EXAMPLE

We can also use **QRCode** class to create a QR Code and change its details. It takes the following parameters:

- **Version:** This parameter is an integer from 1 to 40 that controls the size of the QR Code (the smallest, version 1, is a 21×21 matrix).
- **error_correction:** This parameter controls the error correction used for the QR Code. There are following four constants available for this :
    - ***qrcode.constants.ERROR_CORRECT_L* :** About 7% or fewer errors can be corrected.
    - ***qrcode.constants.ERROR_CORRECT_M*** (default) **:** About 15% or fewer errors can be corrected.
    - ***qrcode.constants.ERROR_CORRECT_Q*:** About 25% or fewer errors can be corrected.
    - ***qrcode.constants.ERROR_CORRECT_H*:** About 30% or fewer errors can be corrected.
- **box_size:** This parameter controls how many pixels each “box” of the QR code is.
- **border:** The border parameter controls how many boxes thick the border should be (the default is 4, which is the minimum in the specification).
- **add_data():** This method is used to add data to the QRCode object. It takes the data to be encoded as a parameter.
- **make():** This method with **(fit=True)** ensures that the entire dimension of the QR Code is utilized, even if our input data could fit into less number of boxes.
- **make_image():** This method is used to convert the QRCode object into an image file. It takes the ***fill_color*** and ***back_color*** optional parameters to set the foreground and background color.

Below is the implementation:

- Python3

```
# Importing library
import qrcode


# Data to encode
data = "GeeksforGeeks"


# Creating an instance of QRCode class
qr = qrcode.QRCode(version = 1,
                   box_size = 10,
                   border = 5)


# Adding data to the instance 'qr'
qr.add_data(data)


qr.make(fit = True)
img = qr.make_image(fill_color = 'red',
                    back_color = 'white')
```

```
img.save('MyQRCode2.png')
```

Output :