WEEK 4

ONDERWERPEN

- case: word histogram
- recursion
- case: solving Numbrix
- PySerial
- tkinter GUI-toolkit

OPGAVE 1 : EEN GUI VOOR ARDUINO

In deze opgave gaan we een eenvoudige gui applicatie maken die je wellicht ook in het project kan gebruiken. Op Blackboard is de file simple_plot.py te vinden. In dit voorbeeld worden waarden random gegenereerd, maar op deze manier kun je ook waarden afkomstig van de Arduino plotten. Het programma simple_plot gaan we een beetje aanpassen en uitbreiden.

a) Bekijk de code. Op regel 14 staat een global-statement. Is deze wel noodzakelijk? Motiveer je antwoord
b) De code is niet erg mooi. Vorm de code om in een maak een class Plot met constructor en een functie step().
c) Voeg bij de horizontale en verticale as labels toe, bijvoorbeeld "step" en "value".
d) Voeg een pauzeknop toe, zodat je het plotten even kan stoppen.
e) Voeg twee velden "max" en "min" met een "set"-knop toe, waarmee de gebruiker kan aangeven waartussen de y-waarden kunnen variëren (dus 0 <= min_value < value < max_value <= 100). Teken ook twee horizontale lijnen die deze grenzen aangeven.

OPGAVE 2 : JSON

a) Op Blackboard is de file "books.json" te vinden. Lees de file "books.json" in naar een Python variabele book_list (d.w.z. in een Python object). Bepaal vervolgens de gemiddelde prijs van alle boeken met als onderwerp "Python". Druk deze gemiddelde prijs af met print(), met twee decimalen achter de komma.
b) (1 pt) Sorteer de lijst met boeken op achternaam van de auteur. Druk deze lijst af met pprint(). Je mag hierbij aannemen dat het formaat van de auteursnaam ‘<voornaam><spatie><achternaam>’ is.

OPGAVE 3 : PLOT HISTOGRAM

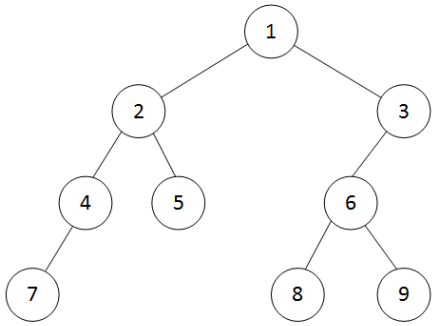
Op Blackboard is de file "show_histogram_start.py" te vinden. In het college van week 4 is besproken hoe je een histogram kunt maken van de woord-frequenties in een tekstbestand. Alleen het plotten van het histogram ontbreekt hier nog. Maak een programma dat op basis van "show_histogram_start.py" de gebruiker een testbestand laat kiezen, en vervolgens de frequenties van de 10 meest voorkomende woorden in een histogram laat zien.

OPGAVE 4 : MARKOV ANALYSE

Deze opgave gaat verder met de vorige opgave (en komt ook overeen met paragraaf 13.8 uit het boek "Thinking Python"). Behalve weten hoe vaak een woord voorkomt, kan het ook interessant zijn welke combinaties van woorden vaak voorkomen. Neem een zin als : "Onze poes heet Tijger", dan zijn (onze, poes), (poes, heet) en (heet, Tijger) 3 paren van woorden. Schrijf een programma dat een histogram maakt van de 10 meest voorkomende *paren van woorden* in een tekstbestand.

OPGAVE 5 : PRE-ORDER TRAVERSAL VAN EEN BINAIRE BOOM (DFS)

In het college van week 4 is besproken hoe je een binaire boom pre-order kunt doorlopen op basis van recursie. Als voorbeeld moet bij onderstaande boom de waarden 1 .. 9 op volgorde worden afgedrukt wanner deze pre-order wordt doorlopen.



Maak het volgende programma af door de functie pre-order te schrijven (zijn maar een paar regels).

```
from collections import namedtuple

Node = namedtuple('Node', 'data, left, right')

tree = Node(1,
            Node(2,
                 Node(4,
                      Node(7, None, None),
                      None),
                 Node(5, None, None)),
            Node(3,
                 Node(6,
                      Node(8, None, None),
                      Node(9, None, None)),
                 None))

def preorder(node):
    // your code

preorder(tree)
```

OPGAVE 6 : NUMBRIX PUZZEL

In het college van week 4 is besproken wat een Numbrix puzzel is en hoe deze kan worden opgelost. Op Blackboard is de file numrix.py te vinden. Dit programma is al voor een deel af, maar de functies neighbors() en solve() moet nog worden geïmplementeerd. De functie solve() roept zichzelf aan, de oplossing is dus recursief.

OPGAVE 7 : EEN BEETJE DATA SCIENCE

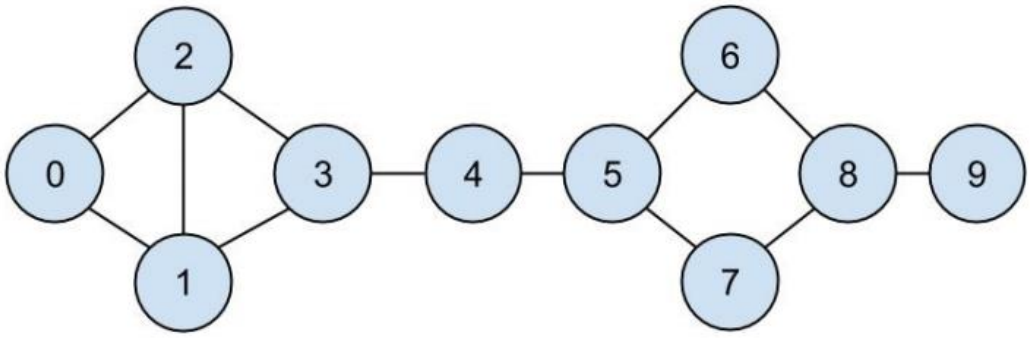
Deze opgave gaat over het in kaart brengen van studenten en vriendschappen. Er zijn 10 studenten, gedefinieerd in een dictionary als volgt :

```
students = { 0: "Nynke", 1: "Lolle", 2: "Jikke", 3: "Popke", 4: "Teake", 5:
"Lieuwe", 6: "Tsjabbe", 7: "Klaske", 8: "Ypke", 9: "Lobke"}
```

De vriendschappen zijn vastgelegd in een lijst van tuples :

```
friendships = [(0, 1), (0, 2), (1, 2), (1, 3), (2, 3), (3, 4), (4, 5), (5,
6), (5, 7), (6, 8), (7, 8), (8, 9)]
```

Deze studenten en hun vriendschappen kunnen ook in een graaf als volgt worden weergegeven:



a) Maak een 2-dimensionale lijst (matrix), waarin voor elke student een lijst welke vrienden/vriendinnen hij/zij heeft. Dus de volgende lijst : [[1, 2], [0, 2, 3], ..., [8]]. Print deze lijst zodat je weet wie welke vrienden/vriendinnen heeft.

b) Maak en print een lijst van studenten gesorteerd op het *aantal* vrienden van elke student. Dus een lijst [(0, 2),(1,3), ... , (8, 3), (9, 1)].

c) Hoe zit het met gemeenschappelijke vrienden ? Bijvoorbeeld, student 0 en student 3 hebben twee gemeenschappelijke vrienden, terwijl 0 en 3 zelf geen vrienden van elkaar zijn. En 3 en 5 hebben één gemeenschappelijke vriend, terwijl ook zij geen vrienden van elkaar zijn. Print voor studenten 3 en 5 af met wie ze gemeenschappelijke vrienden zijn, en hoeveel vrienden ze gemeenschappelijk hebben. Dus voor student 3 is dat "0 : 2, 5 : 1". Voor het tellen is het handig om de klasse Counter uit collections te gebruiken.

d) Stel dat de interesses van studenten als volgt zijn (lijst van tuples) :

```
interests = [
(0, "Hadoop"), (0, "Big Data"), (0, "HBase"), (0, "Java"),(0, "Spark"), (0,
"Storm"), (0, "Cassandra"),
(1, "NoSQL"), (1, "MongoDB"), (1, "Cassandra"), (1, "HBase"),(1, "Postgres"),
(2, "Python"), (2, "scikit-learn"), (2, "scipy"), (2, "numpy"), (2, "statsmodels"),
(2, "pandas"),
```

```
(3, "R"), (3, "Python"),(3, "statistics"), (3, "regression"), (3, "probability"),
(4, "machine learning"), (4, "regression"), (4, "decision trees"),(4, "libsvm"),
(5, "Python"), (5, "R"), (5, "Java"), (5, "C++"),(5, "Haskell"), (5, "programming
languages"),
(6, "statistics"),(6, "probability"), (6, "mathematics"), (6, "theory"),
(7, "machine learning"), (7, "scikit-learn"), (7, "Mahout"),(7, "neural networks"),
(8, "neural networks"), (8, "deep learning"),(8, "Big Data"), (8, "artificial
intelligence"),
(9, "Hadoop"),(9, "Java"), (9, "MapReduce"), (9, "Big Data")
]
```

Maak een dictionary waarin keys=studenten en values=lijst met interesses van die student. Dus een dictionary {0: ['Hadoop', 'Big Data', 'HBase', ...], 1: ['NoSQL', 'MongoDB', 'Cassandra', ...], ... }.

e) Maak een dictionary waarin keys=interesses en values=lijst van studenten met die interesse. Dus een dictionary {'Java': [0, 5, 9], 'Big Data': [0, 8, 9], ...}