LAPORAN AKHIR PRAKTIKUM PEMOGRAMAN BERBASIS FUNGSI

```
Nama   : Alfianri Manihuruk
NIM    : 120450088
Kelas  : RB
Matkul : Pemograman Berbasis Fungsi
```

Jurnal Modul 1

```
### 1. Seorang mahasiswa sains data ingin menyewa buku dari sebuah startup yang menyediakan layanan sewa buku, Startup tersebut memiliki ketentuan sewa dengan aturan sebagai berikut:

- a. Harga sewa buku berbeda-beda sesuai dengan kategorinya
- b. Harga sewa buku dihitung berdasarkan jumlah halaman nya
- c. Harga sewa buku dihitung per hari nya
- d. Maksimal durasi sewa adalah 26 hari

Startup tersebut masih dalam tahap awal pengembangan, sehingga ingin melakukan uji coba penye-
waan 5 kategori buku. Berikut rincian kategori nya:

- Kategori 1 : 100 rupiah per lembar per hari
- Kategori 2 : 200 rupiah per lembar per hari
- Kategori 3 : 250 rupiah per lembar per hari
- Kategori 4 : 300 rupiah per lembar per hari
- Kategori 5 : 500 rupiah per lembar per hari

Startup tersebut memerlukan sebuah program untuk:
- menghitung total biaya dari customer
- mencatat tanggal awal sewa, dan durasi hari
- menampilkan informasi kapan tanggal pengembalian buku dari customer

Format input tanggal adalah yyyy-mm-dd
Bantulah startup tersebut membuat program tersebut dengan menggunakan konsep modularisasi!
```

In [ ]:

```
tanggal = input('tanggal pinjam :')
durasi = int( input ('durasi pin: '))
kategoris ={
    1:100,
    2:200,
    3:250,
    4:300,
    5: 500
}
```

In [36]:

```
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 % 4 == 0 )

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   kabisat ( t [0

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(tgl_p, d, dec_c(tgl_p)) ]
```

In [30]:

```
#tanggal="2022-08-1"
#durasi=1

tgl_p = dtl(tanggal)
wkt_kembali(tgl_p, durasi)
```

Out[30]: [2022, 8, 2]

In [27]:
```
sewaan_all = [[1,5], [2,3], [3,0], [4,1], [5,2]]
def calc_biaya_per_kategori(kategoris, sewaan):
    return sewaan[1] * kategoris.get(sewaan[0])
def calc_all_biaya(kategoris, sewaan_all, durasi):
    return sum( [ calc_biaya_per_kategori(kategoris, sewaan) for sewaan in sewaan_all]) * durasi
```

In [28]:
```
calc_all_biaya(kategoris, sewaan_all, durasi)
```

Out[28]: 14400

Jurnal Modul 2

```
### 1. Buatlah sebuah fungsi bernama ulangi_NIM, ulangi memiliki input sebuah bilangan skalar a, dan mengeluarkan vektor 1xn dengan seluruh elemen nya adalah a
```

In [29]:
```
def ulangi_088(a): # membuat fungsi untuk jumlah atau panjang sklar
    # akan me return nilai si x dengan panjangnya tetap nilai tersebut dengan func range
    return list(map(lambda x: a, range(a)))
```

2. Buatlah deret bilangan sebagai berikut dengan input n sebagai panjang deret:

$$\frac{1}{2}, \frac{-1}{4}, \cdots, (-1)^n \frac{1}{2^{n+1}}$$

In [6]:
```
n = 5 # inputan
deret = list(map(lambda x: ((-1)**(x)) * (1/(2**(x+1))), range(0,n))) # Fungsi map untuk iterable n
# Rumus deretnya menggunakan fungsi lambda untuk menghitung deret
#bilangan pecahan diatas dengan batas iterasi 0 sampai n

print(deret)
```

```
[0.5, -0.25, 0.125, -0.0625, 0.03125]
```

In [7]:
```
b = range(1,n+1)

def pola_deret(x):
    return (((-1)**(x+1)) * (1/(2**x)))
```

3. Jumlahkan deret bilangan tersebut!

In [8]:
```
from functools import reduce
# Import library functools.reduce(function, iterable, initializer) dengan initializer adalah opsional.
```

In [9]:
```
print(reduce(lambda x,y: x+y, deret))
# Menyimpan nilai sebelumnya dan menambahkan nilai yang baru dari jumlah deret bilangan
```

```
0.34375
```

4. Sebuah DNA dimodelkan dalam sebuah string menjadi sequence TCGA dan disimpan ke dalam data :

https://drive.google.com/file/d/18C1ylsTXrY9pglqqlhijoS8LYmcxdIjM/view?usp=sharing (https://drive.google.com/file/d/18C1ylsTXrY9pglqqlhijoS8LYmcxdIjM/view?usp=sharing) hitunglah jumlah kemunculan pola berikut pada data tersebut:

In [10]:
```
filename= 'DNA.txt' # kita deklarasikan filenya menjadi sebuah variabel (filename)
dat = open(filename, 'r').read() #membuka file, fungsi dari (r) adalah membaca file tersebut
dat=dat[:-1] # melakukan slicing pada data sehingga memotong indexnya hanya 1
seq = 'ACT' # membuat variabel seq dengan isi string("ACT")
dat #menampilkan data
```

Out[10]:
```
'TGTCTTCCGGCTGAGCGGTTCCTAACCAGCAGACTGATACTGGTCGAATATCGACGGGCAAGAGCCCTGGGATTGATGCGTTTCACCATGCGCGTCTCAGTGCAGGCAGGAATGCAGA
GCTTACTTCAAACTAGTTACTGGCAAAAAATACAAATTTTTTCGATCGACCTTGAGTTTATTCATTACCGCACAGTCTTTTACCGCACCTGTTACCGCACATCCGTAAGTTTACCGCAC
GTTACCGCACTACCTCTCTATATTACCGCACTTCGTTTACCGCACGCTGAGGAACGGTTACCGCACTTACCGCACCACAAGGTGCGTGCTCTGTTATTACCGCACCACCATTACCGCAC
GCACTTTTATTACCGCACCAGGGCACAGCCACGTAGGGTAGCGTCGTTCTCACTGTATTGCGGCGACGGTCGTAATTTACCGCATTACCGCACCACTCGTTAGCTTACCGCACCTAGGG
TTGTTACCGCACGACTTACCGCACAGCCGTTACCGCACGTGTTACTTGACGCTCTAACTCCCACTCATATCAGTCTTATTACCGCACACTGGGCTTACCGCACCCGCACCTTAAGTAGG
CAGTTACCGCACGTATTACCGCACGTAATTACCGCACACCTGTAAAGGCAGGGTAAAGTACAGACTTACCGCTTACCGCACGGTTGCACCACGACAAATCTAACGTTAGGTACGTTACC
GCACGGGAAATTACCGCACTCCAGGGTTTTACCGCACAGATATCCATTCGGGAATGTGACCCCTGGAGTGGAGTTGTGCGAAAGATACGGAGTTTTCAAGGGCACACCCAGCTATGTTA
TTAAGCGTTACAGTGGCCGCTGCATCATGTCAATGTTCAGGTCATTCTCTATCTTGCTATGTACGAACCCTCGTTAAGAGGGAGTAAGCGATCTTTTGACAAAATCGTATGCATGTAGG
CGAGGCAATGCCGATTACATTGAACGGCGGGACTTTTCGTATGAGACACCGCGGTTGAAATATTTTTTTATGCAAGAGCGGGATTGGGCGGAAGGAGACTTAACGCAGTGCCTAGCACT
GTTAACTGCGGCATGGCCGGATGGACTACCTATTTTGCAGCTCCAGCGTTTGAGTTCCACGTACTGACGGAACAGTCCCGAGATAGGCCATGTGGTCGATCCCAGTGAGAAATGAGACT
CGAGATGCCGGTACCGGTAGCATCACCACATTGCTCCAGTATGATATCAGTCTTCACTGTCAGCAATTAATGCAGCGATCTTGAAGAGAGTTATTCATCTCTTATCACCTGACAATAAA
TCAATTTACCAGTCAAATTCTCTTTAACATCGTGCCGAACTGCGATGCGTCGTAGTCTAGATTAGGATATATTTTCTTAGCTGGCTTCGATGATTGGCTGTACGCTAAGGTGATTGAAT
TTCGATCTGCATTGGAGCTGTACCCCACCTTGCATGGCATTGACAGCCTAAAGCGTGAAGAATGCAATACAGCTGACAGAAAAATAACGGGCTCGATAACGTTCCAAGATTCTGACTTA
ACGACGGCTAGCGAGCGAGTCATAAATCCCGTCCACACCGGGCAATCGGGTCGGAGTGGAAAGGGCGGGATTTTATTATTACGTGACGCAGATCTCCGTGTCACTATACTCACATCCTC
TCTGTAGATAAAGTTATACCAACCTCCATATTCTTCTTACGCTAAGTTCGGGCTATCCGAGTCTCGGCCCATAGCAGGAGCACTTTAAGGGAAGTCCTATTGCCGAATACAGTACGTTC
CCCGCAATATGTTATACTCACCCAAATATGTTAATATGTTATATGTTAAAACGCAGTGTGGGAATATGTTAATATGTTATGTGAATATGTTAAAATATGTTAAAATATGTTAACGATGT
TAGCCGTGATAAATATGTTATTAACGGCGTGCGTTAATATGTTAGCGACGACTGGGGGTCAATATGTTAGCCAACTTCCTCAATATGTTAACCGGTTAATATGTTAGTTAAGATCAATA
AATATGTTAGCTACGTAGACAATAAAAGCATAAGCAATATGTTATAATATGTTAGACAGTTCTCTAACCGATAATATGTTAAGGCATACTTAACCAGCGAATGACAGAAATATGTTAAT
ATGTTAAATAAATATGTTAGATAATATGTTACGATATTACCCGCACATTGCTCCGAATATGAATATGTTAAGGTGGTTCTCCGTATTTAATATTGTGAGAGATAGCTTGTGAGAGATGT
TGTTGTGAGAGAGCTGTGAGAGCTTTGCGAGCCTTTAAATATTGTGAGAGTTATGTAGTCGGCCTGTGAGAGATGTGTGAGAGAGTTAAATATGTTAGTTAGCTGAGAGCTACGCTGTG
AGAGGCGAATTGACGTAGTGCTTTTTGTTCTGTGAGAGATGTGTGAGAGTGTGAGAGCGCGTGTGAGTGTGAGAGTGATTGTGCATGGTCCAGTAAGATGTGAGAGTGTGAGAGTGATC
TAACGCTATGTGTGAGAGAGGGTGTGAGAGGCTGCGTATGAAGCACAAATGTGAGAGTTGTGAGAGATACGTTAAGAGCCCGGAAGCTCGGCATCATAAGCTGAGCAGATTCAATGTGA
GAGGGCGAGCCGACGGTAGGCTGTGAGAGTCATTATTGTGAGAGTCGCGTGGTGTGAGAGTCCCATTTTATGTGAGATGTGAGAGCTCTGGGGCTGTGATGTGAGAGAGTACGCCGAAG
CGTGTGAGAGTCCTGTGAGAGATTCGGAGGTCTGGATGACATTGTGAGAGCCTGCTTACGCGACGTGATGAACGCGACCGACTAGCGACCGCCCACTACTACTCGCAGTTGGTCTAGAG
GCATTGCTTTACTGAAATACGCAGGATGCTTATGACGCTCGCGCCAATACATCGCGCTCGCACTGTATGTCGCTTCACCTTAATCCTAAAGCTCAAATATAACGGAAAAAGAGAAATTA
GGACGACCGAGGGTCGTCCTCCGGTGGTTTTCACGACTTCGCCAATGGCGTGCTGCGTCGAAATGTGCTCAAAGCCCCGTAAAGCTCAGACACCATGCAGGAATGGGAATGTGTACCCA
GAGATCCCTAGTAAGAGAGATCCAAGACTTAAAGCCGTTCCGAGAGAGATCTAATCACTAGAGATCTTAACACCAATAGAGATCCTCTAAGAGATCAGTAGAGATCGCTTTTCAGAGAT
AGAGATCACTCACCGAGAGATCTTACAGTTTGATATGTCAGTTCGGTTAAAAGCAGAGATCGTCTGCAGAGATCGGTAGCGTAGAGATCCCGTGTCGTACAAAAACTTAGAGATCAGAT
CGCGCCTCGAACTGTACTTAGAGATCTACATTATCTAAGAGAGAGATCAGAGATCACAAGGCCACACACGACAAAGTTAGAGATCTACACACGATAGGTGGTGCCGAACCTGAGAGATC
CGGGTTTTGAGAGAGATCAAGAGATAGAGATCGTTAGAGAAGAGATCTAGAGATCGCACGGGTTTTGGAGAGATCGTTCGGGTTTTGTCGGGAAGAGATTAATGCGGTGAGTTAATGCG
GGTATAATGCGGCAGATAATGTAATGCGGTCTAATGTAATGCGGGAGATAATGCGGTGATGAAACTTAATGCGGCTAATGCGGTTAATGCGGTCGAACGCTAATGCGGAGCTAATGCGG
GCGTAACATAATGTAATGCGGTTGTCAATATTGTTTTCAATATTAATATTCAATTACAATATTCAAACGCAATATTAAACGGCCGGTAATGCGGGGTCAATCAATATTTTCGTAATGCG
GGGTTAATGCGGTTTTCAATATTATCGGTAATGCGGGAGCTGGCAATATTGGTTTTGGTAATGCGGTTTAATAATGCGGGGCGACAATATTGGGTAATGCGGATATTTATCAATATTGT
GTTTCAATATTTAACACAATATTTGCCGTAGGTATGACCTAATTAATGCGGATATTAGGGGCCAATTAATGCGGATCGTAATGCGGGTCGGGCTTATAACAATTAATGCGGTCAATATT
ACTAATGCTAATGCGGCGGACTACAATATTTACAAAAGACTACCAATAATGCGGATAATGCGGTCAATAATGCGGAAGATAACGCGGCAATATTGCCCGACAATATTTGACTACACAAG
ACTACACAATATTCCGTTATTCTGTGCCAACGCCAGGTCAATGCGTCGAACCAATATTCTTGATTGTGATGCAGACTACACGACTACAATATTTACCCCCGGGACTACATATCCACGAC
TACAGGGCGAGACTACATAGGGACTACAGACTACAACAATTATGGTCACATTAACTCTGCCCGGCGGCTCTTCCCTAAATCTCACGTGATGGACTAGACTACACCGACTACAACATACT
TTGCAACGACTACAGTACGTTAAGACTACAGGATTCAGACTACACTTGATTTCCTGACTACACTTCTGACAACCCGCACATTGCCCGCTAACTCTGATGGCCCCCAGAGACTACATACC
ATCGAGCGCGACTACAGGACTACAGCCGTAGACCCTTTAGACTACACGCCAGGGCCAACTGACACGGATAAGGTCTTTGCCCCGCAAGTGCTCGCCGAATGTGATTAATCTCAACATTC
CGACCTGCAAGAGCACACGCATTTGATATGGGTATAAGGAAGATCTCGTCCAGCTATAATGTACAACATTTCCCCGTCATGACTTGCTACATAAACAGAATAAGACGTGACGTCGCCAA
TATAAGACGTACTCGATTGACCGTAAAATTTTTCTAAGAACCAAAATAAGACGTAAGACGTTCACTTAAATAAGACGTAGGGCGTTACCGATAAGACGTTAAGACGTGGATCGCCATCG
CCCGTGAGTCGCTCTCCCGCATAAGACGTTAAGACGTCCCAATAGTGCTCCCTACACTTTACCGGTGGTAGATAAGACGTAGACGTTATAAGACGTCGGGTAAATATAAGACGTTATTC
CCAATAAATAAGACGTAATCCCTGTAACACTGGAAGTGATAAGACGTTTGTTCTAACATAAGACGTTGTAACTGCCCTAACCCTGATAAGACGTTTTAAAAAGTACTATAAGACGTTCG
AGGAATGAGACCATAAGACGTCGTCCCTCCCTCAGCACTGAATTTTTTCGAAGATAAGACGTAAGACGTGTTGGTTTATCGTTAGAAATAAGACGTACGTTTATAAGACGTAATGGTCA
TAAGACGTACGTTAAGACGTAATAAGACGTTATCCATCCCAAAATTACACGTCAGAAATCATGGCAACCGCCGTGATGGAAGAGAGTAGCAACCGACTACATACAGTATACTGTGGGCA
GACTCGTTTGTACACCAACACTTCCGCCGCCATTATTAAATACGATTGGTGCTTTACGCATCTTGATGACCATGGTTACTCACCTCGGGTGCTGACCCGCCTGTCTCCTATGACGTCGG
GCTCCACTACGGCCCCGTTTCGACAGATAGGGGGGAGTTGACCTCGAATGCGGGTTACTTCGCCTGCCTTTCGACGAATCGGTATGGCTAGCTTGGACAAGTATAGGATTGGTCTTTCA
AGCTGCACTGTTTTGCAGCTTCTAGCGAGATAAGGCTGAAGCCTCCAGCGATATTGTCCAGTTGGAAAAAAGTTGGAAAAATGGGGGTTTGGAAAAAAGAAAACGCCCGGGTTACACCG
GGGACATAATTGGAAAAAACCAGTTGGAAAAAGCTTAGAAAGCTTGGAAAAAGTCTTGGGAACAATTATTGGAAAAACGATGGGCGACTGAGAGTTGGAAAAAAAAATTGGAAAAATTG
GAAAAACGTGGCTTTGGAAAAATGGAAAAAGATTGGAAAAATTGGAAAAACTTTTGGAATTGGAAAAAAAAGCCACTGCGGGTGCTTTGGAAAAAATATTTGGAAAAAACTAGCAAAGC
GGCATTCTGAGAGATTGGAAAAACGTGCTAAGCTTCTTTGGAAAAAGAATTGGAAAAAAAAAAGCGCACCACTCAGGAAGACATGTCTGGCACTTTAGCGTTAAAGTTTGGAAAAAACT
CCTCCCACATTTGGAAAAATGGAAAAAGAATCGGTTAGAGCGGCACGTGTCATATTGGAAAAATACTCAGCGCGTTAGCAGTTGGAAAAAAATGATGACTATGTTTGGAAGACAAGGAG
AAAGTCTCCGAACAACATCCATGACAAGGAGGAGGCTGGACAAGGATTCAGGCTGTTCAGACAAGGAGGGACGACAAGGAAGGACTGTTCAGGCTGGACAAGACAAGGACTGTTGACAA
GGACAGGACAAGGACGAAAGGCTGTTCAGGGACAAGGAAGGACAGGCTGTTCAGAGGACGAGGACGACAAGGAAGGCTGTTCAGGCTGTTCAGGAGGACGAGGAAGGATGTTCGACAAG
AGGACGACAGGCTAGGACGACGAAGAGGACGACTGTTCAGGCTGTAGGACGAGGACGAAAGGATGTTGACAAGGAGGAGAGGAGGACGAGGAAGGACGAAAGGAGACAAGGAGACAAGG
AGACAAGGAAGGACGAAGGACGAAGGACGAAGGACGAAGGACGAAGGACGAAGGACAGAGGACAGGACGACGAAGGACGAAGGACGAAGGACAGGACGAAGGACGAAGGACGAAGGACG
ACAATCATCAATCATCAATCATCAATCATCAATCATCAATCATCAATCATCAATCATCAATCATCAATCATCAATCATCAATCATCAATCATCAATCATCAATCATCAATCATCAATCA
TCAATCATCAATCATCAATCATCAATCATCAATCATCAATCATCAATCATCAATCATCAATCATCAATCATCAATCATCAATCATCAATCATCAATCATCAATCATCAATCATCAATCA
TCAATCATCAATCATCAATCATCAATCAT'
```

In [11]:
```
#memodulkan setiap fungsi yang akan digunakan
""" Fungsi append_n adalah menambahkan nilai array pada nilai akhir dan membagi
data DNA sesuai dengan n yang dibutuhkan, Rumusnya a nilai sebelumnya data DNA,
b nilai yang baru dari data DNA dengan batasan index dat"""
def append_n(dat,i,n):
    return reduce(lambda a,b: a+b, dat [i: i+n])

""" Fungsi remap adalah memetakan kembali dari sequence yang baru dan anyaknya urutan dikurang banyaknya kombinasi""
def remap(dat, seq):
    return map( lambda x: append_n(dat, x, len(seq)), range (len(dat) -  len(seq) + 1))

"""pada fungsi ke-2 remap dihitung jumlah sequence yang muncul """
def count_mer(dat, seq):
    return reduce (lambda a,b : a + (1 if b == seq else 0), remap(dat,seq), 0)

```

In [12]:
```
list(remap(dat, 'ACT'))[-1] # mengubah dna "act" ke dna yang lain
```

Out[12]:
```
'CAT'
```

In [13]:
```
len (seq) # mencari panjang seq
```

Out[13]: 3

In [14]:
```
sequences=['A', 'AT', 'GGT', 'AAGC', ' AGCTA']
def count_all(dat, sequences):
    return map(lambda x: count_mer(dat, x), sequences)
#melakukan perhitungan mer pada setiap sequence yang ada dalam list
res = count_all(dat, sequences)
# merupakan jumla pengembalian berapa kali res terjadi dalam dat dan sequencecs
print(*res)
```

```
2112 557 77 22 0
```

5. Reverse complement dari suatu sequence string DNA memiliki aturan sebagai berikut:

In [15]:
```
def komplemen(x):
    return {'A':'T', 'T':'A', 'C':'G', 'G':'C'}.get(x)#Kamus(dictionary) pengubahan huruf diambil x sesuai dengan ke
def reverse_komplemen(dat):
    return map( lambda x: komplemen(x), dat)#Setiap data yang ada diubah sesuai dengan dictionary pengubahan
```

In [16]:
```
res = reverse_komplemen (dat)
print(*res)
```

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

6. Buatlah fungsi feed-forward

In [17]:
```
import math # library yang digunakan
```

In [19]:
```
#memdulkan setiap fungsi yang ada
def aktivasi (x):
    return 1/ (1+math.exp(-x))
'''# untuk fungsi ini merupakan fungsi aktivasi yang telah ada pada soal yang dimana
    kita hanya menuliskan saya, dengan parameternya adalah x'''

def WTi(W,i): # Index yang tercantum dalam W digunakan didalam list
    return list(map( lambda w:w[i] , W))

def WT(W):
    return list( map( lambda i : WTi(W,i) , range(len(W[0]))) )

def XW(X,W):
    return map( lambda w: reduce( lambda a,b:a+b ,map (lambda xx,ww: xx * ww , X,w),0) ,  WT(W) )
#memuat w ke layer pertama

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

def feed_forward(X,W,M):
    return input_to_hidden(input_to_hidden(X,W),M)#membuat fungsi feed forward
```

In [20]:
```
"""pada bagian ini kita menentukan setiap nilai dari si x, w, dan m"""
x= [9, 10, -1]
w= [[0.5, 0.4], [0.3, 0.7], [0.25, 0.9]]
m= [[0.34], [0.45]]
```

In [21]:
```
feed_forward(x, w, m)
"""kita memanggil fungsi feed_forward, diaman Nilai yang masuk ke neuron di hidden layer
adalah penjumlahan antara perkalian weight dengan nilai yang masuk pada input neuron
setelah itu diaktifkan dengan
fungsi aktivasi(yang tela dimodulkan di atas ). """
```

Out[21]:
```
'kita memanggil fungsi feed_forward, diaman Nilai yang masuk ke neuron di hidden layer\nadalah penjumlahan antara perka
lian weight dengan nilai yang masuk pada input neuron setelah itu diaktifkan dengan\nfungsi aktivasi(yang tela dimodulk
an di atas ). '
```

In [22]:
```
WTi(w,0) # memanggiilfungsi WTI
```

Out[22]:
```
[0.5, 0.3, 0.25]
```

In [23]:
```
WT(w) #Memanggil fungsi WT
```

Out[23]:
```
[[0.5, 0.3, 0.25], [0.4, 0.7, 0.9]]
```

Kumpulan Latihan dan Pembahasan

Pertemuan 7 - Higher Order Function (Map)

```
1. Given List P = [‘a’, ‘k’,’u’, ‘l’ , ‘u’ , ‘p’ , ‘a’ ]
We want to make list of tuples of P like this
P’ = [ (1, ’a’ ) , (3, ‘k’), (5,’u’), (7,’l’ ) , (9,’u’), (11,’p’) , (13,’a’) ]
```

In [50]:
```
P = 'akulupa'
print(*map(lambda p: (p[0]*2+1 , p[1]) , enumerate(P)))
```

```
(1, 'a') (3, 'k') (5, 'u') (7, 'l') (9, 'u') (11, 'p') (13, 'a')
```

2. Terdapat bilangan B B = 24 Petakan B menjadi list faktor nya! B’ = [1,2,3,4,6,8,12,24]

In [52]:
```
B = 24
two = map( lambda b: b+1 if B % (b+1) == 0 else -1, range(B))
def dua(two):
    x = []
    for t in two:
        if t != -1:
            x.append(t)
    return x
print( dua(two))

```

```
[1, 2, 3, 4, 6, 8, 12, 24]
```

```
3. Diketahui matriks A,B,C sebagai berikut
A = [ [3 , 4] , [ 5, 6 ] ]
B = [ [1,2] , [ 7 , 8] ]
C = AB
Buatlah program untuk menghitung determinan matriks C menggunakan HOF map!

```

In [53]:
```
A =[[3,4], [5,6]]
B = [[1,2],[7,8]]
C=list(map(lambda ra, rb:list(map(lambda raa,rbb: raa+rbb, ra, rb)), A, B))
def dett (C):
    return C[0][0]*C[1][1]-C[0][1]*C[1][0]
print(dett(C))
```

```
-16
```

Pertemuan 9 Higher order Function Filter

```
Latihan filter, 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 [16]:
```
# cara 1
faktor = lambda n: list(filter(lambda i : n%i == 0, range (1, n+1)))
primes  = lambda n: filter(lambda i: len (faktor(i))==2, range(1, 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
```

In [17]:
```
# cara 2
L = lambda n: range(2, n)
factor = lambda n: list(filter( lambda i: n%i == 0, L(n)))
primes = lambda n: filter( lambda i : len(factor(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
```

In [18]:
```
# cara 3
L= lambda n: range(2, n)
s_factor = lambda n: filter(lambda i : len(factor(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
```

In [22]:
```
# cara 4
L = lambda n : range(2, round(n**0.5)+1)
s_factors = lambda n: map(lambda i: i if n%i == 0 else 0, L(n))
primes = lambda n: filter( lambda i: sum(s_factor(i))== 0, range (2, n+1))

print(*primes(100))
```

In [23]:
```
s_factor = lambda n: map(lambda i: i if n%i==0 else 0, range (2, round(n**0.5)+1))
gjl = lambda n : n%2!=0
L = lambda n:filter( lambda i: sum(s_faktors(i))==0, L(n))
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
```

In [24]:
```
#Latihan Filter
employee= {'Nagao': 35,
           'ishii': 30,
           'Kazutomo':20,
           'Saito': 25,
           'Hidemi': 29}
print(employee.items())
```

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

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

```
Nagao ishii Hidemi
```

Pertemuan 10 Higher Order Function - Reduce

```
1. Buat fungsi mencari jumlah bilangan genap dari list L!

Contoh:
L = [2,1,9,10,3,90,15]
```

In [41]:
```
L= [2, 1, 9, 10, 11, 90, 3, 15]
r(lambda a, b: a+ (1 if b % 2 == 0 else 0), L, 0)
```

Out[41]: 3

```
2. Buat fungsi untuk menghitung n! Menggunakan reduce!
```

In [54]:
```
facto = lambda n: r(lambda a,b: a*b if b>1 else 1, range(1, n+1), 1)
for i in range(10+1):
    print( str(i) + '!=', facto(i) )

```

```
0!= 1
1!= 1
2!= 2
3!= 6
4!= 24
5!= 120
6!= 720
7!= 5040
8!= 40320
9!= 362880
10!= 3628800
```

In [42]:
```
from functools import reduce
n= 4
print(reduce(lambda x, y: x*y, range(1, n+1)))
```

```
24
```

```
3. Hitung euclidian distance dari dua vektor berikut menggunakan higher order
function!
X = [2,5,6,7,10]
Y = [-2,9,2,-1,10]
```

In [44]:
```
X = [2,5,6,7,10]
Y = [-2,9,2,-1,10]

eclid = lambda X, Y: r(lambda a, c: a+c, map(lambda x, y: (x-y)**2, X, Y))**0.5
eclid(X,Y)
```

Out[44]:
```
10.583005244258363
```

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

In [33]:
```
#Latihan Filter
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(),0)
cnt_emp(25, employee)
```

Out[33]:
```
3
```

```
5. Buatlah deret fibonacci menggunakan higher order function!
```

In [3]:
```
# NO 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[0]
```

In [7]:
```
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, 1597]
Fibonacci 18 -> [0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, 377, 610, 987, 1597, 2584]
Fibonacci 19 -> [0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, 377, 610, 987, 1597, 2584, 4181]
```

In [8]:
```
fibo(10)
```

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

In [1]:
```
# dengan menggunakan rekursive
fibo_rec = lambda n: 0 if n ==0 else 1 if (n==1 or n==2) else fibo_rec(n_1) + fibo_rec(n-2)
```

Pertemuan 11 Recursionin FP

In [39]:
```
#contohh
def factorial(x):
    if x == 1:
        return 1
    else:
        return (x * factorial(x-1))
num = 3
print("The factorial of", num, "is", factorial(num))
```

```
The factorial of 3 is 6
```

In [40]:
```
# Recursive Lambda
lambda_factorial = lambda i:1 if i==0 else i*lambda_factorial(i-1)

print(lambda_factorial(4))
```

```
24
```

```
1.Buat sebuah program untuk membuat deret fibonacci dari 0 hingga N dengan
menggunakan fungsi non-rekursif dan rekursif!
Bandingkan keduanya jika nilai N = 500, Manakah yang lebih baik? Jelaskan!
```

In [55]:
```
fibo_rec = lambda n: 0 if n==0 else 1 if (n==1 or n==2) else fibo_rec(n-1) + fibo_rec(n-2)
deret_fibo_rec = lambda n: list( map(lambda i: fibo_rec(i), range(n+1)))
deret_fibo_rec(10)
```

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

```
Jadi kesimpulannya dalam membuat fibonacci, lebih baik menggunakan non
recursive terlihat dari hasil running yang lebih cepat

```

Pertemuan 12 Purity and Immutabillity

In [43]:
```
# Contoh pure function
def min(x, y):
    if x < y:
        return x
    else:
        return y
```

In [46]:
```
# Contoh yang bukan pure function
def pows(L):
    for i in range(len(L)):
        L[i] = L[i]**exponent
    return L
```

In [48]:
```
# mengubah menjadi pure function
exponent = 2
L = [1,2,3,4,5]
print( 'pows(L): ' , pows( L ) )
print( 'L : ' , L )
Res = pows( L )
print( 'Res: ', Res )
print( 'L:' ,L )

```

```
pows(L):  [1, 4, 9, 16, 25]
L :  [1, 4, 9, 16, 25]
Res:  [1, 16, 81, 256, 625]
L: [1, 16, 81, 256, 625]
```

1. Ubah fungsiku menjadi pure function!

In [9]:
```
def fungsiku(L):
    def check_genap(l):
        return l% 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 [10]:
```
n = 3
L = [3, 5, 7, 8]
print (fungsiku(L))
```

```
[10, 16, 22, 4.0]
```

In [11]:
```
print(L)
```

```
[10, 16, 22, 4.0]
```

2. Ubah fungsiku2 menjadi Pure function

In [12]:
```
def fungsiku2(L):
    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 [13]:
```
n= 3
L= [5, 6, 7, 8]
print(list(fungsiku2(L)))
print(L)
```

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

3. Apakah isi dalam tupel tup ada yang dapat diubah? tup = ([3, 4, 5], 'myname')

In [57]:
```
tup = ([3, 4, 5], 'myname')
tup[0]=[3,2,3]
print("tup 2 :", tup)
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
~\AppData\Local\Temp/ipykernel_6500/899989142.py in <module>
      1 tup = ([3, 4, 5], 'myname')
----> 2 tup[0]=[3,2,3]
      3 print("tup 2 :", tup)

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

```
kesimpulan: type data tuple tidak dapat di ubah datanya,
kecuali langsung mengubah dari fungsinya/kodenya
```

Pertemuan 13 Function Building Function

```
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 [15]:
```
#a. fungsi komposisi
addku = lambda x:x +10
powku = lambda x: x**2
kurku = lambda x: x - 2 * x

f_com= lambda f, g: lambda x: f(g(x))
my_f_com = f_com(kurku, f_com(powku, addku))

my_f_com(10)
```

Out[15]: -400

In [16]:
```
#b. fungsi invers
inv_addku = lambda x:x +10
inv_powku = lambda x: x**2
inv_kurku = lambda x: x - 2 * x

my_f_kom_inv= f_com(inv_addku, f_com(inv_powku, inv_kurku))
my_f_kom_inv(-200)
```

Out[16]: 40010

```
c. Latihan Penentuan UKT Mahasiswa
```

In [50]:
```
Latihan Penentuan UKT Mahasiswa
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

1. Ketentuan Jumlah Tanggungan :
Jika lebih >= 5, maka skor = 1
Jika < 5 , maka skor = 5 - jumlah tanggungan

2. Ketentuan token listrik:
Jika rata-rata lebih dari 100 ribu per bulan , maka skor = 3
Jika diantara 50 ribu - 100 ribu per bulan , maka skor = 2
Jika dibawah 50 ribu , maka skor = 1

3. Ketentuan Gaji :
Jika gaji penanggung jawab > 10 juta maka skor = 7
Jika 8 < gaji <= 10 juta, maka skor = 6
Jika 6 < gaji <= 8 juta , maka skor = 5
Jika 4 < gaji <= 6 juta , maka skor = 4
Jika 3< gaji <= 4 juta , maka skor = 3
Jika gaji < 3 juta , maka skor = 2

4. Jika mahasiswa memiliki KIP-K , maka skor = 1, jika tidak maka skor = 5
Skor_total = 20 % * skor_1 + 30 % * skor_2 + 20% skor_3 + 30% skor_4
Jumlah bayar UKT = biaya pokok + skor_total * 500 ribu
Uang Pokok = 750 ribu

Gunakan fungsi komposisi untuk menyelesaikan masalah tersebut!
Hitung berapa biaya UKT yang harus dibayarkan dengan input sebagai berikut:
1. Jumlah tanggungan = 3
2. Listrik 3 bulan terakhir = 120 ribu , 75 ribu , 50 ribu
3. Gaji Penanggung jawab = 5.5 juta per bulan
4. Peserta KIP-K = Tidak

```

In [22]:
```
from functools import reduce as r

mycompose = lambda * funcs: r( lambda f, g:lambda x:f(g(x)), reversed(funcs), lambda x:x)
```

In [23]:
```
# ketentuan jumlah tanggungan
def skor1(jtg):
    return 1 if jtg >= 5 else 5-jtg
```

In [24]:
```
# 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)
```

In [25]:
```
# 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 [26]:
```
# ketentuan KIP-K
def skor4(X=True):
    return 1 if X else 5
```

In [27]:
```
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
```

In [28]:
```
mhs = [3,
     [120000, 75000, 50000],
     5.5 * 10**6,
     False
    ]

datas = [mhs, [skor1, skor2, skor3, skor4] ]
compose_fin = mycompose(combineskor, boboti, toUKT)
compose_fin(datas) /10**6
```

Out[28]: 2.4

d. Turunan Polinom

```
Turunan Polinom
Contoh input: "-3x^5 + 2x^2 + 4x + 5"

output : " 15.0x^4 + 4x -4 "

```

In [30]:
```
#Turunan Polinom

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^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 calctur(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]) +
               'x' if x[1]==1 else str(x[0]) +'x^' + str(int(x[1])))

def prettykan(dat):
    return r (lambda a,b : a+'+' + b if b != '0' else a, dat, '')

def prettysign(dat):
    return dat.replace('+-', ' -'). replace('+', '+')
```

In [17]:
```
dat = '-3x^5 + 2x^2 - 4x + 5'
fss = (splt,chdepan,eqkan,toarr2d,sortdesc,calctur,tostr,prettykan,prettysign)
my_turunan = mycompose(*fss)
```

In [16]:
```
splt(dat)
```

Out[16]: ['', '-3x^5', '2x^2', '-4x', '5']

```
e. 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 [64]:
```
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 [65]:
```
@calc_time_decorator
@pajak_decorator
def hitung_pembayaran_1(keranjang):
    return r(lambda a,b: a + (b['Jumlah_Barang'] * b['Harga']), keranjang, 0)
hitung_pembayaran_1(keranjang)
```

```
Sub Total:  280.0
Pajak:  30.8
Total:  310.8
Waktu eksekusi:  0.0010008811950683594
```

Out[65]:
```
280.0
```

In [68]:
```
@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:  280000.0
Pajak:  30800.0
Total:  310800.0
Waktu eksekusi:  0.010005474090576172
```

Out[68]:
```
280000.0
```

Latihan UAS

In [69]:
```
import numpy as np
from random import randint
from functools import reduce as r
```

In [70]:
```
def fungsiw(a, b):
    W= np.random.uniform(a, b, size=(3, 2))
    return W
W=fungsiw(-1, 1)
W
```

Out[70]:
```
array([[ 0.54260981,  0.07212758],
       [-0.01271604,  0.39396045],
       [ 0.97375111,  0.13003532]])
```

In [ ]:
```

```