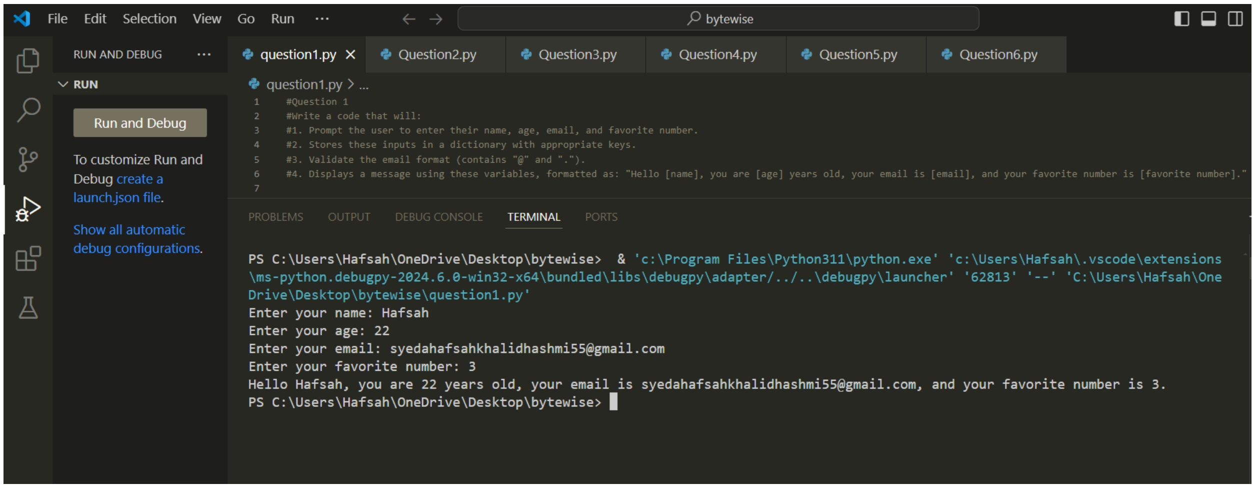
Week 2 (Bytewise fellowship)

Q1: Output:



```python
#Question 1
#Write a code that will:
#1. Prompt the user to enter their name, age, email, and favorite number.
#2. Stores these inputs in a dictionary with appropriate keys.
#3. Validate the email format (contains "@" and ".").
#4. Displays a message using these variables, formatted as: "Hello [name], you are [age] years old, your email is [email], and your favorite number is [favorite number]."
```

```
PS C:\Users\Hafsah\OneDrive\Desktop\bytewise>  & 'c:\Program Files\Python311\python.exe' 'c:\Users\Hafsah\.vscode\extensions
\ms-python.debugpy-2024.6.0-win32-x64\bundled\libs\debugpy\adapter/../..\debugpy\launcher' '62813' '--' 'C:\Users\Hafsah\One
Drive\Desktop\bytewise\question1.py'
Enter your name: Hafsah
Enter your age: 22
Enter your email: syedahafsahkhalidhashmi55@gmail.com
Enter your favorite number: 3
Hello Hafsah, you are 22 years old, your email is syedahafsahkhalidhashmi55@gmail.com, and your favorite number is 3.
PS C:\Users\Hafsah\OneDrive\Desktop\bytewise> 
```

Components Used:

Import:
import re: I import the `re` module, which provides support for working with regular expressions in Python.

Function:
get_user_input(): I define a function get_user_input to collect user information (name, age, email, favorite number) and validate the email format.

Variable:
user_info: I use a dictionary user_info to store the collected user information (name, age, email, favorite number).

Dictionary:
user_info: I store user information in this dictionary, where keys are strings ('name', 'age', 'email', 'favorite_number').

Input:
input("Enter your name: "): I prompt the user to enter their name.
input("Enter your age: "): I prompt the user to enter their age.
input("Enter your email: "): I prompt the user to enter their email address.
input("Enter your favorite number: "): I prompt the user to enter their favorite number.

Function Definition (Nested):
validate_email(email): I define a nested function validate_email within get_user_input to check if an email address matches a specified pattern using regular expressions.

Regular Expression:
pattern = r"[^@]+@[^@]+\.[^@]+": I use this regular expression pattern to validate the format of the email address entered by the user.

Loop:
while not validate_email(user_info['email']):: I use a while loop to repeatedly prompt the user to enter a valid email address until it matches the specified pattern.

Print Statement:

print("Invalid email format. Please try again."): I print an error message if the email address entered by the user does not match the expected format.

Display Function:
display_message(user_info): I define a function display_message to print a formatted message with the collected user information.

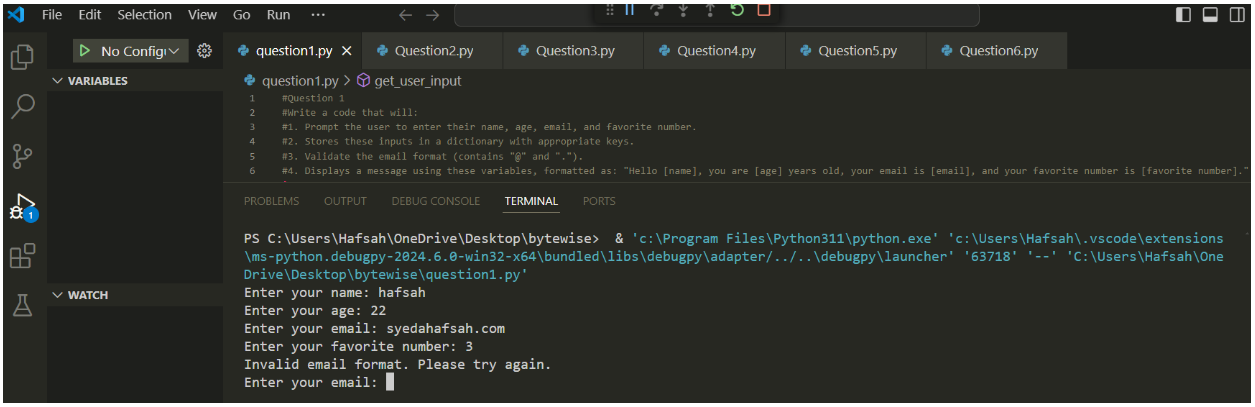
Print Formatting:
print(f"Hello {user_info['name']}, you are {user_info['age']} years old, your email is {user_info['email']}, and your favorite number is {user_info['favorite_number']}."): I use an f-string to format and print a greeting message using the collected user information.

Main Function:
main(): I define the main function to orchestrate the flow of the program by calling get_user_input() to collect user information, validating it, and then displaying the formatted message using display_message().

Execution Check:
if __name__ == "__main__":: I use this conditional statement to ensure that the main function (main()) runs only when the script is executed directly, not when it is imported as a module.

Summary:
In this code, I create a program that prompts the user to enter their name, age, email address, and favorite number. The program validates the email address format using a regular expression. If the email format is invalid, it prompts the user to re-enter it until it matches the expected pattern. After collecting the user information, it displays a formatted message greeting the user with their entered details. The program ensures structured user interaction, validation of input, and proper output formatting.

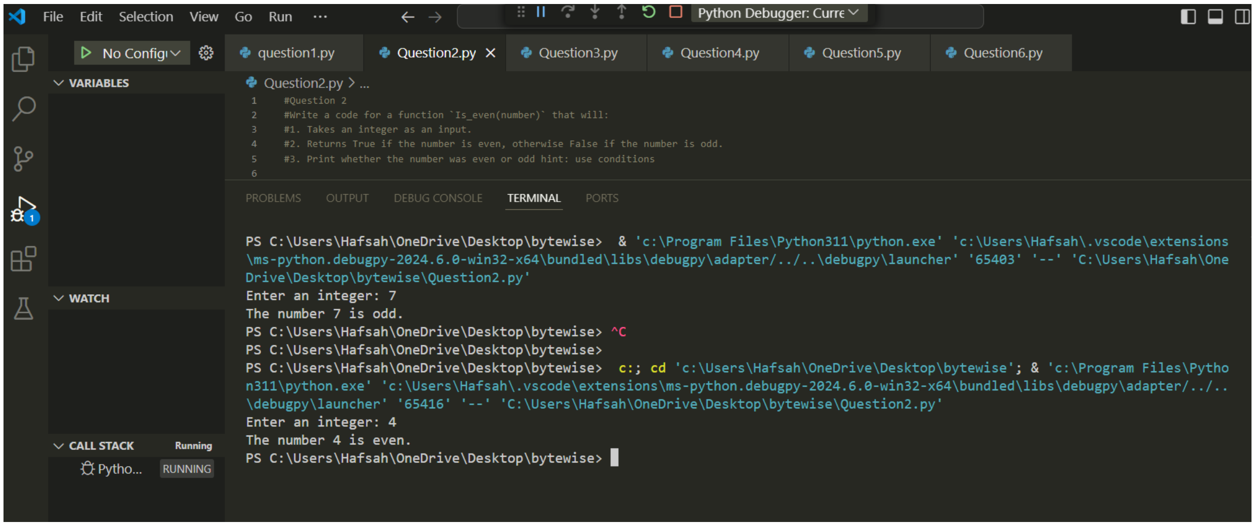
In Case of an invalid email format, it will display the output as:



```
#Question 1
#Write a code that will:
#1. Prompt the user to enter their name, age, email, and favorite number.
#2. Stores these inputs in a dictionary with appropriate keys.
#3. Validate the email format (contains "@" and ".").
#4. Displays a message using these variables, formatted as: "Hello [name], you are [age] years old, your email is [email], and your favorite number is [favorite number]."
```

```
PS C:\Users\Hafsah\OneDrive\Desktop\bytewise>  & 'c:\Program Files\Python311\python.exe' 'c:\Users\Hafsah\.vscode\extensions
\ms-python.debugpy-2024.6.0-win32-x64\bundled\libs\debugpy\adapter/../..\debugpy\launcher' '63718' '--' 'C:\Users\Hafsah\One
Drive\Desktop\bytewise\question1.py'
Enter your name: hafsah
Enter your age: 22
Enter your email: syedahafsah.com
Enter your favorite number: 3
Invalid email format. Please try again.
Enter your email: 
```

Q2 : Output



```python
#Question 2
#Write a code for a function `Is_even(number)` that will:
#1. Takes an integer as an input.
#2. Returns True if the number is even, otherwise False if the number is odd.
#3. Print whether the number was even or odd hint: use conditions

```

```
PS C:\Users\Hafsah\OneDrive\Desktop\bytewise>  & 'c:\Program Files\Python311\python.exe' 'c:\Users\Hafsah\.vscode\extensions
\ms-python.debugpy-2024.6.0-win32-x64\bundled\libs\debugpy\adapter/../..\debugpy\launcher' '65403' '--' 'C:\Users\Hafsah\One
Drive\Desktop\bytewise\Question2.py'
Enter an integer: 7
The number 7 is odd.
PS C:\Users\Hafsah\OneDrive\Desktop\bytewise> ^C
PS C:\Users\Hafsah\OneDrive\Desktop\bytewise>
PS C:\Users\Hafsah\OneDrive\Desktop\bytewise>  c:; cd 'c:\Users\Hafsah\OneDrive\Desktop\bytewise'; & 'c:\Program Files\Pytho
n311\python.exe' 'c:\Users\Hafsah\.vscode\extensions\ms-python.debugpy-2024.6.0-win32-x64\bundled\libs\debugpy\adapter/../..
\debugpy\launcher' '65416' '--' 'C:\Users\Hafsah\OneDrive\Desktop\bytewise\Question2.py'
Enter an integer: 4
The number 4 is even.
PS C:\Users\Hafsah\OneDrive\Desktop\bytewise>
```

Components Used:

Function:
Is_even(number): I use this function to check if a number is even or odd.

Condition:
if number % 2 == 0: I use this conditional statement to check if the number is even.

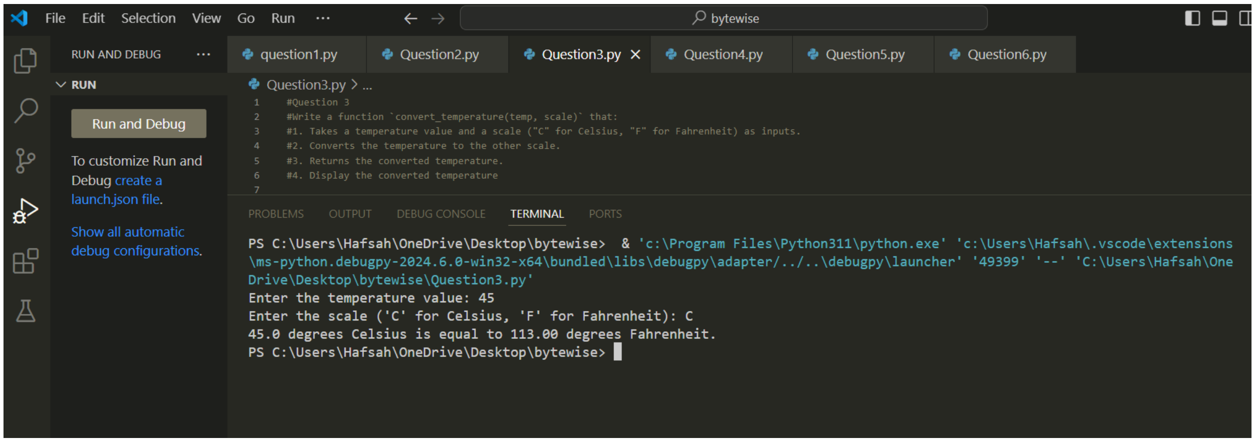
Variable:
number: I use this variable to store the user-provided integer.
result: I use this variable to store the boolean result from the Is_even function.

Input:
input("Enter an integer: "): I use this function to take user input.

Print Statements:
print(f"The number {number} is even."): I use this to print if the number is even.
print(f"The number {number} is odd."): I use this to print if the number is odd.

Summary:
In this code, I define a function Is_even to check whether a number is even or odd using conditional statements. I take an integer input from the user, call the function to determine if the number is even or odd, and print the result. I use variables to store the user's input and the function's result.

Q3 (Output):



```python
#Question 3
#Write a function `convert_temperature(temp, scale)` that:
#1. Takes a temperature value and a scale ("C" for Celsius, "F" for Fahrenheit) as inputs.
#2. Converts the temperature to the other scale.
#3. Returns the converted temperature.
#4. Display the converted temperature

```

```
PS C:\Users\Hafsah\OneDrive\Desktop\bytewise>  & 'c:\Program Files\Python311\python.exe' 'c:\Users\Hafsah\.vscode\extensions
\ms-python.debugpy-2024.6.0-win32-x64\bundled\libs\debugpy\adapter/../..\debugpy\launcher' '49399' '--' 'C:\Users\Hafsah\One
Drive\Desktop\bytewise\Question3.py'
Enter the temperature value: 45
Enter the scale ('C' for Celsius, 'F' for Fahrenheit): C
45.0 degrees Celsius is equal to 113.00 degrees Fahrenheit.
PS C:\Users\Hafsah\OneDrive\Desktop\bytewise> 
```

Components Used:

Function:
convert_temperature(temp, scale): I use this function to convert temperatures between Celsius and Fahrenheit.

Condition:
if scale == "C" and elif scale == "F": I use these conditional statements to check the temperature scale.

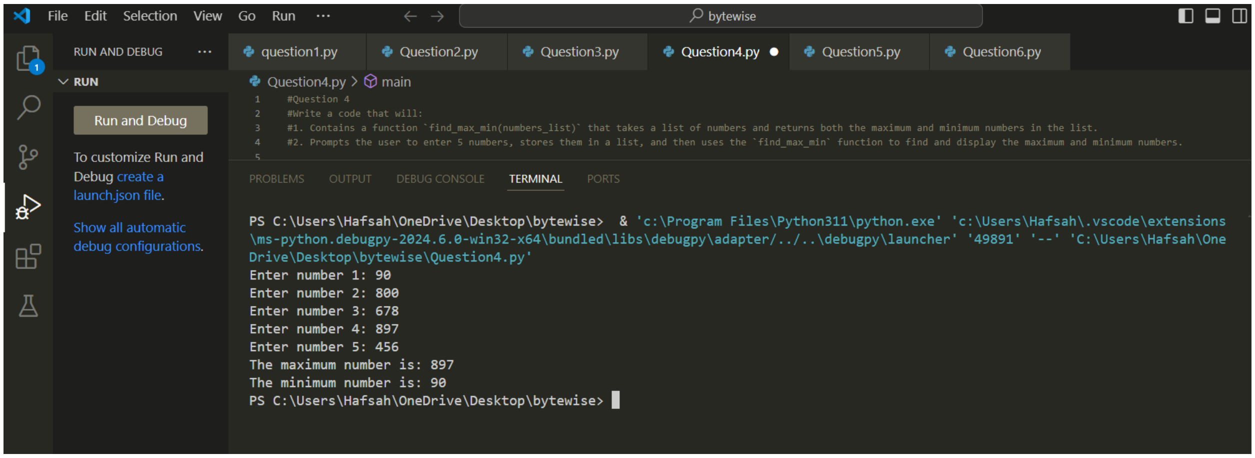
Variable:
temp: I use this variable to store the user-provided temperature.
scale: I use this variable to store the user-provided scale.
converted_temp: I use this variable to store the converted temperature.

Input:
input("Enter the temperature value: "): I use this function to take user input for the temperature value.
input("Enter the scale ('C' for Celsius, 'F' for Fahrenheit): ").upper(): I use this function to take user input for the temperature scale.

Print Statements:
I use print statements to display the converted temperature or an error message.

Summary:
In this code, I define a function convert_temperature to convert temperatures between Celsius and Fahrenheit using conditional statements. I take user input for the temperature value and scale, call the function to perform the conversion, and print the result. I use variables to store the user's input and the converted temperature. If the user enters an invalid scale, I print an error message.

Question 4:



```
#Question 4
#Write a code that will:
#1. Contains a function `find_max_min(numbers_list)` that takes a list of numbers and returns both the maximum and minimum numbers in the list.
#2. Prompts the user to enter 5 numbers, stores them in a list, and then uses the `find_max_min` function to find and display the maximum and minimum numbers.
```

```
PS C:\Users\Hafsah\OneDrive\Desktop\bytewise>  & 'c:\Program Files\Python311\python.exe' 'c:\Users\Hafsah\.vscode\extensions
\ms-python.debugpy-2024.6.0-win32-x64\bundled\libs\debugpy\adapter/../..\debugpy\launcher' '49891' '--' 'C:\Users\Hafsah\One
Drive\Desktop\bytewise\Question4.py'
Enter number 1: 90
Enter number 2: 800
Enter number 3: 678
Enter number 4: 897
Enter number 5: 456
The maximum number is: 897
The minimum number is: 90
PS C:\Users\Hafsah\OneDrive\Desktop\bytewise> 
```

Components Used:

Function:
find_max_min(numbers_list): A function to find the maximum and minimum numbers in a list.
main(): The main function to handle user input and display results.

Variable:
numbers_list: A list to store the user-provided numbers.
max_number: A variable to store the maximum number in the list.
min_number: A variable to store the minimum number in the list.
number: A variable to store each number entered by the user during input.

List:
numbers_list: A list that stores the five numbers entered by the user.

Input:
input(f"Enter number {i+1}: "): A function to prompt the user to enter numbers.

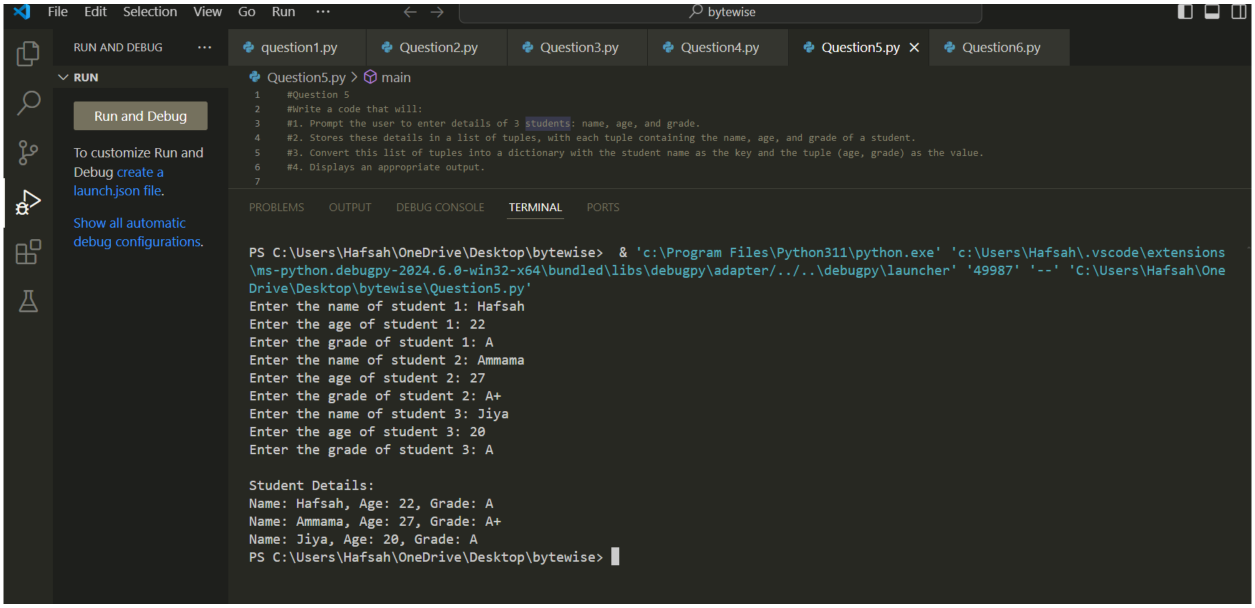
Loop:
for i in range(5): A loop to iterate five times for user input.

Print Statements:
print(f"The maximum number is: {max_number}"): A statement to display the maximum number.
print(f"The minimum number is: {min_number}"): A statement to display the minimum number.

Built-in Functions:
max(): A function to find the maximum number in the list.
min(): A function to find the minimum number in the list.

Summary:
In this code, I define a function find_max_min to find the maximum and minimum numbers in a list. The main function handles user input, prompting the user to enter five numbers, which are stored in a list. The find_max_min function is then called with this list to determine the maximum and minimum numbers. These values are printed to the console. The if __name__ == "__main__": block ensures that the main function is called when the script is run directly.

Question 5:



```python
#Question 5
#Write a code that will:
#1. Prompt the user to enter details of 3 students: name, age, and grade.
#2. Stores these details in a list of tuples, with each tuple containing the name, age, and grade of a student.
#3. Convert this list of tuples into a dictionary with the student name as the key and the tuple (age, grade) as the value.
#4. Displays an appropriate output.
```

```
PS C:\Users\Hafsah\OneDrive\Desktop\bytewise>  & 'c:\Program Files\Python311\python.exe' 'c:\Users\Hafsah\.vscode\extensions\ms-python.debugpy-2024.6.0-win32-x64\bundled\libs\debugpy\adapter/../..\debugpy\launcher' '49987' '--' 'C:\Users\Hafsah\OneDrive\Desktop\bytewise\Question5.py'
Enter the name of student 1: Hafsah
Enter the age of student 1: 22
Enter the grade of student 1: A
Enter the name of student 2: Ammama
Enter the age of student 2: 27
Enter the grade of student 2: A+
Enter the name of student 3: Jiya
Enter the age of student 3: 20
Enter the grade of student 3: A

Student Details:
Name: Hafsah, Age: 22, Grade: A
Name: Ammama, Age: 27, Grade: A+
Name: Jiya, Age: 20, Grade: A
PS C:\Users\Hafsah\OneDrive\Desktop\bytewise>
```

Components Used:

Function:
main(): I define a main function to handle the process of collecting and displaying student details.
Variable:
students_list: I use a list called students_list to store tuples containing details of three students.
students_dict: I create a dictionary students_dict to store student details with names as keys and age/grade as values.
name, age, grade: I use these variables to temporarily store each student's name, age, and grade during input.

List:
students_list: I use this list to store tuples, where each tuple represents the details (name, age, grade) of a student.

Tuple:
(name, age, grade): I use tuples to group together the name, age, and grade of each student before converting them into a dictionary.

Dictionary:
students_dict: I convert students_list into a dictionary where the student's name serves as the key, and the corresponding value is a tuple containing their age and grade.

Input:
input(f"Enter the name of student {i+1}: "): I prompt the user to input the name of each student.
int(input(f"Enter the age of student {i+1}: ")): I prompt the user to input the age of each student.
input(f"Enter the grade of student {i+1}: "): I prompt the user to input the grade of each student.

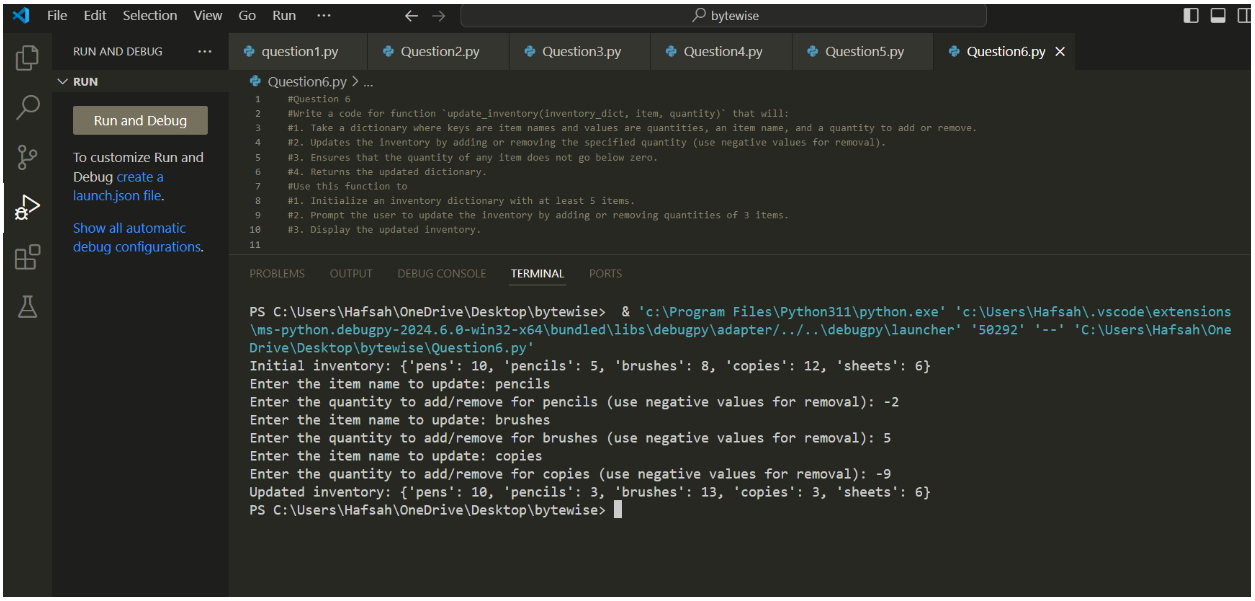
Loop:
for i in range(3): I use a loop to iterate three times, collecting details for three students.

Print Statements:
print("\nStudent Details:"): I print a header indicating the start of the student details section.
print(f"Name: {name}, Age: {details[0]}, Grade: {details[1]}"): I print each student's name, age, and grade after converting them into a dictionary format.

Dictionary Details:
{student[0]: (student[1], student[2]) for student in students_list}: I converted students_list into a dictionary using a dictionary comprehension, where each tuple in students_list becomes an entry in students_dict.

Summary:
In this code, I use a main function to interact with the user and manage student details. I collect information for three students (name, age, and grade) through input prompts, store these details in tuples within a list (students_list), convert this list into a dictionary (students_dict) where each student's name is paired with their age and grade, and finally, I print each student's details formatted neatly. The script executes main() when run directly due to the if __name__ == "__main__": condition.

Question 6:



```
#Question 6
#Write a code for function `update_inventory(inventory_dict, item, quantity)` that will:
#1. Take a dictionary where keys are item names and values are quantities, an item name, and a quantity to add or remove.
#2. Updates the inventory by adding or removing the specified quantity (use negative values for removal).
#3. Ensures that the quantity of any item does not go below zero.
#4. Returns the updated dictionary.
#Use this function to
#1. Initialize an inventory dictionary with at least 5 items.
#2. Prompt the user to update the inventory by adding or removing quantities of 3 items.
#3. Display the updated inventory.
```

```
PS C:\Users\Hafsah\OneDrive\Desktop\bytewise>  & 'c:\Program Files\Python311\python.exe' 'c:\Users\Hafsah\.vscode\extensions
\ms-python.debugpy-2024.6.0-win32-x64\bundled\libs\debugpy\adapter/../..\debugpy\launcher' '50292' '--' 'C:\Users\Hafsah\One
Drive\Desktop\bytewise\Question6.py'
Initial inventory: {'pens': 10, 'pencils': 5, 'brushes': 8, 'copies': 12, 'sheets': 6}
Enter the item name to update: pencils
Enter the quantity to add/remove for pencils (use negative values for removal): -2
Enter the item name to update: brushes
Enter the quantity to add/remove for brushes (use negative values for removal): 5
Enter the item name to update: copies
Enter the quantity to add/remove for copies (use negative values for removal): -9
Updated inventory: {'pens': 10, 'pencils': 3, 'brushes': 13, 'copies': 3, 'sheets': 6}
PS C:\Users\Hafsah\OneDrive\Desktop\bytewise>
```

Components Used:

Function:
update_inventory(inventory_dict, item, quantity): I define a function update_inventory that modifies a dictionary representing inventory by adding or removing items with specified quantities.

Variable:
inventory: I use this dictionary to store the initial inventory items and their quantities.
item: I use this variable to temporarily store the name of each item the user wants to update.
quantity: I use this variable to temporarily store the quantity of an item the user wants to add or remove.

Dictionary:
inventory_dict: I use this dictionary to represent the inventory, where keys are item names (strings) and values are quantities (integers).

Input:
input("Enter the item name to update: "): I prompt the user to input the name of an item they want to update in the inventory.
int(input(f"Enter the quantity to add/remove for {item} (use negative values for removal): ")): I prompt the user to input the quantity to add or remove for a specified item.

Loop:

for _ in range(3): I use a loop to iterate three times, allowing the user to update the inventory for three different items.

Conditional Statement:
if item in inventory_dict:: I use this conditional to check if the item already exists in the inventory dictionary.
else:: I use this else clause to handle cases where the item doesn't exist in the inventory dictionary.

Function Call:
update_inventory(inventory, item, quantity): I call the update_inventory function to update the inventory dictionary based on the user's input.

Print Statements:
print("Initial inventory:", inventory): I print the initial state of the inventory before any updates.
print("Updated inventory:", inventory): I print the updated state of the inventory after the user has made their updates.

Summary:
In this code, I define a function update_inventory to manage updates to an inventory represented by a dictionary. I initialize an inventory dictionary with initial quantities for several items. Then, I prompt the user to update the inventory for three items, allowing them to add or remove quantities of each item. The update_inventory function ensures that item quantities are updated correctly, handling additions, removals (with non-negative results), and initial additions of new items. Finally, I display the initial and updated inventory states to the user. The script runs the main() function when executed directly due to the if __name__ == "__main__": condition.