Joeri van Grimbergen (1244825) | Nursize Bilen (1260235) | Rick van Gorp (1328417)

Zuyd Hogeschool | Heerlen | GRoep 4

MICT1

eXercise week 1

# What we found

Door Zuyd Hogeschool is het bestand week1.data beschikbaar gesteld, waarin enkele geëncodeerde strings zijn verwerkt. Dit betreft de volgende strings:

* Simple text: Zuyd
* Double happiness: 囍
* Pile of Poo: 💩
* BONUS: First name (“Jeroen”) somewhere in the file, using Base64-encoding

Om bovenstaande strings te vinden in het bestand week1.data is er een Python-script ontworpen door Groep *4.* De bedoeling van dit Python-script is om het zo universeel mogelijk te laten functioneren. Dit houdt in dat het ook te gebruiken is voor andere strings en voor alle mogelijke encoderingsmethoden die door Python worden ondersteund. De resultaten die met dit script zijn gegenereerd zijn onderstaand weergegeven: (Dit betreft dus alleen de gevonden resultaten. Alle encoderingsmethoden zijn behandeld, maar hebben geen resultaten opgeleverd op onderstaande encoderingsmethoden na).

**Pile of poo:**

|  |  |  |
| --- | --- | --- |
| Encoderingsmethode | Input in hex-formaat | Aantal keer in week1.data |
| UTF\_32BE | 0001F4A9 | 6 |
| UTF\_8 | F09F92A9 | 7 |

Aantal unieke offsets voor alle encodeermethoden: 13

**Zuyd:**

|  |  |  |
| --- | --- | --- |
| Encoderingsmethode | Input in hex-formaat | Aantal keer in week1.data |
| UTF\_16\_BE | 005A007500790064 | 6 |
| UTF\_16\_LE | 5A00750079006400 | 6 |

Aantal unieke offsets voor alle encodeermethoden: 12

**Double happiness:**

|  |  |  |
| --- | --- | --- |
| Encoderingsmethode | Input in hex-formaat | Aantal keer in week1.data |
| BIG5 | F8A6 | 33 |
| BIG5HKSCS | F8A6 | 33 |
| CP949 | FDEE | 39 |
| CP950 | F8A6 | 33 |
| EUC\_KR | FDEE | 39 |
| GBK | 87D6 | 47 |
| GB18030 | 87D6 | 47 |
| JOHAB | F9EE | 29 |
| SHIFT\_JIS\_2004 | F1D2 | 30 |
| UTF\_16\_BE | 56CD | 28 |
| UTF\_16\_LE | CD56 | 37 |
| UTF\_8 | E59B8D | 7 |

Aantal unieke offsets voor alle encodeermethoden: 250

**Base64 “Jeroen”:**

Onderstaand wordt de initiële encoderingsmethode (die is gebruikt voor de base64 string) weergegeven. Alle resultaten zijn op basis hiervan gegroepeerd. In het script wordt de string 2x gecodeerd met alle mogelijke encoderingsmethoden. De hex-formatted input is ook alleen gebaseerd op de match van de regex string met het binaire bestand.

|  |  |  |
| --- | --- | --- |
| Encoderingsmethode | Input in hex-formaat | Aantal keer in week1.data |
| ASCII | 536D567962325675 | 12 |
| BIG5 | 536D567962325675 | 12 |
| BIG5HKSCS | 536D567962325675 | 12 |
| CP437 | 536D567962325675 | 12 |
| CP737 | 536D567962325675 | 12 |
| CP775 | 536D567962325675 | 12 |
| CP850 | 536D567962325675 | 12 |
| CP852 | 536D567962325675 | 12 |
| CP855 | 536D567962325675 | 12 |
| CP856 | 536D567962325675 | 12 |
| CP857 | 536D567962325675 | 12 |
| CP860 T/M CP869 | 536D567962325675 | 12 |
| CP874 | 536D567962325675 | 12 |
| CP932 | 536D567962325675 | 12 |
| CP949 | 536D567962325675 | 12 |
| CP950 | 536D567962325675 | 12 |
| CP1006 | 536D567962325675 | 12 |
| CP1250 T/M CP1258 | 536D567962325675 | 12 |
| EUC\_JP | 536D567962325675 | 12 |
| EUC\_JIS\_2004 | 536D567962325675 | 12 |
| EUC\_JISX0213 | 536D567962325675 | 12 |
| EUC\_KR | 536D567962325675 | 12 |
| GB2312 | 536D567962325675 | 12 |
| GBK | 536D567962325675 | 12 |
| GB18030 | 536D567962325675 | 12 |
| HZ | 536D567962325675 | 12 |
| ISO2022\_JP + EXTENSIONS | 536D567962325675 | 12 |
| ISO2022\_KR | 536D567962325675 | 12 |
| ISO8859\_ + EXTENSIONS | 536D567962325675 | 12 |
| JOHAB | 536D567962325675 | 12 |
| KOI8\_R | 536D567962325675 | 12 |
| KOI8\_U | 536D567962325675 | 12 |
| MAC\_ + EXTENSIONS | 536D567962325675 | 12 |
| PTCP154 | 536D567962325675 | 12 |
| SHIFT\_JIS | 536D567962325675 | 12 |
| UTF\_7 | 536D567962325675 | 12 |
| UTF\_8 | 536D567962325675 | 12 |

Aantal unieke offsets voor alle encodeermethoden: 12

# Where we found it

In dit gedeelte worden screenshots weergegeven van de output die is gegenereerd door het Python script in het volgende formaat:

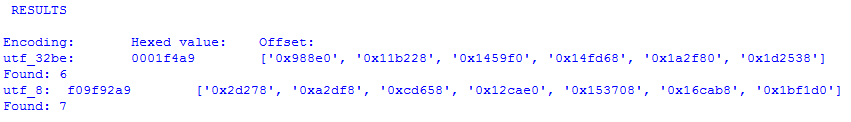
*Encoding: Hexed Value: Offset:*

*Encoderingsmethode Input string in hexadecimaal Array offsets van string in bestand*

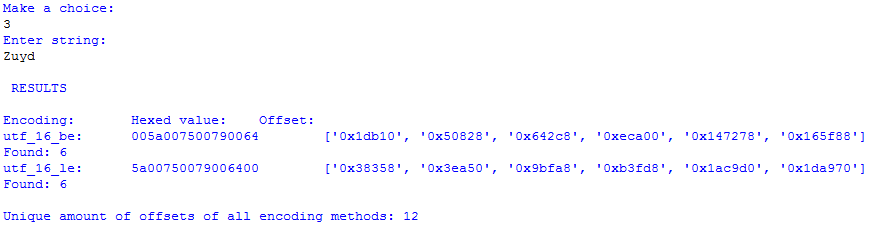
*Found: [gevonden aantal offsets]*

De weergegeven offsets zijn te gebruiken in bijvoorbeeld een hex-viewer om de cursor naar de exacte positie van de start van de binaire string te verplaatsen. Op de onderstaand weergegeven offsets zijn de “hexed values” te vinden. De “hexed values” zijn gevormd uit de initiële zoekstring.

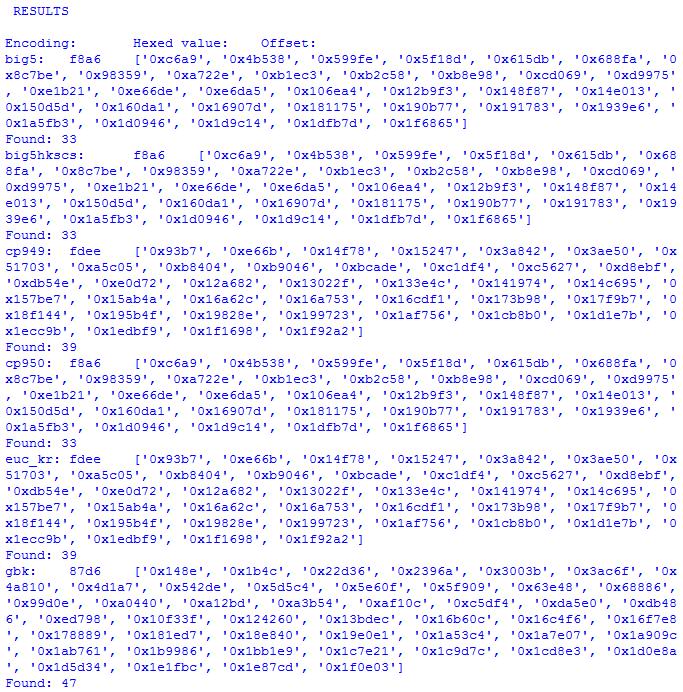
**Pile of poo:**

****

**Zuyd:**

****

**Double happiness:**

****

****

**Base64 “Jeroen”:**

****

Jeroen valt in heel veel encodings te vinden, voor alle resultaten kan het script gedraaid worden en op Jeroen gezocht worden.

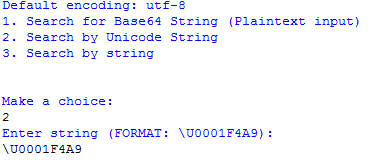
# How we found it

Om tot de hierboven beschreven resultaten te komen is er een Python script ontworpen door groep 4*.* In dit gedeelte zal beschreven worden hoe dit Python script tot stand is gekomen en het zal een korte handleiding bevatten om het Python script te bedienen.

## 3.1 Handelingen voor resultaten

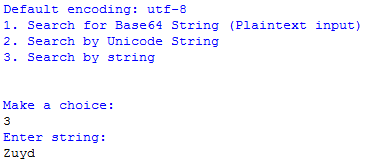
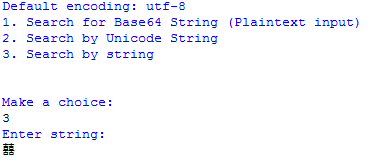
In dit gedeelte worden per karakter de handelingen weergegeven die zijn uitgevoerd om tot het resultaat te komen. Het script moet uitgevoerd worden in Python versie 3.4.3 en alle bestanden moeten in dezelfde map als het Py-bestand staan. Dit betreft de bestanden *codecs.txt* en *week1.data*. In *codecs.txt* kunnen encoderingsmethoden worden gedefinieerd die door Python worden ondersteunt en die door Python gebruikt moeten worden.

**Pile of Poo**



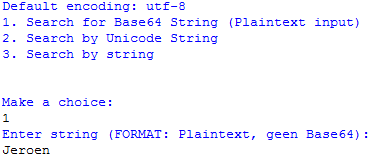
Als het script wordt uitgevoerd, wordt er een keuze aan de gebruiker geboden. Omdat Python problemen heeft met het verwerken van het Pile of Poo karakter, hebben we gebruik gemaakt van de unicode notatie van Pile of Poo. Dit is \U0001F4A9.

**Zuyd & Double happiness**

Voor beide karakters wordt gekozen voor de standaard string invoer en wordt vervolgens het karakter of de set van karakters ingevoerd.

**Base64 “Jeroen”:**



Voor base64 wordt gebruik gemaakt van optie 1 en wordt de string “Jeroen” ingevuld.

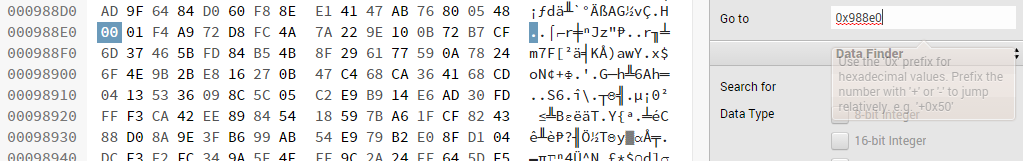
## Controle uitvoer Python script

De uitvoer van het python script is gecontroleerd aan de hand van de website: <https://hexed.it/>

Op deze website is het mogelijk een extern bestand te laden in de interface en in hexadecimaal formaat weer te geven. Door enkele offsets die uit het Python script komen in te voeren in het “Go to”-tekstveld, is de bijbehorende hexadecimale string te vinden.

Vervolgens kan worden gecontroleerd of de bijbehorende hexadecimale string correct is en voldoet aan het ingevoerde karakter en de bijbehorende encoderingsmethode. Voor het karakter Pile of Poo is bijvoorbeeld deze website gebruikt om de hexadecimale uitvoer te controleren: <http://www.fileformat.info/info/unicode/char/1F4A9/charset_support.htm>.

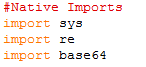
Voorbeeld Pile of Poo (UTF32\_BE):



In bovenstaande afbeelding is op de specifieke offset de hex-string die correspondeert aan de hex-string van Pile of Poo weergegeven, namelijk: 00 01 F4 A9.

## 3.3 Uitleg Python script

Het Python script is te vinden in de bijbehorende Github-repository van groep *4*. Onderstaand worden delen van het script kort uitgelegd.

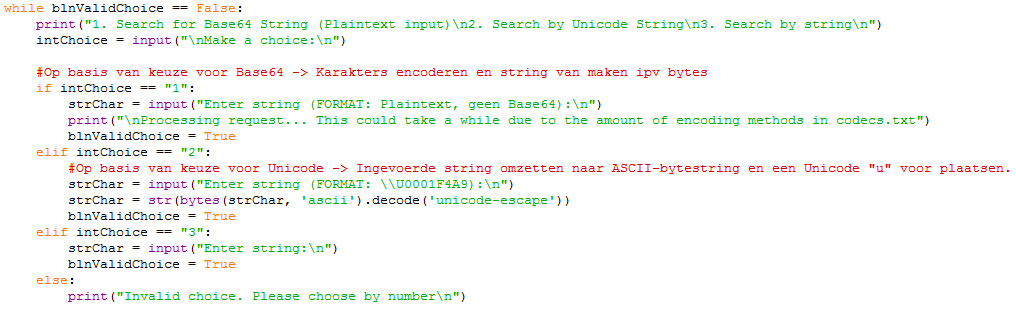


Bovenstaande imports worden op hun beurt gebruikt voor:

* Sys: Verkrijgen van default encoding binnen Python;
* Re: Controleren van regex-strings aan de hand van in het script gedefinieerde gegevens;
* Base64: Converteren van ingevoerde string naar een Base64 string.



Bovenstaand wordt een bestand gelezen in binary mode. Dit houdt in dat het bestand en de inhoud van het bestand behandeld worden als binair object. Hier is voor gekozen om een zo nauwkeurig mogelijk resultaat te bereiken.



In bovenstaande afbeelding is te zien dat de gebruiker keuzes heeft om gebruik te maken van verschillende functionaliteiten binnen de applicatie:

* 1. Search for Base64 String: Bij deze keuze wordt gevraagd om een plaintext string als invoer. Dit zal later geconverteerd worden naar een Base64-string. Deze Base64-string zal daarna nog geëncodeerd worden. Deze optie wordt gebruikt bij het zoeken naar “Jeroen”.
* 2. Search by Unicode String: Deze zoekoptie maakt het mogelijk om naar karakters als Pile of Poo te zoeken. Python kan standaard geen karakter als Pile of Poo verwerken, dus wordt er gebruik gemaakt van de Unicode-notatie van dit karakter. In dit gedeelte wordt de Unicode string geconverteerd naar een Unicode binair object en vervolgens weer terug naar een string (heeft als toegevoegde waarde om de “u” toe te voegen).
* 3. Search by string. Hier kan een gebruiker elke andere normale zoekstring invullen. Deze optie wordt gebruikt bij het zoeken naar het woord Zuyd en Double Happiness.



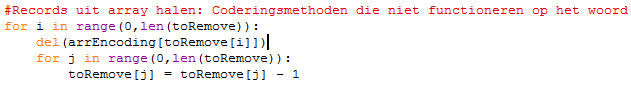
In dit script is het mogelijk om custom encoderingsmethoden (die door Python worden ondersteund) te gebruiken. Deze worden gedefinieerd in codecs.txt die ook is bijgevoegd aan deze Github-repository. Het bestand wordt per regel uitgelezen en toegevoegd aan het array *arrEncoding*.



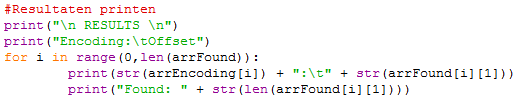
Het subscript in de bovenstaande afbeelding vervult de belangrijkste functie van het Python script om de uitkomsten te genereren. Er wordt een *for-loop* uitgevoerd om alle encoderingsmethoden te verwerken. Als de gebruiker van het script heeft gekozen voor optie 1, worden er extra taken uitgevoerd om de initiële string te vertalen via base64 naar een binaire string. De ingevoerde string wordt geconverteerd naar een base64-geëncodeerde string aan de hand van alle mogelijke encoderingsmethoden die in *codecs.txt* zijn gedefinieerd. Vervolgens wordt de base64-geconverteerde string geconverteerd naar een binaire string aan de hand van alle mogelijke encoderingsmethoden die in *codecs.txt* zijn gedefinieerd.

Met het resultaat in de vorm van een regex-string wordt gecontroleerd of de binaire string bestaat in het bestand. Als dit het geval is wordt de hexadecimale offset opgeslagen in een array en wordt de offset toegevoegd aan een set dat gebruikt wordt om duplicaten binnen een tupel te voorkomen. Als de gebruiker kiest voor de overige opties wordt enkel de binaire string op alle mogelijke encoderingsmethoden gecontroleerd. Er is hier geen sprake van extra codering.

Als er waardes zijn gevonden worden deze toegevoegd aan het 2D-array *arrFound*. Als dit niet het geval is wordt de huidige *i* (index van *arrEncoding*) opgenomen in een array (*toRemove)* die gegevens bevat om bepaalde records uit arrays te verwijderen. Dit gebeurt ook als er een fout optreedt bij het encoderen van een string.



In bovenstaande *for-loop* worden de encoderingsmethoden waarmee de string niet geëncodeerd kon worden verwijderd uit het array *arrEncodering*.



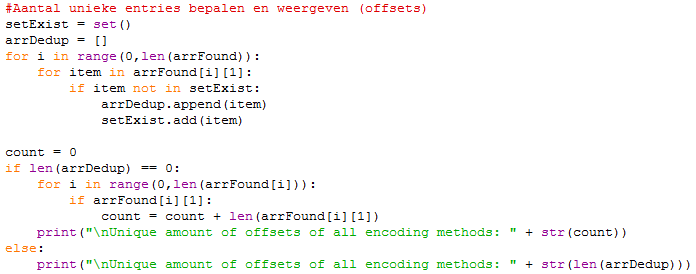
In dit gedeelte worden de resultaten afgedrukt in het formaat:

*Encoding: Offset:*

*Encoderingsmethode: [encoderingsmethode] [array van offsets]*

*Found: [Aantal offets in array]*

*Etc.*



Als laatste wordt voor alle encoderingsmethoden die een resultaat hebben opgeleverd bepaald of er sprake is van duplicaten. Er wordt voor het gehele array gecontroleerd of een specifiek item uit het array al is toegevoegd aan een set. Als dit niet het geval is wordt de offset toegevoegd aan *arrDedup*.

Als laatste wordt bepaald of er sprake is van duplicaten. Als dit niet het geval is, wordt de lengte van het sub-array *arrFound[i][1]* opgeteld bij de huidige waarde voor alle waarden in *arrFound[i]*. Dit wordt vervolgens het totaal aantal unieke offsets van alle encoderingsmethoden voor de ingevoerde string.

Als er wel sprake is van waardes binnen *arrDedup* wordt de lengte van dit array gebruikt om het totaal aantal unieke offsets van alle encoderingsmethoden bepaald voor de ingevoerde string.