Københavns Universitet
PoP Assignment 4

Victor Vangkilde Jørgensen - kft410

January 31, 2025

Contents

1 Question 1

1.a Give a list of the external sources you used during the assignment and how you used them. If you use a generative AI tool (e.g., copilot, chatGPT, AITutor, ...) you should keep a log of all the prompts you use.

Alle mine promts og den generede kode kan findes i "promts.txt".
Jeg har benyttet ChatGPT4o til at opskrive type definitions'ne i "colouring.py," da jeg havde glemt hvordan dette bør gøres i python.
Derudover har jeg brugt Claude 3.5 Sonnet til at forklare hvad mine inputs og outputs i "colouring.py" har af betydning. Dette har jeg brugt til at lave mine specifications for funktionerne i "colouring.py".

2 Question 2

2.a What is a data type? Explain what it is using the definitions given in PoP. Illustrate the role data types play when representing data in Python, using examples from the thursday worksheets in week 13, 14 or 15.

En datatype er en form for struktur, som bestemmer hvordan data skal repræsenteres. Disse strukturer har begrænsninger for, hvordan deres data kan defineres.

```
1  class Repeater:
2      "Creates a list with 'num' elements of value: 'input'"
3
4      def __init__(self, num):
5          self.num = num
6
7      def apply(self, input):
8          l = []
9          i = self.num
10         if i <= 0:
11             return l
12         else:
13             while i > 0:
14                 l.append(input)
15                 i += -1
16             return l
17
18 print(Repeater.__doc__)
19 print(Repeater(0).apply(10))
20 print(Repeater(5).apply(20))
21 // OUTPUT:
22 Creates a list with 'num' elements of value: 'input'
23 []
24 [20, 20, 20, 20, 20]
```

I det viste kode eksempel fra torsdags worksheet i uge 15, har jeg eksempelvis brugt den datatype, der kaldes for 'int', hvilket i python er int ræprsenteret som 32-bit heltal.
Vi ser også, at der er angivet en kort docstring, som beskriver funktionens specification. Dette er af typen 'string', og tillader alle tegn, dog skal dette angives i citationstegn, for at markere det som en string.

I mange tilfælde kan sådanne datatyper ikke interagere med hinanden, uden at blive konverteret til en af samme type først. Dette håndteres dog automatisk i python.

2.b Representing colours

Consider the colouring problem that Ken introduced in class. In the context of this problem, describe three possible way to represent colour in Python. Explain in detail how the different representations impact the canExtendColour function (both its signature and its body).

1. Strings

Farver kan repræsenteres som strings, hvor hver farve er en unik strings, eksempelvis: "rød", "grøn", og "blå". Signaturen forbliver:

```
1 canExtendColour(nr: NeighbourRelation, country1: Country, colour:
    Colour) -> bool
```

Funktionens body kræver ingen ændringer, da Colour allerede er en liste af strings.

2. RGB-list

Endnu en måde er at repræsentere farver som RGB-værdier, eksempelvis: [255, 0, 0] for rød, [0, 255, 0] for grøn, og [0, 0, 255] for blå. Dette giver mulighed for at arbejde med mere præcise farver. Signaturen ændres til:

```
1 canExtendColour(nr: NeighbourRelation, country1: Country, colour: list[
    list[int, int, int]]) -> bool
```

hvor Colour er en liste med RGB-lister. Funktionens body skal opdateres til at sammenligne RGB-lister i stedet for strings.

3. Heltal

Man kunne også repræsentere farver som unikke heltal, hvor eksempelvis: 1 er rød, 2 er grøn, og 3 er blå. Signaturen ændres til:

```
1 canExtendColour(nr: NeighbourRelation, country1: Country, colour: list[
    int]) -> bool
```

hvor Colour er en liste med heltal.
Funktionens body skal opdateres til at sammenligne heltal i stedet for strings.

Jeg har valgt disse 3 måder at repræsentere farver på, at det var dem jeg højst sandsynligt selv ville havde benyttet, hvis det var mig der havde lavet colour.
Jeg kan især godt lide RGB-listen, da det giver mulighed for at arbejde med mere præcise farver, og kanExtendColour ville kunne bruges til at sammenligne farver, der er meget tæt på hinanden.

2.c Representing Neighbour Relation

Consider the colouring problem that Ken introduced in class (you can refer to the announcement describing the problem and to his F# solution Download F# solution). In Ken's solution, the type NeighbourRelation is a list of pairs of Countries. A list of pairs is a way to represent a graph (each pair represents an edge between the components of the pair that are vertices). Define a Country type as a recursive data structure, in Python that represents the same graph as NeighbourRelation.

Nederst i 'colouring.py', har jeg lavet følgende:

```
1  class Country:
2      def __init__(self, countryName: str):
3          self.countryName = countryName
4          self.neighbours = []
5
6      def addNeighbour(self, neighbour):
7          if neighbour not in self.neighbours:
8              self.neighbours.append(neighbour)
9              neighbour.addNeighbour(self)
10
11     def __repr__(self):
12         return f"{self.countryName}: {[neighbour.countryName for
               neighbour in self.neighbours]}"
13
14 de = Country("de")
15 da = Country("da")
16 se = Country("se")
17 no = Country("no")
18
19 # Not sure if 'no' and 'da' should be neighbours
20 de.addNeighbour(da)
21 da.addNeighbour(se)
22 se.addNeighbour(no)
23
24 print(de)
25 print(da)
26 print(se)
27 print(no)
28 # Output:
29 de: ['da']
30 da: ['de', 'se']
31 se: ['da', 'no']
32 no: ['se']
```

2.d Colouring problem

Give your solution to the colouring problem in Python. For each function, describe its specification as a docstring.

Omskrivningen af funktionerne samt deres specifications kan findes i "../src/colouring.py"

3 Question 3
Consider a grid of the following form, where DIKU is written horizontally, vertically, diagonally, backwards, or even overlapping other words:

```
..D...
.UKID.
.K..K.
DIKU.U
.D....
```

3.a Write a program that finds all the instances of DIKU in this file Download this file. Follow Ken's method and give the specification of each function that you define as a docstring.

Kens method:

1. **Write a brief description of what the function should do**
2. **Find a name for the function**
3. **Write down test examples**
4. **Find out the type of inputs and outputs**
5. **Generate code for the function (and possibly helper functions)**
6. **Write test cases**
7. **Write short documentation for the fucntion**

Nedeunder har jeg i forbindelse med Kens method skrevet en kort beskrivelse af, hvad mit program skal kunne gøre, hvordan det skal gøres, hvilke inputs og outputs der er, og hvordan det skal testes.

Programmet bør kunne læse en given teks-fil for bogstaver, og beregne mængden af "DIKU" både horisontalt, vertikalt og baglæns. Herefter skal programmet printe mængden af "DIKU" til terminalen.
Eksempelvis skal programmet kunne læse bogstaverne:

```
1 DDKUD
2 IIKUD
3 IDKUD
4 IDKUD
5 IDKUD
```

og beregne mængden af "DIKU" til 1.
Ideelt bør programmet læse disse rækker af bogstaver som string elementer i en liste, som kan gives til en funktion, der kan tælle mængden af "DIKU". Dette kan testes senere hen.

3.b What programming paradigm dominates in your program. Why?

Funktionel programmering fylder en del, da det er det, jeg bruger til at læse bogstavrækkerne og sortere dem i forskellige lister.
Imperativ programmering bruges til at tælle mængden af "DIKU" i hvert af disse lister, men fylder ud over dette ikke meget.
Nu når jeg ser tilbage på opgaven, bør jeg nok havde brugt imperativ programmering i højere grad, da jeg fra forelæsningen havde hørt, at er bedst til små opgaver. Jeg havde dog troet, at funktionel programmering ville være bedst, da det skulle være bedst, når man har meget data, hvilket ikke var tilfældet i denne opgave i samme grad, som jeg havde troet.

3.c Explain how you test your program.

Neders i min "diku.py" fil, har jeg lavet en funktion og skrevet nogle test cases (kan findes i: '../data'), der gør brug af input partitioning, for at finde inputs, der udløser en specification violation.
Jeg fandt bland andet en violation, når man giver en liste, der ikke har samme antal elementer, som elementernes string længde.

4 Question 4

Consider the following problem involving a collection of cards. Each card has two lists of numbers separated by a vertical bar (|): a list of winning numbers and then a list of numbers you have. The first winning number you have is worth one point. Every other winning number doubles your number of points. For example:

```
Card 1: 41 48 83 86 17 | 83 86 6 31 17 9 48 53
Card 2: 13 32 20 16 61 | 61 30 68 82 17 32 24 19
Card 3: 31 18 13 56 72 | 74 77 10 23 35 67 36 11
```

Card 1 is worth 8 points. Card 2 is worth 2 points. Card 3 is worth no points.

4.a Write a program that counts the total number of points for all the cards in "cards.txt". Follow Ken's method and give the specification of each function that you define as a docstring.

Nedeunder har jeg i forbindelse med Kens method skrevet en kort beskrivelse af, hvad mit program skal kunne gøre, hvordan det skal gøres, hvilke inputs og outputs der er, og hvordan det skal testes.

Programmet bør kunne læse txt-filen "cards.txt" i mappen "data", beregne antallet af points for hvert card, og write resultatet til terminalen.
Eksempler på inputs og tilsvarende outputs kunne være dem som er givet i opgaven:

```
1 Card 1: 41 48 83 86 17 | 83 86 6 31 17 9 48 53 # 8 Points
2 Card 2: 13 32 20 16 61 | 61 30 68 82 17 32 24 19 # 2 Points
```

```
Card 3: 31 18 13 56 72 | 74 77 10 23 35 67 36 11 # 0 Points
```

Jeg har valgt, at definere en class "Card", der indtager en tekst-række som input (på den måde vist i eksempel inputsne).
Class'en har nogle members, der behandler data'en, og omskriver det på en måde, så vi kan tjekke, om værdierne af kortet matche vinderværdierne.
Antallet af points for hvert kort bliver kumuleret, og printet til terminalen.

4.b What programming paradigm dominates in your program. Why?

Jeg vil mene, at objekt-orienteret programmering dominerer i mit program, da jeg har valgt at lave en class, der indeholder alle members, som behandler data'en.
Jeg ville kunne havde lavet alle mine members uden for class'en, hvilket ville få funktionel programmering til at dominere.

4.c Explain how you test your program.

Nederst i 'cards.py' har jeg lavet en test-function, samt nogle test cases, der skal teste hvilke inputs, der resulterer i en specification violation. Disse cases er lavet, så nogle af de ønskede features bag programmet, bliver indkluderet.
Alle disse test kører på 'Card.cardPoints' member'en, da dette er den vigtigeste member for opgaven.

```
# Test function
def testFun(testCase):
    """
    Runs testcases for the member 'cardPoints' in the class Card.

    Inputs:
        list[str]: of a card.

    Outputs:
        int: point amount for the card, or exception if input invalid.
    """
    try:
        return Card(testCase).cardPoints()
    except:
        return Exception

# Test cases
print("Test Cases:")
print(testFun("Card 1: 41 48 83 86 17 | 83 86 6 31 17 9 48 53")) # Test Case 1: Input from assignment. Expected Output: 8
print(testFun("Card 2: 7 8 9 | 7 8 9")) # Test Case 2: Header cut into numbers. Expected Output: 1
print(testFun("Card 100: 1 2 3 | 4 5 6")) # Test Case 3: Long header that is cut short. Expected Output: 0
print(testFun("Card 9:")) # Test Case 4: Empty card. Expected Output: 0
print(testFun("Card 123: 10 20 | 10 20")) # Test Case 5: Dubble numbers After removing 'Card'. Expected Output: 1
print(testFun("Card 4: | 1 2 3")) # Test Case 6: No winning numbers. Expected Output: 0
print(testFun("Card 5: 1 2 3 |")) # Test Case 7: No your numbers. Expected Output: 0
```

```
26 print(testFun("Card 6: 5 | 5 5 5")) # Test Case 8: Duplicate numbers.
      Expected Output: 4
27 print(testFun("Card 7: 1 2 3 | 3 2 1")) # Test Case 9: All numbers
      match but reversed. Expected Output: 4
28 print(testFun("Card 8: 100 200 | 100 200")) # Test Case 10: Dubbles.
      Expected Output: 1
29 print(testFun("Card 999: 41 48 | 83 86")) # Test Case 11: Overlapping.
      Expected Output: 0
30 # Output:
31 Test Cases:
32 8
33 2
34 0
35 <class 'Exception'>
36 2
37 <class 'Exception'>
38 0
39 <class 'Exception'>
40 2
41 16
42 0
```

4.d Explain the role of all variables in your program, using the role of variables framework introduced in PoP.

Fixed Value

cardRows er konstant, da det indeholder alle kortdata, som er læst fra en fil, og ændres ikke.
cardNumber bruges til at holde det aktuelle kortnummer. Selvom værdien ændres i løbet af programmet opdateres den med det samme (inkrementering med 1).

Stepper

"i" Bruges i flere loops til at iterere gennem elementer i lister, eksempelvis i cardRemover, cardNumbers og winningNumbers.

Most Recent Holder

"result" holder den nuværende værdi af kortdata under iterering i cardRemover.
head og tail holder starten og resten af listen, der bliver itereret i cardRemover.

Most Wanted Holder

points holder den aktuelle score, der matches med kortets winningNumbers.

Container

numbers ændrer lister af tal fra kortdata i cardNumbers og winningNumbers.

Organizer

numbers bruges også til at omskrive talene i methodsne.

Temporary

head og tail bruges kun midlertidigt i cardRemover til at ændre kortdata.
result holder midlertidigt det ændrede kort i cardRemover.