Pemrograman Berbasis Fungsi – RA

TA Genap 2021/2022

Lecturer : Riksa Meidy Karim, S.Kom., M.Si., M.Sc.

NAMA : Muhammad Rivan Hasri

NIM : 120450079

Tugas Exercise

>> Exercise 9 >>

1. Hasil screenshot:

```
s_factor = lambda n: map(lambda i: i if n%i == 0 else 0, range(2, round(n**0.55)+1))

gjl = lambda n: n%2 != 0
L = lambda n: filter(lambda i: 2 if i==2 else gjl(i), range(2,n+1))

primes = lambda n: filter(lambda i: sum(s_factor(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
```

2. Hasil screenshot

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

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
```

>> Exercise 10 >>

1. Hasil tangkapan layar code:

```
from functools import reduce

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

print(reduce(lambda x,y: x+1 if y%2 == 0 else x, L, 0))
```

```
3
```

2. Hasil screenshot

```
n = 5
print(reduce(lambda x,y: x*y, range(1,n+1)))
```

```
120
```

3. Hasil screenshot

```
x = [2,5,6,7,10]
y = [-2,9,2,-1,10]

euclid = lambda x,y: reduce(lambda a,b: a+b, map(lambda x,y: (x-y)**2, x,y))**0.5
euclid(x,y)
```

```
10.583005244258363
```

4. Hasil screenshot

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

cnt_emp = lambda lim,employee: reduce(lambda a,b: a+1 if b[1] > lim else a, employee.items(), 0)
cnt_emp(25,employee)
```

```
3
```

5. Hasil screenshot

```
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 = lambda n: list( map( lambda x: fibo_rec(x), range(n+1) ) )
print(deret_fibo(20))
```

```
[0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, 377, 610, 987, 1597, 2584, 4181, 6765]
```

>> Exercise 11 >>

1. Sebelum itu, kita perlu tahu apa itu iterative solution (non-recursive) dan recursive.
   a. Iterative solution artinya kita tidak menggunakan paradigma recursive ataupun functional, kita hanya perlu menggunakan perulangan dan sebuah array. Seperti yang kita ketahui dari definisi Fibonacci, deret pertama dan kedua adalah 1 dan 0. Oleh karena itu, kita bisa menyimpannya dalam sebuah array sebagai pre-computed sequence.
   b. Recursive adalah suatu proses dengan salah satu langkah dalam prosedur tersebut menjalankan prosedur itu sendiri. Jika kita lihat lagi rumus Fibonacci, kita bisa melihat bahwa rumus tersebut adalah rekursif secara alami. Dalam kasus rekursif sendiri terdapat 2 hal, yaitu base condition (terminate condition) dan recurrence condition.

   Lalu, mana yang lebih baik? Rekursif atau non-rekursif? Untuk kasus ini, rekursif secara kompleksitas waktu lebih lambat disbanding iterative (non-recursive). Meskipun, pendekatan recursive jauh lebih sederhana akan tetapi ada hal serius yang perlu diperhatikan, yaitu recursive pada kasus ini akan menghitung bilangan Fibonacci dengan beberapa kali dibandingkan iterative. Hal itulah yang mendasari kenapa recursive dalam kasus ini lebih lambat disbanding iterative. Hasil screenshot code bisa dilihat berikut

```
def iterative_fibo(n):
    computed = [0,1]
    for i in range(2,n+1):
        next_num = computed[i-1] + computed[i-2]
        computed.append(next_num)
    return computed

sequences = iterative_fibo(500)
print(sequences)
```

```
[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, 1
21393, 196418, 317811, 514229, 832040]
```

```python
def iterative_fibo(n):
    computed = [0,1]
    for i in range(2,n+1):
        next_num = computed[i-1] + computed[i-2]
        computed.append(next_num)
    return computed

sequences = iterative_fibo(500)
print(sequences)
```

```
[0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, 377, 61
0, 987, 1597, 2584, 4181, 6765, 10946, 17711, 28657, 46368,
75025, 121393, 196418, 317811, 514229, 832040]
```

>> Exercise 12 >>

1. Hasil screenshot

```
# pure function
def fungsiku(L):
    def check_genap(l):
        return l % 2 == 0
    return list(map(lambda x: x/2 if check_genap(x) else x*n+1, L))

n = 3
L = [5,6,7,8]

print(fungsiku(L))
print(L)
```

```
[16, 3.0, 22, 4.0]
[5, 6, 7, 8]
```

2. Hasil screenshot code:

```
# pure function
def fungsiku2(L):
    def check_faktor(l):
        return l % n == 0
    return list(map(lambda x: x/2 if check_faktor(x) else x*n+1, L))

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

```
[16, 3.0, 22, 25]
[5, 6, 7, 8]
```

3. Tuple bersifat immutable, artinya tidak dapat diubah. Namun, pada kasus tersebut untuk elemen listnya dapat kita ubah sedangkan untuk list nya sendiri tidak dapat kita ubah ke dalam bentuk lain. Begitu pula untuk ‘myname’ tidak dapat kita ubah. Hasil screenshot

>> Exercise 13 >>

1. Hasil screenshot fungsi komposisi

```
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)
```

```
-400
```

Hasil screenshot fungsi 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)
```

```
10.0
```

2. Hasil screenshot fungsi ketentuan jumlah tanggungan:

```
from functools import reduce as r

# Define function composition
mycompose = lambda *funcs: r(lambda f,g: lambda x: f(g(x)), reversed(funcs), lambda x: x)

# Ketentuan jumlah tanggungan
def val1(jtg):
    return 1 if jtg >= 5 else 5-jtg
```

Hasil screenshot fungsi ketentuan token listrik

```
# Ketentuan token listrik
def val2(x):
    def rata_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_val2(x):
        return r(lambda a,b: a + (1 if b == True else 0), x[1], 1)

    compose_cond = mycompose(rata_rata,l_cond_1,l_cond_2,to_val2)
    return compose_cond(x)
```

Hasil screenshot fungsi ketentuan gaji:

```
# 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_val(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 val3(gj):
    comp = mycompose(prep,con_1,*(con_2_to_n for i in range(4)), to_val)
    return comp(gj)
```

Hasil screenshot fungsi ketentuan KIP-K:

```
# Ketentuan KIP K
def val4(x=True):
    return 1 if x else 5

def gabung_val(x):
    return x + [map(lambda f,x: f(x), x[1], x[0])]

def bobot(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 total_ukt(x):
    return 750000 + x*500000
```

Hasil screenshot penentuan UKT:

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

data = [mhs, [val1, val2, val3, val4]]
comp_final = mycompose(gabung_val,bobot,total_ukt)
comp_final(data) / 10**6
```

```
7.75
```

3. Hasil screenshot code:

```
def split(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 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^' + str(x[1]), dat)

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

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

```
dat = '-3x^5 + 2x^2 -4x +5'
fss = (split, chdepan, eqkan, toarr2d, sortdesc, calctur, tostr, prettykan, prettysign)
my_turunan = mycompose(*fss)
my_turunan(dat)
```

```
' -15.0x^4.0+ 4.0x^1.0 -4.0'
```

4. Hasil screenshot code:

```
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 * 0.01)
        print('Total: ', res + res * 0.01)
        return res
    return inner

import time

def calc_time_decorator(func):
    def inner(*args, **kwargs):
        start = time.time()
        res = func(*args, **kwargs)
        end = time.time()
        print('Time: ', end - start)
        return res
    return inner
```

```
@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:  2800.0
Total:  282800.0
Time:  0.0009856224060058594

280000.0
```