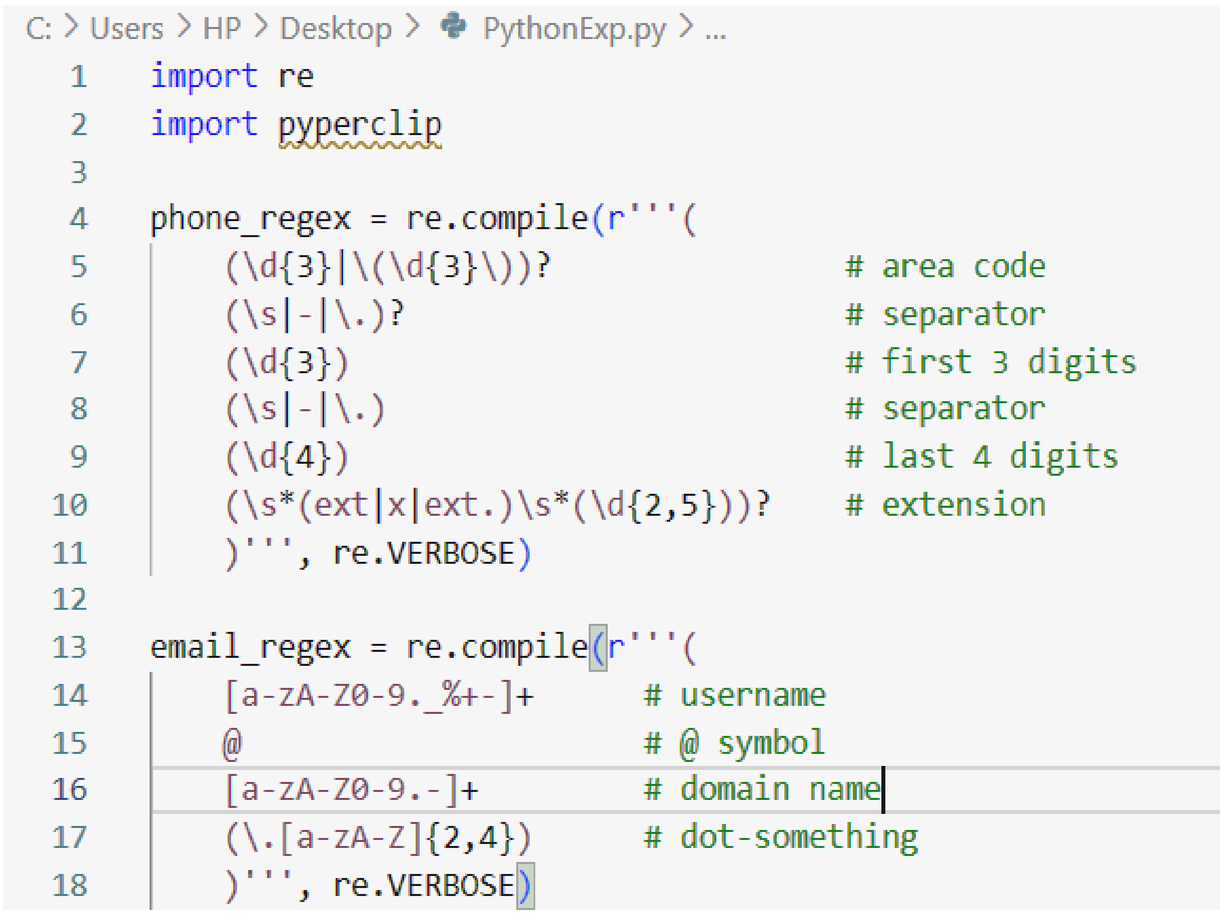
Output :

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS C:\Users\HP\Desktop>  & 'C:\Program Files\Python311\python.exe'
'c:\Users\HP\Desktop\PythonExp.py'
Hello, World!
  eo, Word!
  ello, orld!
PS C:\Users\HP\Desktop>
```

21. Phone Number and Email Address Extractor

The "Phone Number and Email address extractor" program is a Python script that automates the process of extracting phone numbers and email addresses from a large amount of text. The program uses regular expressions (regex) to search the text for patterns that match the format of phone numbers and email addresses, and then outputs the results in a neat and organized manner. This program can be useful for anyone who needs to gather and store contact information from a variety of sources, such as websites, documents, or emails.

Here is a python Program for Phone Number And Email Address Extractor



```python
import re
import pyperclip

phone_regex = re.compile(r'''(
    (\d{3}|\(\d{3}\))?                # area code
    (\s|-|\.)?                        # separator
    (\d{3})                           # first 3 digits
    (\s|-|\.)                         # separator
    (\d{4})                           # last 4 digits
    (\s*(ext|x|ext.)\s*(\d{2,5}))?    # extension
    )''', re.VERBOSE)

email_regex = re.compile(r'''(
    [a-zA-Z0-9._%+-]+      # username
    @                      # @ symbol
    [a-zA-Z0-9.-]+         # domain name
    (\.[a-zA-Z]{2,4})      # dot-something
    )''', re.VERBOSE)
```

```python
text = str(pyperclip.paste())

matches = []
for groups in phone_regex.findall(text):
    phone_num = '-'.join([groups[1], groups[3], groups[5]])
    if groups[8] != '':
        phone_num += ' x' + groups[8]
    matches.append(phone_num)
for groups in email_regex.findall(text):
    matches.append(groups[0])

if len(matches) > 0:
    pyperclip.copy('\n'.join(matches))
    print('Copied to clipboard:')
    print('\n'.join(matches))
else:
    print('No phone numbers or email addresses found.')
```

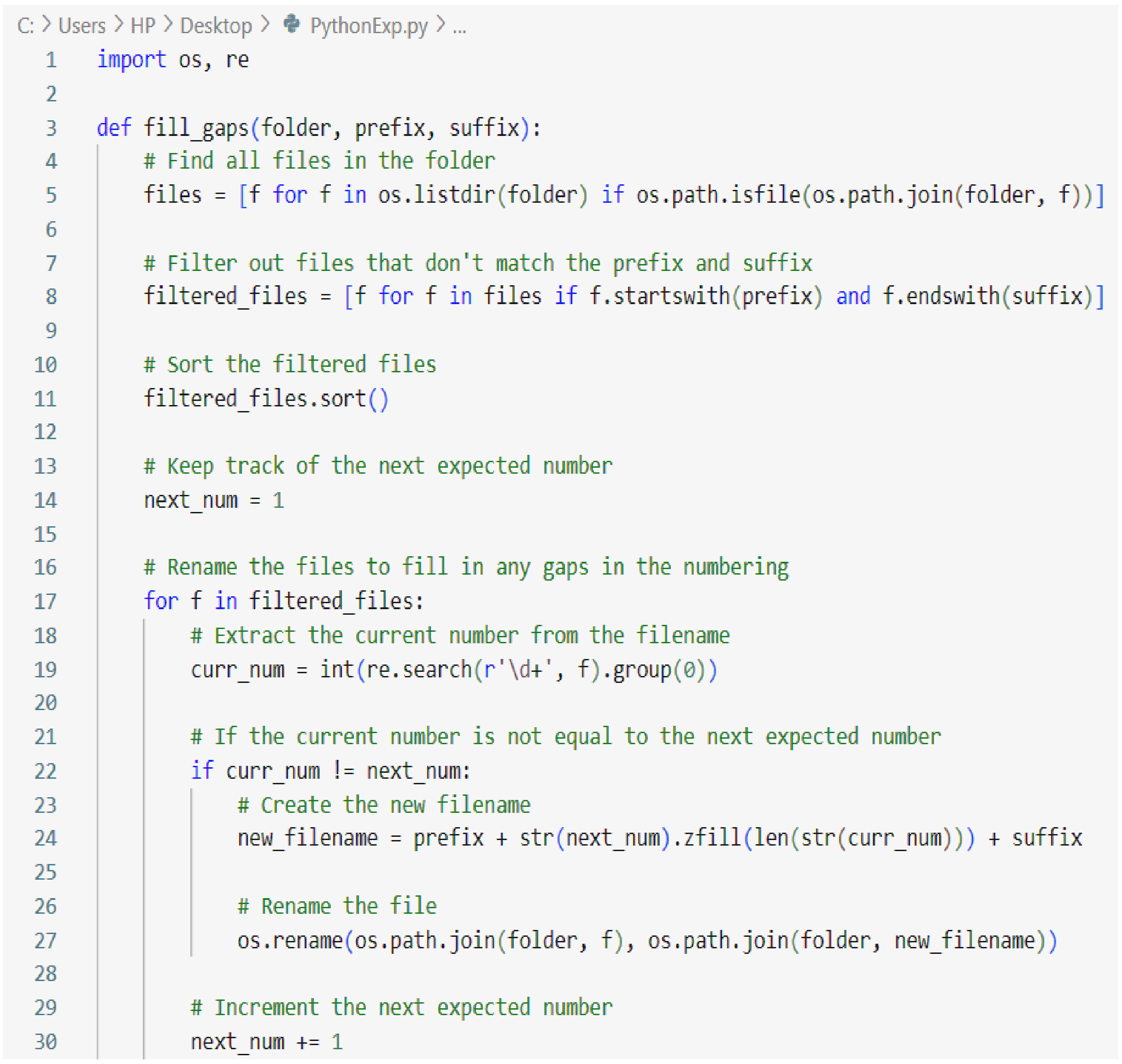
Output :

```
Copied to clipboard:
800-420-7240
415-863-9900
415-863-9950
info@nostarch.com
media@nostarch.com
academic@nostarch.com
help@nostarch.com
```

22. Filling in the Gaps

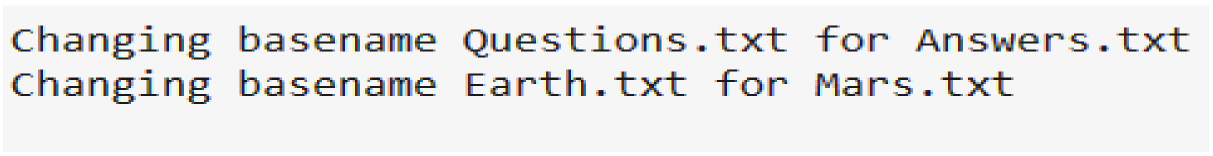
The "Filling In The Gaps" program is a python script that takes a directory path as input and looks for files that match a given pattern (such as "spam001.txt", "spam002.txt", etc.). If a gap is found in the numbering (for example, if "spam001.txt" and "spam003.txt" exist, but "spam002.txt" does not), the program will rename all the files after the gap so that the numbering continues without any missing numbers. This program is useful for organizing a large number of files into a logical, sequential order.

Here is a python program for "Filling in the Gaps "



```python
import os, re

def fill_gaps(folder, prefix, suffix):
    # Find all files in the folder
    files = [f for f in os.listdir(folder) if os.path.isfile(os.path.join(folder, f))]

    # Filter out files that don't match the prefix and suffix
    filtered_files = [f for f in files if f.startswith(prefix) and f.endswith(suffix)]

    # Sort the filtered files
    filtered_files.sort()

    # Keep track of the next expected number
    next_num = 1

    # Rename the files to fill in any gaps in the numbering
    for f in filtered_files:
        # Extract the current number from the filename
        curr_num = int(re.search(r'\d+', f).group(0))

        # If the current number is not equal to the next expected number
        if curr_num != next_num:
            # Create the new filename
            new_filename = prefix + str(next_num).zfill(len(str(curr_num))) + suffix

            # Rename the file
            os.rename(os.path.join(folder, f), os.path.join(folder, new_filename))

        # Increment the next expected number
        next_num += 1
```

Output :



```
Changing basename Questions.txt for Answers.txt
Changing basename Earth.txt for Mars.txt
```

PYTHON MINI PROJECT

VOTING SYSTEM

Voting System Using Python Program

This code is a simple voting system implemented in python. It asks the user to enter the name of two nominees for the election and sets the initial vote count for each nominee to zero. Then, the code takes the voter's ID and verifies if it is a valid voter. If the voter ID is valid, the voter can cast their vote for one of the two nominees by entering either 1 or 2. The voting session continues until all voters have cast their vote. At the end, the program calculates the percentage of votes received by each nominee and declares the winner based on the number of votes received. If both nominees receive an equal number of votes, the result is declared inconclusive.

The python program that implements a simple voting system. The program has the following steps:

1. Welcome Message: The program starts by printing a welcome message to the user "Welcome To The Voting Center".

2. Nominee Information: The user is prompted to enter the names of two nominees for the election. The names are stored in variables nominee1 and nominee2.

3. Initial Vote Count: The initial vote count for each nominee is set to zero. The variables nominee1Votes and nominee2Votes store the vote count for each nominee.

4. Voter ID: The voter ID is used to verify the identity of the voter. A list named "voterId" is created to store the valid voter IDs. The variable numberOfVoters stores the number of voters.

5. Main Loop: A while loop is used to continue the voting session until all voters have cast their vote.

6. Voter ID Verification: The user is prompted to enter their voter ID. The program then checks if the entered ID is present in the voterId list. If the ID is present, the voter is considered a genuine voter and the program moves to the next step.

7. Vote Cast: The program prompts the user to enter their vote for one of the two nominees. The user can enter either 1 or 2 to cast their vote for nominee1 or nominee2, respectively. The vote count for the selected nominee is then incremented.

8. Vote Count Update: The vote count for each nominee is updated after each vote is cast.

9. End of Voting Session: The voting session ends when all voters have cast their vote, and the voterId list becomes empty.

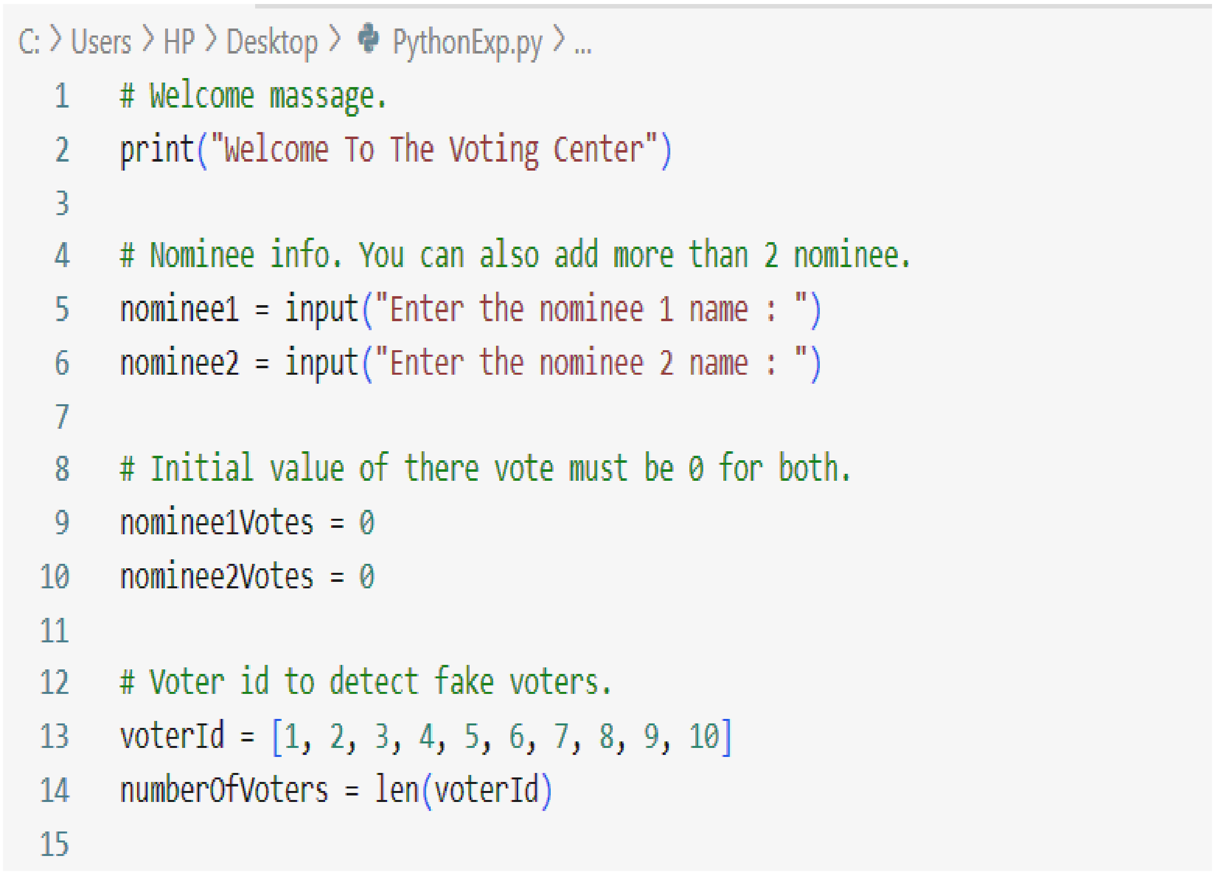
10. Result Declaration: The program calculates the percentage of votes received by each nominee and declares the winner based on the number of votes received. If both nominees receive an equal number of votes, the result is declared inconclusive.

11. Goodbye Message: The program ends with a goodbye message "Thanks for giving your valuable time and vote."

The program doesn't use any modules. It only uses built-in functions and data structures provided by the Python standard library. The following built-in functions and data structures are used in the program:

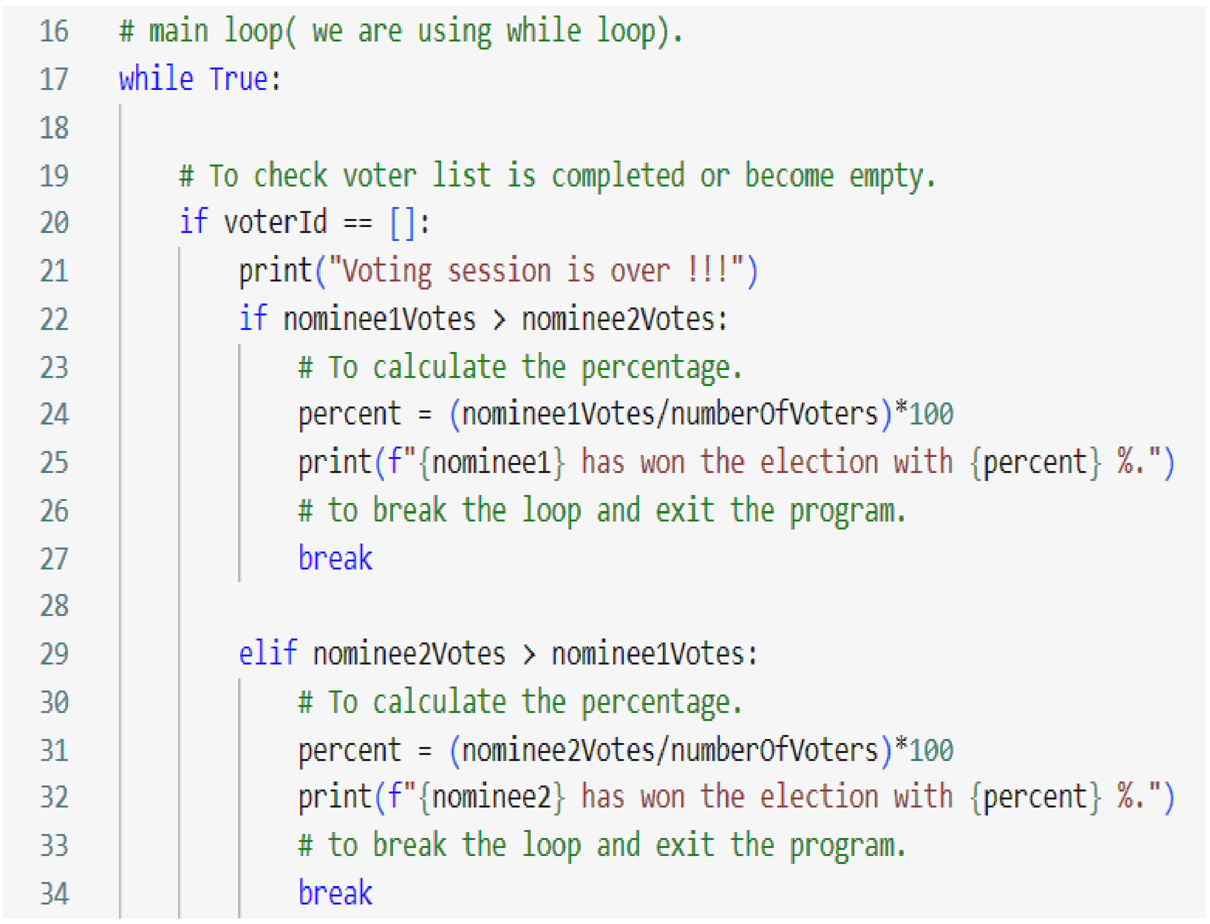
- print() function: used to display output to the user.
- input() function: used to get input from the user.
- int() function: used to convert a string input to an integer.
- list: used to store the voter IDs.
- while loop: used to repeat the voting process until all voters have cast their vote.
- if-elif-else statement: used to implement the voting process and declare the winner.

Here is the Python Program for the " Simple Voting System "

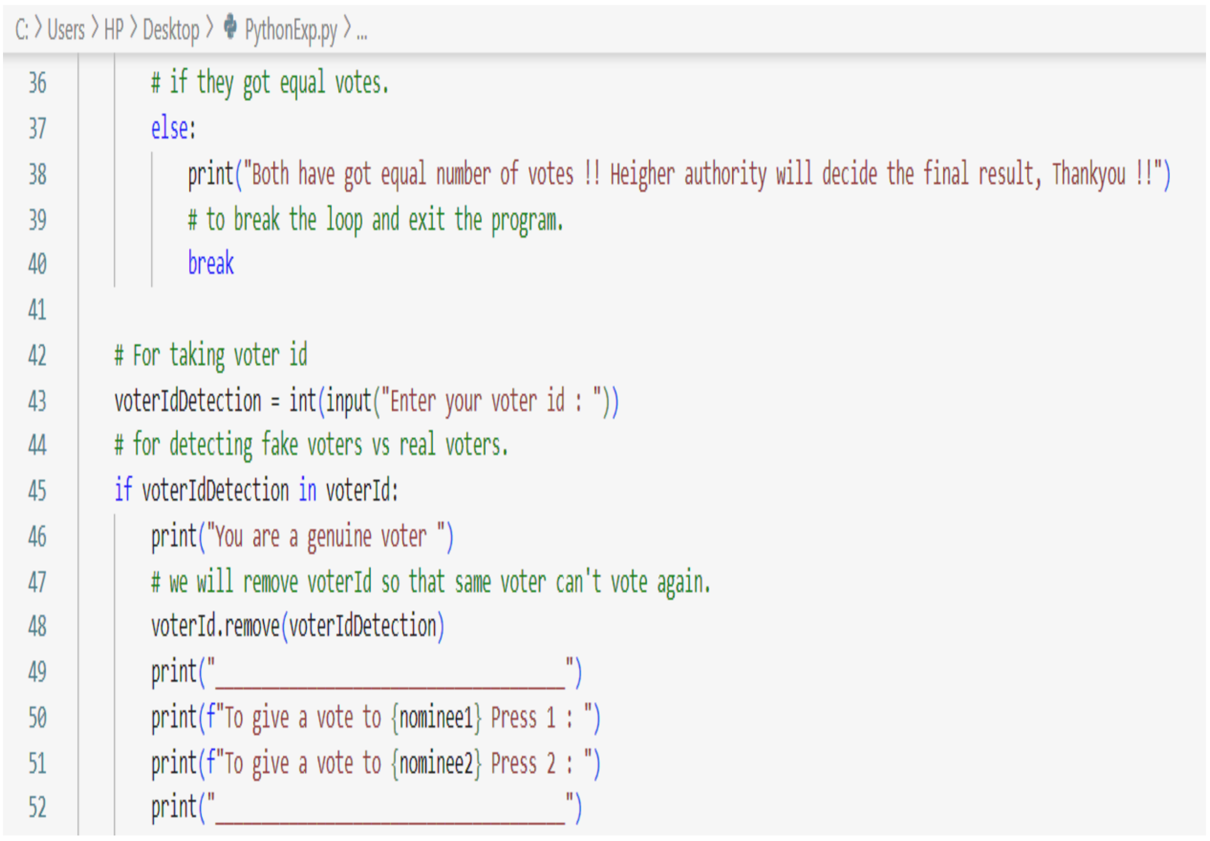


C: > Users > HP > Desktop > PythonExp.py > ...

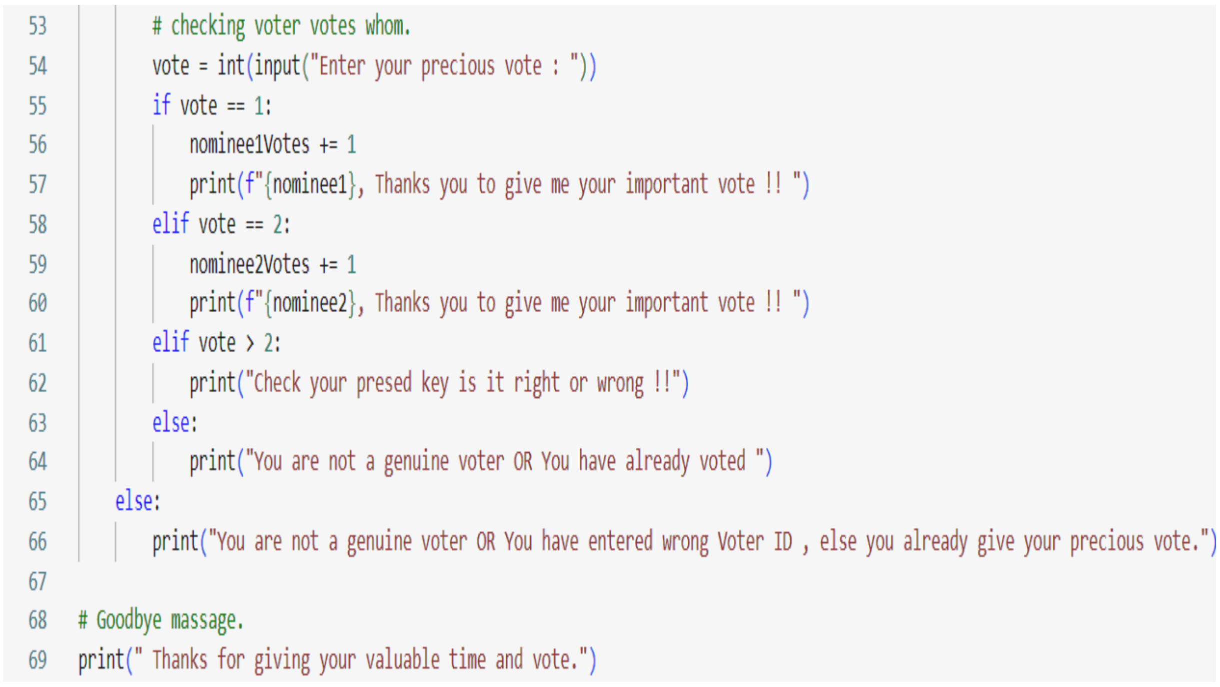
```python
# Welcome massage.
print("Welcome To The Voting Center")

# Nominee info. You can also add more than 2 nominee.
nominee1 = input("Enter the nominee 1 name : ")
nominee2 = input("Enter the nominee 2 name : ")

# Initial value of there vote must be 0 for both.
nominee1Votes = 0
nominee2Votes = 0

# Voter id to detect fake voters.
voterId = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
numberOfVoters = len(voterId)

```



```python
# main loop( we are using while loop).
while True:

    # To check voter list is completed or become empty.
    if voterId == []:
        print("Voting session is over !!!")
        if nominee1Votes > nominee2Votes:
            # To calculate the percentage.
            percent = (nominee1Votes/numberOfVoters)*100
            print(f"{nominee1} has won the election with {percent} %.")
            # to break the loop and exit the program.
            break

        elif nominee2Votes > nominee1Votes:
            # To calculate the percentage.
            percent = (nominee2Votes/numberOfVoters)*100
            print(f"{nominee2} has won the election with {percent} %.")
            # to break the loop and exit the program.
            break
```

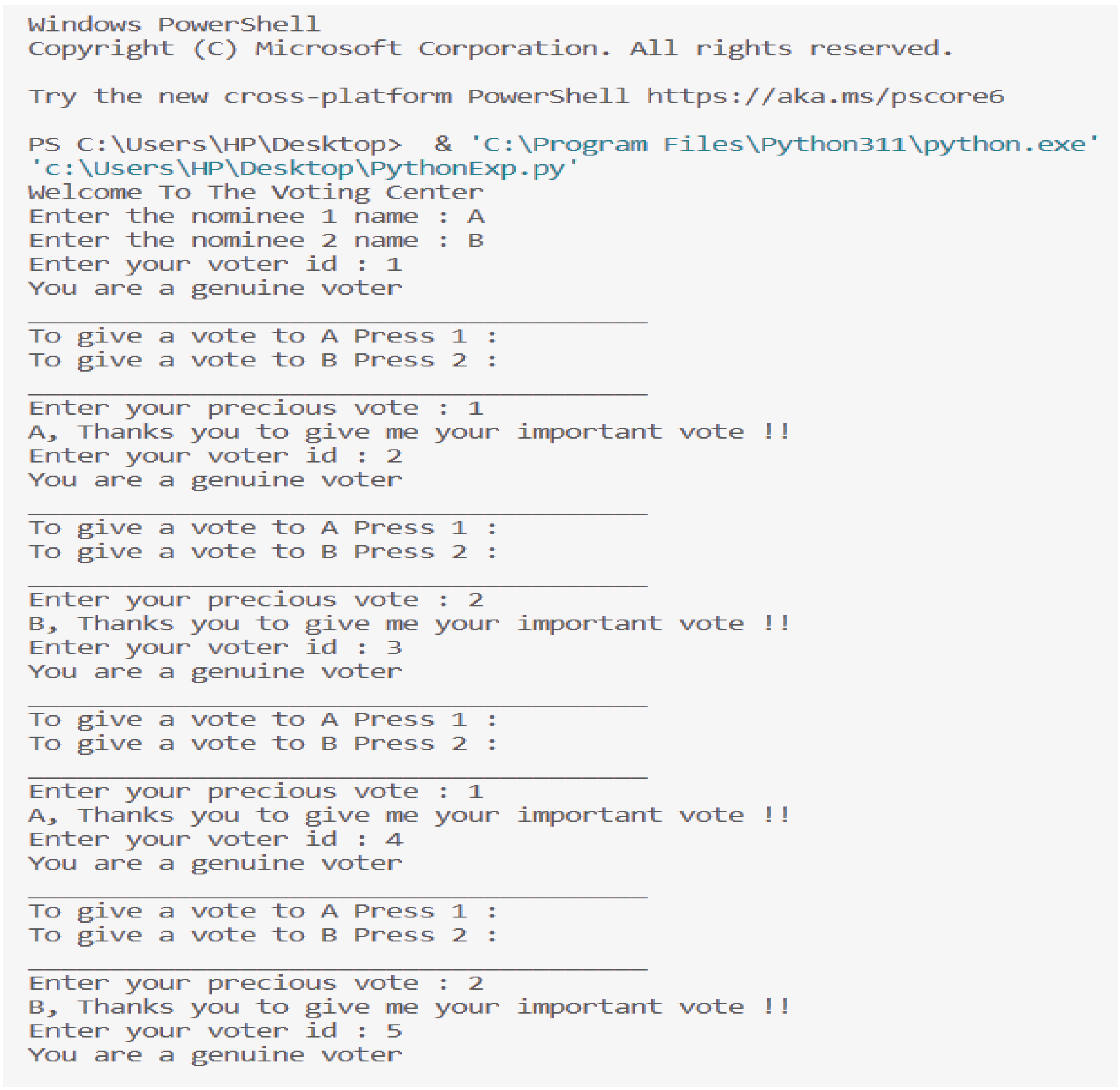


```python
        # if they got equal votes.
        else:
            print("Both have got equal number of votes !! Heigher authority will decide the final result, Thankyou !!")
            # to break the loop and exit the program.
            break

    # For taking voter id
    voterIdDetection = int(input("Enter your voter id : "))
    # for detecting fake voters vs real voters.
    if voterIdDetection in voterId:
        print("You are a genuine voter ")
        # we will remove voterId so that same voter can't vote again.
        voterId.remove(voterIdDetection)
        print("_______________________________________")
        print(f"To give a vote to {nominee1} Press 1 : ")
        print(f"To give a vote to {nominee2} Press 2 : ")
        print("_______________________________________")
```

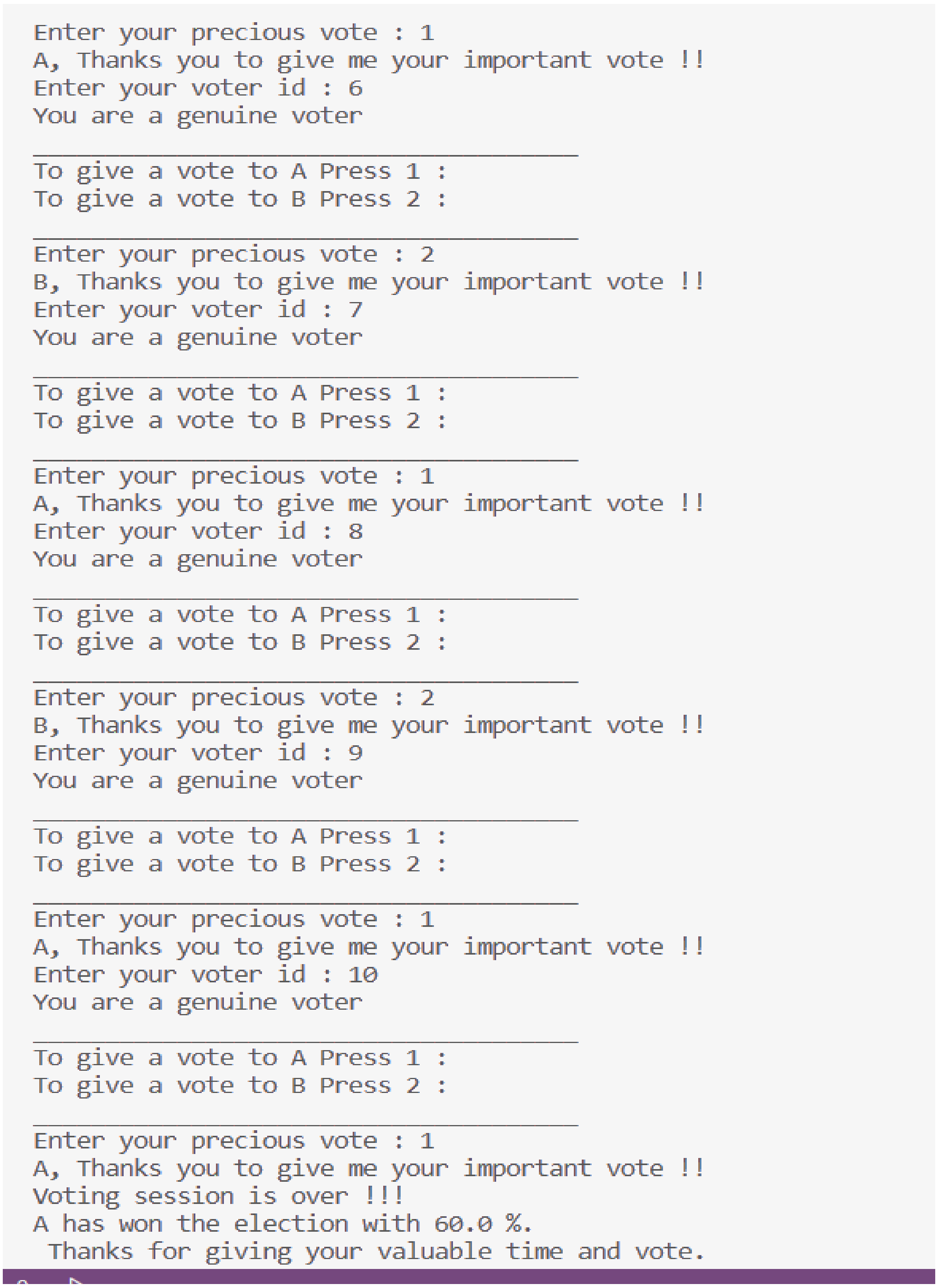


```python
        # checking voter votes whom.
        vote = int(input("Enter your precious vote : "))
        if vote == 1:
            nominee1Votes += 1
            print(f"{nominee1}, Thanks you to give me your important vote !! ")
        elif vote == 2:
            nominee2Votes += 1
            print(f"{nominee2}, Thanks you to give me your important vote !! ")
        elif vote > 2:
            print("Check your presed key is it right or wrong !!")
        else:
            print("You are not a genuine voter OR You have already voted ")
    else:
        print("You are not a genuine voter OR You have entered wrong Voter ID , else you already give your precious vote.")

# Goodbye massage.
print(" Thanks for giving your valuable time and vote.")
```

Output :



```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS C:\Users\HP\Desktop>  & 'C:\Program Files\Python311\python.exe'
'c:\Users\HP\Desktop\PythonExp.py'
Welcome To The Voting Center
Enter the nominee 1 name : A
Enter the nominee 2 name : B
Enter your voter id : 1
You are a genuine voter
______________________________________
To give a vote to A Press 1 :
To give a vote to B Press 2 :
______________________________________
Enter your precious vote : 1
A, Thanks you to give me your important vote !!
Enter your voter id : 2
You are a genuine voter
______________________________________
To give a vote to A Press 1 :
To give a vote to B Press 2 :
______________________________________
Enter your precious vote : 2
B, Thanks you to give me your important vote !!
Enter your voter id : 3
You are a genuine voter
______________________________________
To give a vote to A Press 1 :
To give a vote to B Press 2 :
______________________________________
Enter your precious vote : 1
A, Thanks you to give me your important vote !!
Enter your voter id : 4
You are a genuine voter
______________________________________
To give a vote to A Press 1 :
To give a vote to B Press 2 :
______________________________________
Enter your precious vote : 2
B, Thanks you to give me your important vote !!
Enter your voter id : 5
You are a genuine voter
```



```
Enter your precious vote : 1
A, Thanks you to give me your important vote !!
Enter your voter id : 6
You are a genuine voter
______________________________________
To give a vote to A Press 1 :
To give a vote to B Press 2 :
______________________________________
Enter your precious vote : 2
B, Thanks you to give me your important vote !!
Enter your voter id : 7
You are a genuine voter
______________________________________
To give a vote to A Press 1 :
To give a vote to B Press 2 :
______________________________________
Enter your precious vote : 1
A, Thanks you to give me your important vote !!
Enter your voter id : 8
You are a genuine voter
______________________________________
To give a vote to A Press 1 :
To give a vote to B Press 2 :
______________________________________
Enter your precious vote : 2
B, Thanks you to give me your important vote !!
Enter your voter id : 9
You are a genuine voter
______________________________________
To give a vote to A Press 1 :
To give a vote to B Press 2 :
______________________________________
Enter your precious vote : 1
A, Thanks you to give me your important vote !!
Enter your voter id : 10
You are a genuine voter
______________________________________
To give a vote to A Press 1 :
To give a vote to B Press 2 :
______________________________________
Enter your precious vote : 1
A, Thanks you to give me your important vote !!
Voting session is over !!!
A has won the election with 60.0 %.
 Thanks for giving your valuable time and vote.
```