1 Question 1

https://www.w3schools.com/python/ref$_{s}tring_{f}ind.aspandiuseschatgptaswell$.

2 Question 2

2.1 What is a data type?

A data type defines the kind of data a variable can hold and the operations that can be performed on it.

It could be an string like ”Hallo” or an interger ”3” or a float ”3.2”

Data types can help with structuring data, make working with data more efficient. For example with optimize memory usage so there is allocated the correct amount of space. It can also help with minimaze ”Error Prevention” so that two different types can’t combine.

2.2 Representing colours

There it multiple ways (RGB: Best for precise control of color components, Hexadecimal: Convenient for web design and compatibility with CSS)

examples of this:

color$_{r}gb = (255, 0, 0) Redcolorcolor_{h}ex = "FF0000" Redcolorcolor_{n}ame = "red" Redcolor$

2.3 Representing Neighbour Relation

2.4 Colouring problem

3 Question 3

3.1

First i will use the build in function open(file,mode) where i use the relative file path to the file, so it will work on orther computer. I will call the object file, after that i will call use the build in method read() and call the string content.

Now i will define a function called ”CountWord” that takes as input the text, and then the word that would like to be counted.

then i make 2 variables, one called start another called index.

then i make a while loop, that keeps looping until the condition is false.

the function ”text.find()” returns ”-1” when there is no match. there by the condition ”!= -1” becomes false and the loop stop. and then the function returns the count.

```

f = open('Data\diku.txt', "r")

content = f.read()

```

```
def CountWord(text, word):
    count = 0
    start = 0
    while (index := text.find(word, start)) != -1:
        count += 1
        start = index + len(word)  # Move past the last found word
    return count


#print(CountWord(content, "DIKU"))

#test
def test_CountWord():
    # Test 1: Basic occurrence

    assert CountWord("COMPUTERSCINCE computerscince", "
    computerscince") == 1, "Test 1 Failed"

    assert CountWord("Computerscince ", "Daniel") == 0, "Test 2
    Failed"

    assert CountWord("hello Daniel", "hello") == 1, "Test 3 Failed"

    assert CountWord("COMPUTERSCINCE computerscince", "
    computerscince") == 1, "Test 4 Failed"

    assert CountWord("Computerscince Computerscince", "
    Computerscince") == 2, "Test 5 Failed"

    assert CountWord("", "Computerscince") == 0, "Test 6 Failed"

    assert CountWord("COMPUTERSCINCE computerscince", "
    computerscince") == 1, "Test 7 Failed"

    print("All tests passed!")

# Run tests
test_CountWord()
```

3.2

I used functional programming as the main paradigm, but there is also a bit of object oriented programming in the building methods.

3.3

4 Question 4

4.1

Here I define two functions, one called `calculate_points` and another called `total_points_from_file`.

The first function, `calculate_points`, takes as input a string.

The second function, `total_points_from_file`, takes as input the name of a `.txt` file. First, it initializes a variable `total` to zero. Then, it uses the built-in `open()` function on the filename provided as input to the function. It splits each line into a maximum of two parts using the colon (`:`) as the delimiter, and takes the second element of the list, removes any leading and trailing whitespace using `.strip()`, and saves this string as `card_data`. After this, it appends the value from `calculate_points(card_data)` to `total`, adding the points from one string. The loop continues until there are no more lines to process, thanks to the `for line` statement.

```python
def calculate_points(card):
    """Calculate the points for a single card."""
    # Split the card into winning numbers and numbers you have
    parts = card.split('|')
    vindernummer = parts[0].strip().split()
    egnenummer = parts[1].strip().split()

    # Convert to sets for fast lookups
    vindernummer = set(vindernummer)
    egnenummer = set(egnenummer)

    # Find matching numbers
    matchedenummer = vindernummer & egnenummer

    # Calculate points based on the rules
    point = 0
    for i, _ in enumerate(matchedenummer):
        if i == 0:
            point += 1  # First match gives 1 point
        else:
            point *= 2  # Subsequent matches double the points

    return point

def total_points_from_file(filename):
    """Calculate the total points from cards in a file."""
    total = 0

    with open(filename, 'r') as file:
        for line in file:

            card_data = line.split(':', 1)[1].strip()
            total += calculate_points(card_data)

    return total

filename = "data/cards.txt"  # Replace with your file's name
total = total_points_from_file(filename)
#print(f"Total points for all cards: {total}")

# Tests for calculate_points function
def test_calculate_points():
    # Test case 1: No matching numbers
    card = "1 2 3 | 4 5 6"
    assert calculate_points(card) == 0, "Test 1 Failed"

    # Test case 2: One matching number
    card = "8 9 10 | 7 8 11"
    assert calculate_points(card) == 1, "Test 2 Failed"

    # Test case 3: Two matching numbers
    card = "12 15 20 | 15 12 18"
    assert calculate_points(card) == 2, "Test 3 Failed"


    # Test case 5: Mix of matching and non-matching numbers
    card = "5 6 7 | 7 5 8"
    assert calculate_points(card) == 2, "Test 4 Failed"

    # Test case 6: One matching number with duplicates in one group
    card = "4 4 4 | 4 5 6"
    assert calculate_points(card) == 1, "Test 5 Failed"



    print("All tests passed!")

# Run tests
test_calculate_points()
```

4.2

What programming paradigm dominates in your program. Why?

here i use object ortineted programming, also since the build in methoes is used in this way.

4.3

Explain how you test your program.

5

5.1