**Nama :** Eureka Diaandisy  
**NIM ;** 202110370311345  
**Kelas :** Pemrograman Fungsional H

LATIHAN MODUL 4

# Kegiatan 1

# Higher Order Function (HOF)

def perkalian(a):

    def dengan(b):

        return a \* b

    return dengan

hasil\_hof = perkalian(5)(3)

print(hasil\_hof)

# Currying

def perkalian\_currying(a):

    return lambda b: a \* b

hasil\_currying = perkalian\_currying(5)(3)

print(hasil\_currying)

Higher Order Function

* Fungsi ‘perkalian’ menerima satu argumen yaitu ‘a’ dan fungsi ini me-return fungsi ‘dengan’ yang menerima argumen ‘b’ dan fungsi ini me-return hasil perkalian ‘a \* b’
* ‘hasill\_hof = perkalian(5)(3)’ memanggil fungsi ‘perkalian’ dengan argumen ‘5’, yang kemudian me-return fungsi ‘dengan’ dengan argumen ‘3’ dan menghasilkan hasil dari fungsi ‘dengan’ dengan argumen ‘a \* b’ atau ‘5 \* 3’
* Output = ‘15’

Currying

* Fungsi ‘perkalian\_currying’ mengambil argumen ‘a’ dan me-return fungsi lambda yang mengambil satu argumen ‘b’ dan me-return hasil ‘a \* b’
* ‘hasil\_currying = perkalian\_currying(5)(3)’ menghasilkan hasil dari pemanggilan fungsi lambda dengan argumen ‘3’, yang juga setara dengan ‘5 \* 3’
* Output = ‘15’

# Kegiatan 2

def uppercase\_decorator(function):

    def wrapper():

        func = function()

        return func.upper()

    return wrapper

@uppercase\_decorator

def say\_hi():

    return 'hello there'

result = say\_hi()

print(result)

* ‘uppercase\_decorator’ adalah fungsi dekorator. Fungsi ini mengambil satu argumen, yaitu fungsi ‘(function)’ yang akan didekorasi.
* Fungsi ‘wrapper’ yang ada di dalamnya memanggil fungsi yang didekorasi ‘(function)’, menyimpan hasilnya dalam variabel ‘func’, dan kemudian me-return versi huruf besar ‘upper()’ dari nilai yang dikembalikan oleh ‘func’
* Fungsi ‘say\_hi’ di atas didekorasi dengan menggunakan ‘@uppercase\_decorator’. Ini berarti fungsi ‘say\_hi’ akan diubah oleh fungsi dekorator sebelum digunakan.
* Setelah didekorasi, pemanggilan ‘say\_hi()’ sebenarnya memanggil wrapper yang ditambahkan oleh dekorator.
* Fungsi wrapper kemudian memanggil fungsi asli ‘say\_hi()’ dan me-return hasilnya dalam huruf besar.
* Output = 'HELLO THERE'

# Kegiatan 3

def title\_decorator(function):

    def wrapper():

        func = function()

        make\_title = func.title()

        print(make\_title + " " + " —Data is converted to title case")

        return make\_title

    return wrapper

def split\_string(function):

    def wrapper():

        func = function()

        splitted\_string = func.split()

        print(str(splitted\_string) + " " + "— Then Data is splitted")

        return splitted\_string

    return wrapper

@split\_string

@title\_decorator

def say\_hi():

    return 'hello there'

result = say\_hi()

* Fungsi ‘title\_decorator’ merupakan decorator yang mengambil fungsi lain ‘(function)’ sebagai argumen.
* Di dalamnya, ada fungsi inner ‘wrapper()’ yang pertama kali memanggil fungsi yang didekorasi ‘function()’.
* Hasil dari fungsi tersebut ‘func’ yang kemudian diubah menjadi title case menggunakan metode ‘.title()’.
* Hasilnya dicetak bersamaan dengan pesan yang menjelaskan bahwa data telah diubah ke title case.
* Fungsi ini me-return hasil dari perubahan title case.
* Fungsi ‘split\_string’ merupakan decorator yang kurang lebih irip dengan fungsi ‘title\_decorator’
* Hasil dari fungsi tersebut ‘func’ kemudian diubah menjadi list menggunakan metode ‘.split()’.
* Hasilnya dicetak bersamaan dengan pesan yang menjelaskan bahwa data telah dipecah menjadi list.
* Fungsi ini me-return hasil dari perubahan menjadi list.
* Fungsi ‘say\_hi’ didekorasi terlebih dahulu oleh ‘title\_decorator’ dan kemudian oleh ‘split\_string’.
* Maka ‘say\_hi’ akan diproses terlebih dahulu oleh ‘title\_decorator’, dan hasilnya akan menjadi argumen untuk ‘split\_string’.

# Kegiatan 4

import math

def translasi(tx, ty):

    def transformasi(x, y):

        return x + tx, y + ty

    return transformasi

def dilatasi(sx, sy):

    def transformasi(x, y):

        return x \* sx, y \* sy

    return transformasi

def rotasi(sudut):

    def transformasi(x, y):

        sudut\_rad = math.radians(sudut)

        x\_baru = x \* math.cos(sudut\_rad) - y \* math.sin(sudut\_rad)

        y\_baru = x \* math.sin(sudut\_rad) + y \* math.cos(sudut\_rad)

        return x\_baru, y\_baru

    return transformasi

# Contoh kasus

titik\_awal = (3, 5)

# Translasi

translasi\_func = translasi(2, -1)

titik\_setelah\_translasi = translasi\_func(\*titik\_awal)

print(f"Koordinat setelah translasi: {titik\_setelah\_translasi}")

# Dilatasi

dilatasi\_func = dilatasi(2, -1)

titik\_setelah\_dilatasi = dilatasi\_func(\*titik\_awal)

print(f"Koordinat setelah dilatasi: {titik\_setelah\_dilatasi}")

# Rotasi

rotasi\_func = rotasi(30)

titik\_setelah\_rotasi = rotasi\_func(\*titik\_awal)

print(f"Koordinat setelah rotasi: {titik\_setelah\_rotasi}")

Fungsi ‘transformasi’

* Dalam program ini, ada tiga fungsi transformasi utama: ‘translasi’, ‘dilatasi’, dan ‘rotasi’.
* Setiap fungsi transformasi menerima parameter-parameter spesifik untuk transformasi yang diinginkan, yaitu tx dan ty untuk translasi, sx dan sy untuk dilatasi, serta sudut untuk rotasi.

Translasi

* Fungsi ‘translasi’ me-return fungsi ‘transformasi’ yang menambahkan tx ke koordinat x dan ty ke koordinat y.

Dilatasi

* Fungsi ‘dilatasi’ me-return fungsi ‘transformasi’ yang mengalikan koordinat x dengan sx dan koordinat y dengan sy.

Rotasi

* Fungsi ‘rotasi’ me-return fungsi ‘transformasi’ yang merotasi koordinat (x, y) sejauh sudut tertentu (dalam derajat) menggunakan rumus rotasi matriks.
* Sebuah titik P (3, 5) diberikan sebagai contoh kasus.
* Tiga transformasi (translasi, dilatasi, rotasi) diterapkan pada titik P menggunakan fungsi-fungsi transformasi yang telah didefinisikan.
* Hasil dari setiap transformasi dicetak menggunakan f-string.
* Output menunjukkan koordinat titik setelah masing-masing transformasi.
* Output = ‘Koordinat setelah translasi: (5, 4)  
   Koordinat setelah dilatasi: (6, -5)  
   Koordinat setelah rotasi: (0.09807621135331646, 5.830127018922194)’

# Kegiatan 5

def point(x, y):

    return x, y

def line\_equation\_of(p1, p2):

    def calculate\_slope(p1, p2):

        return (p2[1] - p1[1]) / (p2[0] - p1[0])

    m = calculate\_slope(p1, p2)

    def calculate\_intercept(p, m):

        return p[1] - m \* p[0]

    c = calculate\_intercept(p1, m)

    return f"y = {m:.2f}x + {c:.2f}"

# Titik A dan B

A = point(1, 3)

B = point(4, 5)

# Persamaan garis yang melalui titik A dan B

equation\_AB = line\_equation\_of(A, B)

print("Persamaan garis yang melalui titik A dan B:")

print(equation\_AB)

Fungsi ‘line\_equation\_of(p1, p2)’

* Fungsi ini menghitung persamaan garis lurus yang melalui dua titik, p1 dan p2.
* Fungsi inner ‘calculate\_slope(p1, p2)’ digunakan untuk menghitung gradien (slope) m antara dua titik.
* Fungsi inner ‘calculate\_intercept(p, m)’ digunakan untuk menghitung nilai intercept c pada sumbu y.
* Hasil akhir berupa string yang menyatakan persamaan garis dalam bentuk umum y = mx + c.

Perhitugan

* Gradien (slope), m, dihitung menggunakan rumus: , dengan (1, 3) dan (4, 5) sebagai koordinat titik.
* Intercept, c, dihitung menggunakan rumus: , dengan (x, y) sebagai koordinat salah satu titik.
* Output = ‘y = 0.67x + 2.33’

# Kegiatan 6

def point(x, y):

    return x, y

def line\_equation\_of(p, M):

    x1, y1 = p

    def calculate\_intercept(x1, y1, M):

        return y1 - M \* x1

    C = calculate\_intercept(x1, y1, M)

    return f"y = {M:.2f}x + {C:.2f}"

# Titik (3, 4) dan gradien 5

p = point(3, 4)

M = 5

# Persamaan garis yang melalui titik (3, 4) dan bergradien 5

equation = line\_equation\_of(p, M)

print("Persamaan garis yang melalui titik (3, 4) dan bergradien 5:")

print(equation)

Fungsi ‘line\_equation\_of(p, M)’

* Fungsi ini menghitung persamaan garis lurus yang melalui suatu titik (p) dan memiliki gradien (slope) M.
* Koordinat titik (x1, y1) dari titik p diambil menggunakan unpacking.
* Fungsi inner ‘calculate\_intercept(x1, y1, M)’ digunakan untuk menghitung nilai intercept c pada sumbu y.
* Hasil akhir berupa string yang menyatakan persamaan garis dalam bentuk umum y = mx + c.

Perhitungan

* Intercept, C, dihitung menggunakan rumus: ​, dengan (3, 4) sebagai koordinat titik dengan gradien (5).
* Output = ‘Persamaan garis yang melalui titik (3, 4) dan bergradien 5:  
   y = 5.00x + -11.00’