Wrocław University of Science and Technology

PYTHON
LABORATORY REPORT

Faculty of Electronics, Photonics and MicrosystemsPYTHON LABORATORY

Theme of class: LOOPS , CONDITIONAL STATEMENTS and FUNCTIONS

Student: Hayrettin Aycetin (276807)
Date of class: 23.10.2023 15:15-16:55
Group No:3
Submition Date:

Lab assistant: Aleksander Kubeczek, Alicja Kwaśny GRADE:

Table of contents

Introduction *1*
Theory *2*
Code and Comments *3*
Conclusion *4*

1-Introduction

In these exercises, we'll explore essential Python concepts like loops, conditionals, and functions. These are fundamental tools in programming that we must understand to achieve good results. We have six tasks to complete to help us learn and practice these concepts.

2-Theory

Python Loops

Python has two primitive loop commands:

- `while` loops
- `for` loops

While Loops

- With the `while` loop we can execute a set of statements as long as a condition is true.

For Loops

- A `for` loop is used for iterating over a sequence (that is either a list, a tuple, a dictionary, a set, or a string).

Break,Continue and Else Statements

- With the `break` statement we can stop the loop even if the while condition is true.
- With the `continue` statement we can stop the current iteration, and continue with the next
- With the `else` statement we can run a block of code once when the condition no longer is true but as our professor said I think it is not convenient to use else in while loop.

Range()

- To loop through a set of code a specified number of times, we can use the `range()` function.

Syntax = for x in range(start,end(exclusive),iteration)

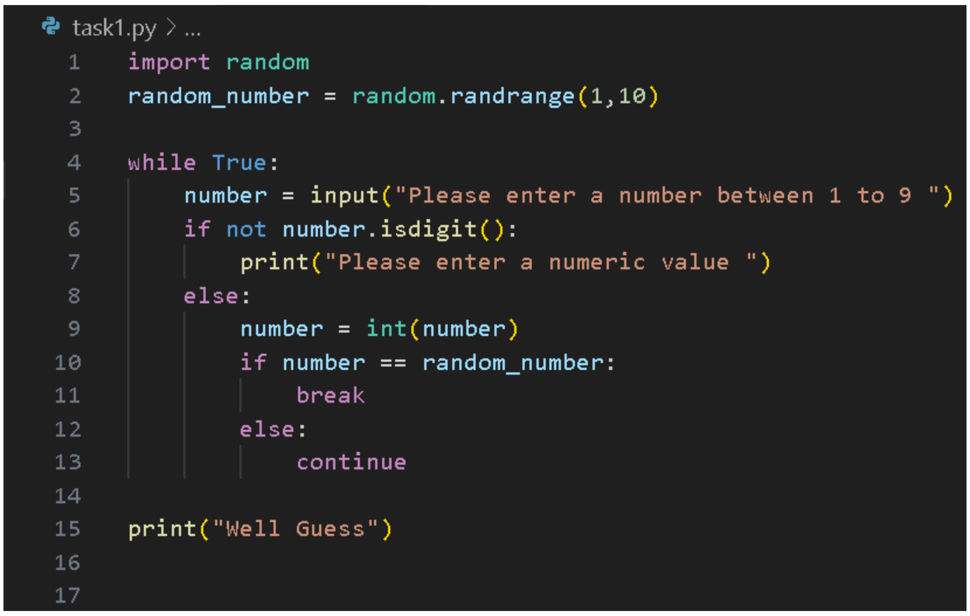
W3Schools. (2023). Python While Loops. https://www.w3schools.com/python/python_while_loops.asp
W3Schools. (2023). Python For Loops. https://www.w3schools.com/python/python_for_loops.asp

3- Code and Comments

Task 1

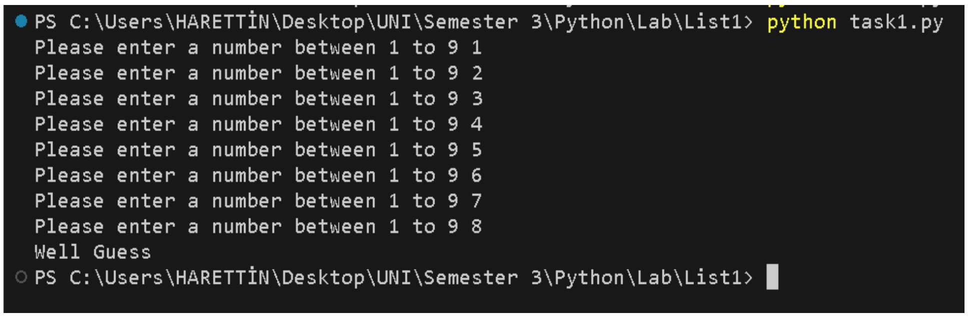
Write a program to guess a number between 1 and 9 which is randomly generated. User enters a number in the range. If the user guesses wrong then the prompt appears again until the guess is correct, then user gets a "Well guessed!" message, and the program exits.

Code:



task1.py > ...

```python
import random
random_number = random.randrange(1,10)

while True:
    number = input("Please enter a number between 1 to 9 ")
    if not number.isdigit():
        print("Please enter a numeric value ")
    else:
        number = int(number)
        if number == random_number:
            break
        else:
            continue

print("Well Guess")


```



```
PS C:\Users\HARETTİN\Desktop\UNI\Semester 3\Python\Lab\List1> python task1.py
Please enter a number between 1 to 9 1
Please enter a number between 1 to 9 2
Please enter a number between 1 to 9 3
Please enter a number between 1 to 9 4
Please enter a number between 1 to 9 5
Please enter a number between 1 to 9 6
Please enter a number between 1 to 9 7
Please enter a number between 1 to 9 8
Well Guess
PS C:\Users\HARETTİN\Desktop\UNI\Semester 3\Python\Lab\List1>
```

Comments:

First, I imported the random module, which allowed me to generate random numbers. I stored my random number in the random_number variable. I used the random.randrange() function with the range between 1 and 10 (exclusive).

Inside the loop, the user is prompted to input a number between 1 and 9. Then, the code checks if the user's input is numeric. If it's numeric, we convert the string input to an integer.

I used another if statement to check whether the user guessed correctly or not. If the user's guess is correct, I used break to exit the while loop. If not, we continue to ask the same question until we get the correct answer."

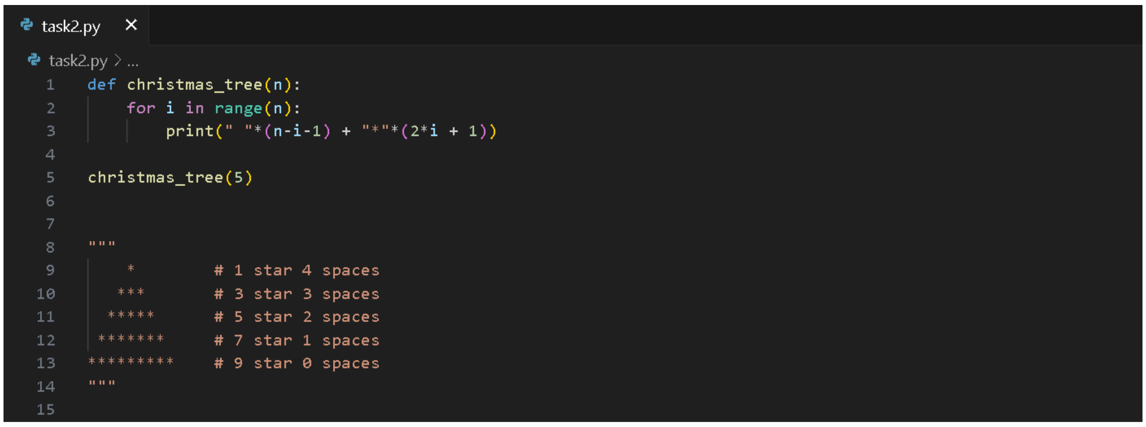
This example was good practice for me in the usage of while loops and if statements.

Task 2

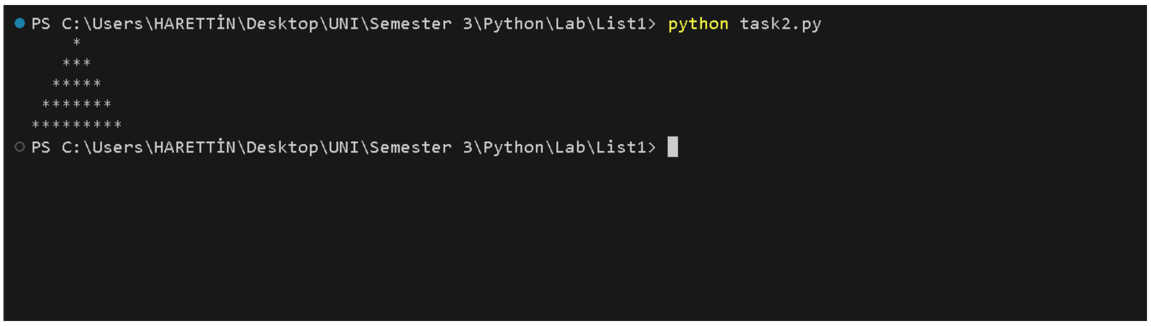
Write a program that displays a tree made of stars using for and/or while loops. The tree should look similar to this:

```
    *
   ***
  *****
 *******
*********
```

Code:



```
task2.py
def christmas_tree(n):
    for i in range(n):
        print(" "*(n-i-1) + "*"*(2*i + 1))

christmas_tree(5)


"""
    *          # 1 star 4 spaces
   ***         # 3 star 3 spaces
  *****        # 5 star 2 spaces
 *******       # 7 star 1 spaces
*********      # 9 star 0 spaces
"""

```



```
● PS C:\Users\HARETTİN\Desktop\UNI\Semester 3\Python\Lab\List1> python task2.py
     *
    ***
   *****
  *******
 *********
○ PS C:\Users\HARETTİN\Desktop\UNI\Semester 3\Python\Lab\List1> █
```

Comments:
First, I defined a function called christmas_tree that takes an integer n as its argument and then prints a Christmas tree with a specified number of levels. I used a for loop to create and display the tree.

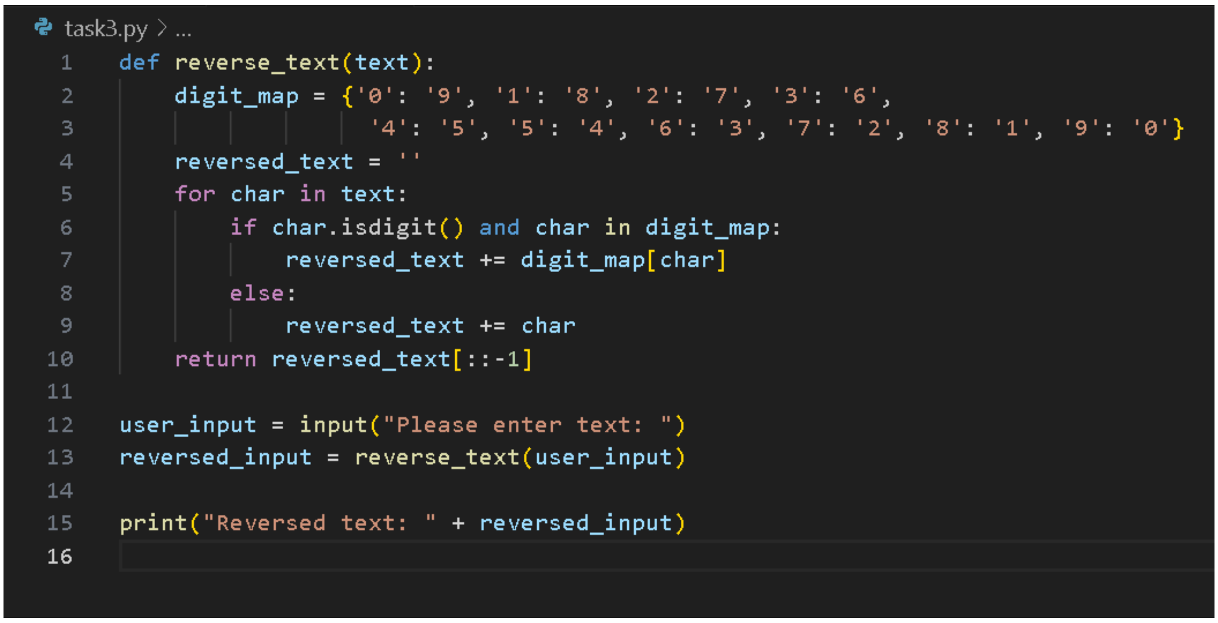
The loop for i in range(n) means that we need to iterate from 0 to n - 1, and then we use the print function to print stars layer by layer. I discovered the equation inside the print brackets when I was trying to find the relationship between stars and spaces.

This topic was a bit challenging, but it was fun. We did a similar task in C++ during my first semester, but I realized that in Python, it's much easier and shorter."

Task 3

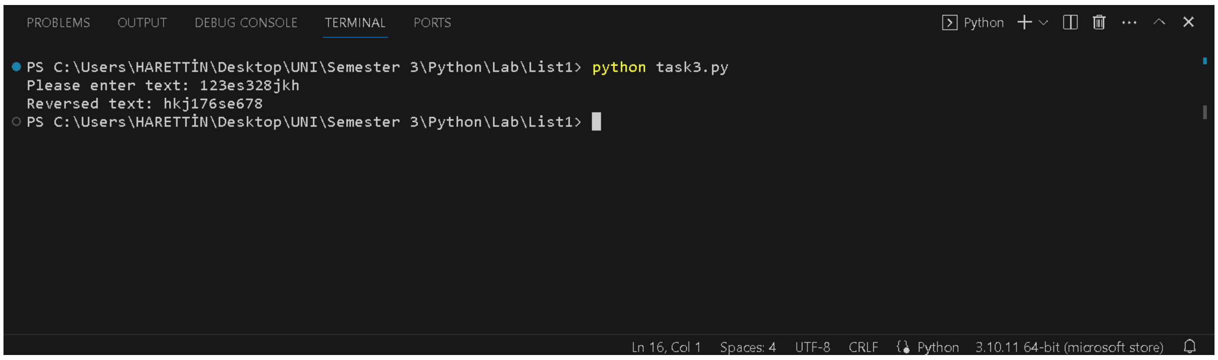
Write a program that takes input from the user and reverses it (prints backwards). It also should detect and reverse any digits given in the input like so: 0 → 9, 1 → 8, etc. (Example: book1 → 9koob, 20lamp →pmal97)

Code:



task3.py > ...

```python
def reverse_text(text):
    digit_map = {'0': '9', '1': '8', '2': '7', '3': '6',
                 '4': '5', '5': '4', '6': '3', '7': '2', '8': '1', '9': '0'}
    reversed_text = ''
    for char in text:
        if char.isdigit() and char in digit_map:
            reversed_text += digit_map[char]
        else:
            reversed_text += char
    return reversed_text[::-1]

user_input = input("Please enter text: ")
reversed_input = reverse_text(user_input)

print("Reversed text: " + reversed_input)

```



```
PS C:\Users\HARETTİN\Desktop\UNI\Semester 3\Python\Lab\List1> python task3.py
Please enter text: 123es328jkh
Reversed text: hkj176se678
PS C:\Users\HARETTİN\Desktop\UNI\Semester 3\Python\Lab\List1>
```

Comments:

First, I've defined a function called reverse_text, which takes a text parameter. Within the reverse_text function, I've created a dictionary called digit_map. This dictionary maps each number to its reversed counterpart.

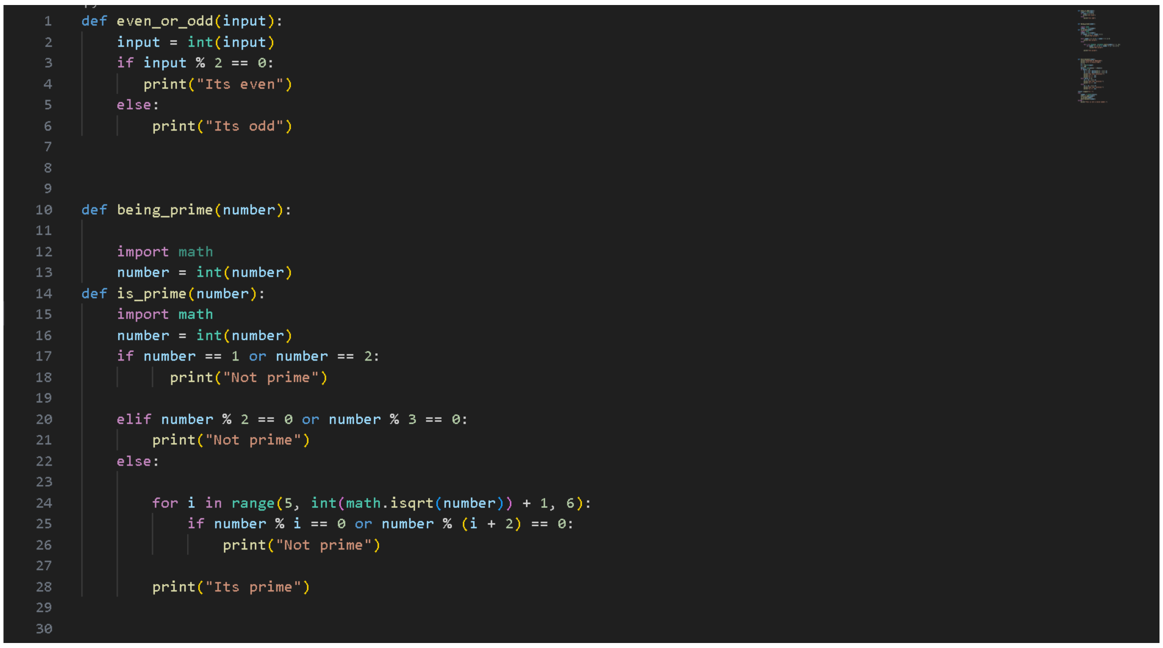
Next, I've initialized an empty string, reversed_text, which will store the reversed version of the input. Inside a for loop, I check whether each character is a digit. If it's a digit, I replace it with its reversed version using the digit_map dictionary. If it's not a digit, I simply append it to reversed_text. To fully reverse the text, I return reversed_text[::-1].

Finally, I print the reversed text. While this example may seem a bit complex, it's both educational and enjoyable. I'm certain there are alternative approaches to achieve the same result as well.

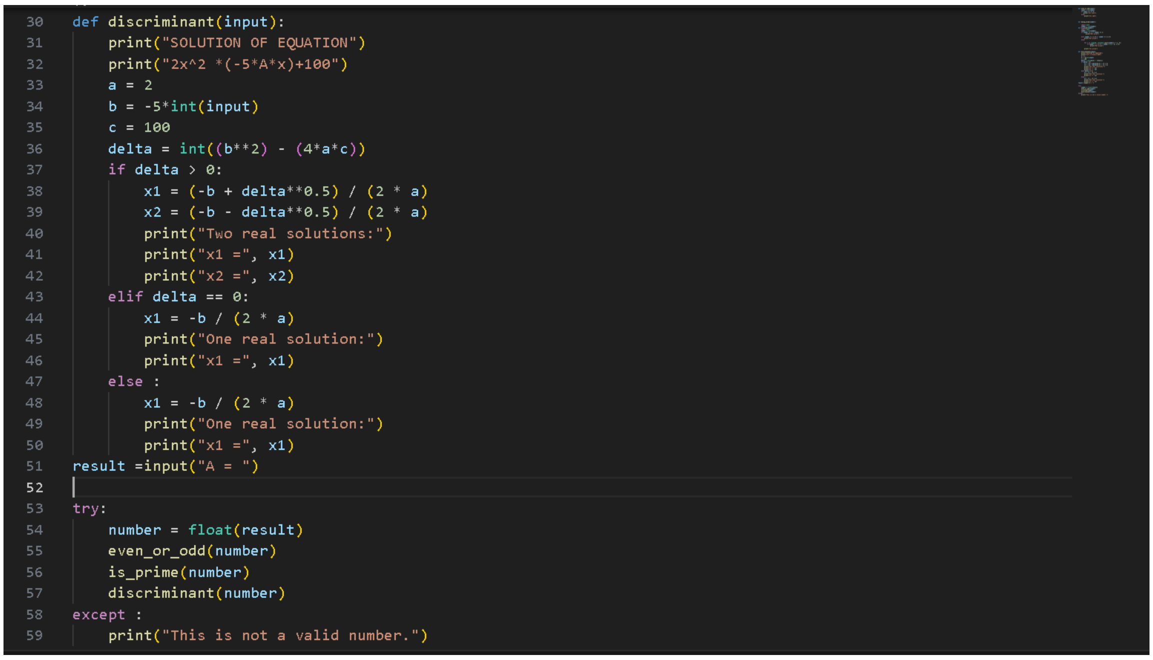
Task 4

Write a program that takes a number as an input from the user (you should check of the input is a valid natural number) and in separate functions checks:
a. If the number is even or odd
b. If the number is prime
c. Solves a quadratic equation 2x2 -5*(input)*x + 100 using discriminant (Δ). It also should consider the imaginary solutions case. The solution might be exact or approximate.

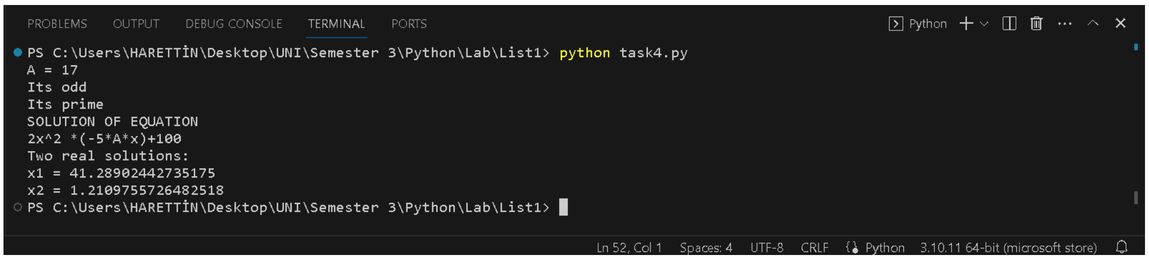
Code:



```python
def even_or_odd(input):
    input = int(input)
    if input % 2 == 0:
       print("Its even")
    else:
        print("Its odd")



def being_prime(number):

    import math
    number = int(number)
def is_prime(number):
    import math
    number = int(number)
    if number == 1 or number == 2:
          print("Not prime")

    elif number % 2 == 0 or number % 3 == 0:
        print("Not prime")
    else:

        for i in range(5, int(math.isqrt(number)) + 1, 6):
            if number % i == 0 or number % (i + 2) == 0:
                print("Not prime")

        print("Its prime")

```



```python
def discriminant(input):
    print("SOLUTION OF EQUATION")
    print("2x^2 *(-5*A*x)+100")
    a = 2
    b = -5*int(input)
    c = 100
    delta = int((b**2) - (4*a*c))
    if delta > 0:
        x1 = (-b + delta**0.5) / (2 * a)
        x2 = (-b - delta**0.5) / (2 * a)
        print("Two real solutions:")
        print("x1 =", x1)
        print("x2 =", x2)
    elif delta == 0:
        x1 = -b / (2 * a)
        print("One real solution:")
        print("x1 =", x1)
    else :
        x1 = -b / (2 * a)
        print("One real solution:")
        print("x1 =", x1)
result =input("A = ")

try:
    number = float(result)
    even_or_odd(number)
    is_prime(number)
    discriminant(number)
except :
    print("This is not a valid number.")
```



```
PS C:\Users\HARETTİN\Desktop\UNI\Semester 3\Python\Lab\List1> python task4.py
A = 17
Its odd
Its prime
SOLUTION OF EQUATION
2x^2 *(-5*A*x)+100
Two real solutions:
x1 = 41.28902442735175
x2 = 1.2109755726482518
PS C:\Users\HARETTİN\Desktop\UNI\Semester 3\Python\Lab\List1>
```

Comments:

First, I divided the operations into smaller functions to handle each specific task. I used conditional statements to determine whether a number is even or odd. To check for prime numbers, I leveraged the math library, conditional statements, and for loops. Initially, I handled cases where the input is 1, 2, or an even number.

For prime number checking, I implemented a loop from 5 to the square root of the number plus one, with a step size of 6. Within this loop, I used conditional statements to test if the number is divisible by i or i + 2. If it is, I printed "Not prime."

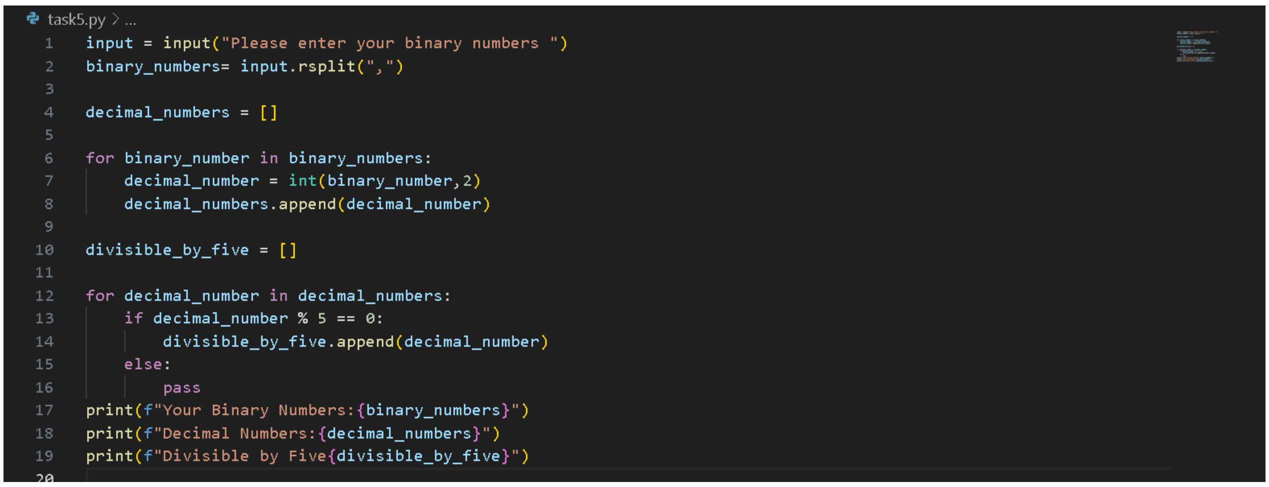
Regarding the discriminant, I separated the variables a, b, and c to compute the delta. I evaluated delta for different situations and returned its value to input A. To ensure the input's validity, I used a try-except method.

This task proved to be quite challenging, particularly in identifying prime numbers, but I managed to find some online assistance.

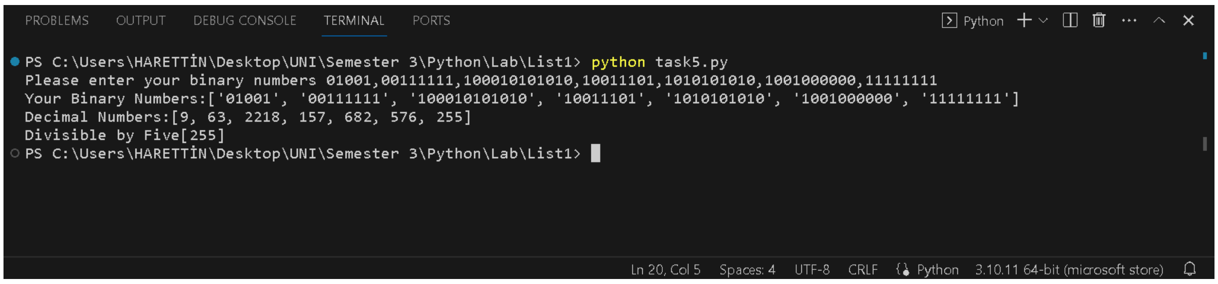
Task 5

Write a program which accepts a sequence of comma separated binary numbers as its input and inside a function prints the numbers that are divisible by 5 in a comma separated sequence.
Sample Data : 01001, 00111111, 100010101010, 10011101, 1010101010, 1001000000, 11111111

Code:



```python
input = input("Please enter your binary numbers ")
binary_numbers= input.rsplit(",")

decimal_numbers = []

for binary_number in binary_numbers:
    decimal_number = int(binary_number,2)
    decimal_numbers.append(decimal_number)

divisible_by_five = []

for decimal_number in decimal_numbers:
    if decimal_number % 5 == 0:
        divisible_by_five.append(decimal_number)
    else:
        pass
print(f"Your Binary Numbers:{binary_numbers}")
print(f"Decimal Numbers:{decimal_numbers}")
print(f"Divisible by Five{divisible_by_five}")
```



```
PS C:\Users\HARETTİN\Desktop\UNI\Semester 3\Python\Lab\List1> python task5.py
Please enter your binary numbers 01001,00111111,100010101010,10011101,1010101010,1001000000,11111111
Your Binary Numbers:['01001', '00111111', '100010101010', '10011101', '1010101010', '1001000000', '11111111']
Decimal Numbers:[9, 63, 2218, 157, 682, 576, 255]
Divisible by Five[255]
PS C:\Users\HARETTİN\Desktop\UNI\Semester 3\Python\Lab\List1>
```

Comments:

First, we receive a sample dataset as input and split it into binary numbers using the rsplit() function. Next, I initialize an empty list to store the decimal equivalents of these binary numbers. Using a for loop, I iterate through each binary number in the list and convert it to decimal using int(binary_number, 2), then append the result to the empty list.

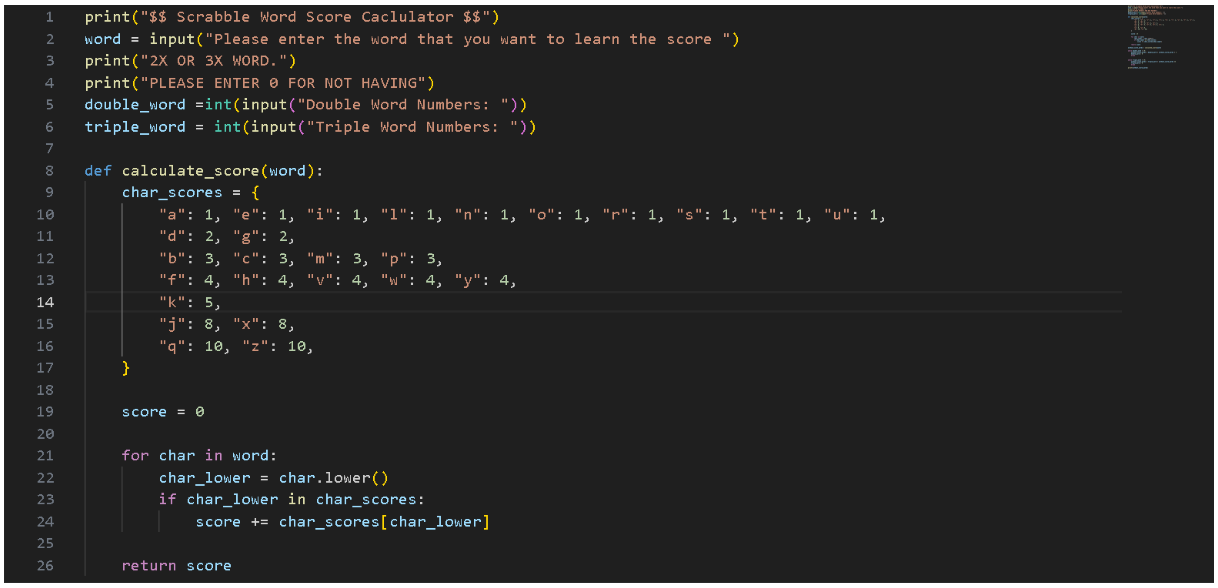
Subsequently, I create another empty list to store the decimal numbers that are divisible by 5. Within a for loop, I check each decimal number to determine if it is divisible by 5 or not. If it is divisible by 5, I append it to the second empty list. If not, I simply skip it.

Finally, I print the contents of both lists.

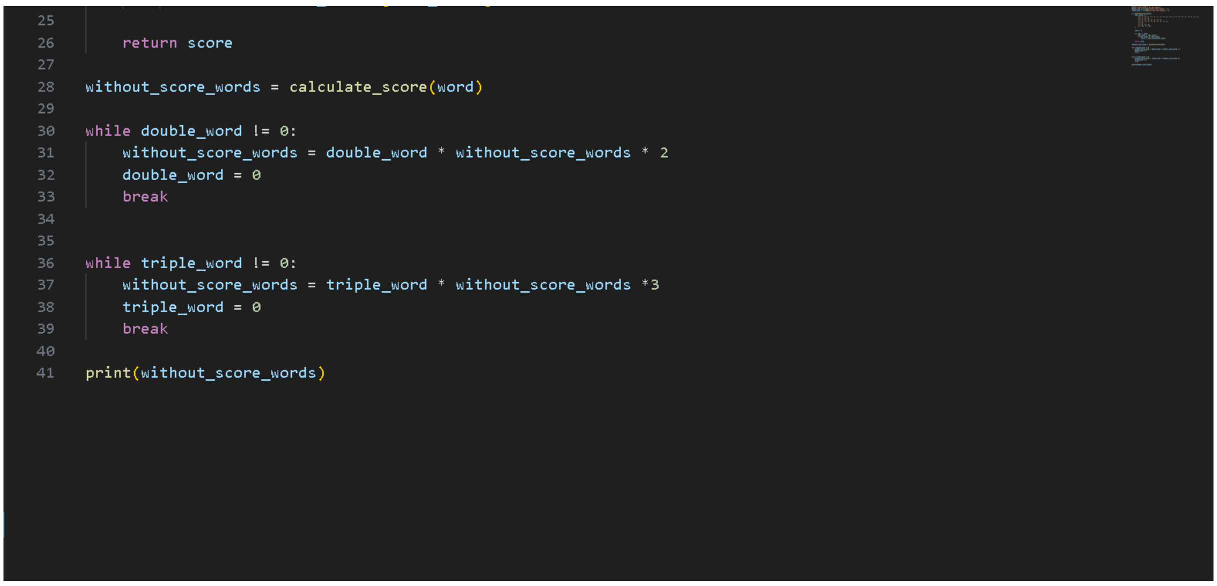
Task 6

Write a Scrabble (link, link2) game score calculator, a function that calculates score for a given word, taking into account also double and triple word score. Make sure, it accepts both lowercase as well as uppercase letters.

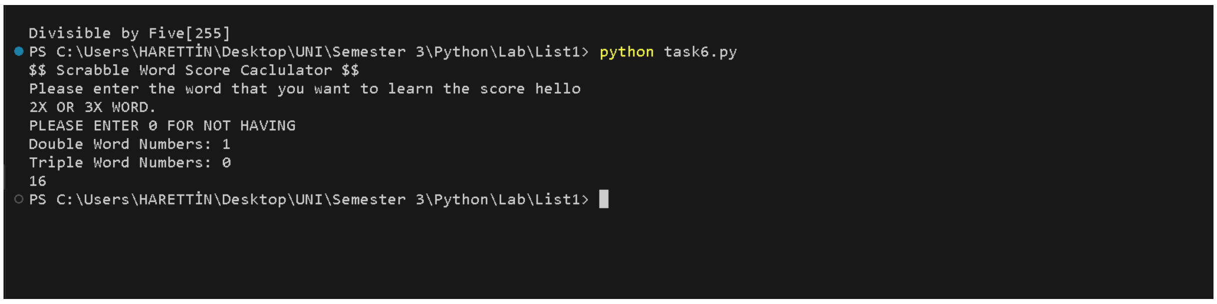
Code:



```python
print("$$ Scrabble Word Score Caclulator $$")
word = input("Please enter the word that you want to learn the score ")
print("2X OR 3X WORD.")
print("PLEASE ENTER 0 FOR NOT HAVING")
double_word =int(input("Double Word Numbers: "))
triple_word = int(input("Triple Word Numbers: "))

def calculate_score(word):
    char_scores = {
        "a": 1, "e": 1, "i": 1, "l": 1, "n": 1, "o": 1, "r": 1, "s": 1, "t": 1, "u": 1,
        "d": 2, "g": 2,
        "b": 3, "c": 3, "m": 3, "p": 3,
        "f": 4, "h": 4, "v": 4, "w": 4, "y": 4,
        "k": 5,
        "j": 8, "x": 8,
        "q": 10, "z": 10,
    }

    score = 0

    for char in word:
        char_lower = char.lower()
        if char_lower in char_scores:
            score += char_scores[char_lower]

    return score
```



```
    return score

without_score_words = calculate_score(word)

while double_word != 0:
    without_score_words = double_word * without_score_words * 2
    double_word = 0
    break


while triple_word != 0:
    without_score_words = triple_word * without_score_words *3
    triple_word = 0
    break

print(without_score_words)
```



```
Divisible by Five[255]
PS C:\Users\HARETTİN\Desktop\UNI\Semester 3\Python\Lab\List1> python task6.py
$$ Scrabble Word Score Caclulator $$
Please enter the word that you want to learn the score hello
2X OR 3X WORD.
PLEASE ENTER 0 FOR NOT HAVING
Double Word Numbers: 1
Triple Word Numbers: 0
16
PS C:\Users\HARETTİN\Desktop\UNI\Semester 3\Python\Lab\List1>
```

Comments:

First, I took an input word to check. Then, I asked whether it's a 2X or 3X word. I obtained these coefficients, or if none were given, I prompted the user to enter "0". To calculate the score, I defined a function called "calculate_score," which takes the word as a parameter. In this function, I created a dictionary that matches letters to their scores. I initialized the score to 0 and then, for each character in the word, I made the character lowercase to ensure that both uppercase and lowercase letters are considered. I added the letter values to the score and returned the final score.

Afterwards, I checked the 2X and 3X coefficients to apply the respective multiplications, and finally, I printed the result.

This task was quite fun and interesting.I didn’t play Scrabble Game but I hope I understood it correct.

4- Conclusion

"In this lab, we practiced using conditional statements, loops, and functions in Python. Loops enable us to repeatedly execute a block of code, while conditional statements help us check whether a condition is true or not. We also utilized functions to improve code organization and reduce its length.

In summary, to create basic programs in Python, it is essential to be familiar with loops, conditionals, and functions