Københavns Universitet
PoP Assignment 4

Victor Vangkilde Jørgensen - kft410

January 14, 2025

Contents

1 Question 1 **4**

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. 4

2 Question 2 **4**

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. 4

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). 5

2.c URepresenting NeighbourRelation
Consider the colouring problem that Ken introduced in class. 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 NeighbourRelation type as a recursive data structure, in Python. 5

2.d Colouring problem
Give your solution to the colouring problem in Python. For each function, describe its specification as a docstring. 6

3 Question 3
Consider a grid of the following form, where DIKU is written horizontally, vertically, diagonally, backwards, or even overlapping other words: **6**

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. 6

3.b What programming paradigm dominates in your program. Why? 7

3.c Explain how you test your program. 7

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: **7**

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. 7

4.b What programming paradigm dominates in your program. Why? 8

4.c Explain how you test your program. 8

4.d Explain the role of all variables in your program, using the role of variables framework introduced in PoP. 8

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 vi ikke havde lært, at lave ADTs i python til forelæsningen.

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.
I mange tilfælde kan sådanne datatyper ikke interagere med hinanden, uden at blive konverteret til en af samme type først.

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:

```
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:

```
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:

```
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.

2.c URepresenting NeighbourRelation

Consider the colouring problem that Ken introduced in class. 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 NeighbourRelation type as a recursive data structure, in Python.

```
class NeighbourRelation:
    def __init__(self, country1: Country, country2: Country, rest=None)
        :
        self.pair = (country1, country2)
        self.rest = rest

    def __repr__(self):
        return f"{self.pair}, {self.rest}"
```

```
8
9  nr = NeighbourRelation("de", "da", NeighbourRelation("da", "se",
       NeighbourRelation("se", "no")))
10 print(nr)
11 #OUTPUT:
12 ('de', 'da'), ('da', 'se'), ('se', 'no'), None
```

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**

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:

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".

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.

3.c Explain how you test your program.

Neders i min "diku.py" fil, har jeg skrevet nogle test cases, 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.

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
3 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 beregnes, og printes til terminalen for hvert kort.

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

Jeg vil sige, at funktionel programmering domminerer i mit program, da jeg sagtens kunne have undgået, at lave funktionerne i mit object Card.
Hvis jeg havde benyttet flere classes, ville object-orienteret programmering fylde mere.

4.c Explain how you test your program.

En af mine members hedder "cardStats". Denne member tager et kort, og viser resultatet af de fleste andre members. Jeg valgte at lave denne member, da jeg så kunne fodre den nogle inputs, og se om/hvor der sker errors.

```
1 print(Card("Card   7: 51 64 21 49 30 93 60  4 10 44 |").cardStats()) #
    No winning numbers
2 print(Card("Card   8:  | 58 64 83 41 69 63 11 30 73 28 75 87 76 15  6
    2 10 51 20 47 82 48 61 46 53").cardStats()) # No card numbers
    numbers
3 print(Card("Card   9:  | ").cardStats()) # Only vertical bar
4 print(Card("Card   10000: 49 98 13  3 61 66 16 46 22 44 | 58 26 35 75
    96 64 18  9 92 45 94 67 49 89 84 20 48 39 43 38 55 90  6 69 30").
    cardStats()) # Large card index
5 print(Card("Card   9: ").cardStats()) # Empty card
```

Kun den sidste input forårsager en specification violation.

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

Fixed Value

- cardNumber: Denne variabel bruges til at holde det aktuelle kortnummer. Selvom værdien ændres i løbet af programmet, har den en fast opdateringslogik (inkrementering med 1).

Stepper

-

Walker

-

Most Recent Holder

-

Most Wanted Holder

-

Container

-

Organizer

-

Temporary

-