Project / groepswerk Programmeren I: Temperatuursensor

1 Inleiding

Dit document bevat de beschrijving van het project voor het opleidingsonderdeel "Programmeren I". Studenten kunnen dit project uitvoeren in groepjes, met een **maximum van 4 studenten per groep**. Per groep wordt één document met broncode ingediend (eventueel aangevuld met bijkomende bestanden indien je dit nodig acht). Dit wordt digitaal (een zip-file met het .py bestand met broncode en evt. aanvullende bestanden) ingediend via `indianio.ugent.be`[1]. De **deadline** voor indienen is **zondag 24 november om 23u59**. Projecten die te laat worden ingediend, worden niet aanvaard. Voor algemene richtlijnen en tips m.b.t. het opmaken van het verslag en de broncode, zie verder in dit document.

2 Situering

In vele toepassingen worden temperaturen quasi continu gemeten en gelogd. Indien de metingen gebeuren op afgelegen plaatsen, kan men ervoor kiezen om een draadloze verbinding te gebruiken om deze metingen door te sturen naar een centraal data-centrum. Op zeer afgelegen plaatsen (zonder GSM-dekking of Wifi) maakt men voor de transfer soms gebruik van LEO (Low Earth Orbit) satellieten. De grootte van de datapaketten die deze satellieten kunnen verwerken is echter beperkt. Daarom zal men gemeten temperaturen hercoderen als volgt:

1. Deel de gemeten temperatuur door 10.
2. Rond de bekomen waarde af naar beneden, tot een geheel getal.

Voorbeeld 1:

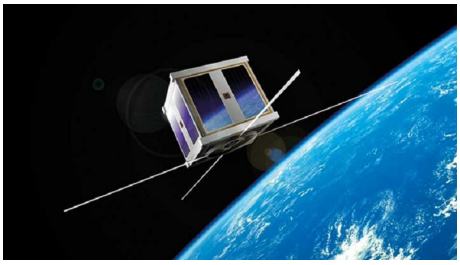
$$62.1°C \text{ wordt } 6$$

Temperaturen tussen 0 en 79 graden celcius kunnen op deze manier voorgesteld worden door de cijfers 0–7. De binaire voorstelling van gehele getallen laat toe om deze cijfers te coderen d.m.v. **drie bits**. Deze binaire sequenties worden vervolgens doorgestuurd naar de LEO satelliet.

Voorbeeld 2:
De rij van temperaturen 25.8, 48.1 en 11.2 wordt:

$$25.8, 48.1 \text{ en } 11.2 \rightarrow \texttt{'241'} \rightarrow \texttt{'010100001'}$$

Merk op dat de sequenties van cijfers `'241'` en `'010100001'` worden voorgesteld als *strings*.



In dit project zal aan de hand van een reeks opdrachten het coderings- en decoderingsprotocol geïmplementeerd worden.

[1] Kies voor *Project Programmeren 20192020*.

3 Beschrijving van de opdrachten

In deze sectie worden de verschillende opdrachten die deel uitmaken van het project beschreven. Het is de bedoeling dat per onderdeel van deze taak een afzonderlijk stuk broncode wordt geschreven, deze broncode wordt aangevuld in de file `taak_informatica.py` die beschikbaar is op Ufora in de map `Project`. Dit bestand wordt ingediend via `indianio.ugent.be`. Gebruik voor deze opdrachten (met uitzondering van Opdracht 6) GEEN bijkomende modules (met uitzondering van `infoFun`).

Opdracht 1

Schrijf een programma dat de hercodering van temperaturen naar een string van cijfers implementeert zoals hiervoor beschreven:

$$25.8, 48.1 \text{ en } 11.2 \rightarrow \texttt{'241'}.$$

Schrijf een programma dat:

1. De gebruiker een temperatuur vraagt (tussen 0 en 79 °C)
2. De temperatuur omzet in een cijfer
3. Stappen 1–2 herhaalt tot de gebruiker STOP ingeeft.
4. De sequentie van cijfers als string weergeeft op het scherm (zie voorbeeld hieronder).

Gebruik voor Opdracht 1 **GEEN ingebouwde string/list-methoden zoals `str.count()` of externe modules**, maar uiteraard wel while/for/else statements (cfr. hoofdstukken 1, 2 en 3).

Een mogelijke input/output is:

```
>>> Geef een temperatuur: 25.3
Geef een temperatuur: 48.3
Geef een temperatuur: 12.4
Geef een temperatuur: 7.3
Geef een temperatuur: STOP
De sequentie is '2410'
```

Opdracht 2

Implementeer een functie `codeer_T_seq` die een lijst van gemeten temperaturen (datatype *float*) als input aanvaardt en de binaire voorstelling van de sequentie (bestaande uit 3 bits per gemeten temperatuur) als string retourneert. Een mogelijke input/output is:

```
>>> codeer_T_seq([12.1])
'001'
>>> codeer_T_seq([26.3,34.6,38.9,40.0])
'010011011100'
```

Tip: voor de omzetting naar de binaire voorstelling kan je inspiratie halen op *stackoverflow*[2].

[2] `https://stackoverflow.com/questions/16926130/convert-to-binary-and-keep-leading-zeros-in-python`

Opdracht 3

Op het centraal data-centrum dienen de gecodeerde data, afkomstig van de LEO, weer omgezet worden naar sequenties van cijfers, alvorens ze geanalyseerd kunnen worden. Implementeer een functie `decodeer_bin_T_seq` die een binaire sequentie naar een string van cijfers decodeert.

Een mogelijke input/output is:

```
>>> decodeer_bin_T_seq('001')
'1'
>>> decodeer_bin_T_seq('010011011100')
'2334'
```

Het bestand `temperatuursequenties.txt` bevat een reeks sequenties van cijfers samen met hun binaire voorstelling, gescheiden door een spatie. Implementeer een script dat voor deze sequenties nagaat of de overeenkomstige binaire voorstelling correct is. Indien er fout gecodeerde sequenties voorkomen, dienen deze te worden uitgeschreven naar het scherm. Nadat alle sequenties gecontroleerd zijn, dient ook het totaal aantal codeerfouten gemeld te worden.

De laatste twee regels van de output zouden er als volgt moeten uitzien (met cijfers op de plaats van de ...):

```
Temperatuur-sequentie ... werd fout gecodeerd.
In totaal werden ... van de ... sequenties fout gecodeerd.
```

Tip 1: voor deze opdracht kan je weer inspiratie halen op *stackoverflow*[3].
Tip 2: gebruik de module `infoFun` om de gegevens in te lezen.

Opdracht 4

In een onderzoek is men geïnteresseerd in het aantal keer dat de temperatuur stijgt. Bij voorbeeld in de sequentie

$$36733675 \text{ heeft men } 3 \uparrow 6 \uparrow 7 \downarrow 3 = 3 \uparrow 6 \uparrow 7 \downarrow 5$$

en kan men 4 keer een stijging terugvinden.
Implementeer een script voor elke string van cijfers in een gegeven lijst bepaalt hoeveel temperatuurstijgingen erin voorkomen. Dit aantal wordt vervolgens uitgeschreven samen met de oorspronkelijke sequentie.

Een mogelijke output is:

```
Sequentie '36733675'
telt 4 stijgingen.

Sequentie '1234224567'
telt 7 stijgingen.

Sequentie '535645567522402233202224'
telt 10 stijgingen.
```

[3] `https://stackoverflow.com/questions/8928240/convert-base-2-binary-number-string-to-int`

Opdracht 5

Om te controleren of er tijdens de data-overdracht of het coderen geen fouten zijn opgetreden, kan een pariteitsbit worden toegevoegd aan de binaire voorstelling. Een pariteitsbit geeft aan of een binaire code een even of oneven aantal logische enen heeft. Indien het aantal enen even is, neemt de (even) pariteitsbit de waarde 0 aan, indien het oneven is, neemt deze de waarde 1 aan. Dit wordt geïllustreerd in onderstaande tabel.

binaire code	**aantal enen**	**pariteitsbit**
001	1	1
101	2	0

Opdracht 5.a Pas de functie `codeer_T_seq` uit Opdracht 2 aan zodat deze **voor** de binaire voorstelling van elke temperatuur in de sequentie ook een pariteitsbit plaatst. Noem je nieuwe functie `codeer_T_seq_PB`.
Een mogelijke input/output is:

```
>>> codeer_T_seq_PB([12.1])
'1001'
>>> codeer_T_seq_PB([26.3,34.6,38.9,40.0])
'1010001100111100'
```

Opdracht 5.b Pas de functie `decodeer_bin_T_seq` uit Opdracht 3 aan zodat deze rekening houdt met de pariteitsbits in de sequentie. Noem je nieuwe functie `decodeer_bin_T_seq_PB`. Indien een fout gedetecteerd wordt (nl. een pariteitsbit die niet overeenkomt met het aantal enen van de bijhorende temperatuur), dient de functie dit aan te geven met een opgegeven symbool in de sequentie. Wanneer geen symbool wordt opgegeven, wordt `'X'` gebruikt.
Een mogelijke input/output is:

```
>>> decodeer_bin_T_seq_PB("1001")
'1'
>>> decodeer_bin_T_seq_PB('010011011100')
'XX4'
>>> decodeer_bin_T_seq_PB("10100011001011001001")
'23X41'
```

Opdracht 6

In de voorgaande opdracht waren *fouten* het gevolg van bits die een foutieve waarde hadden. Een ander type fouten ontstaat wanneer, tijdens het verzenden/ontvangen van data, bits verdwijnen uit een (binaire) sequentie. In het bestand `missing_bits.txt` vinden jullie 150 verschillende sequenties waarbij dit het geval is. Implementeer een script dat zoveel mogelijk van de originele temperatuur-sequenties probeert te reconstrueren. In de eerste 50 sequenties ontbreekt telkens 1 bit, maar komen verder geen fouten voor. In de sequenties 51–100 ontbreken telkens twee bits, maar komen verder geen fouten voor. In de laatste 50 sequenties ontbreken 1 of 2 bits en kunnen bovendien ook bits van waarde zijn gewijzigd. Voor deze opdracht zijn jullie volledig vrij in de gekozen oplossingsstrategie (en dus ook ook in het gebruik van methodes, functies en modules).