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

LATIHAN MODUL 4

Kegiatan 1

```python
# 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

```python
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: $m = \frac{y2-y1}{x2-x1}$, dengan (1, 3) dan (4, 5) sebagai koordinat titik.

- Intercept, c, dihitung menggunakan rumus: $c = y - mx$, dengan (x, y) sebagai koordinat salah satu titik.
- Output = ‘y = 0.67x + 2.33’

Kegiatan 6

```python
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: $C = y1 - M\ .x1$, 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’