GATHAN OKTARIANSYAH

- 120450010
- gathan.120450010@student.itera.ac.id

JURNAL PRAKTIKUM DAN LATIHAN

PEMROGRAMAN BERBASIS FUNGSI

DOSEN PENGAMPU

Riksa Meidy Karim, M.Sc

Amalya Citra Pradana, M.Sc

DAFTAR ISI

1 JURNAL PRAKTIKUM MODUL 1

2 JURNAL PRAKTIKUM MODUL 2

3 LATIHAN YANG TELAH DIBAHAS

Gathan Oktariansyah / 120450010 / RB / Jurnal

In [40]:
```
tanggal = input('Tanggal Pinjam:')
durasi = int( input('Durasi Pinjam: '))

kategoris = {
    1: 100,
    2: 200,
    3: 250,
    4: 300,
    5: 500
}
```

In [41]:
```
def dtl (s_tgl):
    return [ int(k) for k in s_tgl.split('-')]

def is_cm (tgl_p,d,c):
    return tgl_p[2]+ d > c

def thn_back (tgl_p,d,c):
    return tgl_p[0]+1 if ( is_cm(tgl_p,d,c) and tgl_p[1] == 12) else tgl_p[0]

def bln_back (tgl_p,d,c):
    return ( tgl_p[1] % 12 )+1 if is_cm(tgl_p,d,c) else (tgl_p[1])

def tgl_back (tgl_p,d,c):
    return tgl_p[2] + d - c if is_cm(tgl_p,d,c) else tgl_p[2] + d

def is_awal_abad(thn):
    return thn % 100 == 0

def kabisat (thn):
    return(is_awal_abad(thn) and thn % 400 == 0) or (not is_awal_abad(thn) and thn %

def dec_c(t):
    return 30 +( t[1]%2 if t[1]<= 8 else abs((t[1]%2)-1)) if t[1] !=  2 else(29 if k

def wkt_kembali (tgl_p,d):
    return [thn_back(tgl_p,d, dec_c(tgl_p)),bln_back(tgl_p,d, dec_c(tgl_p)),tgl_back
```

In [42]:
```
tgl_p = dtl(tanggal)
wkt_kembali(tgl_p,durasi)
```

Out[42]:
```
[2020, 1, 27]
```

In [43]:
```
sewaan_all = [ [1,5], [2,3], [3,0], [4,1], [5,2]]

def calc_biaya_per_kategori(kategori,sewaan):
    return sewaan[1] * kategoris.get( sewaan[0])

def calc_all_biaya(kategori,sewaan_all,durasi):
    return sum( [ calc_biaya_per_kategori(kategori,sewaan) for sewaan in sewaan_all]
```

In [44]:
```
calc_all_biaya(kategoris,sewaan_all,durasi)
```

Out[44]:
```
2400
```

No.1

In [1]:
```
import functools

def ulangi_120450010(a,n):
    return list(map(lambda _: a, range(0,n))) #rumus bilangan skalar

ulangi_120450010(10,4)
```

Out[1]:
```
[10, 10, 10, 10]
```

No.2

In [1]:
```
n = 5 #panjang deret
deret = list(map(lambda x: ((-1)**(x+1)) * (1/(2**(x+1))),range(1,n+1)))
print(deret)
```

```
[0.25, -0.125, 0.0625, -0.03125, 0.015625]
```

In [2]:
```
b = range(1,n+1)

def pola_deret(x):
    return ((-1)**(x)) * (1/(2**(x+1)))
#digunakan rumus yang sama dengan tugas pendahuluan, lalu ditambahkan 1 agar mendapa
print(list(map(pola_deret, b)))
```

```
[-0.25, 0.125, -0.0625, 0.03125, -0.015625]
```

No.3

In [3]:
```
from functools import reduce #import reduce
```

In [4]:
```
print(reduce(lambda x,y: x+y, deret, 0)) #reduce harus memiliki dua argumen di lamba
```

```
0.171875
```

No.4

In [6]:
```
filename = 'DNA.txt'
dat = open(filename, 'r').read() #membaca file txt dan menyimpannya dalam variable "
dat = dat[:-1] #[:-1] digunakan untuk menghilangkan "\n" pada data
sec = 'ACT'
dat
```

Out[6]:
```
'TGTCTTCCGGCTGAGCGGTTCCTAACCAGCAGACTGATACTGGTCGAATATCGACGGGCAAGAGCCCTGGGATTGATGCGTTT
CACCATGCGCGTCTCAGTGCAGGCAGGAATGCAGAGCTTACTTCAAACTAGTTACTGGCAAAAAATACAAATTTTTTCGATCGA
CCTTGAGTTTATTCATTACCGCACAGTCTTTTACCGCACCTGTTACCGCACATCCGTAAGTTTACCGCACGTTACCGCACTACC
TCTCTATATTACCGCACTTCGTTTACCGCACGCTGAGGAACGGTTACCGCACTTACCGCACCACAAGGTGCGTGCTCTGTTATT
ACCGCACCACCATTACCGCACGCACTTTTATTACCGCACCAGGGCACAGCCACGTAGGGTAGCGTCGTTCTCACTGTATTGCGG
CGACGGTCGTAATTTACCGCATTACCGCACCACTCGTTAGCTTACCGCACCTAGGGTTGTTACCGCACGACTTACCGCACAGCC
GTTACCGCACGTGTTACTTGACGCTCTAACTCCCACTCATATCAGTCTTATTACCGCACACTGGGCTTACCGCACCCGCACCTT
```

```
AAGTAGGCAGTTACCGCACGTATTACCGCACGTAATTACCGCACACCTGTAAAGGCAGGGTAAAGTACAGACTTACCGCTTACC
GCACGGTTGCACCACGACAAATCTAACGTTAGGTACGTTACCGCACGGGAAATTACCGCACTCCAGGGTTTTACCGCACAGATA
TCCATTCGGGAATGTGACCCCTGGAGTGGAGTTGTGCGAAAGATACGGAGTTTTCAAGGGCACACCCAGCTATGTTATTAAGCG
TTACAGTGGCCGCTGCATCATGTCAATGTTCAGGTCATTCTCTATCTTGCTATGTACGAACCCTCGTTAAGAGGGAGTAAGCGA
TCTTTTGACAAAATCGTATGCATGTAGGCGAGGCAATGCCGATTACATTGAACGGCGGGACTTTTCGTATGAGACACCGCGGTT
GAAATATTTTTTTATGCAAGAGCGGGATTGGGCGGAAGGAGACTTAACGCAGTGCCTAGCACTGTTAACTGCGGCATGGCCGGA
TGGACTACCTATTTTGCAGCTCCAGCGTTTGAGTTCCACGTACTGACGGAACAGTCCCGAGATAGGCCATGTGGTCGATCCCAG
TGAGAAATGAGACTCGAGATGCCGGTACCGGTAGCATCACCACATTGCTCCAGTATGATATCAGTCTTCACTGTCAGCAATTAA
TGCAGCGATCTTGAAGAGAGTTATTCATCTCTTATCACCTGACAATAAATCAATTTACCAGTCAAATTCTCTTTAACATCGTGC
CGAACTGCGATGCGTCGTAGTCTAGATTAGGATATATTTTCTTAGCTGGCTTCGATGATTGGCTGTACGCTAAGGTGATTGAAT
TTCGATCTGCATTGGAGCTGTACCCCACCTTGCATGGCATTGACAGCCTAAAGCGTGAAGAATGCAATACAGCTGACAGAAAAA
TAACGGGCTCGATAACGTTCCAAGATTCTGACTTAACGACGGCTAGCGAGCGAGTCATAAATCCCGTCCACACCGGGCAATCGG
GTCGGAGTGGAAAGGGCGGGATTTTATTATTACGTGACGCAGATCTCCGTGTCACTATACTCACATCCTCTCTGTAGATAAAGT
TATACCAACCTCCATATTCTTCTTACGCTAAGTTCGGGCTATCCGAGTCTCGGCCCATAGCAGGAGCACTTTAAGGGAAGTCCT
ATTGCCGAATACAGTACGTTCCCCGCAATATGTTATACTCACCCAAATATGTTAATATGTTATATGTTAAAACGCAGTGTGGGA
ATATGTTAATATGTTATGTGAATATGTTAAAATATGTTAAAATATGTTAACGATGTTAGCCGTGATAAATATGTTATTAACGGC
GTGCGTTAATATGTTAGCGACGACTGGGGGTCAATATGTTAGCCAACTTCCTCAATATGTTAACCGGTTAATATGTTAGTTAAG
ATCAATAAATATGTTAGCTACGTAGACAATAAAAGCATAAGCAATATGTTATAATATGTTAGACAGTTCTCTAACCGATAATAT
GTTAAGGCATACTTAACCAGCGAATGACAGAAATATGTTAATATGTTAAATAAATATGTTAGATAATATGTTACGATATTACCC
GCACATTGCTCCGAATATGAATATGTTAAGGTGGTTCTCCGTATTTAATATTGTGAGAGATAGCTTGTGAGAGATGTTGTTGTG
AGAGAGCTGTGAGAGCTTTGCGAGCCTTTAAATATTGTGAGAGTTATGTAGTCGGCCTGTGAGAGATGTGTGAGAGAGTTAAAT
ATGTTAGTTAGCTGAGAGCTACGCTGTGAGAGGCGAATTGACGTAGTGCTTTTTGTTCTGTGAGAGATGTGTGAGAGAGTGAGA
GCGCGTGTGAGTGTGAGAGTGATTGTGCATGGTCCAGTAAGATGTGAGAGTGTGAGAGTGATCTAACGCTATGTGTGAGAGAGG
GTGTGAGAGGCTGCGTATGAAGCACAAATGTGAGAGTTGTGAGAGATACGTTAAGAGCCCGGAAGCTCGGCATCATAAGCTGAG
CAGATTCAATGTGAGAGGGCGAGCCGACGGTAGGCTGTGAGAGTCATTATTGTGAGAGTCGCGTGGTGTGAGAGTCCCATTTTA
TGTGAGATGTGAGAGCTCTGGGGCTGTGATGTGAGAGAGTACGCCGAAGCGTGTGAGAGTCCTGTGAGAGATTCGGAGGTCTGG
ATGACATTGTGAGAGCCTGCTTACGCGACGTGATGAACGCGACCGACTAGCGACCGCCCACTACTACTCGCAGTTGGTCTAGAG
GCATTGCTTTACTGAAATACGCAGGATGCTTATGACGCTCGCGCCAATACATCGCGCTCGCACTGTATGTCGCTTCACCTTAAT
CCTAAAGCTCAAATATAACGGAAAAAGAGAAATTAGGACGACCGAGGGTCGTCCTCCGGTGGTTTTCACGACTTCGCCAATGGC
GTGCTGCGTCGAAATGTGCTCAAAGCCCCGTAAAGCTCAGACACCATGCAGGAATGGGAATGTGTACCCAGAGATCCCTAGTAA
GAGAGATCCAAGACTTAAAGCCGTTCCGAGAGAGATCTAATCACTAGAGATCTTAACACCAATAGAGATCCTCTAAGAGATCAG
TAGAGATCGCTTTTCAGAGATAGAGATCACTCACCGAGAGATCTTACAGTTTGATATGTCAGTTCGGTTAAAAGCAGAGATCGT
CTGCAGAGATCGGTAGCGTAGAGATCCCGTGTCGTACAAAAACTTAGAGATCAGATCGCGCCTCGAACTGTACTTAGAGATCTA
CATTATCTAAGAGAGAGATCAGAGATCACAAGGCCACACACGACAAAGTTAGAGATCTACACACGATAGGTGGTGCCGAACCTG
AGAGATCCGGGTTTTGAGAGAGATCAAGAGATAGAGATCGTTAGAGAAGAGATCTAGAGATCGCACGGGTTTTGGAGAGATCGT
TCGGGTTTTGTCGGGAAGAGATTAATGCGGTGAGTTAATGCGGGTATAATGCGGCAGATAATGTAATGCGGTCTAATGTAATGC
GGGAGATAATGCGGTGATGAAACTTAATGCGGCTAATGCGGTTAATGCGGTCGAACGCTAATGCGGAGCTAATGCGGGCGTAAC
ATAATGTAATGCGGTTGTCAATATTGTTTTCAATATTAATATTCAATTACAATATTCAAACGCAATATTAAACGGCCGGTAATG
CGGGGTCAATCAATATTTTCGTAATGCGGGGTTAATGCGGTTTTCAATATTATCGGTAATGCGGGAGCTGGCAATATTGGTTTT
GGTAATGCGGTTTAATAATGCGGGGCGACAATATTGGGTAATGCGGATATTTATCAATATTGTGTTTCAATATTTAACACAATA
TTTGCCGTAGGTATGACCTAATTAATGCGGATATTAGGGGCCAATTAATGCGGATCGTAATGCGGGTCGGGCTTATAACAATTA
ATGCGGTCAATATTACTAATGCTAATGCGGCGGACTACAATATTTACAAAAGACTACCAATAATGCGGATAATGCGGTCAATAA
TGCGGAAGATAACGCGGCAATATTGCCCGACAATATTTGACTACACAAGACTACACAATATTCCGTTATTCTGTGCCAACGCCA
GGTCAATGCGTCGAACCAATATTCTTGATTGTGATGCAGACTACACGACTACAATATTTACCCCCGGGACTACATATCCACGAC
TACAGGGCGAGACTACATAGGGACTACAGACTACAACAATTATGGTCACATTAACTCTGCCCGGCGGCTCTTCCCTAAATCTCA
CGTGATGGACTAGACTACACCGACTACAACATACTTTGCAACGACTACAGTACGTTAAGACTACAGGATTCAGACTACACTTGA
TTTCCTGACTACACTTCTGACAACCCGCACATTGCCCGCTAACTCTGATGGCCCCCAGAGACTACATACCATCGAGCGCGACTA
CAGGACTACAGCCGTAGACCCTTTAGACTACACGCCAGGGCCAACTGACACGGATAAGGTCTTTGCCCCGCAAGTGCTCGCCGA
ATGTGATTAATCTCAACATTCCGACCTGCAAGAGCACACGCATTTGATATGGGTATAAGGAAGATCTCGTCCAGCTATAATGTA
CAACATTTCCCCGTCATGACTTGCTACATAAACAGAATAAGACGTGACGTCGCCAATATAAGACGTACTCGATTGACCGTAAAA
TTTTTCTAAGAACCAAAATAAGACGTAAGACGTTCACTTAAATAAGACGTAGGGCGTTACCGATAAGACGTTAAGACGTGGATC
GCCATCGCCCGTGAGTCGCTCTCCCGCATAAGACGTTAAGACGTCCCAATAGTGCTCCCTACACTTTACCGGTGGTAGATAAGA
CGTAGACGTTATAAGACGTCGGGTAAATATAAGACGTTATTCCCAATAAATAAGACGTAATCCCTGTAACACTGGAAGTGATAA
GACGTTTGTTCTAACATAAGACGTTGTAACTGCCCTAACCCTGATAAGACGTTTTAAAAAGTACTATAAGACGTTCGAGGAATG
AGACCATAAGACGTCGTCCCTCCCTCAGCACTGAATTTTTTCGAAGATAAGACGTAAGACGTGTTGGTTTATCGTTAGAAATAA
GACGTACGTTTATAAGACGTAATGGTCATAAGACGTACGTTAAGACGTAATAAGACGTTATCCATCCCAAAATTACACGTCAGA
AATCATGGCAACCGCCGTGATGGAAGAGAGTAGCAACCGACTACATACAGTATACTGTGGGCAGACTCGTTTGTACACCAACAC
TTCCGCCGCCATTATTAAATACGATTGGTGCTTTACGCATCTTGATGACCATGGTTACTCACCTCGGGTGCTGACCCGCCTGTC
TCCTATGACGTCGGGCTCCACTACGGCCCCGTTTCGACAGATAGGGGGGAGTTGACCTCGAATGCGGGTTACTTCGCCTGCCTT
TCGACGAATCGGTATGGCTAGCTTGGACAAGTATAGGATTGGTCTTTCAAGCTGCACTGTTTTGCAGCTTCTAGCGAGATAAGG
CTGAAGCCTCCAGCGATATTGTCCAGTTGGAAAAAAGTTGGAAAAATGGGGGTTTGGAAAAAAGAAAACGCCCGGGTTACACCG
GGGACATAATTGGAAAAAACCAGTTGGAAAAAGCTTAGAAAGCTTGGAAAAAGTCTTGGGAACAATTATTGGAAAAACGATGGG
CGACTGAGAGTTGGAAAAAAAAATTGGAAAAATTGGAAAAACGTGGCTTTGGAAAAATGGAAAAAGATTGGAAAAATTGGAAAA
ACTTTTGGAATTGGAAAAAAAGCCACTGCGGGTGCTTTGGAAAAAATATTTGGAAAAAACTAGCAAAGCGGCATTCTGAGAGA
```

```
TTGGAAAAACGTGCTAAGCTTCTTTGGAAAAAGAATTGGAAAAAAAAAAGCGCACCACTCAGGAAGACATGTCTGGCACTTTAG
CGTTAAAGTTTGGAAAAAACTCCTCCCACATTTGGAAAAATGGAAAAAGAATCGGTTAGAGCGGCACGTGTCATATTGGAAAAA
TACTCAGCGCGTTAGCAGTTGGAAAAAAATGATGACTATGTTTGGAAGACAAGGAGAAAGTCTCCGAACAACATCCATGACAAG
GAGGAGGCTGGACAAGGATTCAGGCTGTTCAGACAAGGAGGGACGACAAGGAAGGACTGTTCAGGCTGGACAAGACAAGGACTG
TTGACAAGGACAGGACAAGGACGAAAGGCTGTTCAGGGACAAGGAAGGACAGGCTGTTCAGAGGACGAGGACGACAAGGAAGGC
TGTTCAGGCTGTTCAGGAGGACGAGGAAGGATGTTCGACAAGAGGACGACAGGCTAGGACGACGAAGAGGACGACTGTTCAGGC
TGTAGGACGAGGACGAAAGGATGTTGACAAGGAGGAGAGGAGGACGAGGAAGGACGAAAGGAGACAAGGAGACAAGGAGACAAG
GAAGGACGAAGGACGAAGGACGAAGGACGAAGGACGAAGGACGAAGGACAGAGGACAGGACGACGAAGGACGAAGGACGAAGGA
CAGGACGAAGGACGAAGGACGAAGGACGACAATCATCAATCATCAATCATCAATCATCAATCATCAATCATCAATCATCAATCA
TCAATCATCAATCATCAATCATCAATCATCAATCATCAATCATCAATCATCAATCATCAATCATCAATCATCAATCATCAATCA
TCAATCATCAATCATCAATCATCAATCATCAATCATCAATCATCAATCATCAATCATCAATCATCAATCATCAATCATCAATCA
TCAATCATCAATCATCAATCATCAATCATCAATCATCAATCAT'
```

In [7]:
```
def append_n(dat, i,n): #digunakan untuk menambahkan karakter n pada i
    return reduce( lambda a,b:a+b , dat[i:i+n] ) #digunakan untuk menghitung jumlah
def remap(dat, seq): #untuk me-remap semua fungsi seq
    return map( lambda x: append_n(dat,x,len(seq)) , range(len(dat) - len(seq) ) )
def count_mer(dat,seq):
    return reduce( lambda a,b: a + (1 if b==seq else 0) , remap(dat,seq), 0)
```

In [8]:
```
append_n(dat,1,3) #menghasilkan panjang seq sesuai dengan index
```

Out[8]:
```
'GTC'
```

In [29]:
```
list(remap(dat, 'ACT'))
len(dat)
```

Out[29]:
```
6930
```

In [9]:
```
sequences = ['A', 'AT', 'GCT', 'AAGCT', 'AGCTA'] #dictionary untuk menyimpan hasil c

def count_all(dat, sequences):
    return map(lambda x: count_mer(dat,x) , sequences) #digunakan untuk menghitung s

res = count_all(dat,sequences) #memanggil semua jumlah seq yang kita cari
print(*res)
```

```
2112 556 86 8 5
```

No.5

In [10]:
```
def komplemen(x):
    return {'A':'T', 'T':'A', 'C':'G', 'G':'C'}.get(x) #dictionary untuk memanggil k

def reverse_komplemen(dat): #untuk mengubah komplemen menjadi reverse
    return map(lambda x:komplemen(x), dat)
```

In [11]:
```
res = reverse_komplemen(dat) #untuk memanggil semua komplemen
print(*res)
```

```
A C A G A A G G C C G A C T C G C C A A G G A T T G G T C G T C T G A C T A T G A C
C A G C T T A T A G C T G C C C G T T C T C G G G A C C C T A A C T A C G C A A A G
T G G T A C G C G C A G A G T C A C G T C C G T C C T T A C G T C T C G A A T G A A
G T T T G A T C A A T G A C C G T T T T T T A T G T T T A A A A A A G C T A G C T G
```

```
G A A C T C A A A T A A G T A A T G G C G T G T C A G A A A A T G G C G T G G A C A
A T G G C G T G T A G G C A T T C A A A T G G C G T G C A A T G G C G T G A T G G A
G A G A T A T A A T G G C G T G A A G C A A A T G G C G T G C G A C T C C T T G C C
A A T G G C G T G A A T G G C G T G G T G T T C C A C G C A C G A G A C A A T A A T
G G C G T G G T G G T A A T G G C G T G C G T G A A A A T A A T G G C G T G G T C C
C G T G T C G G T G C A T C C C A T C G C A G C A A G A G T G A C A T A A C G C C G
C T G C C A G C A T T A A A T G G C G T A A T G G C G T G G T G A G C A A T C G A A
T G G C G T G G A T C C C A A C A A T G G C G T G C T G A A T G G C G T G T C G G C
A A T G G C G T G C A C A A T G A A C T G C G A G A T T G A G G G T G A G T A T A G
T C A G A A T A A T G G C G T G T G A C C C G A A T G G C G T G G G C G T G G A A T
T C A T C C G T C A A T G G C G T G C A T A A T G G C G T G C A T T A A T G G C G T
G T G G A C A T T T C C G T C C C A T T T C A T G T C T G A A T G G C G A A T G G C
G T G C C A A C G T G G T G C T G T T T A G A T T G C A A T C C A T G C A A T G G C
G T G C C C T T T A A T G G C G T G A G G T C C C A A A A T G G C G T G T C T A T A
G G T A A G C C C T T A C A C T G G G G A C C T C A C C T C A A C A C G C T T T C T
A T G C C T C A A A A G T T C C C G T G T G G G T C G A T A C A A T A A T T C G C A
A T G T C A C C G G G C G A C G T A G T A C A G T T A C A A G T C C A G T A A G A G A
T A G A A C G A T A C A T G C T T G G G A G C A A T T C T C C C T C A T T C G C T A
G A A A A C T G T T T A G C A T A C G T A C A T C C G C T C C G T T A C G G C T A
A T G T A A C T T G C C G C C C T G A A A A G C A T A C T C T G T G G C G C C A A C
T T T A T A A A A A A A T A C G T T C T C G C C C T A A C C C G C C T T C C T C T G
A A T T G C G T C A C G G A T C G T G A C A A T T G A C G C C G T A C C G G C C T A
C C T G A T G G A T A A A A C G T C G A G G T C G C A A A C T C A A G G T G C A T G
A C T G C C T T G T C A G G G C T C T A T C C G G T A C A C C A G C T A G G G T C A
C T C T T T A C T C T G A G C T C T A C G G C C A T G G C C A T C G T A G T G G T G
T A A C G A G G T C A T A C T A T A G T C A G A A G T G A C A G T C G T T A A T T A
C G T C G C T A G A A C T T C T C T C A A T A A G T A G A G A A T A G T G G A C T G
T T A T T T A G T T A A A T G G T C A G T T T A A G A G A A A T T G T A G C A C G G
C T T G A C G C T A C G C A G C A T C A G A T C T A A T C C T A T A T A A A A G A A
T C G A C C G A A G C T A C T A A C C G A C A T G C G A T T C C A C T A A C T T A A
A G C T A G A C G T A A C C T C G A C A T G G G G T G G A A C G T A C C G T A A C T
G T C G G A T T T C G C A C T T C T T A C G T T A T G T C G A C T G T C T T T T T A
T T G C C C G A G C T A T T G C A A G G T T C T A A G A C T G A A T T G C T G C C G
A T C G C T C G C T C A G T A T T T A G G G C A G G T G T G G C C C G T T A G C C C
A G C C T C A C C T T T C C C G C C C T A A A A T A A T A A T G C A C T G C G T C T
A G A G G C A C A G T G A T A T G A G T G T A G G A G A G A C A T C T A T T T C A A
T A T G G T T G G A G G T A T A A G A A G A A T G C G A T T C A A G C C C G A T A G
G C T C A G A G C C G G G T A T C G T C C T C G T G A A A T T C C C T T C A G G A T
A A C G G C T T A T G T C A T G C A A G G G G C G T T A T A C A A T A T G A G T G G
G T T T A T A C A A T T A T A C A A T A T A C A A T T T T G C G T C A C A C C C T T
A T A C A A T T A T A C A A T A C A C T T A T A C A A T T T T A T A C A A T T T T A
T A C A A T T G C T A C A A T C G G C A C T A T T T A T A C A A T A A T T G C C G C
A C G C A A T T A T A C A A T C G C T G C T G A C C C C C A G T T A T A C A A T C G
G T T G A A G G A G T T A T A C A A T T G G C C A A T T A T A C A A T C A A T T C T
A G T T A T T T A T A C A A T C G A T G C A T C T G T T A T T T T C G T A T T C G T
T A T A C A A T A T T A T A C A A T C T G T C A A G A G A T T G G C T A T T A T A C
A A T T C C G T A T G A A T T G G T C G C T T A C T G T C T T T A T A C A A T T A T
A C A A T T T A T T T A T A C A A T C T A T T A T A C A A T G C T A T A A T G G G C
G T G T A A C G A G G C T T A T A C T T A T A C A A T T C C A C C A A G A G G C A T
A A A T T A T A A C A C T C T C T A T C G A A C A C T C T C T A C A A C A A C A C T
C T C T C G A C A C T C T C G A A A C G C T C G G A A A T T T A T A A C A C T C T C
A A T A C A T C A G C C G G A C A C T C T C T A C A C A C T C T C T C A A T T T A T
A C A A T C A A T C G A C T C T C G A T G C G A C A C T C T C C G C T T A A C T G C
A T C A C G A A A A A C A A G A C A C T C T C T A C A C A C T C T C A C A C T C T C
G C G C A C A C T C A C A C T C T C A C T A A C A C G T A C C A G G T C A T T C T A
C A C T C T C A C A C T C T C A C T A G A T T G C G A T A C A C A C T C T C T C C C
A C A C T C T C C G A C G C A T A C T T C G T G T T T A C A C T C T C A A C A C T C
T C T A T G C A A T T C T C G G G C C T T C G A G C C G T A G T A T T C G A C T C G
T C T A A G T T A C A C T C T C C C G C T C G G C T G C C A T C C G A C A C T C T C
A G T A A T A A C A C T C T C A G C G C A C C A C A C T C T C A G G G T A A A A T A
C A C T C T A C A C T C T C G A G A C C C C G A C A C T A C A C T C T C T C A T G C
G G C T T C G C A C A C T C T C A G G A C A C T C T C T A A G C C T C C A G A C C T
A C T G T A A C A C T C T C G G A C G A A T G C G C T G C A C T A C T T G C G C T G
G C T G A T C G C T G G C G G G T G A T G A T G A G C G T C A A C C A G A T C T C C
```

```
G T A A C G A A A T G A C T T T A T G C G T C C T A C G A A T A C T G C G A G C G C
G G T T A T G T A G C G C G A G C G T G A C A T A C A G C G A A G T G G A A T T A G
G A T T T C G A G T T T A T A T T G C C T T T T T C T C T T T A A T C C T G C T G G
C T C C C A G C A G G A G G C C A C C A A A A G T G C T G A A G C G G T T A C C G C
A C G A C G C A G C T T T A C A C G A G T T T C G G G G C A T T T C G A G T C T G T
G G T A C G T C C T T A C C C T T A C A C A T G G G T C T C T A G G G A T C A T T C
T C T C T A G G T T C T G A A T T T C G G C A A G G C T C T C T C T A G A T T A G T
G A T C T C T A G A A T T G T G G T T A T C T C T A G G A G A T T C T C T A G T C A
T C T C T A G C G A A A A G T C T C T A T C T C T A G T G A G T G G C T C T C T A G
A A T G T C A A A A C T A T A C A G T C A A G C C A A T T T T C G T C T C T A G C A G
A C G T C T C T A G C C A T C G C A T C T C T A G G G C A C A G C A T G T T T T T G
A A T C T C T A G T C T A G C G C G G A G C T T G A C A T G A A T C T C T A G A T G
T A A T A G A T T C T C T C T C T A G T C T C T A G T G T T C C G G T G T G T G C T
G T T T C A A T C T C T A G A T G T G T G C T A T C C A C C A C G G C T T G G A C T
C T C T A G G C C C A A A A A C T C T C T C T A G T T C T C T A T C T C T A G C A A T
C T C T T C T C T A G A T C T C T A G C G T G C C C A A A A C C T C T C T A G C A A
G C C C A A A A C A G C C C T T C T C T C T A A T T A C G C C A C T C A A T T A C G C C
C A T A T T A C G C C G T C T A T T A C A T T A C G C C A G A T T A C A T T A C G C
C C T C T A T T A C G C C A C T A C T T T G A A T T A C G C C G A T T A C G C C A A
T T A C G C C A G C T T G C G A T T A C G C C T C G A T T A C G C C C G C A T T G T
A T T A C A T T A C G C C A A C A G T T A T A A C A A A A G T T A T A A T T A T A A
G T T A A T G T T A T A A G T T T G C G T T A T A A T T T G C C G G C C A T T A C G
C C C C A G T T A G T T A T A A A A G C A T T A C G C C C C A A T T A C G C C A A A
A G T T A T A A T A G C C A T T A C G C C C T C G A C C G T T A T A A C C A A A A A C
C A T T A C G C C A A A T T A T T A C G C C C C G C T G T T A T A A C C C A T T A C
G C C T A T A A A T A G T T A T A A C A C A A A G T T A T A A A T T G T G T T A T A
A A C G G C A T C C A T A C T G G A T T A A T T A C G C C T A T A A T C C C C G G T
T A A T T A C G C C T A G C A T T A C G C C C A G C C C G A A T A T T G T T A A T T
A C G C C A G T T A T A A T G A T T A C G A T T A C G C C G C C T G A T G T T A T A
A A T G T T T T C T G A T G G T T A T T A C G C C T A T T A C G C C A G T T A T T A
C G C C T T C T A T T G C G C C G T T A T A A C G G G C T G T T A T A A A C T G A T
G T G T T C T G A T G T G T T A T A A G G C A A T A A G A C A C G G T T G C G G T C
C A G T T A C G C A G C T T G G T T A T A A G A A C T A A C A C T A C G T C T G A T
G T G C T G A T G T T A T A A A T G G G G G C C C T G A T G T A T A G G T G C T G A
T G T C C C G C T C T G A T G T A T C C C T G A T G T C T G A T G T T G T T A A T A
C C A G T G T A A T T G A G A C G G G C C G C C G A G A A G G A T T T A G A G T G
C A C T A C C T G A T C T G A T G T G G C T G A T G T T G T A T G A A A C G T T G C
T G A T G T C A T G C A A T T C T G A T G T C C T A A G T C T G A T G T G A A C T A
A A G G A C T G A T G T G A A G A C T G T T G G G C G T G T A A C G G G C G A T T G
A G A C T A C C G G G G G T C T C T G A T G T A T G G T A G C T C G C G C T G A T G
T C C T G A T G T C G G C A T C T G G G A A A T C T G A T G T G C G G T C C C G G T
T G A C T G T G C C T A T T C C A G A A A C G G G G C G T T C A C G A G C G G C T T
A C A C T A A T T A G A G T T G T A A G G C T G G A C G T T C T C G T G T G C G T A
A A C T A T A C C C A T A T T C C T T C T A G A G C A G G T C G A T A T T A C A T G
T T G T A A A G G G G C A G T A C T G A A C G A T G T A T T T G T C T T A T T C T G
C A C T G C A G C G G T T A T A T T C T G C A T G A G C T A A C T G G C A T T T T A
A A A A G A T T C T T G G T T T T A T T C T G C A T T C T G C A A G T G A A T T T A
T T C T G C A T C C C G C A A T G G C T A T T C T G C A A T T C T G C A C C T A G C
G G T A G C G G G C A C T C A G C G A G A G G G C G T A T T C T G C A A T T C T G C
A G G G T T A T C A C G A G G G A T G T G A A A T G G C C A C C A T C T A T T C T G
C A T C T G C A A T A T T C T G C A G C C C A T T T A T A T T C T G C A A T A A G G
G T T A T T T A T T C T G C A T T A G G G A C A T T G T G A C C T T C A C T A T T C
T G C A A A C A A G A T T G T A T T C T G C A A C A T T G A C G G G A T T G G G A C
T A T T C T G C A A A A T T T T T C A T G A T A T T C T G C A A G C T C C T T A C T
C T G G T A T T C T G C A G C A G G G A G G G A G T C G T G A C T T A A A A A A G C
T T C T A T T C T G C A T T C T G C A C A A C C A A A T A G C A A T C T T T A T T C
T G C A T G C A A A T A T T C T G C A T T A C C A G T A T T C T G C A T G C A A T T
C T G C A T T A T T C T G C A A T A G G T A G G G T T T T A A T G T G C A G T C T T
T A G T A C C G T T G G C G G C A C T A C C T T C T C T C A T C G T T G G C T G A T
G T A T G T C A T A T G A C A C C C G T C T G A G C A A A C A T G T G G T T G T G A
A G G C G G C G G T A A T A A T T T A T G C T A A C C A C G A A A T G C G T A G A A
C T A C T G G T A C C A A T G A G T G G A G C C C A C G A C T G G G C G G A C A G A
G G A T A C T G C A G C C C G A G G T G A T G C C G G G G C A A A G C T G T C T A T
C C C C C C T C A A C T G G A G C T T A C G C C C A A T G A A G C G G A C G G A A A
```

```
G C T G C T T A G C C A T A C C G A T C G A A C C T G T T C A T A T C C T A A C C A
G A A A G T T C G A C G T G A C A A A A C G T C G A A G A T C G C T C T A T T C C G
A C T T C G G A G G T C G C T A T A A C A G G T C A A C C T T T T T T C A A C C T T
T T T A C C C C C A A A C C T T T T T T T C T T T T T G C G G G C C C A A T G T G G C C
C C T G T A T T A A C C T T T T T T T G G T C A A C C T T T T T C G A A T C T T T C G
A A C C T T T T T C A G A A C C C T T G T T A A T A A C C T T T T T T G C T A C C C G
C T G A C T C T C A A C C T T T T T T T T T T A A C C T T T T T T A A C C T T T T T G C
A C C G A A A C C T T T T T T A C C T T T T T T C T A A C C T T T T T T A A C C T T T T T T
G A A A A C C T T A A C C T T T T T T T T T T C G G T G A C G C C C A C G A A A C C T T
T T T T A T A A A C C T T T T T T T G A T C G T T T C G C C G T A A G A C T C T C T A
A C C T T T T T G C A C G A T T C G A A G A A A C C T T T T T T C T T A A C C T T T T T
T T T T T T C G C G T G G T G A G T C C T T C T G T A C A G A C C G T G A A A T C G
C A A T T T C A A A C C T T T T T T T G A G G A G G G T G T A A A C C T T T T T T A C C
T T T T T C T T A G C C A A T C T C G C C G T G C A C A G T A T A A C C T T T T T T A
T G A G T C G C G C A A T C G T C A A C C T T T T T T T T A C T A C T G A T A C A A A
C C T T C T G T T C C T C T T T C A G A G G C T T G T T G T A G G T A C T G T T C C
T C C T C C G A C C T G T T C C T A A G T C C G A C A A G T C T G T T C C T C C C T
G C T G T T C C T T C C T G A C A A G T C C G A C C T G T T C T G T T C C T G A C A
A C T G T T C C T G T C C T G T T C C T G C T T T C C G A C A A G T C C C T G T T C
C T T C C T G T C C G A C A A G T C T C C T G C T C C T G C T G T T C C T T C C G A
C A A G T C C G A C A A G T C C T C C T G C T C C T T C C T A C A A G C T G T T C T
C C T G C T G T C C G A T C C T G C T G C T T C T C C T G C T G A C A A G T C C G A
C A T C C T G C T C C T G C T T T C C T A C A A C T G T T C C T C C T C T C C T C C
T G C T C C T T C C T G C T T T C C T C T G T T C C T C T G T T C C T C T G T T C C
T T C C T G C T T C C T G C T T C C T G C T T C C T G C T T C C T G C T T C C T G C
T T C C T G T C T C C T G T C C T G C T G C T T C C T G C T T C C T G C T T C C T G
T C C T G C T T C C T G C T T C C T G C T T C C T G C T G T T A G T A G T T A G T A
G T T A G T A G T T A G T A G T T A G T A G T T A G T A G T T A G T A G T T A G T A
G T T A G T A G T T A G T A G T T A G T A G T T A G T A G T T A G T A G T T A G T A
G T T A G T A G T T A G T A G T T A G T A G T T A G T A G T T A G T A G T T A G T A
G T T A G T A G T T A G T A G T T A G T A G T T A G T A G T T A G T A G T T A G T A
G T T A G T A G T T A G T A G T T A G T A G T T A G T A G T T A G T A G T T A G T A
```

In [12]:

```
import math

def aktivasi(x): #digunakan untuk menghitung fungsi aktivasi
  return 1/(1+math.exp(-x))

def WTi(W,i): #digunakan untuk men-transpose matrix
  return list(map(lambda w:w[i],W))

def WT(W): #digunakan untuk menampung perhitungan
  return list(map(lambda i: WTi(W,i),range(len(W[0]))))

def XW(X,W): #digunakan untuk perhitungan satu input
  return map(lambda w: reduce(lambda a,b:a+b,map(lambda xx,ww:xx*ww,X,w),0),WT(W))

def input_to_hidden(X,W): #menjalankan fungsi aktivasi
  return list(map(lambda x:aktivasi(x),XW(X,W)))

def feed_forward(X,W,M):
  return input_to_hidden(input_to_hidden(X,W),M)
```

In [13]:

```
X = [9,10,-4]
W =[[0.5,0.4],[0.3,0.7],[0,25,0.9]]
M = [[0.34],[0.45]]
#bentuk W harus sama dengan M untuk menyesuaikan pola yang sudah dibuat
feed_forward(X,W,M) #untuk menjalankan fungsi feed_forward
```

Out[13]:

```
[0.5841448681895914]
```

In [ ]:

LATIHAN LECT 9

No. 1

Buat program untuk menghitung deret bilangan prima dari 2 hingga N menggunakan HOF filter dan map

Contoh primes (100): 2 3 5 7 11 13 17 19 23 29 31 37 41 43 47 53 59 61 67 71 73 79 83 89 97

In [1]:
```
L = lambda n:range(2,n)
faktor = lambda n: list(filter(lambda i:n%i==0,  L(n) ) )
primes = lambda n: filter(lambda i:len(faktor(i))==0, range(2,n+1))

print(* primes(100))
```

```
2 3 5 7 11 13 17 19 23 29 31 37 41 43 47 53 59 61 67 71 73 79 83 89 97
```

No.2

Latihan Filter

employee = { 'Nagao':35, 'Ishii':30, 'Kazutomo':20, 'Saito':25, 'Hidemi':29 } Terdapat dictionary employee berisi nama dan umur pegawai, lakukan filter untuk mengetahui pegawai yang berumur > 25 tahun !

In [2]:
```
employee = {
    'Nagao':35,
    'Ishii':30,
    'Kazutomo':20,
    'Saito':25,
    'Hidemi':29
}

print( employee.items())
```

```
dict_items([('Nagao', 35), ('Ishii', 30), ('Kazutomo', 20), ('Saito', 25), ('Hidem
i', 29)])
```

In [3]:
```
print(list(employee.items()))
```

```
[('Nagao', 35), ('Ishii', 30), ('Kazutomo', 20), ('Saito', 25), ('Hidemi', 29)]
```

In [4]:
```
print( * filter(lambda x:x[1] > 25, employee.items()))
```

```
('Nagao', 35) ('Ishii', 30) ('Hidemi', 29)
```

In [5]:
```
filter_by_age = lambda age,employee: list(filter(lambda x:x[1]>25,employee.items()))
print(* map(lambda d:d[0], filter_by_age(25,employee)) )
```

```
Nagao Ishii Hidemi
```

LATIHAN LECT 10

No.1 Buat fungsi mencari jumlah bilangan genap dari list L!

contoh

- L = [2,1,9,10,3,90,15]
- Output = 3

In [6]:
```
from functools import reduce as r
```

In [7]:
```
L = [2,1,9,10,3,90,15]
r(lambda a,b: a + (1 if b % 2 == 0 else 0), L, 0)
```

Out[7]:
```
3
```

In [8]:
```
def addku(a,b):
    res = a+b
    print('a:',a,',b:',b,'a+b: ', res)
    return res
```

In [9]:
```
L = [1,2,3,4,5]
r(addku, L)
```

Out[9]:
```
a: 1 ,b: 2 a+b:  3
a: 3 ,b: 3 a+b:  6
a: 6 ,b: 4 a+b:  10
a: 10 ,b: 5 a+b:  15
15
```

No.2 Buat fungsi untuk menghitung n! Menggunakan reduce!

In [10]:
```
from functools import reduce

n = 4
print(reduce(lambda x,y: x*y, range(1,n+1)) )
```

Out[10]:
```
24
```

No. 3 Hitung euclidian distance dari dua vektor berikut menggunakan higher order function!

In [11]:

```
X = [2,5,6,7,10]
Y = [-2,9,2,-1,10]
euclid = lambda X,Y: r(lambda a,c:a+c, map(lambda x,y: (x-y)**2 ,X,Y) )**0.5
euclid(X,Y)
```

Out[11]:
```
10.583005244258363
```

No.4 Terdapat dictionary employee berisi nama dan umur pegawai, lakukan reduce untuk mengetahui berapa jumlah pegawai yang berumur > 25 tahun !

In [12]:
```
employee = {
'Nagao':35,
'Ishii':30,
'Kazutomo':20,
'Saito':25,
'Hidemi':29
}

cnt_emp = lambda lim,employee: r(lambda a,b: a+1 if b[1]>lim else a, employee.items(
cnt_emp(25,employee)
```

Out[12]:
```
3
```

No.5 Buatlah deret fibonacci menggunakan higher order function!

In [13]:
```
# Non recursive menggunakan reduce

fibo = lambda n: r( lambda a,b: a if b[0] <=1 else a + [ a[ b[0]-1 ] + a[b[0] - 2] ]
                    enumerate( [0,1] + list( range(1,n) ) ) , [0,1] ) if n > 0 else
```

In [14]:
```
for i in range(20):
    print('Fibonacci',i,'->', fibo(i))
```

```
Fibonacci 0 -> [0]
Fibonacci 1 -> [0, 1]
Fibonacci 2 -> [0, 1, 1]
Fibonacci 3 -> [0, 1, 1, 2]
Fibonacci 4 -> [0, 1, 1, 2, 3]
Fibonacci 5 -> [0, 1, 1, 2, 3, 5]
Fibonacci 6 -> [0, 1, 1, 2, 3, 5, 8]
Fibonacci 7 -> [0, 1, 1, 2, 3, 5, 8, 13]
Fibonacci 8 -> [0, 1, 1, 2, 3, 5, 8, 13, 21]
Fibonacci 9 -> [0, 1, 1, 2, 3, 5, 8, 13, 21, 34]
Fibonacci 10 -> [0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55]
Fibonacci 11 -> [0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89]
Fibonacci 12 -> [0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144]
Fibonacci 13 -> [0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233]
Fibonacci 14 -> [0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, 377]
Fibonacci 15 -> [0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, 377, 610]
Fibonacci 16 -> [0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, 377, 610, 987]
```

```
Fibonacci 17 -> [0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, 377, 610, 987, 1
597]
Fibonacci 18 -> [0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, 377, 610, 987, 1
597, 2584]
Fibonacci 19 -> [0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, 377, 610, 987, 1
597, 2584, 4181]
```

In [15]:
```
fibo(10)
```

Out[15]:
```
[0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55]
```

LATIHAN LECT 11

Buatlah program untuk membuat deret fibconacci dari 0 hingga ke N dengan menggunakan fungsi non-rekursif dan rekursif!

In [3]:
```
from functools import reduce as r
```

In [4]:
```
# Non recursive menggunakan reduce

fibo = lambda n: r( lambda a,b: a if b[0] <=1 else a + [ a[ b[0]-1 ] + a[b[0] - 2] ]
                    enumerate( [0,1] + list( range(1,n) ) ) , [0,1] ) if n > 0 else
```

In [5]:
```
for i in range(20):
    print('Fibonacci',i,'->', fibo(i))
```

```
Fibonacci 0 -> [0]
Fibonacci 1 -> [0, 1]
Fibonacci 2 -> [0, 1, 1]
Fibonacci 3 -> [0, 1, 1, 2]
Fibonacci 4 -> [0, 1, 1, 2, 3]
Fibonacci 5 -> [0, 1, 1, 2, 3, 5]
Fibonacci 6 -> [0, 1, 1, 2, 3, 5, 8]
Fibonacci 7 -> [0, 1, 1, 2, 3, 5, 8, 13]
Fibonacci 8 -> [0, 1, 1, 2, 3, 5, 8, 13, 21]
Fibonacci 9 -> [0, 1, 1, 2, 3, 5, 8, 13, 21, 34]
Fibonacci 10 -> [0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55]
Fibonacci 11 -> [0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89]
Fibonacci 12 -> [0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144]
Fibonacci 13 -> [0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233]
Fibonacci 14 -> [0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, 377]
Fibonacci 15 -> [0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, 377, 610]
Fibonacci 16 -> [0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, 377, 610, 987]
Fibonacci 17 -> [0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, 377, 610, 987, 1
597]
Fibonacci 18 -> [0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, 377, 610, 987, 1
597, 2584]
Fibonacci 19 -> [0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, 377, 610, 987, 1
597, 2584, 4181]
```

In [6]:
```
fibo(500)
```

Out[6]:
```
[0,
 1,
 1,
 2,
 3,
 5,
 8,
 13,
 21,
 34,
 55,
 89,
 144,
 233,
 377,
 610,
 987,
 1597,
 2584,
 4181,
 6765,
 10946,
 17711,
 28657,
 46368,
 75025,
 121393,
 196418,
 317811,
 514229,
 832040,
 1346269,
 2178309,
 3524578,
 5702887,
 9227465,
 14930352,
 24157817,
 39088169,
 63245986,
 102334155,
 165580141,
 267914296,
 433494437,
 701408733,
 1134903170,
 1836311903,
 2971215073,
 4807526976,
 7778742049,
 12586269025,
 20365011074,
 32951280099,
 53316291173,
 86267571272,
 139583862445,
 225851433717,
 365435296162,
 591286729879,
 956722026041,
 1548008755920,
 2504730781961,
```

```
4052739537881,
6557470319842,
10610209857723,
17167680177565,
27777890035288,
44945570212853,
72723460248141,
117669030460994,
190392490709135,
308061521170129,
498454011879264,
806515533049393,
1304969544928657,
2111485077978050,
3416454622906707,
5527939700884757,
8944394323791464,
14472334024676221,
23416728348467685,
37889062373143906,
61305790721611591,
99194853094755497,
160500643816367088,
259695496911122585,
420196140727489673,
679891637638612258,
1100087778366101931,
1779979416004714189,
2880067194370816120,
4660046610375530309,
7540113804746346429,
12200160415121876738,
19740274219868223167,
31940434634990099905,
51680708854858323072,
83621143489848422977,
135301852344706746049,
218922995834555169026,
354224848179261915075,
573147844013817084101,
927372692193078999176,
1500520536206896083277,
2427893228399975082453,
3928413764606871165730,
6356306993006846248183,
10284720757613717413913,
16641027750620563662096,
26925748508234281076009,
43566776258854844738105,
70492524767089125814114,
114059301025943970552219,
184551825793033096366333,
298611126818977066918552,
483162952612010163284885,
781774079430987230203437,
1264937032042997393488322,
2046711111473984623691759,
3311648143516982017180081,
5358359254990966640871840,
8670007398507948658051921,
14028366653498915298923761,
22698374052006863956975682,
36726740705505779255899443,
59425114757512643212875125,
```

```
961518554630184224687745 68,
155576970220531065681649693,
251728825683549488150424261,
407305795904080553832073954,
659034621587630041982498215,
106634041749171059581457216 9,
172537503907934063779707038 4,
279171545657105123361164255 3,
451709049565039187140871293 7,
730880595222144310502035549 0,
118258964478718349764290684 27,
191347024000932780814494239 17,
309605988479651130578784923 44,
500953012480583911393279162 61,
810559000960235041972064086 05,
131151201344081895336534324 866,
212207101440105399533740733 471,
343358302784187294870275058 337,
555565404224292694404015791 808,
898923707008479989274290850 145,
145448911123277268367830664 1953,
235341281824125267295259749 2098,
380790192947402535663090413 4051,
616131474771527802958350162 6149,
996921667718930338621440576 0200,
161305314249045814157979073 86349,
260997481020938848020123131 46549,
422302795269984662178102205 32898,
683300276290923510198225336 79447,
110560307156090817237632754 212345,
178890334785183168257455287 891792,
289450641941273985495088042 104137,
468340976726457153752543329 995929,
757791618667731139247631372 100066,
122613259539418829300017470 2095995,
198392421406191943224780607 4196061,
321005680945610772524798077 6292056,
519398102351802715749578685 0488117,
840403783297413488274376762 6780173,
135980188564921620402395544 77268290,
220020566894662969229833221 04048463,
356000755459584589632228765 81316753,
576021322354247558862061986 85365216,
932022077813832148494290752 66681969,
150804340016807970735635273 952047185,
244006547798191185585064349 218729154,
394810887814999156320699623 170776339,
638817435613190341905763972 389505493,
103362832342818949822646359 5560281832,
167244575904137984013222756 7949787325,
270607408246956933835869116 3510069157,
437851984151094917849091873 1459856482,
708459392398051851684960989 4969925639,
114631137654914676953405286 26429782121,
185477076894719862121901385 21399707760,
300108214549634539075306671 47829489881,
485585291444354401197208056 69229197641,
785693505993988940272514728 17058687522,
127127879743834334146972278 486287885163,
205697230343233228174223751 303346572685,
332825110087067562321196029 789634457848,
538522340430300790495419781 092981030533,
871347450517368352816615810 882615488381,
140986979094766914331203559 1975596518914,
```

```
228121724146503749612865140285821200729 5,
369108703241270663944068699483380852620 9,
597230427387774413556933839769202053350 4,
966339130629045077501002539252582905971 3,
156356955801681949105793637902178495932 17,
252990868864586456855893891827436786529 30,
409347824666268405961687529729615282461 47,
662338693530854862817581421557052068990 77,
107168651819712326877926895128666735145 224,
173402521172797813159685037284371942044 301,
280571172992510140037611932413038677189 525,
453973694165307953197296969697410619233 826,
734544867157818093234908902110449296423 351,
118851856132312604643220587180785991565 7177,
192306342848094413966711477391830921208 0528,
311158198980407018609932064572616912773 7705,
503464541828501432576643541964447833981 8233,
814622740808908451186575606537064746755 5938,
131808728263740988376321914850151258073 74171,
213271002344631833494979475503857732749 30109,
345079730608372821871301390354008990823 04280,
558350732953004655366280865857866723572 34389,
903430463561377477237582256211875714395 38669,
146178119651438213260386312206974243796 773058,
236521166007575960984144537828161815236 311727,
382699285659014174244530850035136059033 084785,
619220451666590135228675387863297874269 396512,
100191973732560430947320623789843393330 2481297,
162114018899219444470188162576173180757 1877809,
262305992631779875417508786366016574087 4359106,
424420011530999319887696948942189754844 6236915,
686726004162779195305205735308206328932 0596021,
111114601569377851519290268425039608377 66832936,
179787201985655771049810841955860241270 87428957,
290901803555033622569101110380899849648 54261893,
470689005540689393618911952336760090919 41690850,
761590809095723016188013062717659940567 95952743,
123227981463641240980692501505442003148 737643593,
199387062373213542599493807777207997205 533596336,
322615043836854783580186309282650000354 271239929,
522002106210068326179680117059857997559 804836265,
844617150046923109759866426342507997914 076076194,
136661925625699143593954654340236599547 3880912459,
221123640630391454569941296974487399338 7956988653,
357785566256090598163895951314723998886 1837901112,
578909206886482052733837248289211398224 9794889765,
936694773142572650897733199603935397111 1632790877,
151560398002905470363157044789314679533 61427680642,
245229875317162735452930364749708219244 73060471519,
396790273320068205816087409539022898778 34488152161,
642020148637230941269017774288731118023 07548623680,
103881042195729914708510518382775401680 142036775841,
168083057059453008835412295811648513482 449585399521,
271964099255182923543922814194423915162 591622175362,
440047156314635932379335110006072428645 041207574883,
712011255569818855923257924200496343807 632829750245,
115205841188445478830259303420656877245 2674037325128,
186406966745427364422585095840706511626 0306867075373,
301612807933872843252844399261363388871 2980904400501,
488019774679300207675429495102069900497 3287771475874,
789632582613173050928273894363433289368 6268675876375,
127765235729247325860370338946550318986 59556447352249,
206728493990564630953197728382893647923 45825123228624,
334493729719811956813568067329443966910 05381570580873,
```

```
541222223710376587766765795712337614833512066938094 97,
875715953430188544580333863041781581743565882643903 70,
141693817714056513234709965875411919657707794958199 867,
229265413057075367692743352179590077832064383222590 237,
370959230771131880927453318055001997489772178180790 104,
600224643828207248620196670234592075321836561403380 341,
971183874599339129547649988289594072811608739584170 445,
157140851842754637816784665852418614813344530098755 0786,
254259239302688550771549664681378022094505404057172 1231,
411400091145443188588334330533796636907849934155927 2017,
665659330448131739359883995215174659002355338213099 3248,
107705942159357492794821832574897129591020527236902 65265,
174271875204170666730810232096414595491256061058212 58513,
281977817363528159525632064671311725082276588295115 23778,
456249692567698826256442296767726320573532649353327 82291,
738227509931226985782074361439038045655809237648443 06069,
119447720249892581203851665820676436622934188700177 088360,
193270471243015279782059101964580241188515112465021 394429,
312718191492907860985910767785256677811449301165198 482789,
505988662735923140767969869749836918999964413630219 877218,
818706854228831001753880637535093596811413714795418 360007,
132469551696475414252185050728493051581137812842563 8237225,
214340237119358514427573114482002411262279184322105 6597232,
346809788815833928679758165210495462843416997164669 4834457,
561150025935192443107331279692497874105696181486775 1431689,
907959814751026371787089444902993336949113178651444 6266146,
146910984068621881489442072459549121105480936013821 97697835,
237706965543724518668151016949848454800392253878966 43963981,
384617949612346400157593089409397575905873189892788 41661816,
622324915156070918825744106359246030706265443771754 85625797,
100694286476841731898333719576864360661213863366454 327287613,
162926777992448823780908130212788963731840407743629 812913410,
263621064469290555679241849789653324393054271110084 140201023,
426547842461739379460149980002442288124894678853713 953114433,
690168906931029935139391829792095612517948949963798 093315456,
111671674939276931459954180979453790064284362881751 2046429889,
180688565632379924973893363958663351316079257878131 0139745345,
292360240571656856433847544938117141380363620759882 2186175234,
473048806204036781407740908896780492696442878638013 2325920579,
765409046775693637841588453834897634076806499397895 4512095813,
123845785297973041924932936273167812677324937803590 86838016392,
200386689975542405709091781656657576085005587743380 41350112205,
324232475273515447634024717929825388762330525546971 28188128597,
524619165249057853343116499586482964847336113290351 69538240802,
848851640522573300977141217516308353609666638837322 97726369399,
137347080577163115432025771710279131845700275212767 4672646 4610201,
222232244629420445529739893461909967206666939096499 764990979600,
359579325206583560961765651721890990523672143092672 32255589801,
581811569836004006491505558634099066259034153405766 997246569401,
941390895042587567453271223806288165311401367715034 229502159202,
152320246487859157394477678244038723157043552112080 1226748728603,
246459335992117914139804800624667539688183688883583 5456250887805,
398779582479977071534282478868706262845227240995663668299961 6408,
645238918472094985674087279493373802533410929879247213925 0504213,
104401850095207205720836975836208006537863817087491088222 50120621,
168925741942416704288245703785545386791204910075415809615 00624834,
273327592037623910009082679621753393329068727162906897837 50745455,
442253333980040614297328383407298780120273637238322707452 51370289,
715580926017664524306411063029052173449342364401229605290 02115744,
115783425999770513860373944643635095356961600163955231274 253486033,
187341518601536966291015050946540312701895836604078191803 255601777,
303124944601307480151388995590175408058857436768034323077 509087810,
490466463202844446442404046536715720760753273372111614880 764689587,
793591407804151926593793042126891128819610710140145037958 273777397,
```

```
128405787100699637303619708866360684958036398351225652839038466984,
207764927881114829962999013079049797839997469365240169079731224438 1,
336170714981814467266618721945410482798033867716465834363650711365,
543935642862929297229617735024460280638031337081706003443366295574 6,
880106357844743764496236456969870763436065204798171837807001366711 1,
142404200070767306172585419199433104407409654187987784125036766228 57,
230414835855241682622209064896420180751016174667804967907357369028 9968,
372819035926008988794794484095853285158425828855792752030773669128 25,
603233871781250671417003548992273465909442003523597719936510572027 93,
976052907707259660211798033088126751067867832379390471967284241156 18,
157928677948851033162880158208040021697730983590298819190379481318 411,
255533968719576999184059961516852696804517766828237866387107905434 029,
413462646668428032346940119724892718502248750418536685577487386752 440,
668996615388005031531000081241745415306766517246774551964595292186 469,
108245926205643306387794020096663813380901526766531123754208267893 8909,
175145587744443809540894028220838354911578178491208578950667797112 5378,
283391513950087115928688048317502168292479705257739702704876065006 4287,
458537101694530925469582076538340523204057883748948281655543862118 9665,
741928615644618041398270124855842691496537589006687984360419927125 3952,
120046571733914896686785220139418321470059547275563626601596378924 43617,
194239433298376700826612232625002590619713306176232425037636371636 97569,
314286005032291597513397452764420912089772853451796051639234750561 41186,
508525438330668298340009685389423502709486159628028476676873122198 38755,
822811443362959895853407138153844414799259013079824528316107872759 79941,
133133688169362819419341682354326791750874517270785300499298099495 818696,
215414832505658809004682396169711233230800418578767753330908886771 798637,
348548520675021628424024078524038024981674935849553053830206986267 617333,
563963353180680437428706474693749258212475354428320807161115873039 415970,
912511873855702065852730553217787283194150290277873860991322859307 033303,
147647522703638250328143702791153654140662564470619466815243873234 6449273,
238898710089208456913416758112932382460077593498406852914376159165 3482576,
386546232792846707241560460904086036600740157969026319729620032399 9931849,
625444942882055164154977219017018419060817751467433172643996191565 3414425,
101199117567490187139653767992110445566155790943645949237361622396 53346274,
163743611855695703555151489893812287472237566090389266501761241553 06760699,
264942729423185890694805257885922733038393357034035215739122863949 60106973,
428686341278881594249956747779735020510630923124424482240884105502 66867672,
693629070702067484944762005665657753549024280158459697980006969452 26974645,
112231541198094907919471875344539277405965520328288418022089107495 493842317,
181594448268301656413948075911105052760867948344134387820089804440 720816962,
293825989466396564333419951255644330166833468672422805842178911936 214659279,
475420437734698220747368027166749382927701417016557193662268716376 935476241,
769246427201094785080787978422393713094534885688979999504447628313 150135520,
124466686493579300582815600558914309602223630270553719316671634469 0085611761,
201391329213688779090894398401153680911677118839451719267116397300 3235747281,
325858015707268079673709998960067990513900749110005438583788031769 3321359042,
527249344920956858764604397361221671425577867949457157850904429069 6557106323,
853107360628224938438314396321289661939478617059462596434692460838 9878465365,
138035670554918179720291879368251133336505648500891975428559688990 86435571688,
223346406617740673564123319000380099530453510206838235072028935074 76314037053,
361382077172658853284415198368631232866959158707730210500588624065 62749608741,
584728483790399526848538517369011332397412668914568445572617559140 39063645794,
946110560963058380132953715737642565264371827622298656073206183206 01813254535,
153083904475345790698149223310665389766178449653686710164582374234 640876900329,
247694960571651628711444594884429646292615632415916575771902992555 242690154864,
400778865046997419409593818195095036058794082069603285936485366789 883567055193,
648473825618649048121038413079524682351409714485519861708388359345 126257210057,
104925269066564646753063223127461971841020379655512314764487372613 5009824265250,
169772651628429551565167064435414440076161351104064300935326208548 0136081475307,
274697920694994198318230287562876411917181730759576615699813581161 5145905740557,
444470572323423749883397351998290851993343081863640916635139789709 5281987215864,
719168493018417948201627639561167263910524812623217532334953370871 0427892956421,
116363906534184169808502499155945811590386789448685844897009316058 05709880172285,
188280755836025964628665263112062537981439270711007598130504653145 16137773128706,
```

```
 3046446623702101344371677622680083495718260601596934430275139692032184765330099
 1,
 4929254182062360990658330253800708875532653308707010411580186223483798542642969
 7,
 7975700805764462335030007876480792371250913910303944841855325915515983307973068
 8,
 1290495498782682332568833813028150124678356721901095525343551213899978185061603
 85,
 2088065579359128566071834600676229361803448112931490009529083805451576515858910
 73,
 3378561078141810898640668413704379486481804834832585534872635019351554700920514
 58,
 5466626657500939464712503014380608848285252947764075544401718824803131216779425
 31,
 8845187735642750363353171428084988334767057782596661079274353844154685917699939
 89,
 1431181439314368982806567444246559718305231073036073662367607266895781713447936
 520,
 2315700212878644019141884587055058551781936851295739770295042651311250305217930
 509,
 3746881652193013001948452031301618270087167924331813432662649918207032018665867
 029,
 6062581865071657021090336618356676821869104775627553202957692569518282323883797
 538,
 9809463517264670023038788649658295091956272699959366635620342487725314342549664
 567,
 1587204538233632704412912526801497191382537747558691983857803505724359666643346
 2105,
 2568150889960099706716791391767326700578165017554628647419837754496891100898312
 6672,
 4155355428193732411129703918568823891960702765113320631277641260221250767541658
 8777,
 6723506318153832117846495310336150592538867782667949278697479014718141868439971
 5449,
 1087886174634756452897619922890497448449957054778126990997512027493939263598163
 04226,
 1760236806450139664682269453924112507703843833044921918867259928965753450442160
 19675,
 2848122981084896117579889376814609956153800887823048909864771956459692714040323
 23901,
 4608359787535035782262158830738722463857644720867970828732031885425446164482483
 43576,
 7456482768619931899842048207553332420011445608691019738596803841885138878522806
 67477,
 1206484255615496768210420703829205488386909032955899056732883572731058504300529
 011053,
 1952132532477489958194625524584538730388053593825001030592563956919572392152809
 678530,
 3158616788092986726405046228413744218774962626780900087325447529650630896453338
 689583,
 5110749320570476684599671752998282949163016220605901117918011486570203288606148
 368113,
 8269366108663463411004717981412027167937978847386801205243459016220834185059487
 057696,
 1338011542923394009560438973441031011710099506799270232316147050279103747366563
 5425809,
 2164948153789740350660910771582233728503897391537950352840492951901187165872512
 2483505,
 3502959696713134360221349745023264740213996898337220585156640002180290913239075
 7909314,
 5667907850502874710882260516605498468717894289875170937997132954081478079111588
 0392819,
 9170867547216009071103610261628763208931891188212391523153772956261768992350663
 8302133,
 1483877539771888378198587077823426167764978547808756246115090591034324707146225
 1869495 2,
 2400964294493489285308948103986302488658167666629995398430467886660501606381291
 5699708 5,
 3884841834265377663507535181809728656423146214438751644545558477694826313527516
 7569203 7,
 6285806128758866948816483285796031145081313881068747042976026364355327919908808
 3268912 2,
 1017064796302424461232401846760575980150446009550749868752158484205015423343632
 5083811 59,
 1645645409178311156114050175340179094658577397657624573049761120640548215334513
 3410702 81,
 2662710205480735617346452022100755074809023407208374441801919604845563638678145
 8494
```

```
51440,
 4308355614659046773460502197440934169467600804865999014851680725486111854012659190521721,
 6971065820139782390806954219541689244276624212074373456653600330331675492690805039973161,
 1127942143479882916426745641698262341374422501694037247150528105581778734670346423049488 2,
 1825048725493861155507441063652431265802084922901474592815888138614946283939426927046804 3,
 2952990868973744071934186705350693607176507424595511839966416244196725018609773350096292 5,
 4778039594467605227441627769003124872978592347496986432782304382811671302549200277143096 8,
 7731030463441349299375814474353818480155099772092498272748720627008396321158973627239389 3,
 1250907005790895452681744224335694335313369211958948470553102500982006762370817390438248 61,
 2024010052135030382619325671771076183328879189168198297827974563682846394486714753162187 54,
 3274917057925925835301069896106770518642248401127146768381077064664853156857532143600436 15,
 5298927110060956217920395567877846701971127590295345066209051628347699551344246896762623 69,
 8573844167986882053221465463984617220613375991422491834590128693012552708201779040363059 84,
 1387277127804783827114186103186246392258450358171783690079918032136025225954602593712568353,
 2244661544603472032436332649584708114319787957314032873538930901437280496774780497748874337,
 3631938672408255859550518752770954506578238315485816563618848933573305722729383091461442690,
 5876600217011727891986851402355662620898026272799849437157779835010586219504163589210317027,
 9508538889419983751537370155126617127476264588285666000776628768583891942233546680671759717,
 1538513910643171164352422155748227974837429086108551543793440860359447816173771026988207674 4,
 2489367799585169539506159171260889687585055544937118143871103737217837010397125695055383646 1,
 4027881710228340703858581327009117662422484631045669687664544597577284826570896722043591320 5,
 6517249509813510243364740498270007350007540175982787831535648334795121836968022417098974966 6,
 1054513122004185094722332182527912501243002480702845751920019293237240666353891913914256628 71,
 1706238072985536119058806232354913236243756498301124535073584126716752850050694155624154125 37,
 2760751194989721213781138414882825737486758979003970286993603419953993516404586069538410754 08,
 4466989267975257332839944647237738973730515477305094822067187546670746366455280225162564879 45,
 7227740462964978546621083062120564711217274456309065109060790966624739882859866294700975633 53,
 1169472973094023587946102770935830368494778993361415993112797851329548624931514651986354051298,
 1892247019390521442608211077147886839616506438992322504018876947992022613217501281456451614651,
 3061719992484545030554313848083717208111285432353738497131674799321571238149015933442805665949,
 4953967011875066473162524925231604047727791871346061001150551747313593851366517214899257280600,
 8015687004359611503716838773315321255839077303699799498282226546635165089515533148342062946549,
 1296965401623467797687936369854692530356686917504586049943277829394875894088205036 3
```

```
241320227149,
 209853410205942894805962024718622465594059464787456599977150048405839240303975835115
83383173698,
 339549950368289674574755661704091718629728156537915204971477831345326829712796338748
24703400847,
 549403360574232569380717686422714184223787621325371804948627879751166070016772173866
40808657454 5,
 888953310942522243955473348126805902853515777863287009920105711096492899729568512612
32789975392,
 143835667151675481333619103454952008707730339918865881486873359084765896974634068647
640876549937,
 232730998245927705729166438267632598993081917705194582478883930194415186947590919908
873666525329,
 376566665397603187062785541722584607700812257624060463965757289279181083922224988556
514543075266,
 609297663643530892791951979990217206693894175329255046444641219473596270869815908465
388209600595,
 985864329041134079854737521712801814394706432953315510410398508752777354792040897021
902752675861,
 159516199268466497264668950170301902108860060828257055685503972822637362566185680548
7290962276456,
 258102632172579905250142702341582083548330704123588606726543823697915098045389770250
9193714952317,
 417618831441046402514811652511883985657190764951845662412047796520552460611575450799
6484677228773,
 675721463613626307764954354853466069205521469075434269138591620218467558656965221050
5678392181090,
 109334029505467271027976600736535005486271223402727993155063941673902001926854067185
02163069409863,
 176906175866829901804472036221881612406823370310271420068923103695748757792550589290
07841461590953,
 286240205372297172832448636958416617893094593712999413223987045369650759719404656475
10004531000816,
 463146381239127074636920673180298230299917964023270833292910149065399517511955245765
17845992591769,
 749386586611424247469369310138714848193012557736270246516897194435050277231359902240
27850523592585,
 121253296785055132210628998331901307849293052175954107980980734350044979474331514800
545696516184354,
 196191955446197556957565929345772792668594307949581132632670453793550007197467505024
573547039776939,
 317445252231252689168194927677674100517887360125535240613651188143594986671799019821
19243555961293,
 513637207677450246125760857023446893186481668075116373246321641937144993869266524849
692790595738232,
 831082459908702935293955784701120993704369028200651613859972830080739980541065544674
812034151699525,
 134471966758615318141971664172456788689085069627576798710629447201788497441033206952
4504824747437757,
 217580212749485611671367242642568888059521972447641960096626730209862495495139761419
9316858899137282,
 352052179508100929813338906815025676748607042075218758807256177411650992936172968372
3821683646575039,
 569632392257586541484706149457594564808129014522860718903882907621513488431312729792
3138542545712321,
 921684571765687471298045056272620241556736056598079477711139085033164481367485698164
46960226192287360,
 149131696402327401278275120573021480636486507112094019661502199265467796979879842795
70098768737999681,
 241300153578896148408079626200283504792160112771901967432616107768784245116628412612
17058994930287041,
 390431849981223549686354746773304985428646619883995987094118307034252042096508255407
87157763668286722,
 631732003560119698094434372973588490220806732655897954526734414803036287213136668 02
```

```
00421675859857376 3,
 1022163853541343247780789119746893475649453352539893941620852721837288329309644923427
9137452226686048 5,
 1653895857101462945875223492720481965870260085195791896147587136640324616522781591447
9559128086543424 8,
 2676059710642806193656012612467375441519713437735685837768439858477612945832426514875
8696580313229473 3,
 4329955567744269139531236105187857407389973522931477733916026995117937562355208106323
8255708399772898 1,
 7006015278387075331872487176552328489096869606671635716844668535955505081876346211996
9522887130023714,
 1133597084613134447271848482284309025629966048359864130560049384871348807054284272752
3520797971127752695,
 1834198612451841980590573354049832310520934744426580487728496070230903857873047734872
3216028582577764 9,
 2967795697064976427862421836334141336150900792786444618288545455102252664927320076246
7368282938552910 4,
 4801994309516818408452995190383973646671835537213025106017041525333156522800379742496
9952856876433055 13,
 7769790006581794836315417026718114982822736329999469724305586980435409187727711750121
6689685170288346 17,
 1257178431609861324476841221710208862949457186721249483032262850576856571052809149261
8664254204672140 130,
 2034157432268040808108382924382020361231730819721196455462821548620397489825580324274
0333222721700974 747,
 3291335863877902132585224146092229224181188006442445938495084399197254060878389473535
8997476926373114 877,
 5325493296145942940693607070474249585412918826163642393957905947817651550703969797809
9330699648074089 624,
 8616829160023845073278831216566478809594106832606088332452990347014905611582359271345
8328176574447204 501,
 1394232245616978801397243828704072839500702565876973072641089629483255716228632906915
5576588762225212 94125]
```

In [7]:
```
#Recursive

fibo_rec = lambda n: 0 if n==0 else 1 if (n==1 or n==2) else fibo_rec(n-1) + fibo_re
deret_fibo_rec = lambda n: list(map(lambda i: fibo_rec(i), range(n+1)) )
```

In [ ]:
```
# deret_fibo_rec(500)
```

dapat disimpulkan bahwa apabila N=500, program yang menggunakan fungsi rekursif lebih lama mengeluarkan outputnya dibandingkan non-rekursif

LATIHAN LECT 12

Purity and Immutability

No.1 Ubah Fungsiku menjadi pure function!

In [3]:
```
def fungsiku(L):
    cL = L.copy()
    def check_genap(l):
        return 1 % 2 == 0
```

```
    for i in range(len(L)):
        if check_genap(L[i]):
            L[i] = L[i]/2
        else:
            L[i] = L[i]*n +1
    return L
```

In [4]:
```
n = 3
L = [5,6,7,8]
print(fungsiku(L))
```

```
[16, 19, 22, 25]
```

In [5]:
```
print(L)
```

```
[16, 19, 22, 25]
```

No.2 Ubah fungsiku2 menjadi pure function!

In [6]:
```
def fungsiku2 (L,n):
    cL = L.copy()
    def check_faktor(l):
        return l % n == 0
    for i in range (len(L)):
        if check_faktor(L[i]):
            L[i] = L[i]/2
        else :
            L[i] = L[i]*n + 1
    return L
```

In [7]:
```
n = 3
L = [5,6,7,8]
print(list(fungsiku2(L,n)))
print(L)
```

```
[16, 3.0, 22, 25]
[16, 3.0, 22, 25]
```

No. 3 Apakah isi dalam tupel tup ada yang dapat diubah?

tup = ([3,4,5], 'myname)

In [8]:
```
tup = ([3, 4, 5], 'myname')
print('tuple 1 :', tup)

tup[0]= "9"
print('tuple 2 :', tup)
```

```
tuple 1 : ([3, 4, 5], 'myname')
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
~\AppData\Local\Temp/ipykernel_14708/3784570710.py in <module>
```

```
      2 print('tuple 1 :', tup)
      3
----> 4 tup[0]= "9"
      5 print('tuple 2 :', tup)

TypeError: 'tuple' object does not support item assignment
```

Tup mempunyai sifat immutable, artinya isi tuple tidak bisa kita ubah dan hapus. Namun, dapat diisi dengan berbagai macam nilai dan objek. Jadi, isi pada tuple tup tidak dapat diubah, namun list nya dapat di ubah.

LATIHAN LECT 13

No. 1

Addku = lambda x: x + 10 Powku = lambda x: x*2 *Kurku = lambda x: x - 2* x

a. Buatlah fungsi komposisi menggunakan 3 fungsi diatas yang melakukan hal sebagai berikut secara berurut:

1. Menjumlahkan input dengan nilai 10
2. Mengurangi input dengan 2 kali input nya
3. Mengeluarkan nilai kuadrat dari input nya

B. Buatlah fungsi invers nya!

In [9]:
```
addku =  lambda x: x+10
powku = lambda x: x**2
kurku = lambda x: x-2*x

f_komp = lambda f,g: lambda x: f(g(x))

my_f_kom = f_komp(kurku,f_komp(powku,addku))

my_f_kom(10)
```

Out[9]:
```
-400
```

In [10]:
```
# Invers
inv_addku = lambda x: x-10
inv_powku = lambda x: x**0.5
inv_kurku = lambda x: -1 * x

my_f_kom_inv = f_komp(inv_addku,f_komp(inv_powku,inv_kurku))
my_f_kom_inv(-400)
```

Out[10]:
```
10.0
```

No.2

Universitas di Lampung ITARE, ingin memiliki sistem penentuan golongan UKT dan jumlah biaya UKT yang dibayarkan oleh Mahasiswa berdasarkan Kriteria berikut:

1. Jumlah tanggungan
2. Jumlah token listrik selama 3 bulan terakhir
3. Gaji Orang tua / Penanggung jawab
4. Penerima program KIP-K atau bukan

In [27]:
```
from functools import reduce as r
```

In [28]:
```
mycompose = lambda * funcs: r(lambda f,g: lambda x: f(g(x)), reversed(funcs), lambda
```

In [29]:
```
#Ketentuan Jumlah Tanggungan
def skor1(jtg):
    return 1 if jtg >=5 else 5-jtg
```

In [30]:
```
#Ketentuan token listrik
def skor2(x):
    def rata(X):
        return sum(X) / len(X)

    def l_cond_1(X):
        return [X, [X>100000] ]

    def l_cond_2(X):
        return [X[0] , X[1] + [ X[0] >= 50000] ]

    def to_score2(X):
        return r( lambda a,b: a+ (1 if b==True else 0), X[1], 1 )

    compose_cond = mycompose(rata,l_cond_1,l_cond_2,to_score2)
    return compose_cond(x)

skor2([5000,5000,50000])
```

Out[30]: 1

In [31]:
```
#Ketentuan gaji
def con_1 (X):
    return [X[0], 1 , X[2] , [ X[0] > X[2] [X[1]]] ]

def con_2_to_n(X):
    return [X[0], X[1] + 1 , X[2] , X[3] + [ X[0] > X[2] [X[1]]] ]

def to_score(X):
    return r( lambda a,b: a+ (1 if b==True else 0), X[-1], 2 )

def prep(gj):
    return [gj, 0, list( map( lambda x: x+1000000 , list(range(10,3,-2)) + [3]) )]

def skor3(gaji):
    commpy = mycompose( prep.con_1, *(con_2_to_n for i in range(4)) , to_score )
    return commpy(gaji)
```

In [32]:
```
```

```
#Ketentuan KIP K
def skor4(X=True):
    return 1 if X else 5
```

In [33]:
```
def combineskor(X):
    return X + [map( lambda f,x: f(x), X[1], X[0] )]

def boboti(X):
    return r( lambda a,b:a+b , map( lambda x,y:x*y , X[-1] , [0.2 , 0.3 , 0.2 , 0.3]

def toUKT(X):
    return 750000 + X*500000
```

No.3

Turunan Polinom

In [21]:
```
def splt(dat):
    return dat.replace(' ',' ').replace('-','+-').split('+')
def chdepan(dat):
    return dat[1:] if dat[0]=='' else dat
def eqkan(dat):
    return map( lambda x: x if '^' in x else x+ '^1' if 'x' in x else x+ 'x^0', dat)
def toarr2d(dat):
    return r( lambda a,b:a + [ [float(hurf) for hurf in b.split('x^')] ] ,dat,[] )
def sortdesc(dat):
    return sorted(dat,key=lambda x:x[1],reverse=True)
def culture(dat):
    return map( lambda x: [0,0] if x[1]==0 else [ x[1] * x[0] , x[1]-1 ], dat)
def tostr(dat):
    return map( lambda x: '0' if x[0]==0 else str(x[0]) if x[1]==0 else str(x[0] + '
def prettykan(dat):
    return r( lambda a,b: a+'+' + b if b!='0' else a ,dat,'')
def prettysign(dat):
    return dat.replace('+-',' -').replace('+',' + ')
```

In [22]:
```
dat = '-3x^5 + 2x^2 - 4x + 5'
fss = (splt,chdepan,eqkan,toarr2d,sortdesc,culture,tostr,prettykan,prettysign)
my_turunan = mycompose(*fss)
```

In [23]:
```
splt(dat)
```

Out[23]:
```
['', '-3x^5 ', ' 2x^2 ', '- 4x ', ' 5']
```

No.4

Buatlah fungsi untuk menghitung biaya yang harus dibayar customer pada suatu e-commerce menggunakan higher order function. Buatlah decorator untuk mengeluarkan harga sebelum pajak dan sesudah pajak (pajak = 11%) ! Gunakan decorator untuk menambahkan perhitungan waktu eksekusi!

In [24]:
```
from functools import reduce as r
```

```
keranjang = [
    {'Jumlah Barang' : 5, 'Harga' : 10},
    {'Jumlah Barang' : 7, 'Harga' : 20},
    {'Jumlah Barang' : 20, 'Harga' : 4.5}
]

def pajak_decorator (func):
    def inner(*args , **kwargs):
        res = func(*args, **kwargs)
        print ('Sub Total:', res)
        print ('Pajak:', res * 0.11)
        print ('Total:', res + res*0.11)
        return res
    return inner

import time

def calc_time_decorator(fu):
    def inner (*args, **kwargs):
        waktu_awal = time.time()
        res = fu(*args, **kwargs)
        waktu_akhir = time.time()
        print ('Waktu Eksekusi:', waktu_akhir-waktu_awal)
        return res
    return inner
```

In [25]:

```
@calc_time_decorator
@pajak_decorator

def hitung_pembayaran_1(keranjang):
    return r(lambda a,b: a+ (b['Jumlah Barang']*b['Harga']),keranjang,0) * 1000

hitung_pembayaran_1(keranjang)
```

```
Sub Total: 280000.0
Pajak: 30800.0
Total: 310800.0
Waktu Eksekusi: 0.00099945068359375
```

Out[25]: 280000.0

In [26]:

```
@calc_time_decorator
@pajak_decorator

def hitung_pembayaran_2(keranjang):
    s = 0
    for k in keranjang :
        s = s + k ['Jumlah Barang'] * k ['Harga']
        return s * 1000

hitung_pembayaran_2(keranjang)
```

```
Sub Total: 50000
Pajak: 5500.0
Total: 55500.0
Waktu Eksekusi: 0.0
```

Out[26]: 50000