# **Encryptor():**

## Introductie

De module genaamd encryptor.py bevat een class met de naam Encryptor.

Deze class word gebruikt om met behulp van een computerversleutelingstechniek die AES (Advanced Encryption Standard) heet de nummer plaat informatie te encrypten en decrypten.

## Toepassing

Er zijn twee verschillende soorten functies beschikbaar in de class Encryptor(). De eerste functie is genoemd “encrypt()”en die zorgt voor de encryptie.

Om de functie te kunnen gebruiken heb je eerst de nummer plaat informatie van een auto nodig. Natuurlijk is het ook mogelijk om je eigen gegevens in de functie te gebruiken, maar het uiteindelijke doel zal zijn voor het encrypten van de nummer plaat gegevens.

Eenmaal de nummerplaat gegevens bekend zijn zal hij moeten worden toevoegd bij de module route.py, in de home()-functie, binnen de laatste if-statement waar er staat:

encoded = encryptor.encrypt(str(car\_info))

De informatie moet worden geplaatst waar er “car\_info” staat. Daarna worden de gegevens verwerkt.

De tweede functie is genoemd “decrypt()” en die zorgt voor de decryptie.

Om deze functie te gebruiken heb je eerst de secret key nodig. Er is gebruikt gemaakt van een eigen secret key in plaats van een randomly generated key, omdat het uiteindelijk handiger was. De secret key hoeft dan niet constant te worden verandert per encrypted data. De secret key is in de module route.py en als volgt uit: SECRET = bytes(b’\*\xd0c\xaeIz\xa3<Y\xf7O\x14\x9el\xc4\xfa’).

## Uitwerking

Het moment dat je encryptor = Encryptor() aanroept word er ook gebruik gemaakt van de \_\_init\_\_ functie. Die zorgt ervoor dat de volgende functies binnen de class kunnen worden gebruikt met behulp van het “self”-keyword:

- encoded  
- decoded  
- secret  
- block\_size  
- padding

Zoals eerder vermeld bestaat de class uit twee verschillende functies: een encrypt-functie en een decrypt-functie.

### encrypt(self, private\_info)

De encryptie AES die gebruikt word binnen deze functie is geimporteerd via Crypto.Cipher en om die te kunnen krijgen moet je “pycrypto” downloaden in je Python. AES maakt gebruik van een precies hoeveelheid aantal bytes. Dit betekent dat als er één bit te weinig is tijdens het encrypten dan lukt het hele proces niet. Daarom word er gebruik gemaakt van padding. Die zorgt ervoor dat de bytes precies genoeg zijn om te kunnen encrypten door de afwijkende bits erin te voegen.

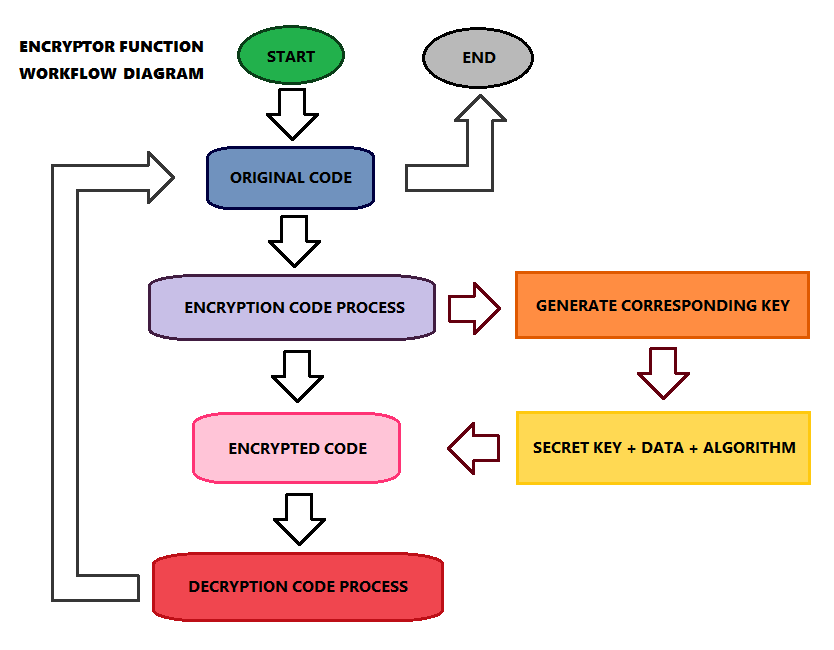
Er is ook gebruik gemaakt van de geïmporteerde base64 die helpt met het proces van “encoding.” Dat is niet te vergelijken met encrypten. Informatie kan niet zomaar worden encrypted en moet eerst in een bepaalde manier worden verandert en opgeslagen voordat die proces begint. Dat heet encoding en met behulp van base64 is dat mogelijk om te doen.

De algoritme die gebruikt word voor het encrypten is opgeslagen in de functie “cipher”. Binnen die functie word de zelf gekozen secret key geïmplementeerd, waardoor er een algoritme word gemaakt die samenhangt met diezelfde key.

### decrypt(self, encrypted\_string)

De decrypt()-functie gebruikt veel van dezelfde processen als de encrypt()-functie en is er hierdoor weinig te vertellen, behalve de verschillen tussen de twee functies.  
Tijdens het decode-proces moet eerst de padding worden weggehaald of populairder gezegd “weggestript”. Daardoor heet de functie ook .rstrip. Python kan de functie .rstrip niet herkennen en is er daarom gebruik gemaakt van .decode(UTF-8) om dat wel mogelijk te maken. De padding moet natuurlijk weggehaald worden omdat de ingevoerde bits anders namelijk de decryptie kunnen verstoren en een foute uitvoer afgeven.   
De key die gebruikt word is natuurlijk ook de zelf-gekozen secret key, afkomstig uit de route.py module.

Op de volgende pagina is een workflow-chart gemaakt die op een simpel manier de proces aanduidt (zie afbeelding 1.2).



Afbeelding 1.2