**Start les** (5 min)

* Leerdoel: Een programma schrijven

**Uitleg Binaire code** (10 min)

* Binaire getallen herhalen. ( Zijn besproken in het hoorcollege)
* Herhaling recursieve oplossing methode op papier dat in hoorcollege langs is gekomen.

**Opdrachten 1 + 2**

* Studenten maken zelfstandig de opdrachten. Docent begeleid waar nodig en geeft feedback.
* Bespreek oplossingen voordat studenten aan ternary systeem beginnen.

**Opdrachten 3 - 7**

* Studenten maken zelfstandig de opdrachten. Docent begeleid waar nodig en geeft feedback.
* Bespreek oplossingen

**Afronding.**

* Wijs ze op de binary opdrachten in het huiswerk.

**Antwoorden**

def main():

    print(binary\_to\_num("01001001"))

def testing():

    assert num\_to\_binary(0) == ""

    assert num\_to\_binary(42) == "101010"

    assert binary\_to\_num("") == 0

    assert binary\_to\_num("101010") == 42

#opdracht 1

def is\_odd(n):

    return n % 2 == 0

#opdracht 2

def num\_to\_binary(n):

    """

    """

    if n == 0:

        return ""

    elif n % 2 == 1:

        return num\_to\_binary(n//2) + "1"

    else:

        return num\_to\_binary(n//2) + "0"

#opdrracht 3

def binary\_to\_num(s):

    """

    """

    if s == "":

        return 0

    # als het laatste cijfer een '1' is...

    elif s[-1] == "1":

        return binary\_to\_num(s[:-1]) \* 2 + 1

    else:  # laatste cijfer moet een '0' zijn

        return binary\_to\_num(s[:-1]) \* 2 + 0

#opdracht 4

def increment(s):

    x = num\_to\_binary(binary\_to\_num(s) + 1)

    return "0" \* (len(s) - len(x)) + x

#opdracht 5

def count (s, n):

    print (s)

    if n == 0:

        return

    count(increment(s), n-1)

#opdracht 6

def num\_to\_ternary(n):

    """

    """

    if n == 0:

        return ""

    return num\_to\_ternary(n//3) + str(n % 3)

#opdracht 7

def ternary\_to\_num(s):

    """

    """

    if s == "":

        return 0

    return ternary\_to\_num(s[:-1]) \* 3 + int(s[-1])

**Oplossing.**

Deze oplossing gebruikt het idee dat de letter ‘e’ de meest voorkomende letter is in de nederlandse taal. Het telt welke letter in de code het meeste voorkomt, gaat er dan vanuit dat dit de ‘e’ is en berekent de gebruikt rotatie.

def rot(c, n):

    hoofdletters = "ABCDEFGHIJKLMNOPQRSTUVWXYZ"

    letters = "abcdefghijklmnopqrstuvwxyz"

    if c in hoofdletters:

        c = chr(ord(c) + n)

        if c not in hoofdletters:

            c = chr(ord(c) - 26)

    elif c in letters:

        c = chr(ord(c) + n)

        if c not in letters:

            c = chr(ord(c) - 26)

    return c

def encipher(s, n):

    if len(s) == 0:

        return s

    else:

        return rot(s[0], n) + encipher(s[1:], n)

def count\_letter(s, c):

    if len(s) == 0:

        return 0

    elif s[0] == c :

        return 1 + count\_letter(s[1:], c)

    else:

        return count\_letter(s[1:], c)

def count\_letters(alphabet, s):

    if len(alphabet)==0:

        return[]

    return [count\_letter(s, alphabet[0])] + count\_letters(alphabet[1:], s)

def max(L):

    if len(L) == 1:

        return L[0]

    M = max(L[1:])  # De max van de REST van L

    if L[0] > M:

        return L[0]

    else:

        return M

def decipher(s):

    letters = "abcdefghijklmnopqrstuvwxyz"

    lc = count\_letters(letters, s)

    freq\_letter\_index = lc.index(max(lc))

    if 5 - freq\_letter\_index >= 0:

        rotation = (4 - freq\_letter\_index)

    else:

        rotation = 26 + (4 - freq\_letter\_index)

    sdecoded = encipher(s, rotation)

    return sdecoded

print(decipher("Bzdrzqbhiedq? Hj gda khdudq Bzdrzqrzkzcd."))